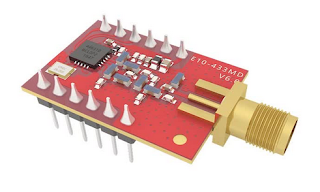
<http://strefapic.blogspot.com/2019/04/si4463-transciver-rf-firmy-silicon-labs.html>

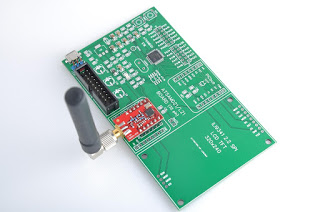
**SI4463 - Transceptor RF Silicon Labs para aplicaciones de batería. Primero ejecute en PIC32MM de Microchip.**

*[](https://4.bp.blogspot.com/-Mz8UnG7DBzk/XLSacjiqWJI/AAAAAAAAGBE/JTcHWKu64JcTdNl0Xrs93m0qm2JShCuuQCLcBGAs/s1600/si4463_obrazek.png)*

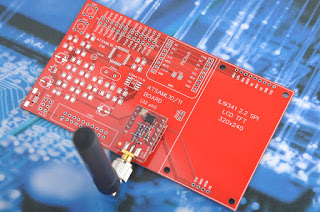
*Después de un día de éxito con la radio****SI4432****compañía de****Silicon Labs****decidió sondear una versión mejorada del módulo con la designación*[***SI4463***](https://www.silabs.com/documents/public/data-sheets/Si4463-61-60-C.pdf)*. Fue muy por delante por número :). Este módulo se basa en el diseño exitoso y bien revisado de****HC12,****es decir****,****el puerto de radio****UART de****largo alcance.*

*En el módulo*[***SI4463***](https://www.silabs.com/documents/public/data-sheets/Si4463-61-60-C.pdf)*, la parte de radio se*[***ha***](https://www.silabs.com/documents/public/data-sheets/Si4463-61-60-C.pdf)*mejorado significativamente, la sensibilidad aumentó a****126 dBm****, la velocidad de transmisión se incrementó cuatro veces a****1 Mbps****. Modos de modulación:****2GFSK****y****4GFSK se han ampliado****. Además, tenemos tiempos de****activación más****rápidos y opciones de****activación****más amplias. Sensor de temperatura y monitoreo de voltaje de suministro como estándar. El módulo con potencia máxima****+ 20dB****transportará fácilmente datos a una distancia de más de****2 km****. Tengo aquí la confirmación de este hecho de dos fuentes independientes. Uno de ellos se puede encontrar en los enlaces debajo del artículo.****30-50nA corrientes****latentes. A modo de comparación, el módulo****nRF24L01 +****, popular entre nosotros,  extrae****900 nA****en modo de suspensión . Módulos*[***SI4463***](https://www.silabs.com/documents/public/data-sheets/Si4463-61-60-C.pdf)*compró sin problemas en el mercado del Lejano Oriente para la letra****A ....****. . Me gustaría basar una construcción inteligente de ahorro de energía en estos módulos. Actualmente los estoy agregando a mis nuevos mosaicos de desarrollo.*

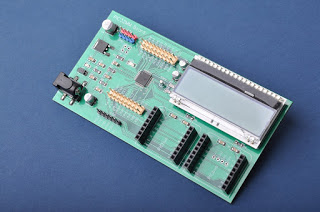
*A continuación, se muestra una imagen de cómo se ve el módulo en mi placa de desarrollo para* ***ATSAMD21***

*[](https://1.bp.blogspot.com/-NeiFC06oEuc/XK-IEyAk7RI/AAAAAAAAF9I/D9SSZfb5lAM-QudrObnqZsxjpYVV3PIrACLcBGAs/s1600/si4463.JPG)*

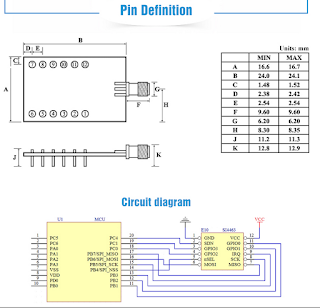
***ATSAML10 / 11***

*[](https://2.bp.blogspot.com/-_EAxoWAIDM4/XLRivx8SRTI/AAAAAAAAF_I/DjkQ6fNC98wDwmSqDkWwiyQL2pdV-cmvgCLcBGAs/s1600/atsaml11.JPG)*

*Por ahora, sin embargo, ejecuto el módulo en mi placa de desarrollo para****PIC32MM****. Esa es la forma más fácil para mí, primero pruebe el software en****PIC32****y luego transfiéralo a****ARM****. A continuación se muestra el tablero para****PIC32MM****en el que haré mis primeras pruebas. Hay****PIC32MM0256GPM048****en el****tablero****:*

*[](https://3.bp.blogspot.com/-IoUDakw-hyU/XLTFaFYAJGI/AAAAAAAAGBQ/4Tinr-RjYEAZ5os0blCp6XB4UfgkApRLACLcBGAs/s1600/pic32mm.JPG)*

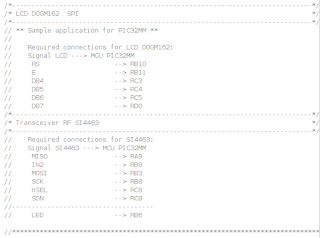
*Al principio, una descripción de los cables y el tamaño del módulo de****RF****que tengo:*

*[](https://1.bp.blogspot.com/-Men9E_3AhUU/XLBGnJ69aVI/AAAAAAAAF9g/6Bu6MFO-3hIPnqPucfB5JEIPpEiAhWmOgCLcBGAs/s1600/so4463_pin.png)*

*El módulo es realmente pequeño para la "power" que representa :)  
Antena dedicada para****433 MHz****comprada por separado pero del mismo proveedor que el módulo de radio. La ganancia de energía dada por el productor del Lejano Oriente es de****2.5 dBm,****pero según yo es una cal y el parámetro real es de aproximadamente****1.5 dBm****. No menos hecho es estéticamente agradable e incluso diría que es bonito.*

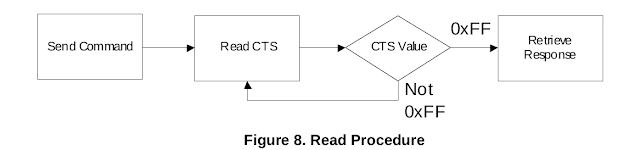
*[](https://4.bp.blogspot.com/-vJnXdHDzCnU/XLYOs8JpHFI/AAAAAAAAGDk/MsorgW4aLCMm4xpKK164n-rzMeJnsBh2gCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-04-16%2B19-19-05.png)*

*Para controlar el módulo,* ***usaré SPI2****y pinologías como se muestra a continuación:*

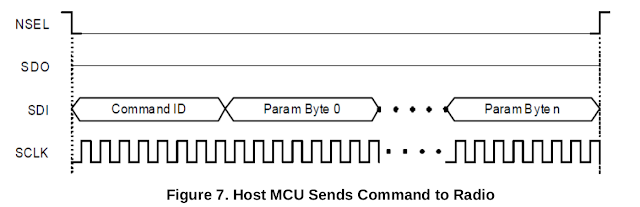
*[](https://4.bp.blogspot.com/-TKNO2l3Gdc4/XMV89rGBWbI/AAAAAAAAGJk/sQYQoVMA0Ekm6aXCXZ3ju8_yud_AsuSQgCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-04-28%2B12-13-48.png)*

*La capa de hardware en****PIC32MM****, es decir, la configuración del reloj, pines,****SPI2****, etc. se realiza utilizando****MCC****en****MPLABX-IDE 5.15****. El módulo de****RF****se puede perseguir después de un****SPI de 10 MHz****. Pero estos son al comienzo de los asuntos secundarios primero, necesitamos reconocer cómo será la función para intercambiar datos después de****SPI****con el módulo****RF****. Bueno, aquí no haremos ningún esfuerzo y utilizaremos la función de listo que ofrece****MCC****. El nombre de esta función es****SPI2\_Exchange8bit****() y lo encontraremos en el archivo****spi2.c****generado automáticamente por****MCC****. En esta función baso todo el intercambio de datos****SPI****con nuestro módulo****RF,****es decir, transmitiendo y recibiendo al mismo tiempo.*

*Ahora debe saber aproximadamente cómo es la estructura de los datos transferidos desde / hacia el módulo****RF****. Y aquí nos sorprenderemos porque es completamente diferente de los que conocemos de los módulos más antiguos de la serie****SI4432****.*

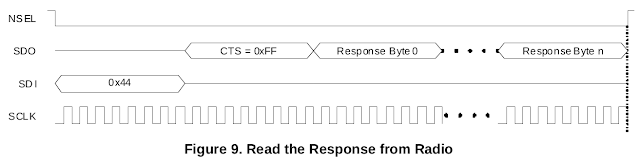
*[](https://1.bp.blogspot.com/-mLnVLdRfjls/XLRopnABjdI/AAAAAAAAF_g/SBtmR8hk46swIRMIQ4aZHf1Y-JblY0GawCLcBGAs/s1600/si4463_ctsread1.png)*

*La imagen de arriba muestra la idea general del flujo de datos. El comando se envía primero. El comando puede contener argumentos que enviamos directamente después del comando, que se simboliza en la imagen a continuación:*

*[](https://1.bp.blogspot.com/-A_BEVTgJ9Ww/XLRptslF_NI/AAAAAAAAF_s/4bmNY8icerYR4Uo3W1CtLj-81mRyV2YSQCLcBGAs/s1600/si4463_ctsread2.png)*

*Después de enviar el comando con argumentos (si los hay), el módulo de****RF****procesa los datos recibidos y establece la confirmación del final de su procesamiento.Después de la confirmación, el módulo de****RF****enviará un flujo de respuestas (si corresponde para el comando dado). Tenemos tres métodos de confirmación a nuestra disposición:*

***Método de software****: consiste en enviar (después del comando con argumentos, vea la imagen anterior) al módulo****RF****una secuencia de dos bytes, uno****0x44****(comando****READ\_CMD\_BUFF****) y el otro lo que sea, por ejemplo,****0x00.****Si el comando enviado anteriormente se procesa positivamente, en el segundo byte de la secuencia enviada al módulo****RF,****es decir****,****a****0x00****, recibiremos del módulo un byte de confirmación****0xFF****(****CTS = 0xFF****) seguido de un flujo de datos de respuesta.*

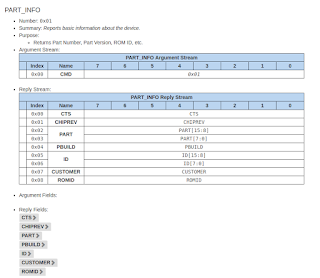
*[](https://4.bp.blogspot.com/-DCEJEbVOP2M/XLRuTH4Xq1I/AAAAAAAAF_4/z1TMebsBQX4COqxHPCmkaR6YrxSMTWYKwCLcBGAs/s1600/si4463_ctsread3.png)*

***Método GPIO****: consiste en mostrar un estado, por ejemplo, alto, como confirmación en un pin configurado para esto.*

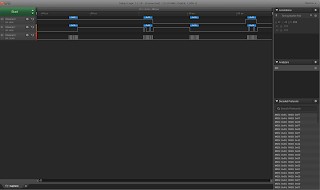
***Método de interrupción****: consiste en generar una interrupción como señal de confirmación.*

*Pero cuidado, no todos los comandos requieren confirmación. Los que lo requieren tienen en la****API****en las tablas de la sección****Flujo de respuesta****el primer byte****0x00****descrito como****CTS****. En otros casos no se requiere confirmación.*

*Para una mejor comprensión del tema de la confirmación, mostraré cómo se ve en señales reales en el caso de enviar al módulo****RF****, por ejemplo, el comando****PART\_INFO****(hex****0x01****)****.****Se puede encontrar una lista y una descripción de todos los comandos en el enlace*[***SI4463 API rev C2***](http://www.silabs.com/documents/public/application-notes/EZRadioPRO_REVC2_API.zip)*. La estructura del comando****PART\_INFO****se ve así:*

*[](https://2.bp.blogspot.com/-vjXSxjxr0_M/XLSK3pWW8SI/AAAAAAAAGAE/mEU75DAbbMcooXTmat-TuZukGb5k3XIrwCLcBGAs/s1600/si4463_ctsread5.png)*

*El envío del comando****PART\_INFO****comienza con el código de este comando, es decir, el envío a la línea****MOSI 0x01****. El comando****PART\_INFO****no tiene argumentos, solo devuelve 8 bytes (byte No.****0x00****-****CTS****no se cuenta ni se omite). Es decir, después de enviar el byte****0x01****que representa el comando****PART\_INFO****enviar de inmediato la solicitud de reconocimiento de secuencia / o estado de los siguientes dos bytes enviados a la línea****MOSI****es****0x44 y 0x00****. Si se envía el byte****0x00****get en la línea****MISO****byte****0x00****significa que el comando****PART\_INFO****aún no se ha procesado y el flujo de comentarios aún no está listo para enviarse a la****MCU****. Esta es la situación en la imagen a continuación:*

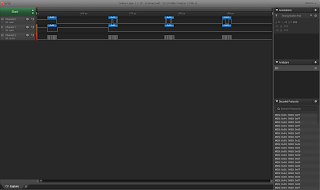
*[](https://1.bp.blogspot.com/-PG4k960raR4/XLSNU6StNpI/AAAAAAAAGAQ/dwJalUkeOnUeCSuV-MAfXExKNzrJtdW4QCLcBGAs/s1600/si4463_info.png)*

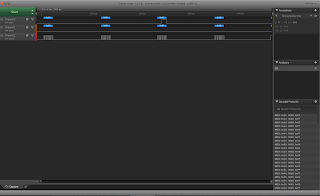
*Dado que el módulo de****RF****aún no procesa los datos , debemos solicitar nuevamente el estado de confirmación, es decir, enviar la secuencia****0x44 y 0x00 nuevamente****a la línea****MOSI****. En la segunda secuencia es enviada al byte****0x00****tenemos en la línea****MISO 0x00****. Somos tercos y enviamos la tercera secuencia donde finalmente obtenemos el byte de confirmación****CTS = 0xFF****.*

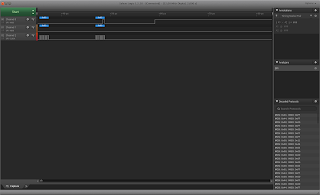
*[](https://2.bp.blogspot.com/-0lHjwHrwRbE/XLSOnoi38AI/AAAAAAAAGAc/Pon-QmB5oXU0qf2Pzy2VFxu53s5THqg4ACLcBGAs/s1600/si4464_info1.png)*

*La confirmación debe aparecer a****0x00****bytes y no a****0x44****. Ahora esperamos que los siguientes bytes que se emitirán en la línea****MISO****sean el flujo de datos de retorno que recibiremos del módulo. Entonces obtenemos****8 bytes****en la línea****MISO****:*

***0x11, 0x44, 0x63, 0x00, 0x00, 0x0F, 0x00, 0x03****y el final del flujo de datos de retroalimentación. Los dibujos a continuación lo muestran:*

*[](https://3.bp.blogspot.com/-8bxLvlWgKW0/XLSPlgcgTZI/AAAAAAAAGAo/ZRqs61LdaE460eqYIDFb-pTXIukNywE3wCLcBGAs/s1600/si4464_info2.png)*

*[](https://1.bp.blogspot.com/-6_WM3wdswIw/XLSPp8hIO4I/AAAAAAAAGAs/cGLxbreZq-wUOEIOXEQats5q87FsbGHZgCLcBGAs/s1600/si4464_info3.png)*

*[](https://2.bp.blogspot.com/-tsU6k2z1zAM/XLSPvbZwwSI/AAAAAAAAGAw/LNeT8ZRLq74uKri_pbY0rFYFHyTIjOdVgCLcBGAs/s1600/si4464_info4.png)*

*No pude ajustar toda la transmisión en una imagen, por eso se dividió en 5 partes arriba. Lo que significan los bytes individuales de los argumentos devueltos por el módulo****RF****, vea el*[*enlace*](http://www.silabs.com/documents/public/application-notes/EZRadioPRO_REVC2_API.zip)*con la lista de comandos. En total, la información más importante es el byte 2 y 3 o****0x44, 0x63,****que forma sospechosamente el modelo de nuestro módulo****RF****o****SI 4463****. Desde aquí sabemos que nuestro hermano del Lejano Oriente no nos hizo un globo y vendió el modelo de módulo que queríamos tener :)*

*Si el concepto de confirmación aún no se comprende, me remito a la*[***Guía de programación***](https://www.silabs.com/documents/public/application-notes/AN633.pdf)*y allí debe leer este problema. Tampoco comencé desde la primera mirada hasta que se iluminó el segundo intento. Y se ve mejor en el analizador, solo necesita tener una biblioteca escrita primero :). Tal vez esta confirmación persistente parece ser algo molesto, en cierto modo. Pero estamos tratando aquí con la confirmación de la operación realizada por el módulo de****RF,****es decir, adquirimos la confianza de que el comando se ejecutó con éxito y esta es una característica positiva en sistemas con un alto grado de confiabilidad requerida. En total, si la biblioteca se basa en la confirmación del estado de la señal en el pin****GPIO****del módulo****RF****entonces podemos aumentar la conveniencia de actuar de esta manera. El módulo****RF****tiene cuatro pines****GPIO****configurables . Pero le aseguro que la biblioteca presentada por mí para****SI4463****implementará fácilmente la confirmación del software de la operación realizada por el módulo****RF****.*

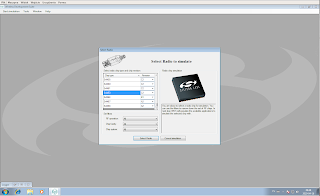
*Al final del tema, mencionaré que el módulo de****RF****tiene cuatro****registros de respuesta rápida****configurables  , que no requieren el procedimiento de esperar la confirmación de la operación. Podemos conectar las operaciones realizadas con mayor frecuencia y el caso resuelto bajo estos registros. Me gusta****SI4463****cada vez más y aún no he comenzado a trabajar con él.*

*Terminamos el problema y vamos más allá. Ahora configuraremos el módulo****RF****. Tenemos aquí*[***WDS***](https://www.silabs.com/documents/login/software/WDS3-Setup.exe)*elegante*[***- Suite de desarrollo inalámbrico***](https://www.silabs.com/documents/login/software/WDS3-Setup.exe)*preparada por****Silicon Labs****. El único inconveniente es la operación solo después del elevador. Debido a que me separé del ascensor hace mucho tiempo, tuve que instalar****Virtual Box****en mi****sistema Linux****y poner el Elevator en él, pero en general no fue un problema porque usaremos****WDS****algunas veces y nos olvidaremos de eso. Configuramos el módulo****RF****usando****WDS****de acuerdo con nuestras propias visiones y finalmente generamos un archivo de configuración que adjuntamos a nuestra biblioteca. Admito que esta es una solución muy conveniente. El programa es simple e intuitivo de usar, no causará ningún problema e incluso causará mucha diversión. Pasemos juntos por la etapa de configuración porque la pila es más brillante :). Si al principio no tenemos nuestra propia visión en cuanto a la configuración de, por ejemplo, la ruta de radio porque no la conocemos, podemos apoyarnos con personas más inteligentes que nosotros (no es una pena que sean así :)) y, por ejemplo, usar visiones prefabricadas desde este*[*enlace*](https://github.com/zkemble/Si446x/blob/master/WDS/README.md)

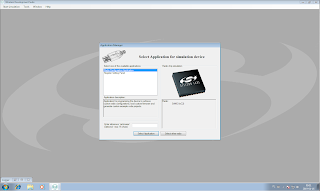
*Después de instalar la aplicación****WDS****en****Windows, la****ejecutamos.*

*[](https://3.bp.blogspot.com/-95VxUbcGDWs/XLWISP7BkrI/AAAAAAAAGCA/X6Stzl6k05YcRRGoAEo4jr21_HVodY5NgCLcBGAs/s1600/wds1.png)*

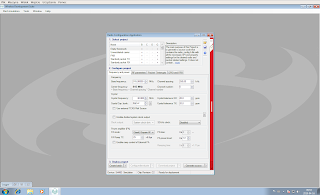
*Seleccione la opción****Iniciar simulación****en la barra superior . En la ventana que aparece, encontramos****SI4463****y lo marcamos.*

*[](https://2.bp.blogspot.com/-4-5quY32gnQ/XLWJHFzDxnI/AAAAAAAAGCM/e5rqK-brRcQsHL6dLAr6jdqZyIhUiXVLgCLcBGAs/s1600/wds2.png)*

*En la parte inferior de la ventana interna, "haga clic" en****Seleccionar radio****, aparecerá otra ventana (ver a continuación) en la que "hacemos clic" en****Seleccionar aplicación****.*

*[](https://4.bp.blogspot.com/-po5QUGR0gtQ/XLWJxBolctI/AAAAAAAAGCU/HOj3CSw0nYMRyhBSan8KhNWcjSVt-j-tQCLcBGAs/s1600/wds3.png)*

*La culminación de nuestro "clic" es la aparición de la ventana de configuración principal de nuestro módulo****RF****:*

*[](https://1.bp.blogspot.com/-aPvVA-8QdpU/XLWKbta7CaI/AAAAAAAAGCc/pI9XkEXkkw8ppZQmxQfjJgwy-3JuRf-6wCEwYBhgL/s1600/wds4.png)*

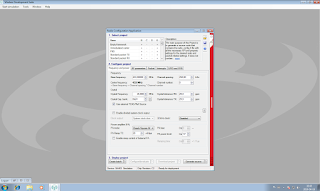
*En general, la aplicación****WDS****tiene muchas más posibilidades que la configuración pasiva en sí, pero supongamos que no nos damos cuenta de que nuestra cabeza no duele por el exceso de novedad :) Nos remangamos y comenzamos a jugar configurando el módulo****RF****. Es bueno que me guste :)*

*Para comenzar, necesitamos reconocer dos parámetros en qué frecuencia base está sintonizado nuestro módulo, en otras palabras, qué frecuencia de portadora usa y qué cuarzo tiene a bordo (por defecto es un cuarzo interno de****30 MHz****). En nuestro caso son****433 MHz****y cuarzo externo****26 MHz****. El cuarzo externo, según el vendedor, es de alta precisión y pequeña deriva de temperatura, si esto es cierto, entonces el módulo debería representar una alta cultura de trabajo en términos de parámetros de frecuencia y esta es la base para un funcionamiento estable de la ruta de****RF.****Cabe señalar aquí que el módulo puede funcionar en el****rango de 425-525 MHz****y tiene****255****disponiblescanales. Si quisiéramos trabajar en, por ejemplo,****255****canales, nuestra frecuencia de funcionamiento sería de****250 kHz****por canal de****496,75 MHz****. En la configuración nos quedamos en el canal****0****. Por lo tanto establecemos:*

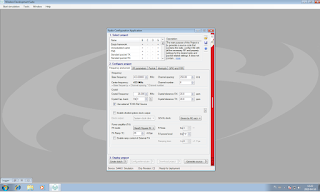
***Frecuencia base = 433 MHz****,*

***Frecuencia de cristal = 26 MHz ,***

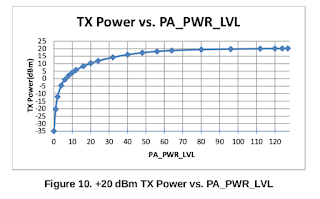
*Debido a que utilizamos cuarzo externo, utilice la opción "****Usar TCXO externo / Fuente de referencia****" . Nuestra configuración actual se presenta en la siguiente imagen:*

*[](https://2.bp.blogspot.com/-Gl0r5as2JKA/XLWV3iLZ3LI/AAAAAAAAGCo/qDzyr4wD16chZe_snGJHFfGgHzPJDfUOQCLcBGAs/s1600/wds6.png)*

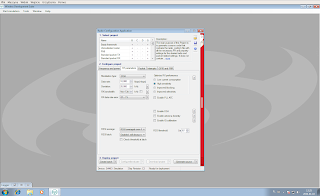
*La última configuración que haremos en la pestaña****Frecuencia y Potencia****es seleccionar****Impulsado por un oscilador RC****para****32kHz****. Oscillator****RC 32kHz****será útil, por ejemplo, para despertarse. A continuación se muestra un resumen de todas nuestras configuraciones en la pestaña****Frecuencia y Potencia****:*

*[](https://3.bp.blogspot.com/-HpYfVrBQmgs/XLWtIAqqARI/AAAAAAAAGC0/MVKLXWhVSQc_0zX4e2LtxTnj3bvYlPvEwCLcBGAs/s1600/wds7.png)*

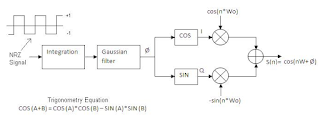
*Mencionaré la configuración de potencia. Vemos que el número máximo para ella en la ventana de****nivel de potencia de PA****es****0x7F****o****127****, la potencia mínima corresponde al dígito****0****. Desafortunadamente, no existe una fórmula simple para asignar un dígito en particular al nivel de potencia porque es un valor no lineal. Posiblemente pueda apoyar aquí los gráficos que encontramos en la hoja de datos. Sabemos que la potencia máxima es de****+20 dBm****y la mínima es de****-20 dBm****. Es cierto que este es un rango relativamente amplio de cambios de poder. Dejamos esta configuración en el valor máximo, es decir,****0x7F****.*

*[](https://3.bp.blogspot.com/-yjC4ajzjWkQ/XLgywjgc-4I/AAAAAAAAGGg/_DUP0zyGKkURa1phL1dAEUPHLd2yGxYxwCLcBGAs/s1600/wds26.png)*

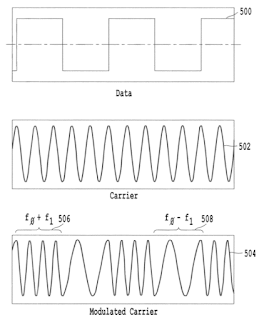
*Es hora de cambiar la pestaña porque la actual ya está aburrida :). Cambiamos a la pestaña****Parámetros de RF****y encontramos la imagen allí:*

*[](https://3.bp.blogspot.com/-JoHtjOL4guY/XLW75I8rQWI/AAAAAAAAGDA/2N_ODqNWNpg-DDep5SFP8C22m3D-NDzUwCLcBGAs/s1600/wds8.png)*

*Lo primero que configuramos es el tipo de modulación a****2GFSK****. Supongo que no somos expertos en tipos de modulación. Sin embargo, vale la pena saber que la modulación****GFSK****está asociada con el Sr.****Gauss****y es una muy buena opción si, por ejemplo, estamos limitados por el ancho de banda y queremos usar una gran cantidad de canales dentro de él. Un rasgo característico de esta modulación es, entre otros, una rápida disminución del espectro en los bordes. Y la señal****0****o****1****se crea mediante una desviación de frecuencia hacia la izquierda o hacia la derecha, por ejemplo,****-50 kHz****para****0****y****+50 kHz****para****1****. En el lado del modulador, un diagrama de bloques para la modulación****GFSK****se ve así:*

*[](https://2.bp.blogspot.com/-hHRbhFSALec/XLXAQWDqboI/AAAAAAAAGDM/-pO0zGzzy78zZh42tSP2j0ceCUC4HDtigCLcBGAs/s1600/wds9.png)*

*El espectro que obtenemos a la salida del modulador es la "campana" del Sr.****Gauss****. Y el principio pictórico de convertir una señal digital en componentes de frecuencia en la figura a continuación:*

*[](https://3.bp.blogspot.com/-SviAnBK0IuY/XLXCX0adonI/AAAAAAAAGDY/fnZUhDSJlWEKVS-EBXmEQaGw46EXigM8ACLcBGAs/s1600/wds10.png)*

***La****modulación****2GFSK****es una optimización, entre otros, en el rango de ancho de banda, lo que significa que, por ejemplo, en el mismo ancho de banda enviaremos más datos que para la modulación****GFSK****o la misma cantidad de datos en el medio ancho de banda. Vale la pena señalar que nuestro módulo también tiene modulación****4GFSK****. Pero no exageraremos con el recorte y la optimización del ancho de banda en términos de velocidad de bits binaria porque todo tiene su precio y, en este caso, por ejemplo, una disminución de la sensibilidad en la recepción.*

*Ahora enfocamos nuestra atención en tres parámetros:****velocidad de datos****,****desviación****y****ancho de banda RX****.****La velocidad de datos****es la velocidad binaria, es decir, cuántos bits volarán en una unidad de tiempo.****La desviación****es la distancia entre dos portadoras y las frecuencias de modulación.****RX Bandwidth****es el ancho de banda que será utilizado por un canal de transmisión. Los parámetros anteriores son aproximados por la fórmula:*

***BW = 2Fd + Rb****,*

***BW****-****Ancho de banda RX***

***Fd - Desviación de frecuencia***

***Rb - Velocidad de datos***

*La fórmula es útil para determinar los tres parámetros de radio enumerados anteriormente para la modulación****GFSK****. El punto de partida para nosotros es la****velocidad de datos****binarios , si queremos obtener grandes rangos de transmisión, este parámetro debe acortarse, algo por algo. Supongamos que no necesitamos una tasa de bits binaria alta porque no enviamos grandes flujos de datos. Supongo que****10 kbps****está dimensionado para mis necesidades no demasiado altas. El segundo paso es especificar el parámetro de****desviación****para la tasa de bits binaria seleccionada. Aquí el principio es simple, dividimos la tasa de bits binaria por****2****. Desde aquí obtenemos****10 kbps / 2 = 5 kHz****. Si la tasa de bits binaria fue, por ejemplo,****500 bps****, establecemos la desviación en****250 Hz****.*

*El último paso es establecer el ancho de banda o el ancho de banda****RX****. Lo calculamos a partir de la fórmula anterior para****BW****. Entonces en nuestro caso:*

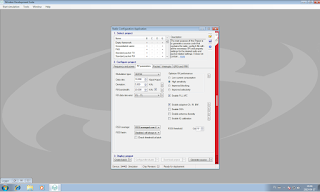
***BW****=****2****\*****5 (kHz)****+****10 ( kbps) = 20 kHz***

*Las unidades aquí pueden fascinar que agregue****kbps****y****kHz****y salga de este****kHz****. Pero es importante y no puedo evitarlo :). Sin embargo, debe tenerse en cuenta que hay una opción en el programa para calcular automáticamente el ancho de banda (sin proporcionar el valor de cálculo) pero quiero tener control sobre él y calcularlo yo mismo.*

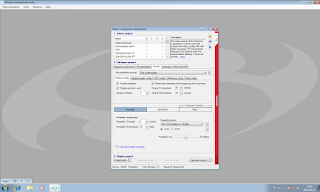
*Por lo tanto, hemos seleccionado de manera óptima y consciente los parámetros básicos para la ruta de radio. En general, estoy disfrutando este juego de diseñador de "ondas de radio". Como regla general, las personas que juegan con módulos de radio evitan entrar en los detalles de los parámetros de radio, a veces incluso estos parámetros están ocultos para el usuario como un cuadro negro. Tiramos el conejo a la entrada y tenemos paté listo en la salida. Todo es genial, pero así es como crece la generación de consumidores mentalmente entrenados y perezosos. Ese es un ciudadano y consumidor ideal para la elite gobernante. Digo este estado definitivamente no y donde sea que pueda tratar de entrar en detalles y usar el cerebro para que mis neuronas no desaparezcan en él.*

*Entonces, por curiosidad, miré los parámetros de radio del módulo****nRF24L01 +****, que es popular entre nosotros  . En comparación con el módulo****SI4463****, el módulo****nRF24L01 +****se ve muy pálido. De hecho, en cada punto de los parámetros de radio,****SI4463****supera al competidor y la única ventaja del módulo****nRF24L01 +****es el precio, y por esta razón probablemente se deba a la popularidad de estos módulos en nuestro mercado.*

*Aplicamos nuestra configuración en el programa. También verificamos las****opciones Activar PLL AFC****y****Activar Ch.Fil.BW adaptativo****.****AFC****(****Control automático de frecuencia****) es la sintonización automática de la frecuencia a la señal recibida. Los filtros de recepción se sintonizan y los parámetros se mantienen en términos de resonancia. Si usamos****AFC,****necesitamos extender el preámbulo, pero vale la pena hacerlo para mejorar la recepción. Resumamos nuestra configuración en la pestaña****Parámetros de RF****:*

*[](https://1.bp.blogspot.com/-RbEjb4JWKUM/XLbmNlxvVDI/AAAAAAAAGDw/1dV8ST7GuOs7w-NUezmsrGWk-JY0uwSGwCLcBGAs/s1600/wds11.png)*

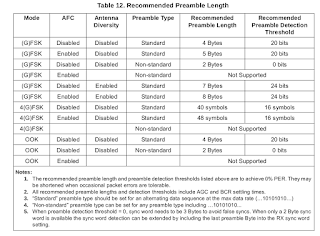
*Vamos a la siguiente pestaña****Paquete****, el estado que encontramos a continuación:*

*[](https://4.bp.blogspot.com/-9KQ4eriZ9y0/XLbmqRTOaiI/AAAAAAAAGD4/VCthXeu_xMESaBtuppt1iEfoauO2R22NgCLcBGAs/s1600/wds12.png)*

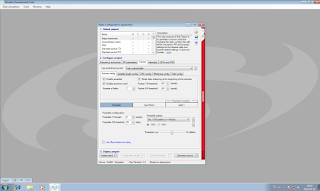
*En esta pestaña, modelamos la imagen y los parámetros del paquete de datos que se envía. Las posibilidades de modelar la estructura del marco son realmente impresionantes. La configuración inicial es el campo de****Preámbulo****, el campo con la palabra de sincronización****SyncWord****y el contenedor de datos del campo****1****.*

*Primero, nos enfocamos en la****configuración****del preámbulo, el preámbulo es el comienzo de la trama y existe la secuencia****1010****repetida a una longitud definida . Esto es para sintonizar el receptor con la señal transmitida e informarle que la trama de datos está volando :)*

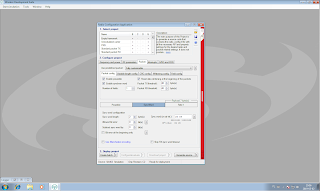
*Para establecer el preámbulo, para no hacerlo en un inconveniente, podemos usar la tabla con la hoja de datos:*

*[](https://3.bp.blogspot.com/-JDdhbY3W1XE/XLbttSEkwwI/AAAAAAAAGEE/ICc7IRAQUAUyJ9gnaHpMhw-Rv0ccyo50wCLcBGAs/s1600/wds13.png)*

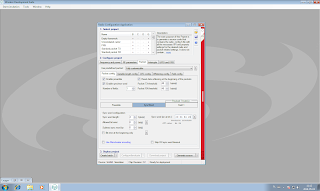
*Recordamos que tenemos****AFC****habilitado. Sin embargo, la tabla no tiene datos para la modulación****2GFSK****que hemos elegido. Pero no se preocupe por este hecho, ciertamente la longitud del preámbulo debe estar en el nivel de modulación****GFSK****y podemos tomar una reserva contra ella, es decir, la longitud en lugar de****5 bytes****para dar****8 bytes****. Nuestro objetivo para la brecha se ve así:*

*[](https://4.bp.blogspot.com/-DJEDt1h-XBg/XMqwMHqrsJI/AAAAAAAAGKk/XaHGyBBDJew56AR7eHHDIdMBO4xhsupFQCLcBGAs/s1600/wds36.png)*

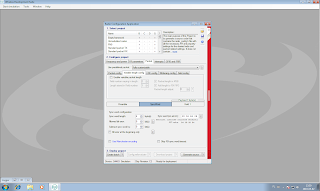
*en total, como encontramos al principio.  
Haga clic en el campo****SyncWord****y mire la palabra de sincronización:*

*[](https://3.bp.blogspot.com/-Ru5qIN5F9PE/XMqyswZa0WI/AAAAAAAAGKw/wS9nXrY3P8k1hONHtHB8s_7_KrFGX5uFACLcBGAs/s1600/wds37.png)*

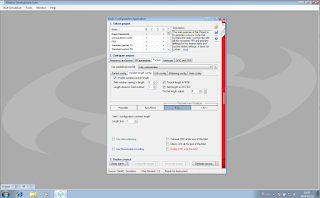
*Podemos ver dos bytes****2D****y****D4****. La palabra de sincronización puede tener un tamaño de hasta****4 bytes.****Usaremos este hecho y expandiremos la palabra existente con dos bytes adicionales a****2D D4 B4 2B.****Para hacer esto, cambie el valor en el campo****Sincronizar longitud de palabra****a****4****y agregue dos bytes adicionales****B4****y****2B****. El efecto a continuación:*

*[](https://1.bp.blogspot.com/-3ilNXpuyodY/XMq0mwFj5tI/AAAAAAAAGK8/EkGy3jwNsXsKlbH8xZY4ZqxFlIwVDGG8ACLcBGAs/s1600/wds38.png)*

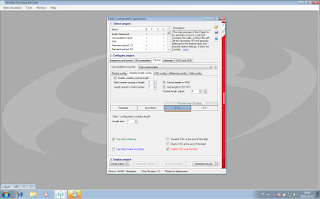
*Porque nuestro marco es demasiado pobre y el módulo le permite modelarlo. Utilizaremos esta opción y ampliaremos nuestro marco, incluido un campo adicional en el que enviaremos información sobre la longitud de los datos transmitidos. Para hacer esto, cambie de la pestaña****Configuración de paquete****a la pestaña****Configuración de longitud variable****. La vista existente a continuación:*

*[](https://2.bp.blogspot.com/-bS35n-rZH5E/XMrWoMYRBCI/AAAAAAAAGLI/J0howmRJZZwHulHPf6KgFwKsI87IohsKgCLcBGAs/s1600/wds39.png)*

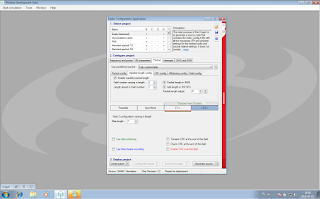
*Marcamos la opción****Habilitar longitud de paquete variable****.  
Podemos ver que la estructura de nuestro marco ha cambiado ahora que tenemos, aparte de los campos****Preámbulo****y****Sincronización****, dos campos****F1L****y****<F2>****. El campo****F1L****es un contenedor en el que se****enviará****información sobre la longitud del paquete de datos en el campo****<F2>****. Haz clic en los gráficos del campo****F1L****y mira lo que tenemos allí.*

*[](https://2.bp.blogspot.com/-MX6pdmJvwjI/XMsDSb2T6YI/AAAAAAAAGLU/rkZ5Novv_o0tUAPUOw9rPRllJWHiL1Q1ACLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-02%2B16-47-35.png)*

*Podemos ver que el campo se establece en una longitud fija de****1 byte****y así lo dejamos. Marque la opción****Usar blanqueamiento de datos****. Este mecanismo convierte el estado alto o bajo en mucho tiempo en datos transmitidos en secuencias de cambios cortos de señal****0/1****. Esto da como resultado una mejor imagen de espectro. Subrayado en verde, los campos****F1L****significa que el mecanismo de****blanqueamiento****se utilizará en este campo. También queremos que el mecanismo de suma de verificación****CRC****tenga en cuenta el campo****F1L****. Marcamos el rojo****Habilitar CRC sobre esta****opción de****campo****. El efecto de la siguiente manera:*

*[](https://1.bp.blogspot.com/-Aw4pRkafmfE/XMsFWf7WcvI/AAAAAAAAGLg/KnhR6ix-ZOMYmUAalZ2FTA-8FoJ8RLLqgCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-02%2B16-56-41.png)*

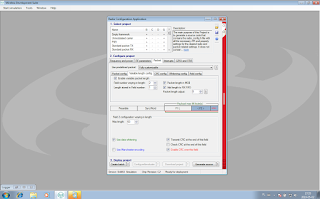
*Ahora es el momento de buscar en el campo****<F2>****. Haga clic en el icono con este símbolo. La vista que encontramos a continuación:*

*[](https://3.bp.blogspot.com/-u6x0yIknEmI/XMsFxyw_EcI/AAAAAAAAGLo/FQLMNAxtmUUhvF9YAnF5pFUtqxvjK_XQACLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-02%2B16-58-33.png)*

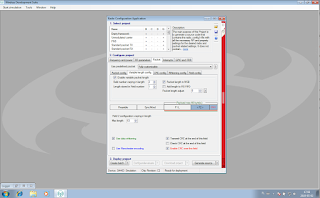
*El****campo <F2>****es un contenedor para nuestros datos pero de longitud variable. Longitud variable porque físicamente el tamaño del campo en la trama transmitida depende de la cantidad de información enviada, si enviamos, por ejemplo,****4 bytes de****datos, entonces la trama de****63 bytes****no vuela (porque esta es la longitud máxima que declararemos en un momento) incluyendo nuestros****4 bytes****con datos, pero solo una trama con una longitud de****4 bytes****. Es decir, la longitud de la trama transmitida físicamente cambia según la cantidad de datos transferidos. De ahí el campo****<F2>****es un campo de longitud variable con nuestros datos. Esta es una funcionalidad muy interesante del módulo. Por supuesto, si tenemos tal solicitud, podemos establecer el campo como una longitud fija, independientemente de la cantidad de información enviada, el marco a lo largo de toda la longitud se enviará incluso con campos vacíos. Establezcamos la longitud máxima de nuestro contenedor en datos, es decir, campos****<F2>****. El valor máximo disponible es****63 bytes****y establecemos este valor en el campo****Longitud máxima****.****1 byte****fue robado por el campo****F1L****. Aquí, una pequeña digresión, el módulo tiene la capacidad de enviar más de****64 bytes.****datos en un cuadro y no estoy hablando de transmisión de datos aquí porque también es posible. Tenemos la capacidad de enviar hasta****128 bytes****en un cuadro, pero no lo configuraremos en el programa****WDS****, debemos hacerlo manualmente configurando la opción adecuada. A continuación muestro un fragmento de la****API****donde configuraremos esta opción:*

*[](https://3.bp.blogspot.com/-Qt9TOPWEfVI/XMsJVZGY8bI/AAAAAAAAGL0/ZOOWzlT08HgQIYSYcSLyBXXmaT1eFTYaACLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-02%2B17-12-10.png)*

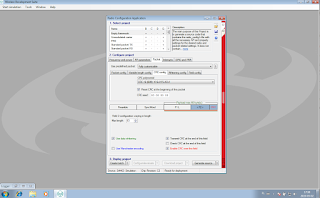
*Volvemos a la configuración de campo****<F2>****. Seleccione la opción****Usar blanqueamiento de datos****, queremos que este campo también tenga este mecanismo. Luego, conecte el mecanismo****CRC****marcando la****casilla Habilitar CRC sobre este campo****. Además, seleccione la opción****Transmitir CRC al final de este campo****. Esto hará que la suma de verificación****CRC****se inserte después del campo****<F2>****. El efecto final de la configuración del campo****<F2>****. Como abajo:*

*[](https://4.bp.blogspot.com/-M7U66wIDce8/XMsOdoaDJ1I/AAAAAAAAGMA/-RkT0TBNaQcmgnUdd3Qwti5t9T2UYIaJgCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-02%2B17-35-55.png)*

*En la vista de pestaña de****configuración de longitud variable****, desmarque la opción****Agregar longitud a RX FIFO****. Si he entendido correctamente, desmarcar esta opción hará que el flujo de datos recibido, que estará en el búfer de recepción de hardware****RX FIFO****del  módulo, omitirá los datos sobre la longitud de los datos recibidos. El punto es descargar solo los datos del búfer****RX FIFO****, no otros datos técnicos sobre el marco. ¿Por qué necesitamos datos de longitud de datos perdidos en los datos mismos? Pero esto no significa que no tendremos acceso a los datos con la longitud de los datos recibidos.*

*[](https://4.bp.blogspot.com/-9i9aDiTyGFg/XMsTR74MJiI/AAAAAAAAGMU/JYUJc-FSGkITQDHNkkchh6yVe1y68pQ3ACLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-02%2B17-56-43.png)*

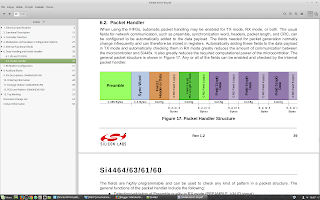
*Por si acaso, mostraré la configuración de****CRC****, para hacer esto, haga clic en la pestaña de****configuración de CRC****y la imagen debería estar a continuación:*

*[](https://1.bp.blogspot.com/-SY7Rvw85TQk/XMsT9OnyJyI/AAAAAAAAGMc/RyLVNEGdeicnPRuHRLIAVUNo-spo42__QCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-02%2B17-59-36.png)*

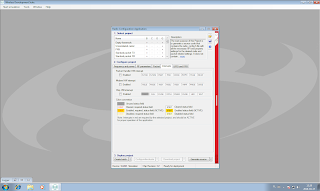
*Resumamos la estructura física de nuestra trama de transmisión, que consistirá en:*

* ***Preámbulo****-****8 bytes***
* ***SyncWord****-****4 bytes****(esto se puede usar para direccionar dispositivos)*
* ***F1L****(número de datos transmitidos físicamente en el campo****<F2>****) -****1 byte***
* ***<F2>****(campo de datos) -****63 bytes***
* ***CRC - 2 bytes***

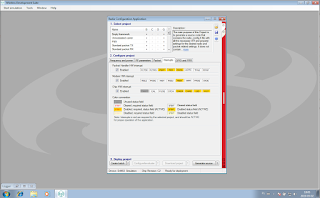
*Entonces, si no cometí un error al agregar, la longitud total de la trama transmitida es de****78 bytes****, no es un golpe :)  
Vale la pena mencionar que la estructura de la trama puede ser mucho más complicada y esto se muestra en la siguiente imagen:*

*[](https://2.bp.blogspot.com/-NKUU5pmCpQI/XLczgKYvnjI/AAAAAAAAGF8/CRY9j5Bl3L0pBHlUkMiwqMP5E47CMkjlACLcBGAs/s1600/wds23.png)*

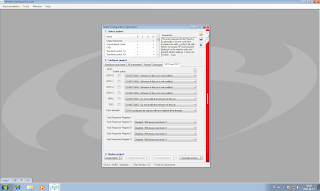
*El único límite en el diseño del marco es nuestra imaginación y posiblemente una limitación en la forma de un máximo de****5****bloques funcionales separados dentro de los****64 bytes****disponibles . Podemos, por ejemplo, solicitar un bloque separado para la dirección del dispositivo, la dirección del mensaje como en****CAN,****por ejemplo, y para cada uno de estos bloques un****CRC****separado, etc. La flexibilidad en la configuración de la estructura del marco es un punto fuerte de****SI4463****.  
  
Terminamos las operaciones en la pestaña****Paquete****y pasamos a la siguiente pestaña****Interrupciones****configurando interrupciones. La configuración existente en la imagen a continuación:*

*[](https://1.bp.blogspot.com/-A7obJSjECek/XLcOlm6TU1I/AAAAAAAAGFk/D-UTv4HxmEMXcNod4hbLfuVNRIB_htBQgCLcBGAs/s1600/wds21.png)*

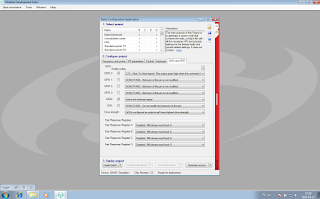
*Nuestro módulo tiene un pin marcado como****nIRQ****y reconoceremos este pin, por ejemplo, si se ha recibido o dado un marco ¿Y por qué necesitamos información sobre el marco dado? para, por ejemplo, poner el módulo en suspensión después de la transmisión. Sería estúpido dormir durante la transmisión. Hay muchos posibles eventos a los que responderá la interrupción, pero nos enfocamos en incluir algunos básicos. Mi sugerencia para habilitar las interrupciones a continuación:*

*[](https://2.bp.blogspot.com/-ph-sef2xWu0/XMsVT7MVCLI/AAAAAAAAGMo/s4NHpRMbRNoLE6aDE0jv9Cv8THe8aea9QCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-02%2B18-05-15.png)*

*Nos estamos acercando lentamente al final de todas las configuraciones. Cambiamos la pestaña a****GPIO y FRR****. En esta pestaña configuramos, entre otras, funciones para pines****GPIO****(****GPIO0 ... GPIO3****) y para registros****FRR****rápidos , que no requieren verificar el estado de la operación procesada (****CTS****).*

*[](https://2.bp.blogspot.com/-NiRUAKwKCl8/XLc1xNDXtrI/AAAAAAAAGGI/xglmanpFl0svc_-k2d3IrmHLonOg9NLQgCLcBGAs/s1600/wds24.png)*

*Cuando abrimos la lista de funcionalidades para, por ejemplo****, GPIO0,****tomaremos nuestra cabeza tanto como podamos. No temas el perro de la madre, el hombre es un hombre se acercó y lo abrazan :)  
Me puse****GPIO0****para dar la señal****CTS****y****NIRQ****para interrumpir una seña el estado bajo y****el pull-up****para****nIRQ****. El resto en esta etapa no lo toco hasta que reconozco lo que aún puede ser útil para mí y mi conocimiento aún es incompleto a este respecto.*

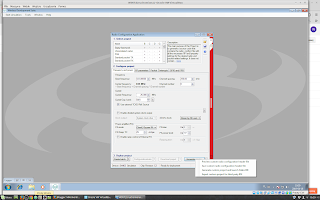
*[](https://4.bp.blogspot.com/-brdYYhm1dmw/XMR1b_WeZcI/AAAAAAAAGJM/3ANa_dkLbg8frJuB50t6aiJNhhwjdRFbACLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-04-27%2B17-29-39.png)*

*Bien hecho, llegamos al final de****la****configuración de nuestro módulo****RF****. Si alguien llegó a este lugar, lo felicito sinceramente :). La configuración de nuestro módulo****RF****en el programa****WDS****es realmente muy cómoda y rápida, solo escribiendo sobre eso :). Te sugiero que no te desanimes si aún no entiendes algo.****Vale la****pena aprender el módulo****SI4463****porque fue creado por una compañía estadounidense que tiene una amplia experiencia con módulos****RF****y más. Vale la pena mencionar que****Silicon Labs****tiene en su cartera uno de los módulos****[WiFi más](https://www.silabs.com/products/wireless/wi-fi/wi-fi-transceivers" \t "_blank)****ahorradores de energía del mundo, pero ya es una curiosidad natural. Bueno, pero incluso el módulo que ahorra más energía****WiFi****no tiene ninguna posibilidad con nuestro****SI4463 de todos modos****.*

*Todavía miro al final de nuestro conjunto o cualquier przeoczyłem.Rzuca pestaña mis ojos****frecuencia y potencia****el valor de la****separación de canales = 250 kHz****. Con nuestro ancho de banda de****20 kHz****es*

*demasiado pero lo dejamos como está.*

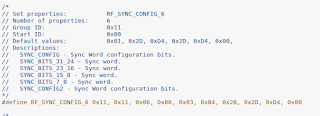
*La culminación de nuestro trabajo es generar un archivo de configuración en el****lenguaje C****, que se unirá a nuestra biblioteca. Para generar este archivo, haga clic en la parte inferior derecha****Generar fuente****. Saltamos fuera de la caja, seleccione la opción de****archivo de cabecera de configuración de radio personalizada Guardar****.*

*[](https://4.bp.blogspot.com/-AAHRyie3r58/XLg4_5b_HFI/AAAAAAAAGGs/2wL_k8BaoAMYCBiXc5aK-_Hgps-zYqQ7gCLcBGAs/s1600/wds27.png)*

*Se****creará****un archivo llamado****radio\_config\_Si4463.h****en el que está contenida toda la configuración de nuestro módulo. Esta configuración debe cargarse en el módulo en el proceso de inicialización de nuestro programa. Y se pone muy simple. Por ahora, sin embargo, necesitamos descifrar dos cosas. En primer lugar, cuál es el formato para enviar propiedades / comandos individuales y cómo extraer datos de una estructura de matriz tan extraña en el formulario****#define****. Primero, trataremos el formato de los comandos y las propiedades que envía, puede encontrar una descripción de este formato en la*[***Guía de programación***](https://www.silabs.com/documents/public/application-notes/AN633.pdf)

*[](https://3.bp.blogspot.com/-ScZo7g1cXbg/XLg-xBEd1yI/AAAAAAAAGG4/NA54yiOi578hVrnUe5lfgtmR3TqOQgBhACLcBGAs/s1600/wds28.png)*

*Ahora correlacionemos con lo que vemos en el archivo de encabezado****radio\_config\_Si4463.h****para el grupo de propiedades seleccionado. Elegí el grupo de propiedades como se muestra a continuación , configurando, entre otros,****SyncWord.****Recordamos que nuestra configuración para la palabra de sincronización es****2D D4 B4 2B***

*[](https://3.bp.blogspot.com/-cl3z-_B3VhE/XLhIR3IvoWI/AAAAAAAAGHE/yD1HLwoohnYdeDuOcyQARpTv2Y82IZmCACLcBGAs/s1600/wds29.png)*

*En la imagen de arriba vemos el formato en el que se construyen los siguientes bytes del comando enviado y las propiedades. Nuestro conjunto de propiedades de muestra se describe como****RF\_SYNC\_CONFIG\_6****. Intentemos descifrar este cifrado. Primero,****veamos****la propiedad específica de la palabra****SyncWord****desde adentro, para esto necesitamos abrir el*[***SI4463 API rev C2***](http://www.silabs.com/documents/public/application-notes/EZRadioPRO_REVC2_API.zip)*descargado o la lista de comandos y propiedades de****API****. Encontramos nuestra propiedad allí bajo el nombre****Sync\_Bits***

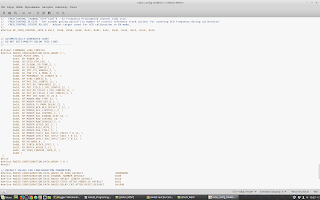
*[](https://4.bp.blogspot.com/-kMgerHUSS28/XLhS9prRQzI/AAAAAAAAGHQ/_-yCS-k64gw_9gaaXMopDIvGNY5qbKoZACLcBGAs/s1600/wds30.png)*

*En la parte superior de la tabla tenemos la estructura de bits de nuestra propiedad y en la parte inferior los****valores predeterminados****.  
Bueno, volvemos a nuestro cifrado, que queremos descifrar, es decir:****#define RF\_SYNC\_CONFIG\_6 0x11****,****0x11****,****0x06****,****0x00****,****0x03****,****0xB4****,****0x2B****,****0x2D****,****0xD4****,****0x00***

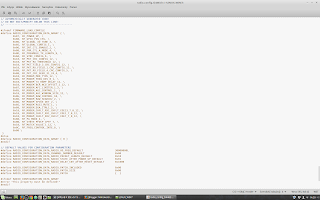
*¿Y cuál es el comando aquí y cuál es la propiedad, etc.?  
  
Byte 1 =****0x11****-> código de comando****SET\_PROPERTY****( Establece el valor de una o más propiedades), lo encontraremos en la lista de comandos. Comando para establecer una o varias propiedades.  
Byte No. 2 =****0x11****->****ID de grupo*** *Byte No. 3 =****0x06****->****Número de propiedades****(el número de propiedades****6,****es decir****,****lo que comienza con****SYNC\_CONFIG****hacia abajo )  
Byte n. ° 4 =****0x00****->****Start ID*** *Byte n. ° 5 =****0x03****->****SYNC\_CONFIG*** *Byte n. ° 6 =****0xB4****->****SYNC\_Bits*** *Byte n. ° 7 =****0x2B****->****SYNC\_Bits*** *Byte n. ° 8 =****0x2D****->****SYNC\_Bits*** *Byte n. ° 9 =****0xD4****->****SYNC\_Bits*** *Byte No. 10 =****0x00****->****SYNC\_CONFIG2***

*El ejemplo anterior se refiere al caso cuando establecemos algo y luego usamos el comando****SET\_PROPERTY (0x11)****. ¿Qué pasa si queremos leer los valores establecidos? luego use el comando****GET\_PROPERTY (0x12)****y envíe otros 3 bytes. Se vería así:  
  
Byte No. 1 =****0x12****->****GET\_PROPERTY****( Obtiene el valor de una o más propiedades) código de comando , lo encontraremos en la lista de comandos. El comando para preguntar acerca de la configuración de propiedad  
Byte No. 2 =****0x11****->****ID de grupo*** *Byte No. 3 =****0x06****->****Número de propiedades****(el número de propiedades que pedimos es****6****)  
Byte No. 4 =****0x00****->****Start ID*** *El resultado será una secuencia datos compuestos de****6 bytes****como respuesta. Recuerde, sin embargo, la señal de confirmación****CTS****.  
  
Al principio puede parecer un poco confuso, pero no se agrietará. Recién comencé cuando escribí el artículo sobre lo que está sucediendo aquí y es realmente simple cómo entiendes :) Para estar más cerca de entender, deberías haber abierto el archivo de configuración****radio\_config\_Si4463.h****y la lista de comandos y****las****propiedades de la****API****y parpadear tanto tiempo que las chanclas en el cerebro se abran :)*

*Bueno, ahora descifremos lo que está en la definición****#define RADIO\_CONFIGURATION\_DATA\_ARRAY****{}*

*[](https://4.bp.blogspot.com/-bnThU90jujA/XLhfMsOaodI/AAAAAAAAGHc/uVpxmrc9qFArIOBkie98R-ra8E5ssLvXgCLcBGAs/s1600/wds32.png)*

*En esta extraña y pequeña definición de la matriz, se contiene toda la esencia de todas las configuraciones y debemos enviar esta matriz con un truco en el proceso de inicialización del módulo. Primero, sin embargo, eliminaremos de esta definición el primer elemento descrito como****SI446X\_PATCH\_CMDS****, no jugamos el módulo "****parchear****". Un formulario de definición de matriz con el primer elemento eliminado:*

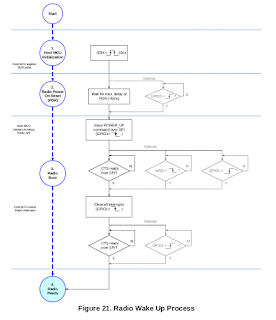
*[](https://1.bp.blogspot.com/--TEfNZhZrTU/XLhruBI18aI/AAAAAAAAGHo/y-zZhbcA6lsFot_iPLhiRtPrwqVmNiEQwCLcBGAs/s1600/wds34.png)*

*Veamos el fragmento en el que vemos que el****RF\_SYNC\_CONFIG\_6****considerado anteriormente****se****ve así: ....****0x0A****,****RF\_SYNC\_CONFIG\_6****....*

*Hasta donde sabemos, cómo se ve la definición de****RF\_SYNC\_CONFIG\_6****, pero cuál es el byte****0x0A****que está en****10 de****diciembre . Este es el número de bytes que se enviarán, en este caso es cuántos bytes contiene****RF\_SYNC\_CONFIG\_6****, si contamos, obtendremos****10****. Ot y toda la filosofía.   
Ahora, cómo****reemplazar la****definición de tabla****RADIO\_CONFIGURATION\_DATA\_ARRAY****{} del archivo de encabezado con una tabla que podemos usar normalmente en el programa, se muestra a continuación:****uint8\_t Radio\_Conf\_Array [] = RADIO\_CONFIGURATION\_DATA\_ARRAY ;***

*¿No es eso fácil? Bueno, tan fácil como sabes hacerlo :)*

***ATENCIÓN****cosa muy importante.****WDS****genera un archivo de configuración por defecto preparado para "****parchar****", eliminamos algunos elementos asociados con él. Pero al momento de iniciar el módulo, descubrí que en la definición de****Power Up****en el segundo byte está el valor****0x81****, que informa al módulo que necesita "parchear", esto efectivamente bloquea la posibilidad de inicializar el módulo . Debemos cambiar este valor manualmente a****0x01****. Me llevó casi****tres días****detectarlo. Probablemente el usuario promedio dependerá de este problema y no podrá ejecutar el módulo.  
  
Resumiré aquí lo que necesita hacer con el archivo de configuración recién generado y estas son exactamente tres cosas:****1****. elimine o comente el siguiente fragmento del archivo de configuración:****#include ".. \ drivers \ radio \ Si446x \ si446x\_patch.h" 2****. eliminar de la definición de la tabla RADIO\_CONFIGURATION\_DATA\_ARRAY el fragmento****SI446X\_PATCH\_CMDS,****es decir****,****la primera fila de la tabla****3****. en la definición****#define RF\_POWER\_UP 0x02, 0x81, 0x01, 0x01, 0x8C, 0xBA, 0x80****cambie el segundo byte a****0x01****, después de cambiar la apariencia de la siguiente manera:****#define RF\_POWER\_UP 0x02, 0x01******, 0x01, 0x01, 0x8C, 0xBA, 0x80*** *La segunda cosa importante es la estructura del proceso de inicialización y la función del pin****SDN****en él. Usemos el dibujo:*

*[](https://2.bp.blogspot.com/-JJoYEAUYKek/XMRF1GfPktI/AAAAAAAAGJA/J8Qml1FUQIUfNfLqxERFx6StK2nq6XCjACLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-04-27%2B14-06-27.png)*

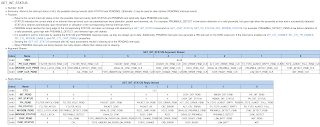
*Después de encender la alimentación, debemos configurar el pin****SDN****para****10uS de****alto. Luego, debe dar un retraso máximo de****6 ms****para realizar el procedimiento interno para****POR (reinicio de encendido)****. Entonces podemos enviar datos de configuración con****Power Up****a la vanguardia. Esperamos****CTS****después de cada comando enviado . De la figura anterior, se puede ver que en cada etapa de inicialización podemos usar la comprobación del estado en los****pines GPIO1****y****nIRQ****, pero en mi opinión es un esfuerzo innecesario. Administrar el cadáver derecho es igualmente efectivo. Todo el proceso de configuración desde el encendido hasta el modo de espera completo dura****15 ms****.*

*Ahora un poco sobre las interrupciones generadas por el módulo****RF****en el pin****nIRQ****.*

*Establecemos a qué debe reaccionar la interrupción en****WDS****y hay un poco de eso.*

*El****pin del****módulo****nIRQ****debe configurarse como se muestra arriba, la****configuración de pull-up****es necesaria cuando se duerme con el pin****SDN****. Desde el lado de****MCU****, necesitamos encontrar un pin y monitorear el estado en él. Elegí el pin****RB9****para esto . Después de detectar el estado bajo en el pin, debemos ejecutar el comando****GET\_INT\_STATUS****(código****0x20****), el comando se envía sin argumentos o con tres argumentos con el valor****0x00****. Esto restablecerá todos los estados que indican las interrupciones generadas y al mismo tiempo restableceremos el pin****nIRQ****, llevándolo a un estado alto. A continuación se muestra un ejemplo del código, ya que puede verse en el caso de, por ejemplo, enviar datos y responder al estado de la trama****TX****enviada :*

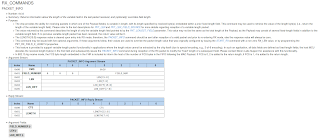
*if (! IO\_nIRQ\_GetValue ()) { // nIRQ Low?  
           SI4463\_Get\_Interrupt (inter\_buff); // obtener el estado de interrupción y borrar  
        if (inter\_buff [2] & 0x20) { // interrupción PACKET\_SENT\_PEND, ver la documentación de la API GET\_INT\_STATUS?  
          }  
       }  
  
En el bucle principal del programa verificamos si el pin****nIRQ****está en un estado bajo (desde el lado de****MCU****usando el estado en el pin****RB9****), si es así, hacemos el comando****GET\_INT\_STATUS****(código****0x20****) oculto en la función SI4463\_Get\_Interrupt () , que descarga todo interrumpir estados. El siguiente paso es verificar Valor en****PACKET\_SENT\_PEND****de p que ayuda expresión si (inter\_buff [2] y 0x20) . Si el resultado de la comparación es verdadero, sabemos que la brecha no generó un envío de trama****TX****. Y debajo de la documentación de la****API****, el contenido de****GET\_INT\_STATUS***

*[](https://1.bp.blogspot.com/-431qRaIHOO0/XMR_9DgyGsI/AAAAAAAAGJY/jtCp-DAwoJExtlivIrS0islIvJ9vrujkQCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-04-27%2B18-08-52.png)*

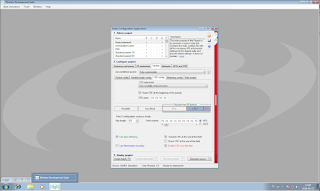
***TX Broadcasting:*** *veamos un ejemplo simple para ver cómo se ve la implementación práctica de la transmisión.  
Primero,  
  
creamos un búfer de envío al que arrojaremos nuestros datos para enviar: uint8\_t send\_buff [] = "DATA";  
  
Luego creamos una función conveniente para enviar datos en su formulario a continuación:  
  
// ------------------------------- SI4463\_Send\_Data ---- --------------------------  
void SI4463\_Send\_Data (uint8\_t \* send\_dt , uint8\_t size , uint8\_t channel )  
{  
   // ENVÍO de diodos rojos -> EN  
    SI4463\_nSEL = 0;                                    
     
            SI4463\_Clear\_TX\_FIFO ();  
            SI4463\_TX\_FIFO (send\_dt, tamaño);  
            SI4463\_TX\_Start (canal, tamaño);  
     
    SI4463\_nSEL = 1;  
    // Diodo rojo ENVIANDO -> APAGADO  
}  
  
Los argumentos de la función son: puntero de memoria intermedia con nuestros datos para enviar, cuántos datos enviaremos y el número de canal en el que se realizará la transmisión, tenemos****255 canales de****radio a nuestra disposición . Llamar a la función que nos enviará los datos automáticamente se ve así:****SI4463\_Send\_Data ( send\_buff******, 4 , 0);*** *La función de envío de datos realiza tres operaciones: limpia el búfer****FIFO TX****ubicado en el módulo****RF****, reescribe nuestros datos en el búfer en el módulo****FIFO TX****, inicia la transmisión en el canal seleccionado.****Recolección TX:*** *echemos un vistazo a un ejemplo simple para ver cómo se ve una recolección práctica. Primero, configuramos el búfer de recepción al que copiaremos los datos del búfer****FIFO****delmódulo****RF****:  
  
uint8\_t Receive\_buff [64];  
  
El siguiente paso es poner el módulo en modo de recepción: SI4463\_RX\_Start (0, 0); El primer argumento de la función es el número de canal en el que escuchará el módulo y el segundo parámetro representa la longitud del marco de datos esperado, si ingresamos cero, entonces un marco de datos con la longitud definida en el campo****F1L****y no serecibirámás byte. Luego necesitamos rastrear el estado del pin****nIRQ*** *, el estado bajo nos informa que se ha producido un evento que ha cambiado el estado en****GET\_INT\_STATUS.****Por lo tanto, es necesario verificar si no se trata de un cambio en el campo PACKET\_RX\_PEND (valor de lectura 1), de ser así, sabemos que los datos se recibieron en el canal seleccionado. Necesitamos asegurarnos de que los datos tengan****CRC****sin errores. El campo CRC\_ERROR\_PEND nos informará al respecto si el****CRC****para los datos es correcto, es decir, leemos el valor 0 en este campo , podemos reescribir los datos del búfer****FIFO****del módulo a nuestro búfer de usuario. Para verificar los estados disponibles después de detectar un estado bajo en el pin****nIRQ****Tenemos que llamar a la función que lee estos estados para nosotros desde el módulo. Y esta es la función****SI4463\_Get\_Interrupt (), el****argumento de la función es el búfer que contendrá los datos leídos sobre todos los estados del módulo.  
  
Toda la operación se presenta en el siguiente ejemplo de código:  
  
 if (! IO\_nIRQ\_GetValue ()) { // nIRQ Low?  
            SI4463\_Get\_Interrupt (inter\_buff); // obtener el estado de interrupción y borrar  
            if (inter\_buff [2] & 0x10) { // interrupción PACKET\_SENT\_PEND, ver la documentación de la API GET\_INT\_STATUS ??  
               if (! (inter\_buff [2] & 0x08)) { // ¿Error de interrupción de datos CRC?)  
                SI4463\_RX\_FIFO (recibir\_buff, 4);  
                LED\_TOG;  
              }  
            }  
           SI4463\_Clear\_RX\_FIFO ();  
           SI4463\_RX\_Start (0, 0);   
           SI4463\_Get\_Interrupt (inter\_buff); // obtener estado de interrupción y borrar  
    }  
  
Es importante borrar el búfer****FIFO RX****al final de las operaciones de recepción , reiniciar****RX START****y restablecer los estados utilizando la función****SI4463\_Get\_Interrupt ()****, esta función además de recuperar los estados actuales restablece el pin****nIRQ****.****Un ejemplo de uso de la API:*** *Vale la pena mostrar cómo navegar por la****API****que se implementa en el módulo. Si lo conseguimos, podemos escribir nuestras propias funciones según sea necesario. Mostraré un esquema simple que podemos replicar en otra función****API****.****API****se ha dividido en comandos y propiedades (propiedad). Los comandos se utilizan para realizar operaciones y cambiaremos la configuración del módulo utilizando las propiedades. En la documentación de la****API****, que tiene la forma de un documento****html****, tenemos todo organizado de forma agradable y legible. Tome el****comando PACKET\_INFO (0x16)****en la bifurcación , este es uno de los comandos en la sección****COMANDOS DE RX***

*[](https://3.bp.blogspot.com/-RwLZUyah4WQ/XMm7KfKPtcI/AAAAAAAAGJ0/wzQu4T817acUjFLg09Q6oykUXa5_C7N9QCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-01%2B17-24-46.png)*

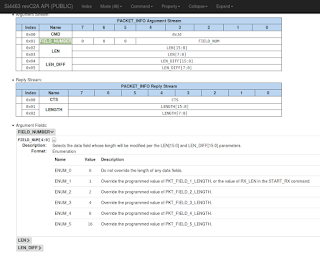
*Su forma detallada a continuación:*

**

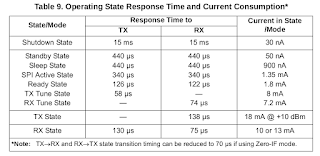
*Leemos en el original que el comando se usa para:  
Resumen: Devuelve información sobre la longitud del campo variable en el último paquete recibido y (opcionalmente) anula la longitud del campo.  
  
En general, el punto es que este comando puede devolver el valor que se nos envió en el campo que especifica la longitud de los datos recibidos en el campo de longitud variable. Pero, ¿qué es un campo de longitud variable? Recordemos la estructura de trama que establecemos en el programa de configuración****WDS****:*

*[](https://4.bp.blogspot.com/-EVxTIpgbuws/XMm98ZX1bdI/AAAAAAAAGKM/YSUMOvcsytgnKqKpqkh2fqTt_lKMSGwbQCLcBGAs/s1600/wds20.png)*

*El campo de longitud variable se representa en la imagen****<F2>****, longitud variable porque físicamente el campo en la trama transmitida depende de la cantidad de información enviada, si enviamos, por ejemplo,****4 bytes de****datos, la trama de****63 bytes****no está volando (porque esta es la longitud máxima que hemos declarado) incluyendo nuestros****4 bytes****con datos pero solo vuela un cuadro de****4 bytes****. Por lo tanto, la longitud de la trama transmitida físicamente cambia según la cantidad de datos que se envían. Por lo tanto, el campo****<F2>****es un campo de longitud variable con nuestros datos. Debido a que el módulo no tiene una máquina automática para calcular la longitud de los datos enviados en el campo****<F2>****, hemos creado un segundo campo etiquetado****F1L****longitud fija****1 byte****. En el campo****F1L****enviamos información sobre la cantidad de datos enviados en el campo****<F2>****. Usando el comando****PACKET\_INFO****, recuperamos fácilmente los datos contenidos en el campo****F1L****en el lado del receptor. La idea de usar este comando es que antes de reescribir y descargar datos del búfer de hardware****FIFO RX****en el proceso de recepción, la longitud de los datos recibidos es la información deseada. En base a esta información, sabemos cuántos datos tenemos que reescribir desde el búfer de hardware****FIFO RX al****búfer de recepción del usuario para su posterior análisis o visualización. Aquí un comentario más, dada la longitud en el campo****F1L****En el lado del transmisor, tenemos que calcularnos e ingresar la primera posición del búfer de transmisión, pero es una operación trivial e incluso un principiante no debería causar ningún problema. En mi programa de muestra se mostrará cómo hacerlo. En caso de que haya un fragmento de código que haga esto: uint8\_t data\_buff [] = "****DATE****"; uint8\_t send\_buff [ 64 ]; uint8\_t i, len;  len = sizeof (data\_buff); // calcula la longitud de los datos send\_buff [ 0 ] = len;  // ingrese la longitud de los datos en la primera posición del búfer de transmisión     para (i = 1; i <len; i ++) { send\_buff [i] =   
  
  
  
  
     
  
   
            
  
       data\_buff [i-1]; // ingrese los datos en el búfer de transmisión comenzando desde la segunda posición  
    } El comando****PACKET\_INFO****debe enviarse con****5****argumentos de****0x01****a****0x05****. Para nosotros, el argumento más importante es el campo****FIELD\_NUMBER****, su estructura a continuación:*

*[](https://1.bp.blogspot.com/-W7koQJuUaD8/XMnE8DszkAI/AAAAAAAAGKY/kXqDtNINIn8_dwnGYwZZiBu_5rJO2qAkwCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-01%2B18-08-35.png)*

*Como queremos que el comando devuelva los datos en el campo****F1L****, que es el campo****número 1****(el campo****<F2>****es el****número 2****), el argumento que usaremos es el valor****0x01****( Anular el valor programado de PKT\_FIELD\_1\_LENGTH, o el valor de RX\_LEN en el Comando START\_RX )  
El resto de los argumentos tendrá el valor****0x0****. Por lo tanto, la forma de bytes del comando****PACKET\_INFO****con los argumentos debería verse así:****0x16 , 0x01, 0x00, 0x00, 0x00, 0x00****Es decir, el código de comando****PACKET\_INFO- 0x16****y cinco argumentos. El primer argumento****0x01****informa al módulo que debe devolver información sobre el contenido del campo****1,****es decir****, F1L****. El formulario de byte anterior se envía al módulo de****RF****y él sabe lo que queremos decir en un ataque de inteligencia. Los datos devueltos por el módulo son dos bytes ( no incluimos el byte****CTS a****pesar de que se proporciona como byte No.****0x00*** *stream.) que representa el valor transmitido en el campo****F1L****. Ahora, ¿cómo se verá físicamente nuestra función enviando el comando****PACKET\_INFO****y devolviendo el contenido del campo****F1L****? Mi sugerencia para esta función se ve así: // ------------------------------- SI4463\_Get\_Packet\_Info --------- -------------------****uint8\_t SI4463\_Get\_Packet\_Info (nulo)****{     uint8\_t tx\_buff [] = { 0x16, 0x01, 0x00, 0x00, 0x00, 0x00 };     uint8\_t rx\_buff [ 2 ] ; SI4463\_Send\_With\_CTS ( tx\_buff , 6 ); SI4463\_Read\_Buffor (  
  
  
  
  
  
  
     
    rx\_buff , 2 );  
    devuelve rx\_buff [ 1 ] ;  
} Como podemos ver en la función, creamos dos buffers. El búfer tx\_buff [] contiene la forma de byte del comando****PACKET\_INFO****enviado al módulo****RF,****y la respuesta del módulo, es decir, el valor del campo****F1L****, irá al búfer rx\_buff [ 2 ] . En este caso, estamos interesados ​​en el valor del byte inferior, es decir, el segundo byte transmitido. La función****SI4463\_Send\_With\_CTS****nos****envía físicamente****el búfer tx\_buff [] después de****SPI*** *. Su segundo argumento es el número de bytes enviados. La función****SI4463\_Read\_Buffor ()****intercepta la respuesta del módulo al comando****PACKET\_INFO****enviado . ¿Cómo podemos poner nuestra función en el proceso de recepción que muestra el fragmento de código a continuación: if (! IO\_nIRQ\_GetValue () ) { // nIRQ Low? SI4463\_Get\_Interrupt (inter\_buff); // obtener el estado de interrupción y borrar             if ( inter\_buff [ 2 ] & 0x10) { // PACKET\_SENT\_PEND interruption ??, ver la documentación de la API GET\_INT\_STATUS ??                 if (! (inter\_buff [ 2 ] & 0x08)) { // ¿Error de interrupción de datos CRC?)  
  
  
             
  
  
                 
                SI4463\_RX\_FIFO ( recibir\_buff , SI4463\_Get\_Packet\_Info ()   );  
                 
                LED\_TOG;                         
                
              }  
            }  
           SI4463\_Clear\_RX\_FIFO () ;  
           SI4463\_RX\_Start ( 0, 0 ) ;   
           SI4463\_Get\_Interrupt (inter\_buff); // obtener el estado de interrupción y borrar  
    }****Dormir y reactivar el módulo: a****continuación se muestra una tabla con los modos disponibles que tenemos en el módulo:*

*[](https://1.bp.blogspot.com/-b4IZy4JuphM/XMwponnjHyI/AAAAAAAAGM0/EJud_WijbN8SufQpZGChuM0twIeSlrVpwCLcBGAs/s1600/Zrzut%2Bekranu%2Bz%2B2019-05-03%2B13-43-36.png)*

*El modo más óptimo en términos de consumo de corriente y tiempo de****activación****es el****estado de espera****. El consumo de energía en este modo es de****50 nA****. y tiempo de recuperación a plena actividad de****440 uS****. Comparemos estos parámetros con el módulo****nRF24L01 +****, que es popular en nuestro mercado , que extrae****900 nA****en el modo más bajocon un tiempo de****activación****de****1.5 ms****. A partir de esta comparación, es evidente que el uso de módulo de batería****SI4463****late****nRF24L01 +****.  
  
Para poner el módulo en modo de****estado de****espera , use el comando****CAMBIAR ESTADO****(código****0x34****). Para llevar el módulo desde el modo de reposo, tenemos que enviar cualquier cosa en el****TPI****evento desencadenado****el SPI****lado del módulo. La****ocurrencia de****eventos****se puede detectar con el****pin****nIRQ****y la verificación de estado****CAMBIAR ESTADO PENDIENTE****usando el comando****GET INT STATUS****. Desde el momento en que se produce el evento****SPI****y se detecta, podemos ingresar al módulo en cualquier modo****TX****o****RX****. Parece que dormir y despertarse en modo de****estado de****espera es relativamente muy simple. Por supuesto que podemos usar el automático "wubudzaczem"****WUT****y ver****RC 32 kHz,****pero creo que es****MCU****él debe supervisar el despertar y controlar todo el puesto. La idea de despertarse con****SPI****es genial. Deliberadamente, no menciono el pin****SDN****y el modo de reposo profundo, donde obtenemos un consumo de energía de****30 nA****porque el tiempo de activación de este modo a****15 ms****es poco aceptado, especialmente en aplicaciones de red donde funcionan más módulos y queremos conectarlo mediante sincronización de tiempo.*

*Bueno, tenemos la parte teórica detrás de nosotros y es hora de ir al grano. Entonces, cómo convertir nuestros conocimientos teóricos en prácticos :) Así que es hora de escribir un software completo que hable con nuestro módulo. No puedo esperar los efectos, es decir, la primera comunicación exitosa y el intercambio de datos.  
  
El programa de prueba ya lo he hecho todo funciona como frambuesa, las primeras pruebas de combate han quedado atrás. La prueba que hago es simple como construir un mayal. Envío****100 paquetes separados****en un módulo a intervalos de****200 ms****. El módulo receptor verifica la corrección de la suma****CRC****(automáticamente sin nuestra interferencia, verificamos solo el estado de esta operación) de la información recibida, si está bien, contamos el paquete como correcto e incrementamos el contador de los paquetes recibidos correctamente. Mostramos el número de paquetes recibidos correctamente en la****pantalla LCD****. Envío datos a una velocidad de****10 kbps****, los datos son físicamente la cadena****"DATOS". Antena 7 cm****aproximadamente****1,5 dB****de ganancia.****Potencia + 20dB****. Modulación****2GFSK****. Portador****433 MHz****. Traté de optimizar los parámetros de la pista de radio de acuerdo con mi mejor conocimiento. Los efectos excedieron mis expectativas.****ATENCION !!!!****. Estoy muy impresionado con las primeras pruebas del módulo****SI4463****. En una gran planta y sótano, de unos****400 m2,****obtuve cobertura total en los recovecos más distantes del objeto. Los paquetes se abrieron paso sin problemas a través de****dos techos sólidos de medio metro de hormigón armado****y****4 paredes****de****30 cm****cada uno. ¡No se ha perdido un solo paquete! de cada****100 enviados****en varios intentos.  No obtuve resultados tan espectaculares en  
  
el módulo****SI4432****probado una vez . Mejora de la ruta de transmisión en****SI4463****y adición de mejores****modulaciones 2GFSK****y****4GFSK****haciendo lo suyo. Ya después de estas primeras pruebas simples, no solo puedo recomendar estos módulos, sino con toda responsabilidad****"maldita sea" recomendarlos :)****. En mi opinión, realmente puede extraer mucho de estos módulos, especialmente si les agrega un conocimiento mínimo de la ruta de radio y sus parámetros. Una herramienta****WDS****muy conveniente para crear configuraciones y generar un archivo de encabezado listo que adjuntamos a nuestro proyecto completa una imagen muy positiva del conjunto. Lo que más me gusta es la gran flexibilidad en la configuración del marco. Tenemos cinco campos a nuestra disposición, que pueden modelarse libremente como bytes de longitud constante o variable. Cada campo puede tener su propia suma****CRC****separada**. Si queremos crear una red, vale la pena crear un campo de dirección como el primer campo en el marco. La base para el funcionamiento de dicha red debe ser la sincronización de tiempo, es decir, cada módulo se comunica, por ejemplo, con el****maestro****en un túnel de tiempo designado y el módulo se puede poner en suspensión solo después de recibir la trama con sincronización de tiempo y el parámetro de túnel de tiempo. Podemos implementar una idea para confirmar que los datos enviados no son problemas difíciles pero sí muy interesantes e inspiradores. ¡También vale la pena señalar que el módulo tiene un modo en el que podemos extender el área máxima a datos de****64 bytes****a****128 bytes****! y enviar tantos datos en un cuadro.*

*Sin embargo, debo mencionar las dificultades que encontré en el camino para iniciar el módulo. Si bien el proceso de configuración fue simple e incluso genial, lograr que el módulo se inicialice correctamente sin usar "parches" me robó mis 3 días de vida. Sin la corrección manual del segundo byte en el comando****Power Up****de****0x81****a****0x01****no****iniciaremos****el módulo. En este tema, probablemente ningún hombre caerá si no encuentra una gran voluntad de lucha y determinación :) Sin embargo, si superamos este esfuerzo, el módulo nos encantará con sus posibilidades.*

*También debe tenerse en cuenta que****Silicon Labs****tiene muy buena documentación y está feliz de leer.*

*En los enlaces a continuación hay un proyecto que subí a****GitHub****. Lo descargamos a****MPLABX-IDE****usando****Team -> Remote -> Clone****o en la consola de Linux:*

***git clone https://github.com/PICmajsterPIC32MM\_SI4463\_RF\_module.git***

*En un proyecto hay una entrada para el módulo de envío****TX****y el módulo de recepción****RX****. El proyecto al inicio está configurado para el módulo que recibe datos****RX****. Para preparar un proyecto para una página que envía****TX,****debemos comentar en las líneas del archivo****main.c****con la descripción****// para RX****i block****/ \* RX \* /******/ \* end RX \* /****y descomentar las líneas con la descripción****// para TX****y block****/ \* TX \* / / \* fin TX \* /***

*En el directorio****WDS****hay un archivo desde el cual restauraremos la configuración en este programa y el archivo generado por****WDS****y, además, el archivo en el que se encuentran los datos de los campos de registro individuales.*

*¿Qué hace el fragmento de código para el módulo de envío****TX****?*

*Módulo funcionando en modo de transmisión****TX****. Envía****100 marcos de****datos y apaga su actividad. La pantalla mostrará un contador decrementado a cero.  
  
Al principio creamos dos buffers****data\_buff []****y****send\_buff []****. En el primero ponemos la cadena****" DATOS "****y la tratamos como nuestros datos que queremos enviar por radio al módulo receptor. El segundo búfer servirá como contenedor para formar datos y datos técnicos (longitud del paquete de datos) en un paquete coherente. El primer byte del búfer****send\_buff [] contendrá****datos técnicos, es decir, información sobre la longitud del paquete de datos. Los siguientes bytes de este búfer son nuestros datos o cadena "****FECHA****". Y eso es todo cuando se trata de buffers necesarios para la fiesta de envío.*

*El contador variable se reducirá después de cada cuadro enviado correctamente.*

*En el primer bloque****/ \* TX \* /  / \* end TX \* /****usamos la función de biblioteca****sizeof () para****calcular el número de bytes con nuestros datos en****data\_buff []****. Puede sorprenderle que el resultado de esta operación sea****5 en****lugar de****4****porque nuestra cadena****" DATOS "****consiste ópticamente en****4 bytes****. Esto no es un error, pero con la especificidad de ingresar cadenas en matrices, el último byte siempre es un byte cero. Si ingresamos un carácter en el apóstrofe****'A'****, no veremos este byte adicional al final.*

*La longitud de datos calculada se escribe en el****búfer send\_buff []****en primer lugar****send\_buff [0]****=****len****; En la siguiente etapa, reescribimos los datos del buffer****data\_buff []****a****send\_buff [],****recordando que el primer byte del buffer****send\_buff []****está ocupado por los datos técnicos (longitud del paquete de datos).*

*Entramos en el bucle principal y nos enfocamos en lo que está en el bloque****/ \* TX \* /  / \* end TX \* /****. Debido a que supuse que la siguiente transmisión de 100 transmisiones se realizará cada****200 ms,****es por eso que llamé a un temporizador de software basado en un temporizador de hardware que marca cada****25 ms****. Cada****200 ms se establece****el indicador****Soft\_Timer1****y el primero****si****se cumple la condición . La operación de enviar datos usando una función se realiza:*

***SI4463\_Send\_Data****(****send\_buff, sizeof (data\_buff) +1****,****0****);*

*Los argumentos para esta función son: puntero del búfer para enviar, tamaño del búfer con datos más 1 y número de canal de radio en el que se enviarán los datos. Recto como la cola de un cerdo. En lugar de usar la función****sizeof ()****nuevamente, puede dar la variable****len****previamente calculada allí y aumentarla en****1****. Agregando****1****porque****len = 5****y debemos recordar los datos técnicos en el primer byte del búfer****send\_buff []****. ¿Por qué no puse la función****sizeof ()****en el****búfer send\_buff []****, porque el resultado sería****64****es decir, tanto como hemos declarado en esta tabla. Si alguien piensa que soy aburrida porque toco aspectos tan triviales, deje que mi cabeza golpee la pared y me recuerde cuán****cero****era, parafraseando la contraseña****"de cero al desarrollador"****:). Mi intención es que lo que escribo vaya a las personas que están al comienzo de su aventura con****el lenguaje C****y los microcontroladores. Porque por lo que escribo es una persona avanzada.*

*En el segundo****si****para la sección****TX****, verificamos si el módulo****RF****no ha establecido el pin****nIRQ****en bajo. Si es así, proceda a leer los estados llamando a la función****SI4463\_Get\_Interrupt****(****inter\_buff)****; El argumento de esta función es el búfer creado en el archivo****si4463.c****y declarado en el archivo****si4463.h****. Este búfer contendrá una secuencia de****8 bytes****como respuesta del módulo a la solicitud de estados actuales. En el siguiente****if-e****verificamos si el estado de la trama****PACKET\_SENT\_PEND****enviada positivamente se encuentra entre los estados recibidos del módulo, si es así disminuimos en el siguientecontador variable****if-e****y verifique al mismo tiempo si es cero, de ser así, detenga el****temporizador****y termine de enviar fotogramas.*

*Mostramos el contador variable y****destellamos el LED****.*

*Traté de escribir este suave con bastante facilidad para que fuera comprensible para el consumidor promedio de pan.*

*¿Qué hace el fragmento de código para el módulo de envío****RX****?*

*El módulo que funciona en modo de recepción****RX****espera las tramas enviadas y muestra el contador de datos recibidos correctamente y la forma de los datos.*

*Aquí, configuramos el búfer de recepción****Receive\_buff []****en el que****escribiremos los****datos del búfer de hardware del módulo****FIFO RX****en el proceso de recepción . El contador variable comienza con un valor de****0****, incrementaremos esta variable después de cada cuadro recibido correctamente. Correcto, es decir, con la suma de verificación****CRC****correcta de los****campos F1L****y****<F2>****. El estado de****PACKET RX PEND nos****informará si tenemos la suma de verificación correcta en el marco recibido . Antes de ingresar al ciclo****while ()****, debemos poner el módulo en modo de recepción****RX****. Hacemos esto llamando a la función****SI4463\_RX\_Start****(****0****,****0*** *); el primer argumento se refiere al número del canal de radio en el que escuchará el módulo, tenemos****255 canales****solo para recordarle. El segundo argumento se refiere a la longitud del paquete recibido, en****0****obtenemos tantos datos como indica el campo****F1L****. Ingresamos el ciclo****while ()****en el bloque entre****/ \* RX \* /  / \* end RX \* /***

*El primer paso es verificar si hay un estado bajo en el pin****nIRQ****, lo que indica la ocurrencia de un evento. Desde el lado de la****MCU****, monitoreamos el estado en el pin****RB9****. Si se detecta un estado bajo,****descargamos 8 bytes de****estado del módulo****RF****al búfer****inter\_buf****. Hacemos esto llamando a la función****SI4463\_Get\_Interrupt ()****. Entre los bytes de estado descargados, verificamos el bit****PACKET\_RX\_PEND****, que está en el segundo byte de estado. Este bit supervisa el estado de la trama recibida; si se recibe correctamente, el valor****1****se establecerá en este bit. Verificamos el estado del bit utilizando la expresión de máscara:****inter\_buff [ 2 ]******y 0x10****.*

*Si el estado de la trama recibida nos informa que la trama ha sido recibida, procedemos a verificar si la trama recibida tiene la suma de verificación****CRC****correcta del campo****F1L****y****<F2>****. ¡Lo hacemos físicamente usando la expresión****!****(****inter\_buff [ 2 ]******y 0x08****). Esta vez verificamos si el bit de estado****CRC\_ERROR\_PEND****es****0****porque solo este valor indica el****CRC****correcto . Si el****CRC****es correcto, procedemos a reescribir los datos recibidos desde el búfer de hardware****FIFO RX****del módulo en el búfer de usuario****Receive\_buff []****. Realizamos toda la operación de reescritura con la función:****SI4463\_RX\_FIFO****( recibir\_buff , SI4463\_Get\_Packet\_Info ()   );   
  
La cantidad de datos a reescribir****eliminará****la función****SI4463\_Get\_Packet\_Info ()****del campo****F1L****. Los datos técnicos, es decir, el número de bytes enviados, se omiten automáticamente. Luego incrementamos el contador variable y lo mostramos en la****pantalla LCD****. Además, mostramos el contenido del búfer de usuario****Receive\_buff []****en el que tenemos nuestra cadena " DATOS "*

*Todo funciona sin tartamudear. Lo único que queda es reducir la potencia de transmisión y encontrar lo óptimo para las necesidades individuales. La potencia máxima****+ 20dB****suele ser demasiado alta, como lo demuestran mis pruebas, a menos que queramos conectar objetos a****2 km de distancia****.*

*En mi opinión, el módulo es****el mejor****y vale la pena dedicarle tiempo.*