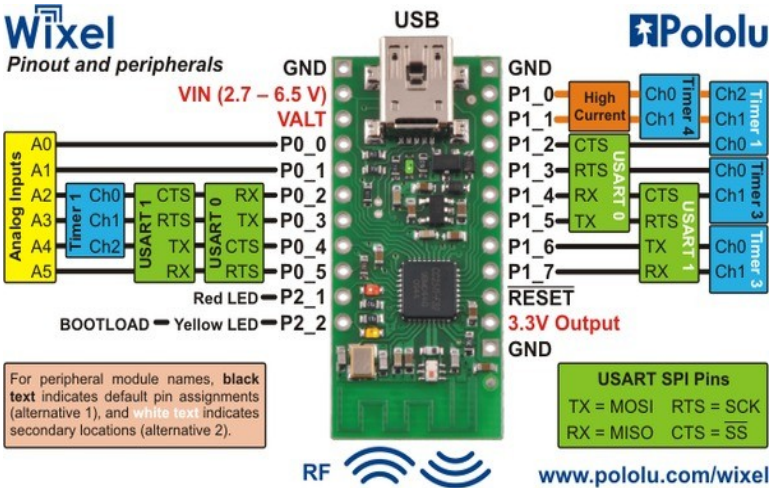
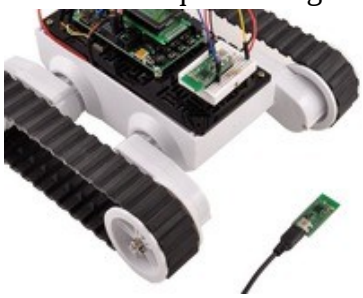
Pololu Wixel Guía de usuario

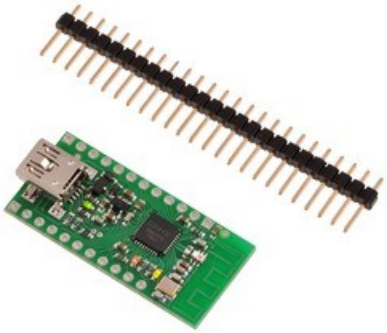


1. Introducción

El Pololu Wixel es un módulo programable que dispone de conexión USB y radiofrecuencia a 2,4 GHz. El Wixel está basado en el CC2511F32 un microcontrolador de Texas Instruments con transceptor de radio integrado, 32 KB de memoria flash, 4 KB de memoria RAM y una interfaz USB de máxima velocidad. Un total de 15 líneas de E/S están disponibles para fines generales, incluyendo 6 entradas analógicas y todo ello en una placa con un espaciado de 0.1" entre pins que lo hace fácil de usar en diversos proyectos.
Proporcionamos aplicaciones libres y de código abierto para que las puedas cargar y configurar con su bootloader USB incorporado, pudiéndolo adaptar a lo que necesites en tus proyectos.
Sólo tienes que descargar una aplicación diferente para reutilizar el Wixel en tu próximo proyecto.



Nuestra aplicación inalámbrica serie convierte a un par de Wixels en un enlace inalámbrico TTL/USB que permite la comunicación entre dos microcontroladores o entre un PC y un microcontrolador, lo que permite utilizarse, por ejemplo, en la comunicación entre dos robots o para la supervisión de forma remota de un robot desde tu equipo. Una versión especial de esta aplicación está diseñada para usarse con nuestra placa Wixel para Arduino que facilita la capacidad de agregar comunicaciones inalámbricas (incluyendo programación) a un Arduino o Arduino clon. Con una tasa de transferencia de bits de RF de 350 kbps, la aplicación es capaz de transmitir o recibir datos hasta 10 KB por segundo y puede llegar a una distancia aproximada de 50 pies (más de 15 metros en condiciones típicas de interior). Pueden utilizarse varios enlaces de forma simultánea con diferentes canales. Para información más detallada, ver la sección 9.b.
Con la aplicación **USB-to-Serial** convertimos un único Wixel en un adaptador USB-a-TTL serie que es capaz de moverse a velocidades de 350,000 bps y soportar cuatro señales de control. Esta aplicación no usa radio. La información detallada está en la sección 9.c.



La aplicación **I/O Repeater** permite extender y replicar con capacidad inalámbrica las líneas de un microcontrolador hasta más de 50 pies con dos o más dispositivos Wixel.
Información detallada sección 9.d.

Estamos trabajando en aplicaciones adicionales para programación inalámbrica de AVR, detección inalámbrica y mucho más. También puedes escribir tus propias aplicaciones mediante el SDK de Wixel de código abierto y compartirlas con la comunidad. Consulta la sección 10.

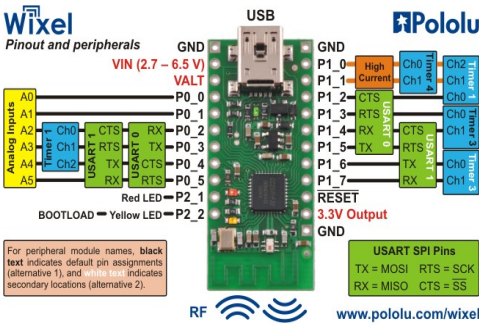
Hardware incluido

El Wixel está disponible en dos versiones:
La versión en kit con una tira de pins de 25×1 de 0.1" machos, ideal para instalaciones compactas y que permite flexibilidad en la elección de los conectores.
La versión ensamblada con los cabezales de pins ya soldados.

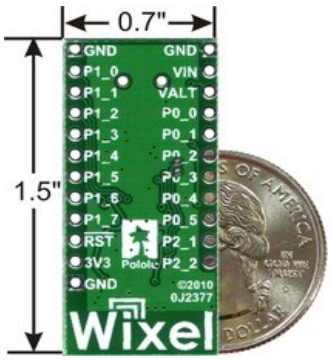
1.a Pins del módulo y componentes

El Wixel se puede conectar a un puerto USB de ordenador mediante un cable USB A a mini-B (no incluido). La conexión se utiliza para su configuración y también para transmitir y recibir datos. La conexión USB también puede proporcionar alimentación al dispositivo. Del lado de la placa junto al

conector USB, hay una antena integrada de 2,4 GHz. Esta antena, junto con los otros circuitos de RF le permite al Wixel enviar y recibir paquetes de datos en la banda de 2,4 Ghz.
El Wixel se basa en el microcontrolador CC2511F32, que hace que sea compatible con el transceptor CC2500, la familia del CC2510Fx y la familia del CC2511Fx, microcontroladores todos ellos de Texas Instruments.
Wixel no es compatible con Bluetooth, Zigbee o Wi-Fi. La antena es de un diseño "*serpenteante invertido F*" descrita en la nota de aplicación *AN043* de Texas Instruments.
Los tres pins **GND** van a 0V por definición.
Cuando conectamos el Wixel con otros dispositivos electrónicos hay que estar seguros de que todas las líneas a masa están conectadas entre si, a no ser que hagas algo muy avanzado.



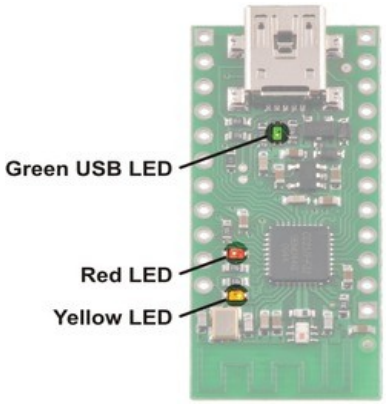
El Wixel puede alimentarse desde el pin **VIN**. Conéctalo a una salida de 2.7–6.5V entre VIN y GND, siendo el positivo VIN. Puedes alimentarlo desde VIN y desde el USB al mismo tiempo.
Mira la sección 5.a para más información sobre las alimentaciones.
El pin **VALT** esta conectado de tres maneras: desde la alimentación de 5V del puerto USB (a través de un diodo), VIN (a través de un diodo) y desde la entrada del Wixel del regulador de 3.3V en la placa. La conexión de 5V se desconecta cuando la alimentación proviene de VIN. Muchos usuarios no necesitan del pin VALT. Mira en la sección 5.a para ver ejemplos de uso.



El pin marcado con 3V3 de la placa (3.3V Output en el diagrama) está conectado a la salida del regulador de 3.3V. La salida de alimentación puede usarse para otros periféricos conectados al módulo. Con una entrada de 5V (lo mismo desde USB, VIN o VALT) esta salida nos da una potencia de 150mA. El pin marcado como RST en la placa (RESET en el diagrama) es una línea reset del micro del Wixel. Este pin se pone bajo al realizar un reset. No es necesario para la mayoría de usuarios, pero puede ser útil cuando desarrollemos aplicaciones para Wixel (mira la sección 5.c). El pin está internamente en alto conectado por pull-up a 3.3V y así permanece. Si cableas algo a este pin, el datasheet del CC2511F32 recomienda agregar un filtro RC externo con valores de 1nF y 2,7kΩ para evitar reinicios accidentales del microcontrolador
El Wixel tiene 15 líneas libres de E/S cuyo comportamiento depende de la aplicación que esté cargada en el Wixel. Concretamente estos son los pins libres, todos los del puerto 0 (P0_0 a P0_5), todos los del puerto 1 (P1_0 a P1_7) y P2_1. El pin P2_1 está vinculado al led rojo, pero los otros 14 están libres y sólo conectados al micro. La línea de P2_2 también es accesible, pero está conectada al led amarillo y se utiliza con el bootloader (véase la sección 5.c). La cantidad de corriente que pueden aguantar las líneas de CC2511F32 no está bien documentada por el fabricante pero según un post en el foro de un empleado de TI, los pins de E/S regulares están diseñados para 4mA mientras que P1_0 y P1_1 están diseñados para 20 mA.
Precaución: Las líneas I/O del Wixel no toleran 5V. Debes usar niveladores, diodos o divisores de voltaje para conectarlas entre líneas de sistemas que utilicen 5V.
El micro CC2511F32 tiene diversos periféricos disponibles para aplicaciones:
2 UARTs que permiten comunicaciones asíncronas serie o SPI.
3 temporizadores para salidas PWM como se verá más adelante y uno más de uso interno.
6 pins analógico-digitales conectados a un ADC de 7–12 bit de resolución
Distintas aplicaciones de Wixel pueden utilizar diferentes conjuntos de estos periféricos. Consulte la documentación para obtener información detallada sobre el comportamiento de las líneas de E/S.
El Wixel tiene tres leds indicadores de estado o de función:
Led Verde
El led verde se nutre del USB, por lo que se enciende cuando conectamos el cable al ordenador y se recibe la alimentación correspondiente para el Wixel. Cuando está en modo bootloader (la

aplicación está parada), este led se usa para indicar el estado del dispositivo USB. Cuando el Bootloader conecta con el USB el led verde se enciende y parpadea despacio. El parpadeo continúa hasta que el bootloader no recibe un mensaje particular desde el PC indicando que los controladores están instalados correctamente (mira la sección 3.a sobre la instalación de los mismos). Después de recoger el mensaje el led empieza a parpadear con un patrón doble. El led se puede apagar al perder la conexión USB ya sea porque el PC ha entrado en modo suspensión o el puerto USB se ha desconectado por cualquier otra razón.

Mientras el Wixel ejecuta su aplicación, el comportamiento del led depende de ella. Las aplicaciones estándar proporcionadas por Pololu los leds en todas ellas se comportan como sigue: cuando la aplicación se conecta con el USB, el led verde comienza a parpadear lentamente. El parpadeo sigue hasta que la aplicación recibe el mensaje del ordenador de que los controladores están correctamente instalados. Después de que eso el led se queda encendido. El led verde también se desactivará al entrar en suspensión el USB, el ordenador o se apaga el puerto por cualquier otro motivo.



Led Rojo

Mientras el Wixel esta en modo bootloader (aplicación parada) el led rojo nos indica si hay una aplicación en el Wixel. Si no hay ninguna, el led permanece encendido, de lo contrario, se apaga. De forma predeterminada, el Wixel no viene con ninguna aplicación, por lo que se encenderá la primera vez que lo alimentemos. El pin P2_1 está conectado al led, por lo que esta línea está alta cuando se enciende y baja en caso contrario. Mientras que el Wixel ejecuta su aplicación, el comportamiento del led depende de la misma. Consulta la documentación particular de la aplicación para obtener más detalles.

Led Amarillo

Mientras que el Wixel está en modo bootloader (aplicación detenida), este led se enciende fijo y parpadea cuando recibe un comando desde el software Wixel en el equipo. La utilidad de configuración de Wixel consulta el estado del bootloader una vez por segundo, por lo que si la utilidad de configuración de Wixel está abierta, el led parpadeará una vez por segundo. Mientras que el Wixel está siendo programado, parpadea de forma constante. El pin P2_2 está conectado al led, con lo que esta línea esta alta cuando el led está encendido, de lo contrario, baja. Mientras que el Wixel ejecuta su aplicación, el comportamiento del led depende de la aplicación. Consulte la documentación correspondiente para obtener más detalles.

1.b Sistemas operativos soportados

Los **Wixel USB drivers y el software** de **configuración** trabajan bajo Windows 7, Windows Vista, Microsoft Windows XP (SP 3) y Linux. Estamos estudiando soporte para Mac OS en un futuro. Cualquier aplicación Wixel que implemente un solo puerto USB virtual COM funcionará en Linux o Mac OS sin necesidad de controlador especial. Cualquier aplicación de Wixel que implemente un dispositivo de interfaz humana (HID) funcionará en Windows, Linux o Mac OS sin necesidad de controlador especial. Por lo tanto, puedes utilizar la mayoría de aplicaciones Wixel con Mac OS, pero necesitas un equipo con Windows o Linux para su programación.

1.c Regulaciones gubernamentales en dispositivos de radiofrecuencia

Aviso sobre el Reglamento de Radiocomunicaciones: El Wixel no ha sido probado ni ha recibido certificado de conformidad con los reglamentos de radiofrecuencia por lo que se envía con sólo un bootloader que no utiliza sus capacidades inalámbricas. La banda de 2,4 GHz es relativamente ilimitada en muchas partes del mundo, pero es tu responsabilidad cumplir con la normativa local si programas el Wixel para utilizar estas capacidades. El Wixel es una plataforma de desarrollo multiuso, no un producto terminado y no está certificado por la FCC o cualquier otra agencia gubernamental. Procura seguir las regulaciones locales y utilizar buenas prácticas en el desarrollo,

instalación y configuración de las aplicaciones para el mismo. El Wixel tiene un radio de baja potencia y utiliza la antena en placa sugerida por TI, por lo que es de esperar que las aplicaciones desarrolladas para Wixel no vayan en contra de las reglamentaciones, ya que no está diseñado para la integración en otros productos. Si estás pensando agregar características Wixel a tu producto, te recomendamos que integres el micro CC2511 directamente, usando la documentación de TI. Para más información de sistemas en la banda de 2.4 GHz mira la AN032 de Texas Instruments sobre regulación de licencias para transceptores en esa banda.

2. Contacto con Pololu

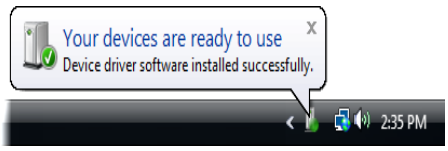
Nos encantaría conocer tu opinión sobre alguno de sus proyectos y sobre tu experiencia con el Wixel. Puede contactar con nosotros directamente o mediante mensaje en el foro. Cuéntanos lo que hacemos bien, lo que podría mejorar, lo que te gustaría ver en el futuro, o cualquier otra cosa.

3. Empezando

3.a Instalación de drivers y software Windows

Antes de conectar el Wixel a tu PC en Windows debes instalar los controladores necesarios:
Descarga el `Wixel Windows Drivers and Software (12MB zip)`.
Abre el archivo ZIP y clic en *setup.exe*. El instalador te guiará a través de pasos sucesivos en la instalación de la utilidad Wixel Configuration, la aplicación Wixel (WixelCmd) y los drivers necesarios para el PC. Si el instalador falla extrae los ficheros a un directorio temporal y clic en *setup.exe* y selecciona "Run como Administrador".

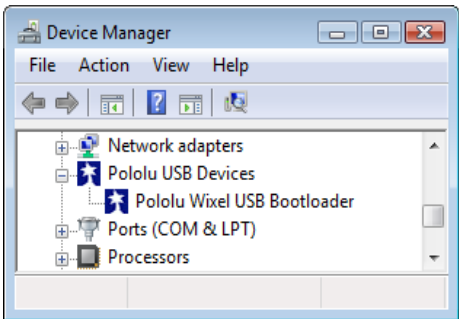
Durante la instalación, Windows sacará un mensaje de aviso de que los controladores no están testeados, tu sigue, clic en *"Continuar"* (Windows XP) o *"Install this driver software anyway"* (Windows 7 y Vista).
Después de finalizar la instalación arranca el menú desde el enlace *Wixel Configuration Utility* (en el directorio *Pololu*).
Es una aplicación Windows que te permite cargar las "apps" dentro del Wixel. Tiene también una utilidad llamada *WixelCmd* para introducir los comandos directamente.



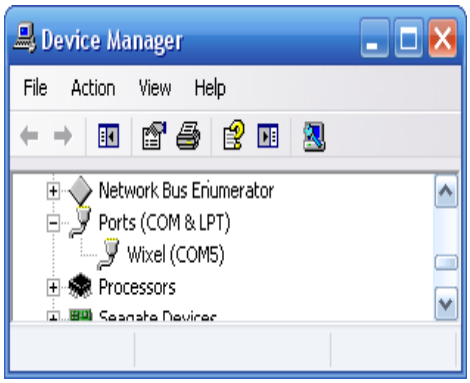
Windows 7 y Windows Vista: El PC debe reconocer de forma automática los controladores necesarios cuando se conecte un Wixel. No hace falta nada más.
Windows XP: Sigue los pasos 5-9 para cada Wixel nuevo que conectes a tu ordenador. También tendrás que seguir estos pasos de nuevo la primera vez que ejecutes una aplicación en el Wixel. Conecta el dispositivo al puerto USB del ordenador y cuando aparezca "Nuevo hardware encontrado", selecciona "No, no esta vez" y clic en "Siguiente".
En la segunda pantalla del *"Nuevo hardware encontrado"*, selecciona *"Instalar el software automáticamente"* y haz clic en *"Siguiente"*.
Windows XP avisa de que los drivers no están testeados tu sigue con clic en *"Continuar"*. Cuando llegues a *"Nuevo hardware encontrado"*, clic *"Finalizar"*.
Después de la instalación puedes ir al Panel de Control-Sistema-Administrador de Dispositivos y expander la lista "Pololu USB Devices", podrás observar una entrada para el Pololu Wixel USB Bootloader. Si se encuentra es que tu Wixel esta en modo bootloader. Debe estar en ese modo la

primera vez que se conecte al puerto USB, ya que no hay aplicación grabada en el Wixel por defecto.

Una vez hayas cargado una aplicación mediante la *Wixel Configuration Utility* y la aplicación se esté ejecutando, entonces ya no verás esa entrada en el Administrador de dispositivos. La entrada depende de la aplicación que se carga en el Wixel.



Algunas aplicaciones no permiten la interfaz USB, en cuyo caso no verás ninguna entrada Wixel en el Administrador. Las aplicaciones típicas para Wixel aparecerán como único puerto COM virtual (con el producto ID 0x2200) en la lista de "Ports (COM & LPT)". Entre paréntesis, verá el nombre del puerto (por ejemplo, COM5 o COM6).



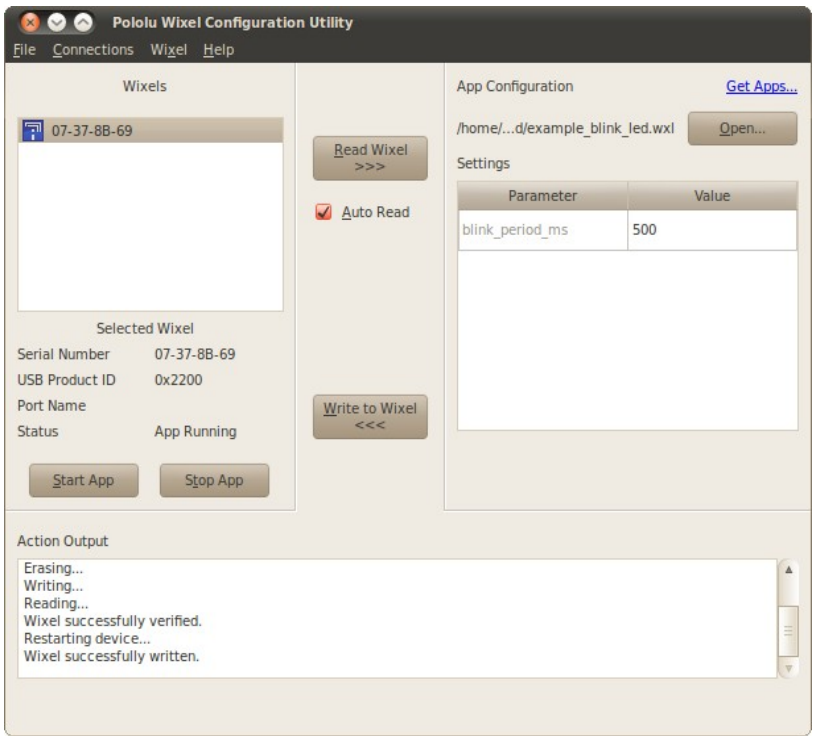
Algunos programas no permiten la conexión de puertos COM con una numeración alta. Si necesita cambiar el número de puerto COM asignado a un Wixel, puedes hacerlo a través del Administrador de dispositivos. Abrir el diálogo de Propiedades del puerto COM y clic en "Opciones avanzadas..." en "Configuración de puerto". Puedes cambiar el número de puerto COM asignado al dispositivo. Windows recuerda el puerto COM que se ha asignado a cada Wixel con la incorporación del número de serie del Wixel, un Wixel dado siempre tendrá asignado al mismo puerto COM, independientemente de que el puerto USB que se conecta.

3.b Instalación de drivers y software en Linux

Debes descargar la *Wixel Configuration Utility* y el *WixelCmd* para Linux:

`Wixel Linux Software for i386 (32-bit) (211k gz)`
`Wixel Linux Software for amd64 (64-bit) (216k gz)`

Desempaqueta el archivo tar/gzip con `"tar -xzvf"` seguido del nombre del fichero. Después sigue las instrucciones en `README.txt` y podrás hacerlos funcionar ejecutando **`wixelconfig`** o **`wixelcmd`**.



Cualquier aplicación Wixel implementa un puerto USB COM virtual o un dispositivo de interfaz humana (HID) que funciona en Linux sin necesidad de instalar controladores especiales.

Los ports COM virtuales son administrados por el kernel de *cdc-acm*, cuyo código fuente se puede encontrar en el código de drivers /usb/clase/cdc-acm.c.

Cuando se conecta un Wixel y se ejecuta cualquier aplicación que implemente un puerto serie virtual en el PC, el puerto debe aparecer como un dispositivo con

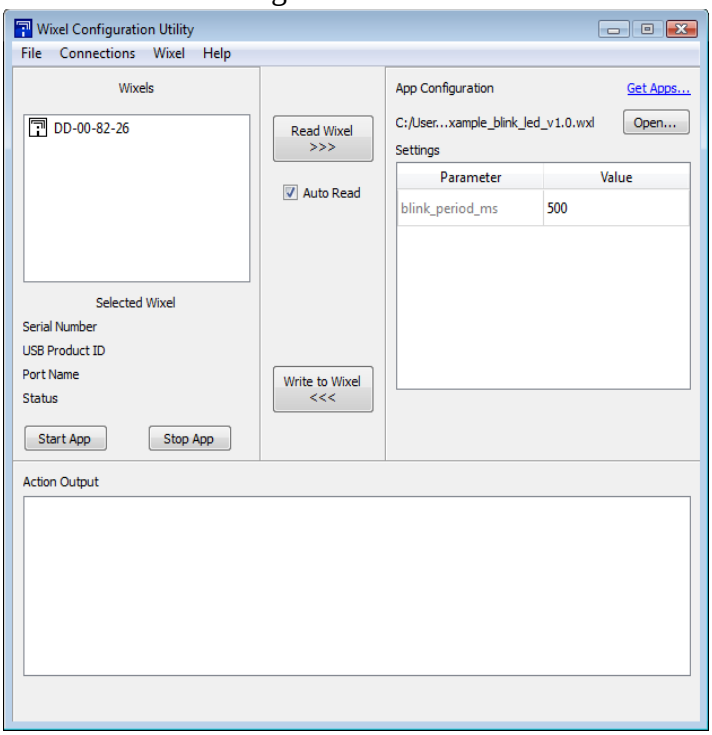
nombres como */dev/ttyACMx* (en donde *x* corresponde a un número que depende de los dispositivos que estén conectados en ese momento).
Podemos utilizar una aplicación de terminal cualquiera (como kermit o screen) para enviar y recibir bytes por esos puertos.

3.c Instalación de drivers y software en Mac

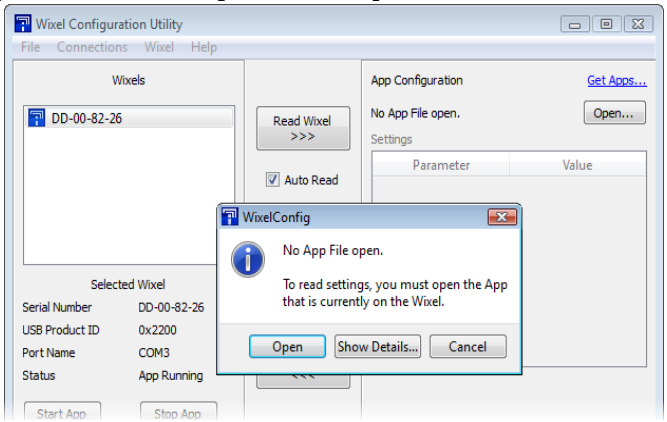
El software no trabaja en Mac OS, pero algunas aplicaciones puede que funcionen sin problemas. Mira la sección 1.b para más detalles.

3.d Cargar una aplicación de ejemplo

La primera vez el Wixel no lleva ninguna aplicación cargada. Para hacer tu Wixel útil, debes cargar una aplicación en el mismo. Esta sección te llevará por los pasos necesarios que te permitirán cargar una aplicación de ejemplo en el Wixel usando la Utilidad de configuración Wixel.



1.- Instala los drivers y el software Wixel siguiendo las instrucciones de las secciones anteriores.
2.- Descarga la aplicación llamada: `example Blink LED App v1.0 (11k wxl)`. Si quieres ver el código lo tienes en Wixel SDK en `apps/example_blink_led`. (Ver sección 10.a)
`3.- Abre la aplicación en el *Wixel Configuration Utility*. Para correr esta aplicación dale al clic en el fichero Wixel App (WXL). De manera alternativa en *Wixel Configuration Utility* puedes abrir, desde "Open…", y seleccionar la aplicación. En Windows, la encontrarás en el directorio de Pololu.`
Conecta un Wixel al PC vía USB. Lo verás aparecer en la ventana de "`Wixels`". Si no aparece necesitas usar el botón o poner tu Wixel en modo bootloader (ver sección 5.c).
Ten en cuenta que en el cuadro Wixels, hay una lista con todos los Wixels conectados al puerto USB que la *Wixel Configuration Utility* puede reconocer. Si hay un Wixel conectado, el número de serie de 32 bits aparecerá en lista.
También debes tener en cuenta que en el cuadro de configuración de la aplicación, hemos abierto la aplicación:
`example_blink_led_v1.0.wxl`.
Esta aplicación de parpadeo de led rojo usa un parámetro, *blink_period_ms* que actualmente está en 500 (por defecto).



Seleccione el Wixel pulsando en el botón izquierdo sobre la entrada de la lista. Si ves un cuadro de diálogo emergente hacia arriba, esto se debe a que el Wixel ya tiene una aplicación grabada y la casilla `Auto Read` está activada.
`Haz clic en el botón "Cancel" en ese diálogo, porque no estamos interesados en leer el contenido de la aplicación Wixel todavía.`

Clic en el botón “Write to Wixel” y escribirá la aplicación actual y sus ajustes en el Wixel seleccionado para luego proceder a su ejecución. El example_blink_led_v1.0.wxl debe funcionar ahora, el led amarillo se apaga y el led rojo parpadea. Si estás en Windows XP y es la primera vez que ejecutas una aplicación en el Wixel, aparecerá la ventana de *“Nuevo hardware encontrado”* y deberás seguir los pasos 5–9 de la sección 3.a para instalar los drivers apropiadamente. Después de su instalación el led verde quedará encendido y felicidades!!, ya tienes configurado el Wixel!

La velocidad de parpadeo esta determinada por la variable *blink_period_ms*. Las unidades están en milisegundos (ms.). Puedes cambiar *blink_period_ms* a 100 haciendo clic dos veces en el número y escribiendo “100”. Ahora debes grabar el Wixel con el botón “Write to Wixel” y observarás como el led rojo cambia el parpadeo cinco veces más rápido que antes.

El *Wixel Configuration Utility* también puede leer la aplicación del Wixel. Para demostrarlo, cierra la utilidad de configuración, vuelve a abrirla y escoge tu Wixel. Siempre que la casilla Auto Read esté marcada y haya una aplicación intentará leerla del Wixel. En el caso de no poder abrir la aplicación correctamente, te aparecerá un cuadro de diálogo como en la imagen.

Para leer la configuración del Wixel, tendrás que abrir la aplicación que se encuentra actualmente en el Wixel. Haz clic en "Open" del diálogo y selecciona example_blink_led_v1.0.wxl.

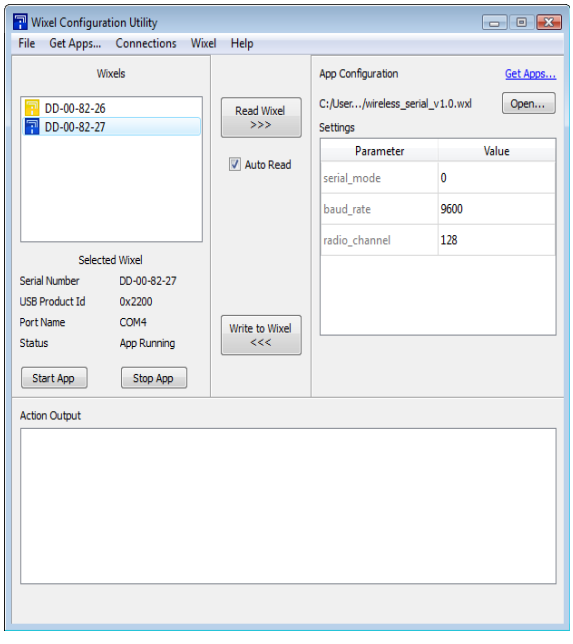
Es necesario abrir el archivo de aplicación que se cargó originalmente en el Wixel debido a que ese archivo contiene los metadatos necesarios para interpretar los valores contenidos en la memoria flash del Wixel. Si abres un archivo de aplicación diferente, incluso una versión diferente de la misma aplicación, la configuración es probable que quede dañada. En este caso, aparecerá un cuadro de advertencia para avisarte y darte algunas opciones. Puesto que todavía tienes el archivo de la aplicación correcta, no deberías ver el diálogo ahora.

Después de abrir el archivo de la aplicación Wixel, la utilidad de configuración Wixel leerá el Wixel y comparará su contenido con lo que está en la aplicación. A continuación, verás la configuración como la muestra en la ventana de la derecha: *blink_period_ms* debe ser de 100.

Tenga en cuenta que el número **100** aparece en negrita. Esto se debe a que difiere de la configuración por defecto, que es de 500. Se puede restablecer el valor predeterminado en cualquier momento con clic en el botón derecho sobre el número y con "Reset to Default Value".

Después de completar estas líneas, debes pues estar cómodo con la escritura y lectura de los ajustes. Esto es todo lo que necesitas saber para configurar tus Wixels.

Cuando se carga una aplicación real, como Wireless Serial App, lo único que variará serán los nombres y significados de los parámetros. Para entender lo que significan los distintos parámetros, consulta la documentación de la aplicación específica. Algunas aplicaciones pueden implementar una interfaz USB no estándar (o ninguna interfaz USB en absoluto). En ese caso, el Wixel no será reconocido por la utilidad de configuración mientras que la aplicación se está ejecutando, por lo que tendrás que entrar en modo de arranque manual (mira la sección 5.c y también la documentación de la aplicación).



4. Configuración de los Wixels

El software *Wixel Configuration Utility* te permite escribir y leer configuraciones para el Wixel. Esta sección expone todas las características de esta utilidad en detalle.

4.a Wixels

A la parte izquierda superior de la pantalla en Wixels, puedes ver una lista de todos los

dispositivos conectados al ordenador que son reconocidos por el software. La aplicación encuentra cualquier Wixel que este en modo bootloader (con la aplicación parada) o ejecutándose siempre que lleve implementado USB Virtual COM port con el ID del USB Vendor 0x1FFB (para Pololu) y el ID de producto 0x2200. Si el Wixel conectado al PC no aparece en *Wixel Configuration Utility*, será porque los drivers no están instalados correctamente o que el Wixel está ejecutando una aplicación que usa diferente tipo de interface USB o que no la usa.

Si hay problemas para que la utilidad de configuración lo reconozca mira la ayuda en la sección 10.

El texto mostrado en la lista Wixel ("07-C2-C8-3A") es el número de serie del Wixel. Cada Wixel tiene un único número de 32-bit que se genera de forma aleatoria y se asigna cuando se crea el dispositivo. El icono mostrado en la lista representa el estado actual del Wixel, según lo que sigue:

Estado	Icono	Descripción
App ejecutándose		La aplicación en el Wixel se está ejecutando
App parada		La aplicación esta parada; el Wixel está en modo bootloader
No App		No hay aplicación en el Wixel; el Wixel está en modo bootloader
Reconexión		El Wixel está en reconexión, desconectando o en estado de transición

Si seleccionas un Wixel, puede ver más información del mismo en el área que hay debajo de la lista. La **USB Product ID** es la identificación del producto tal como se define en la especificación USB. El **Port Name** es el nombre del puerto COM virtual que se ha asignado al Wixel. En Windows, el nombre del puerto también está disponible en el Administrador de dispositivos.

El botón Stop App detiene la aplicación que se ejecuta en el Wixel seleccionado actualmente, poniendo al Wixel en modo bootloader. El botón Start App saca el Wixel del gestor de arranque para ejecutar la aplicación que se encuentra grabada en él.

4.b Configuración de la aplicación app

En la parte derecha de la ventana, en el cuadro de configuración de la aplicación, se puede ver el nombre de la aplicación abierta actualmente y su configuración actual.

Puedes abrir una aplicación diferente, haciendo clic en la opción "Open...". En Windows, también puedes abrir una aplicación simplemente haciendo doble clic en ella. La utilidad de configuración Wixel puede abrir archivos de aplicación, ya sea en formato WXL (documentado en 0J603) o en formato estándar de Intel HEX.

Puedes cambiar la configuración actual, haciendo doble clic en un valor y escribiendo de nuevo.

Los parámetros que están disponibles dependen de la aplicación que esté abierta, otras aplicaciones pueden tener diferentes parámetros disponibles.

Por favor, consulta la documentación específica de cada aplicación para comprender el significado de los parámetros y cuáles son los valores válidos a utilizar. La utilidad de configuración Wixel no te impedirá la entrada de valores erróneos o inconsistentes.

4.c Escribir en la memoria del Wixel

Después de haber elegido la aplicación y la configuración que deseas utilizar selecciona un Wixel, puedes escribir la aplicación y los ajustes haciendo clic en el botón "Write to Wixel". Esto borrará todo lo que había antes y escribirá la nueva aplicación al mismo.

Cuando la operación de escritura se realiza, el Wixel se reinicia y la aplicación debe comenzar a ejecutarse seguidamente.

4.d Leer desde el Wixel

Para leer la configuración de un Wixel que ya ha sido programado, selecciona el Wixel. Si la casilla de verificación Auto Read está activada, la lectura será automática. Desactiva esta casilla si vas a conservar la configuración actual al cambiar de Wixels (por ejemplo, cuando se quiere escribir la misma aplicación y configuración en múltiples Wixels). Si la casilla está desactivada, puedes hacer clic en Read Wixel en cualquier momento para leer la configuración del Wixel seleccionado.

Para leer la configuración de un Wixel, tendrás que abrir la aplicación que se encuentra actualmente en Wixel. Esto es necesario ya que el archivo de aplicación contiene los metadatos que necesita con el fin de interpretar correctamente la configuración contenida en la memoria flash del Wixel.
Si has perdido el archivo de la aplicación y quieres leer el contenido del Wixel, selecciona "Read Flash and Export to HEX File…" en el menú Wixel. A continuación, podrás abrir el archivo HEX exportado como una aplicación y lo podrás utilizar para programar otros Wixels. La configuración del Wixel se incluirá en el archivo HEX exportado, pero no serás capaz de leer estos valores en la Utilidad de configuración Wixel.
Si la aplicación del Wixel se está ejecutando cuando se inicia la operación de lectura, la utilidad de configuración Wixel detendrá temporalmente la aplicación y lo pondrá en modo bootloader para poder leer su contenido. Cuando termine la lectura, la utilidad reinicia la aplicación.

4.e Otros comandos

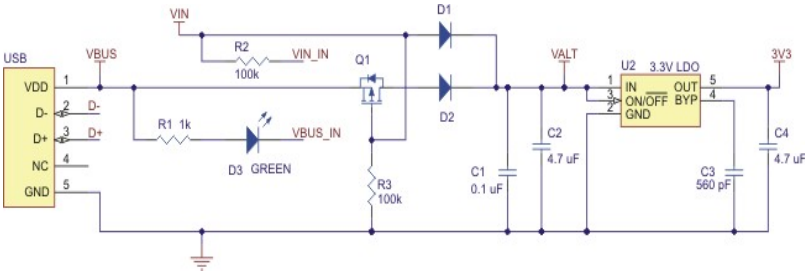
El comando Erase Wixel (en el menú Wixel) borra la aplicación del actual Wixel seleccionado (cada bit en la sección de aplicación de la memoria de pone a 1).
El comando Verify Wixel lee el Wixel seleccionado y comprueba si el contenido actual es idéntico a la aplicación y los ajustes se muestran a la derecha de la caja App Configuration.

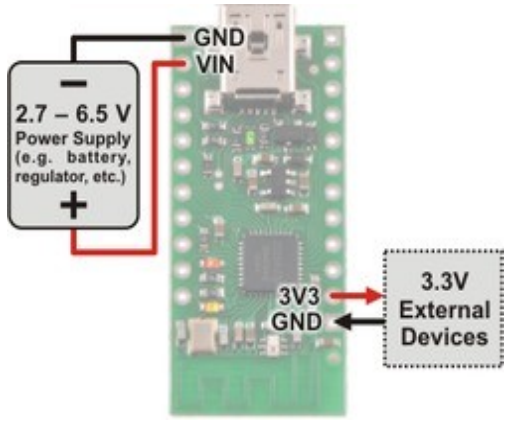
5. Conexión de Wixels

En este capítulo explicaremos como realizar las conexiones eléctricas con otros dispositivos para que el Wixel trabaje de la manera que necesites.

5.a Conexión de alimentación

Hay dos maneras de alimentar el Wixel desde el puerto USB y desde el pin VIN. El esquema de alimentación es el siguiente:





VIN entrada
El Wixel puede alimentarse desde VIN si le suministras un voltaje de entre 2.7–6.5V (baterías o regulador) con GND (negativo) y pin VIN (positivo).
Puedes tener al mismo tiempo la alimentación desde el conector USB y desde la entrada VIN.

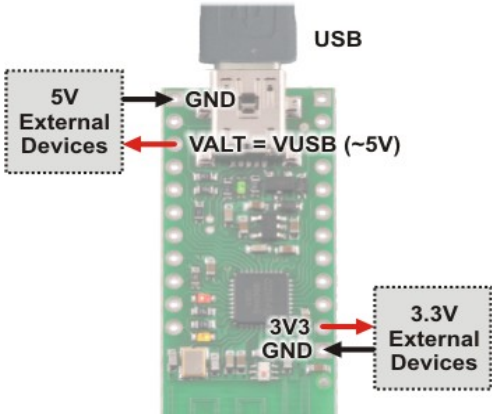
USB entrada
El Wixel puede alimentarse desde USB si conectas el cable correspondiente aún estando VIN desconectado.
El Wixel coge la corriente desde USB si no puede hacerlo desde VIN o en el caso de que esté por debajo de los 4V.
El pin de 3V3 es una salida del regulador interno del Wixel.
Si la alimentación del Wixel baja de 3,5 V esta salida será menor de lo indicado.
Normalmente tiene una potencia de hasta 150mA, pero si la alimentación está en los 5V esta queda limitada a unos 100mA. Puedes usar esta salida para dispositivos externos que trabajen con 3.3V.

VALT salida
VALT permite el acceso al pin de entrada del regulador de 3,3V y se conecta a través de un diodo a VIN o a la tensión del bus USB, dependiendo de la fuente de alimentación que esté conectada. Puedes utilizar VALT para alimentar tus propios circuitos, siempre y cuando su potencia no tenga un consumo superior a 500mA.

Consideraciones de una baja alimentación

El CC2511F32 es capaz de ponerse en modo de suspensión (PM2), con un consumo de 1µA y poder salir del mismo.

Sin modificar el hardware del Wixel, es posible alimentarlo desde VIN y obtener un consumo de corriente alrededor de 100µA. La mayor parte de esa corriente es consumida por regulador de 3,3V. Para deshacerse de él, puedes cortar el pin de salida del regulador y alimentar el Wixel directamente desde el pin 3V3 con una fuente de alimentación de 2.0 a 3.6 V.



Para más detalles sobre cómo hacer esto, por favor póngase en contacto con Pololu.

Tenga en cuenta que en la actualidad ninguna de las aplicaciones usa el modo de bajo consumo, por lo que el Wixel consumirá unos 30mA en todo momento. Para hacer que Wixel opere en bajo consumo tendrías que escribir tu propia aplicación o modificar una de las existentes. Tendrás que asegurarte de que todas las líneas de E/S son salidas que van de alta o baja: una línea con un voltaje intermedio puede consumir varios microamperios extras, de más.

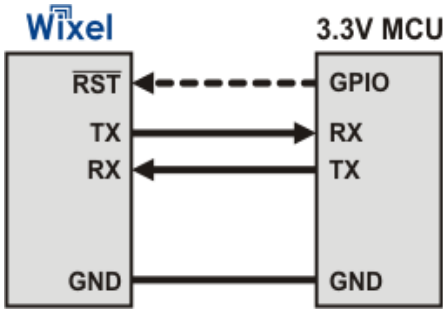
5.b Conexión a un microcontrolador vía TTL serie

Si has cargado una aplicación que emplea una de las dos UARTs de Wixel (como la Wireless Serial App), este puede comunicarse con otro microcontrolador mediante comunicación serie TTL asincrónica, no invertida. La comunicación entre Wixel y un dispositivo con RS-232 requiere de hardware adicional. Para conectar un microcontrolador al Wixel hay que hacer estas conexiones:

GND: Conectar las tierras (GND también conocido como VSS) del microcontrolador a uno de los pins GND de Wixel. Esto es imprescindible.

TX: Si quieres que el micro reciba bytes desde el Wixel, conecta la línea TX del Wixel con la línea RX del microcontrolador.

RX: Si quieres que el microcontrolador pueda enviar bytes al Wixel, conecta la línea TX del micro con la RX del Wixel. La líneas de Wixel **no toleran 5V**, por lo que si el microcontrolador funciona a 5V (o cualquier voltaje superior a 3.3V) necesitas añadir componentes extras para que no puedan superarse estos niveles de voltaje de 3.3 V en el Wixel. Un simple divisor de voltaje a partir de dos resistencias como en el dibujo es suficiente.

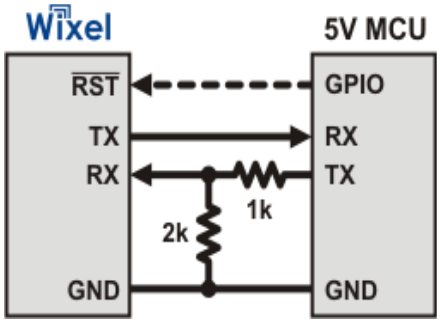


La línea RX de Wixel tiene una resistencia de 20kΩ de pull-up.

RST: Si quieres que el micro pueda resetear el Wixel, debes conectar la línea RST a cualquier línea de E/S del microcontrolador.

El micro debe poner la línea en bajo para resetear el Wixel. La línea RST no tolera 5V, recuérdalo. Si el micro trabaja a 5V (o un voltaje superior a 3.3V) puedes poner un diodo en la línea RST de Wixel para prevenir posibles corrientes en sentido inverso desde las E/S.

La conexión RST es opcional y no se necesita para enviar y recibir datos.



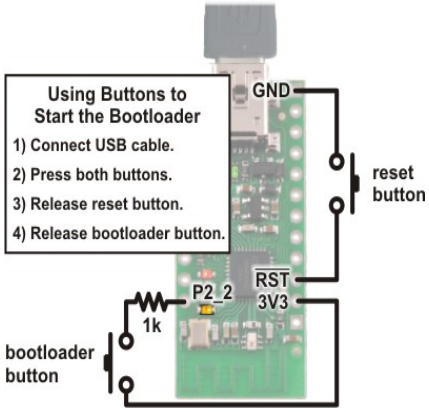
Consulta la documentación de la aplicación Wixel para encontrar información de los pins TX y RX.

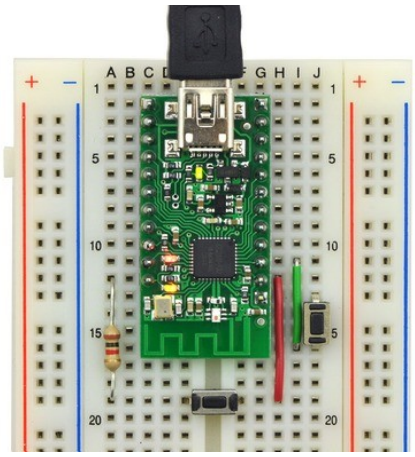
Nota: Wixel **no soporta** los voltajes del modo RS-232 usados en los puertos serie DB9 de los ordenadores. Las líneas de E/S de Wixel incluidas las de TX y RX operan con voltajes de entre 0 y 3.3V y **no toleran los 5V**.

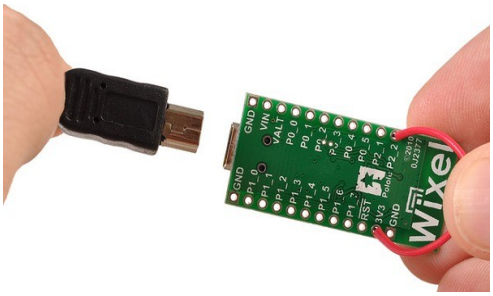
Para conectar la Wixel a señales serie de RS-232 necesitas adaptadores e inversores como la `Pololu 23201a` (RS-232 es invertida y Wixel trabaja en serie no-invertido).

5.c Conexión de pulsadores y arranque del Bootloader

Para cargar una nueva aplicación o realizar nuevos ajustes en tu Wixel (o leer la memoria flash) necesitas entrar en modo bootloader. La mayoría de aplicaciones Wixel soportan un comando especial de USB para poner el Wixel en modo bootloader y con la *Wixel Configuration Utility* puedes enviar ese comando automáticamente cuando se intenta acceder a la memoria flash. Sin embargo puede haber situaciones en donde ese método no funcione. Esto puede suceder por dos razones: que se haya cargado un programa defectuoso en el Wixel que es incapaz de responder al comando especial de USB o por cargar un programa que no utiliza interfaz USB o una diferente. En este caso, consulta la documentación de la aplicación para ver si hay una manera posible para poner el Wixel en modo bootloader. No importa en que estado este el Wixel, puedes ponerlo manualmente en modo bootloader con conexión USB, estableciendo P2_2 en alto y reseteando la Wixel. Hay dos formas para hacerlo. Una forma es desconectando el Wixel de cualquier fuente de energía, conecta el pin P2_2 a 3V3 con un cable y luego al USB.
Otra forma es usando un pulsador para el bootloader y un pulsador de reset para el Wixel.







6. Uso del Virtual COM Port

La mayoría de las aplicaciones de Wixel implementan una interfaz USB que consta de un único puerto (serie) virtual COM. Esta interfaz permite enviar y recibir bytes desde el Wixel de la misma manera que podríamos utilizar para enviar y recibir bytes desde cualquier otro puerto serial del PC.

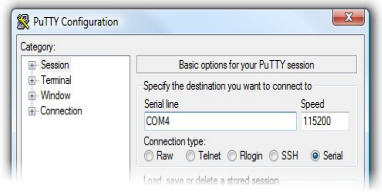
6.a Determinar nombre del puerto

Para conectarse a un puerto COM, normalmente tienes que saber el nombre del puerto. En Windows, el nombre del puerto será algo como "COMx" (siendo x un número). Puedes determinar el nombre del puerto seleccionando por el Wixel en la utilidad de configuración de Wixel, mirando la propiedad de *"Port Name"* en la parte inferior de la ventana. También puedes saber el nombre desde la lista de "Ports (COM & LPT)" en el Administrador de dispositivos de Windows.
Además de contar con nombres como "COM5…" y "COM6", los puertos COM virtuales proporcionados por el Wixel también tienen nombres como "\\.\USBSER000" y "\\.\USBSER001". Estos nombres se asignan secuencialmente cada vez que un dispositivo con un puerto COM virtual se conecta. Si sólo dispone de un dispositivo con un puerto COM virtual conectado al equipo, normalmente se le asignará el nombre "\\.\USBSER000". Estos nombres funcionan con la mayoría de los programas que permiten especificar nombres de puerto arbitrarios.

6.b Usando un programa terminal

Hay muchos programas de terminal gratuitos disponibles que son capaces de enviar y recibir bytes desde un puerto COM virtual. Programas como: PuTTY (Windows o Linux), Tera Term (Windows)

y Br@y Terminal (Windows). Los usuarios avanzados desarrollan scripts que pueden utilizar el programa libre kermit.
Para usar cualquiera de estos programas de terminal con el Wixel, debes especificar el nombre del puerto (mira sección 6.a) y la velocidad de transmisión. La velocidad de transmisión puede o no puede afectar, consulta la documentación de la aplicación. Los caracteres que escribas serán transmitidos por la línea TX. Los bytes recibidos por la línea RX se mostrarán en la pantalla del programa de terminal.
Los programas típicos de terminal te permiten elegir otros ajustes además de la velocidad de transmisión. Si no estás seguro de qué valores usar, entonces configura a 8 bits de datos, 1 bit de parada, sin paridad y sin control de flujo.
Algunos no te permiten utilizar las señales de control, pero en caso de usar Br@y si podrás hacerlo, pulsando en "DTR" y en RTS cambias el estado de estas señales.
El estado de CTS, CD, DSR, RI, DTR y las señales RTS se indican por los colores en los botones correspondientes. PuTTY es otro programa de terminal para Windows para comunicaciones serie.



Si quieres enviar y recibir bytes no-ASCII puedes usar el Pololu Serial Transmitter Utility de Windows o el Br@y.

6.c Escribir PC Software para usar el puerto serie

Puedes escribir tu programa de PC para comunicarte con el puerto serie. El Microsoft.NET Framework contiene la clase SerialPort que hace fácil la lectura y escritura de bytes por ese puerto.
Aquí tienes un ejemplo en código C#.NET que usa el puerto serie:

```
// Escoge puerto y velocidad en baudios.
System.IO.Ports.SerialPort port = new System.IO.Ports.SerialPort("COM4",
115200);
// Abre el puerto.
port.Open();
// Transmite dos bytes por la linea TX: 1, 2
port.Write(new byte[]{1, 2}, 0, 2);
// Espera recibir una byte de la linea RX.
int response = port.ReadByte();
// Muestra el byte recibido.
MessageBox.Show("Recibido byte: " + response);
// Cierra el puerto para que otros programas puedan usarlo.
port.Close();
```

7. Asegurar una buena señal de radiofrecuencia

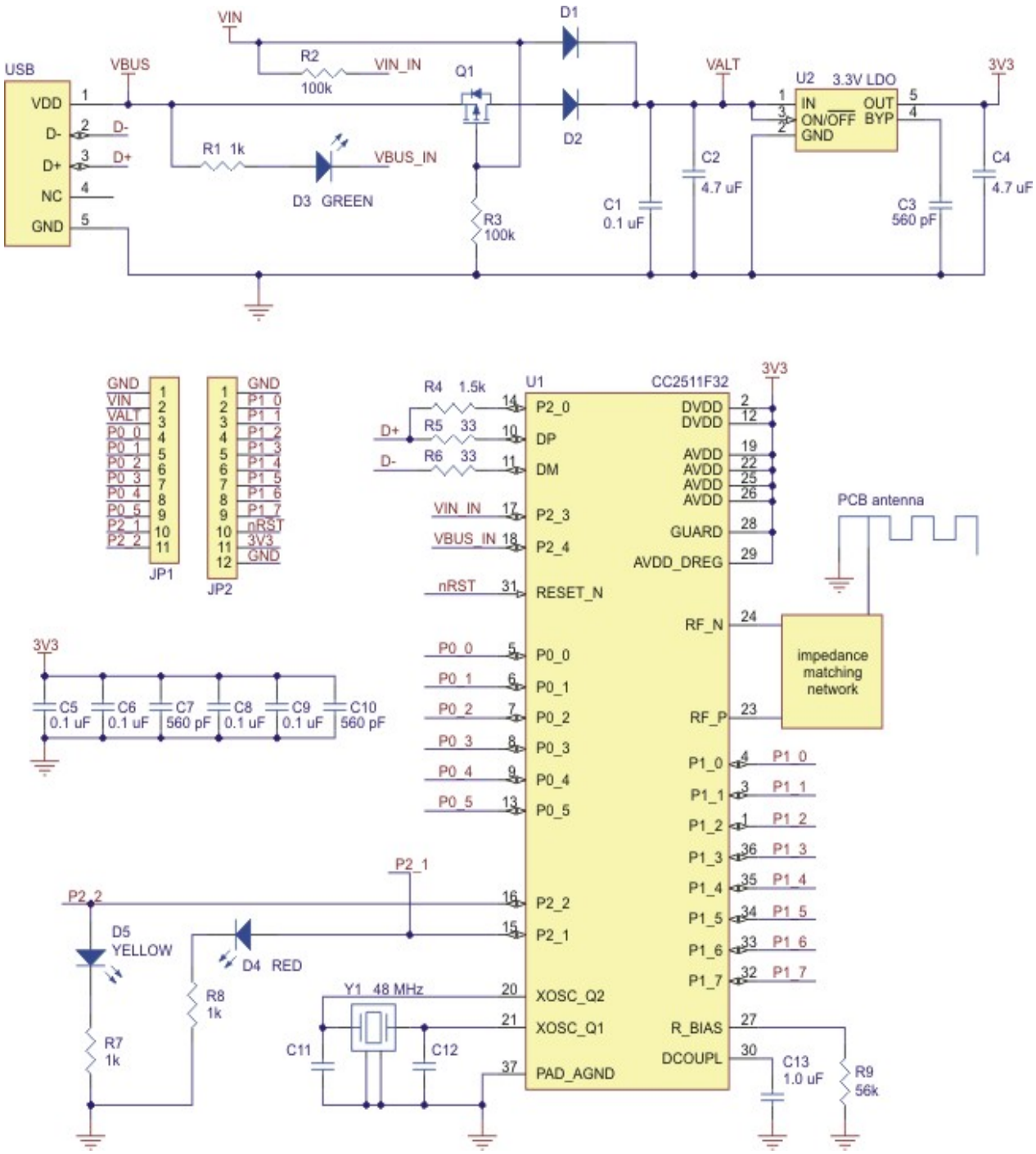
Aquí van unos consejos para mejorar la calidad de las señales de RF entre un par de Wixels:

- Reducir la distancia si hay obstáculos entre Wixels, si es posible.
- Escoger entre diferentes frecuencias. Muchas aplicaciones que usan la RF tienen el parámetro *radio_channel* que determina la frecuencia a usar. Intercambiando canales puedes mejorar la comunicación al eliminar interferencias que pueda haber en la banda de 2.4 GHz. Quizás quieras comprar un analizador de espectro como *Wi-Spy* para encontrar frecuencias con menos actividad o con menos interferencias dentro del área de trabajo.

Quitar los objetos que están muy cerca de la antena del Wixel. Por ejemplo, si el Wixel no esta enclavado en ninguna placa de prototipos y se apoya sobre el escritorio debes encontrar una manera de que el Wixel esté por lo menos una o dos pulgadas por encima de la mesa.
Reducir los obstáculos entre los dos Wixels. Muchos objetos pueden interferir con las ondas de radio de 2,4 GHz, incluyendo paredes, árboles, gente y todo lo que contenga agua.

- Buscar diferentes orientaciones del Wixel. La antena de la Wixel envía y recibe mejor en unas direcciones que en otras.

8. Esquema del circuito Wixel



9. Aplicaciones para Wixel

Vamos a ver diferentes ejemplos de aplicaciones para Wixel.

9.a Blink LED.app

Es un ejemplo de cómo hacer que parpadee el led rojo en función de un periodo configurable. Mira la sección 3.d para saber como configurarlo y hacer que funcione esta aplicación.
Descarga: **`example_bink_led_v1.0.wxl (11k wxl)`**

9.b Wireless Serial.app

Introducción
Esta aplicación sirve para conectar dos Wixels para crear un enlace bidireccional inalámbrico. Usa una RF a 350 kps que es capaz de transmitir 10KB de datos/segundo entre una distancia aproximada de 16 metros (unos 50 pies en condiciones típicas de interior).
También se puede utilizar para convertir Wixel en un adaptador serie USB para TTL.
Esta aplicación se puede ejecutar en varios pares de Wixels siempre que cada par funcione con un canal de radiofrecuencia diferente (la diferencia de canales debe ser de 2 al menos para evitar interferencias).
La aplicación está diseñada para pares de Wixels; no funcionará correctamente si hay tres y están trabajando con el mismo canal de radio.

Instrucciones de instalación
Descarga: **`Wireless Serial App v1.3 (26k wxl)`**. Ábrelo en el *Wixel Configuration Utility*, escoge tus parámetros y escríbelo en los dos Wixels. Mira la sección 4 para más información.

Pin	Función	
P1_0	DTR	Salida de propósito general
P1_1	RTS	Salida de propósito general
P1_2	DSR	Salida de propósito general
P1_3	CD	Salida de propósito general
P1_5	PA_PD	Salida depuración de transmisión radio
P1_6	TX	Transmisión datos serie (0–3.3V)
P1_7	RX	Recepción datos serie (0–3.3 V, no tolera 5V)
P0_0	Arduino DTR	Para programación wireless en Arduino cuando usemos el Wixel shield

Descripción
Este dispositivo aparece al host USB como puerto COM Virtual (con ID de producto USB 0x2200). Si estás utilizando Windows, deberías ver una entrada con la etiqueta "Wixel" en el Administrador de dispositivos de Windows en "Ports (COM & LPT)" mientras se ejecuta la aplicación.
Hay tres modos básicos de comunicación serie que pueden seleccionarse:

1. `USB a Radio`: Bytes serie desde el puerto USB de COM virtual enviados a la radio y viceversa.
2. `UART a Radio`: Bytes desde la línea RX de UART enviados a la radio y bytes de la radio enviados por la línea de TX a la UART.
3. `USB a UART`: Al igual que un adaptador serie normal de USB a TTL, bytes enviados desde el puerto COM virtual en línea de TX de UART y bytes desde línea de RX de UART enviados al puerto COM virtual. Puede seleccionar el modo que vas a utilizar al establecer el parámetro serial_mode con el número adecuado (según lista anterior) o se puede dejar el modo en 0 (valor

por defecto), entonces el Wixel elige automáticamente el modo en función de cómo se trabaje y tal como vemos en la siguiente tabla alternará entre los diferentes modos sobre la marcha.

El pin RX tiene una resistencia de pull-up de 20 kΩ.

El pin PA_PD (P1_5) es una salida de depuración que está baja mientras el Wixel está transmitiendo un paquete por radio. El formato de los datos serie utilizados por esta aplicación es de 8 bits de datos, un bit de parada, sin paridad, que se expresa como 8-N-1. Los datos son no invertida, tal que 0V representa 0 lógico y 3.3V representa 1 lógico.

Auto-Detect Serial Mode (modo serie = 0)	
Alimentación	**Modo serie**
USB solo	USB-a-Radio
VIN solo	UART-a-Radio
USB y VIN	USB-a-UART

Indicadores LEDs

El led **verde** se ha descrito en la sección 1.a y parpadea cuando los datos circulan por el USB

El led **amarillo** representa el estado de funcionamiento en RF. Si el Wixel esta en modo serie y no se usa la radio el led está apagado. En caso contrario se enciende y apaga lentamente mientras la comunicación se establece por primera vez y después parpadea brevemente una vez por segundo mientras los datos se van enviando o recibiendo de forma inalámbrica.

El led rojo indica errores. Parpadeará brevemente si se recibe un byte en línea RX de la UART que debe ser descartado porque los búferes de recepción están llenos. El led se enciende cuando hay un error de paquetes en la línea de RX y permanecerá así hasta que la línea RX vuelva a alta.

Control Señales

Además de la transmisión bidireccional serie, esta aplicación también transmite los valores de cuatro señales de control: DTR, RTS, DSR y CD. Estos nombres provienen del protocolo RS-232, pero en realidad con esta aplicación no tienen el mismo papel: son señales de control digital de propósito general que pueden llevar a cabo cualquier tipo de dato que desees, siempre y cuando estos cambien lentamente (sobre 5 Hz o más lento) y queden limitados a dos bits en cada dirección.

En el modo `USB-a-Radio`, las señales DTR y RTS, del USB se transmiten a los otros Wixel, mientras que las señales de control de recibidas otros Wixel son enviadas al USB como DSR y CD.

En el modo `UART-a-Radio`, las señales DSR y CD de los pines de entrada digital se transmiten de forma inalámbrica a los otros Wixel, mientras que el control de las señales recibidas de otros Wixel se transmiten a los pins de salida DTR y RTS.

En modo `USB-a-UART`, las señales DTR y RTS de USB se transmiten a los pins de salida correspondientes, mientras que los valores de DSR y los pins de entrada CD se retransmiten a USB.

Si dos Wixels se comunican uno con el otro de forma inalámbrica y ambos están en el modo de `UART-a-Radio` o ambos están en modo `USB-a-Radio`, la correspondencia entre las líneas de control es el siguiente: DSR en un Wixel corresponde a DTR en los otros Wixel, mientras que RTS de un Wixel se corresponde a un CD en el otro. La configuración por defecto de esta aplicación (como se muestra en la tabla de arriba) da al Wixel dos pins de salida invertida (DTR y RTS) y dos pins de entrada invertida (DSR y CD). Estos pins al ir invertidos dan un valor lógico de 0 como alto (normalmente 3,3V), mientras 1 es 0V (GND) bajo. Al cambiar los parámetros de configuración (ver más abajo), puedes desactivar estas señales y volver a asignarlas a diferentes líneas E/S o puedes agregar entradas y salidas no invertida. Cualquier pin configurado como entrada tiene una resistencia interna de pull-up de 20k a no ser que las asignes a P1_0 o P1_1 que no tienen resistencias internas. No tienes que conectar nada en los pins de control de señal con el fin de enviar y recibir datos en serie. Estos pins son opcionales.

Parámetros generales

serial_mode: Selecciona el modo serie (1–3, ver lista) o modo auto-detect (0). Por defecto es 0.

baud_rate: El ratio de velocidad usado por la UART, en bits por segundo. Por defecto es 9600. Recomendamos no exceder la velocidad de 115200. Este parámetro no afecta la comunicación a través del puerto virtual COM (USB).

radio_channel: Es el numero de canal que esta entre 0 y 255 y determina la banda de frecuencia. Por defecto es 128. Los Wixels aparejados deben tener el mismo canal para comunicarse entre sí. Los que no deberían tener una diferencia entre los canales apareados (2 o más) para no interferirse.

Por ejemplo, podrías tener un par de Wixels a 128 y otro par en el canal 130.

framing_error_ms: Es el numero aproximado de milisegundos necesario para deshabilitar la recepción UART's al encontrarse con un error en la línea RX. Los valores están entre 0–250. Por defecto es 0 a menos que el receptor no queramos que se deshabilite cuando haya errores.

Parámetros para asignación de pins

Los parámetros siguientes se usan para reasignar el control de las señales a diferentes pins del Wixel. El valor de cada uno puede ser un número de pin no usado. El numero que se computa es por multiplicación del primer digito del nombre del pin por 10 y añadiendo el segundo digito del nombre del pin. Por ejemplo: si buscas asignar el pin DSR al P1_2 debes ajustar el **nDSR_pin** a 12. Para deshabilitar la señal (no asignar a ningún pin), pondrás el parámetro en -1.

nDTR_pin: Pin de asignación de la salida no invertida DTR. Por defecto es **10** (P1_0).
nRTS_pin: Pin de asignación de salida invertida RTS. Por defecto es **11** (P1_1).
nDSR_pin: Pin de asignación de entrada invertida DSR. Por defecto es **12** (P1_2).
nCD_pin: Pin de asignación de entrada invertida CD. Por defecto es **13** (P1_3).
DTR_pin: Pin de asignación de salida no invertida DTR. Por defecto es **-1** (deshabilitado).
RTS_pin: Pin de asignación de salida no invertida RTS. Por defecto es **-1** (deshabilitado).
DSR_pin: Pin de asignación de entrada no invertida DSR. Por defecto es **-1** (deshabilitado).
CD_pin: Pin de asignación de entrada no invertida CD. Por defecto es **-1** (deshabilitado).

arduino_DTR_pin: Es el pin de asignación de salida para programación de un Arduino cuando usamos el Wixel shield. Por defecto es **0** (P0_0).

No se deben permitir entradas invertidas y no invertidas para la misma señal y al mismo tiempo. En concreto, ya sea nCD_pin o CD_pin debe ser -1 y ya sea nDSR_pin o DSR_pin debe ser -1.

Ejemplos de uso

Esta aplicación se puede utilizar para crear un enlace inalámbrico serie entre dos micros, sin usar USB (con excepción de la configuración inicial del Wixel). Para ello, utilizaremos el modo `UART a Radio` para ambos Wixels.

Se puede utilizar para hacer un enlace inalámbrico serie entre un ordenador y un microcontrolador. Usaremos el modo `USB a Radio` para el Wixel que está conectado a la computadora y el modo de `UART a Radio` para el Wixel que esté conectado al microcontrolador. Si ambos Wixels están alimentados de la forma habitual, deben ser capaces de utilizar el auto-detect (serial_mode = 0).

Si estás usando la opción 2 y el modo auto-detect (serial_mode = 0), entonces tienes la opción de (en cualquier momento) conectar el cable USB directamente al Wixel que está conectado al microcontrolador para establecer un conexión más directa (por cable) con el microcontrolador. (Por supuesto, los puertos COM deberán ser distintos al hacer esto.)

Advertencias

Los datos se perderán si el Wixel que recibe bytes en la línea RX va más rápido de lo que la radio pueda transmitir a los otros Wixel. Si tienes problemas, intenta reducir la cantidad de datos enviados a la línea RX mediante la reducción de velocidad de transferencia o la adición de retrasos en el código de su microcontrolador.

Precaución: Las líneas de E/S de Wixel **no son tolerantes a 5V**. Debes utilizar niveladores, diodos, o divisores de tensión para conectar Wixels a salidas de sistemas con 5V. Además, evita el consumo de más en una línea de E/S de lo esta pueda ofrecer (véase el análisis de P1_0 y P1_1 en Sección 1.a). Evita conectar varios pins de salida juntos.

El Wixel no es compatible con los niveles de voltaje RS-232 utilizado habitualmente por los puertos serie DB9 de los ordenadores. Las líneas de Wixel de E/S, incluyendo la línea RX y TX, funcionan con voltajes de entre 0 y 3,3V. Para hacerlo se necesita un sistema de nivelación y de inversión de las señales como la placa Pololu 23201a (RS-232 es invertido, el Wixel espera no-invertido).

Versiones

`Wireless Serial App v1.3` (26k wxl), actualizada en 2011-06-20: Añadido el parámetro de framing_error_ms y su acción oportuna para deshabilitar la recepción UART después del error. Cambia el comportamiento de los leds rojo y amarillo; en versiones anteriores, el rojo estaba apagado y el amarillo simplemente indicaba la alimentación en VIN.

`Wireless Serial App v1.2` (24k wxl), actualizada en 2011-04-06: Añadido soporte para el control de señales. Como resultado el protocolo de radio usado **NO es compatible con versiones anteriores**. También se fija un fallo en la línea TX que ocurría al empezar o resetear. Añade parpadeo al led verde para indicar transferencia de datos en la línea USB.
`Wireless Serial App v1.1` (18k wxl), actualizada en 2011-03-23: Mejora del protocolo radio fijando un problema de la v1.0 que resultaba de resetear un Wixel pero no los otros, cuando (dependiendo del estado del otro Wixel) este tenia el 50% de probabilidades de enviar el siguiente paquete radio en cualquier dirección y que este fuera ignorado por el receptor.
`Wireless Serial App v1.0` (18k wxl), actualizada en 2011-03-22: Inicial

Versiones configuradas para el Wixel Shield para Arduino

Se trata de versiones especiales de la aplicación que tienen el mismo código que las versiones estándar correspondiente, pero tienen diferentes configuraciones para que funcionen correctamente con el Wixel Shield para Arduino. La velocidad predeterminada se cambió a 115200, velocidad de transmisión utilizada por el bootloader de Arduino Uno. Todos los parámetros de asignación de pins se establecen en -1 (desactivados), excepto el arduino_DTR_pin, que se queda en 0 (P0_0). El parámetro framing_error_ms está en 5. Los únicos parámetros de estas aplicaciones que pueden ser modificados por el usuario son los de canal y el baud_rate. Para más información sobre la configuración de esta versión, consulte la Sección 2.c de la Wixel Shield User's Guide.
`Wireless Serial App (versión 1.3)` configurada para el Wixel Shield de Arduino (25k wxl)

9.c USB-to-Serial App

Introducción

Esta aplicación te permite convertir un Wixel en un adaptador `USB-a-TTL` capaz de velocidades de hasta 350.000 bps. Aunque esta aplicación no hace uso de la radio, tiene más características que el modo de `USB-a-UART` de la aplicación Wireless Serial App (ver Sección 9.b).

Instrucciones de instalación

Descarga **`USB-to-Serial App v1.0 (13k wxl)`**. Ábrela con el *Wixel Configuration Utility* y grábala en el Wixel. Ver sección 4 para detalles.

Pin	Función	
P1_0	DTR	Salida de propósito general controlada por el PC
P1_1	RTS	Salida de propósito general controlada por el PC
P1_2	DSR	Entrada de propósito general para el ordenador
P1_3	CD	Entrada de propósito general para el ordenador
P1_6	TX	Transmisión datos serie desde el ordenador
P1_7	RX	Recepción de datos y envío al ordenador

Descripción

Después de haber cargado la aplicación en un Wixel, aparecerá en el equipo como puerto de COM Virtual (con ID de producto USB 0x2200). Si está utilizando Windows, deberías ver una entrada "Wixel" en el Administrador de dispositivos en "Ports (COM & LPT)" mientras se ejecuta la aplicación. Puedes conectarte a este puerto COM mediante un programa de terminal para enviar y recibir datos sobre las líneas de TX y RX. Los programas de terminal típicos permiten establecer la velocidad en baudios, tipo de paridad y bits de parada. Otros permiten utilizar las señales de control (DTR, RTS, DSR y CD). Para más detalle consulta sección 6.

Soporta velocidades entre 23 y 350,000 bps.
Soporta diferentes tipos de paridad: None, Odd, Even, Mark y Space.
Soporta el modo de 1 o 2 stop bits.
RX tiene resistencia interna de pull-up que puedes desconectar.
Los pins DSR y CD de entrada tienen resistencias de pull-up internas que puedes desconectar cuando la lectura este en alto (lógica 0).

Las salidas DTR y RTS están diseñadas para niveles altos de corriente (ver información en P1_0 y P1_1 en la sección 1.a).
El control de las señales esta invertido en todas, por lo que al recibir un 0 lógico se corresponderá con un voltaje alto (3.3V) y un 1 con el voltaje bajo (0 V).
Esta aplicación descartará los bytes recibidos en la línea RX que tengan errores de paquete o de paridad y también rechazará los bytes si existe saturación en el búfer RX. Un desbordamiento del búfer no debería ocurrir si se utiliza una velocidad de 350.000 baudios-por-segundo o inferior.

Ejemplos de uso

La línea TX puede usarse para enviar comandos a un microcontrolador o a otro dispositivo serie.
La línea RX puede usarse para recibir datos desde el microcontrolador o de otro dispositivo serie.
Las líneas DTR y RTS son salidas que pueden usarse de varias maneras (como leds) desde el PC.
Las líneas DSR y CD son entradas de propósito general a las que se les pueden conectar sensores u otros circuitos y leerse en el ordenador.

Advertencias

Las UARTs de CC2511 no admiten 1,5 bits de parada, por lo que si se intenta establecer el número de bits de parada en 1,5, esta aplicación utilizará 1 stop bit en su lugar. Las UARTs de CC2511 no trabajan bien con 2 bits de parada, por lo que si se establece el número de bits de parada en 2, la aplicación puede no detectar errores de paquetes que se produzcan durante el segundo bit de parada. Además, el siguiente byte recibido después del error de paquete puede ser descartado aun siendo este válido. Este problema sólo se aplica a la recepción de bytes por la línea RX; la aplicación no tiene problemas de transmisión de bytes con la línea TX en modo de 2 bits de parada.
Precaución: La misma, las líneas de E/s del Wixel no toleran 5V por lo que debes usar niveladores, divisores de tensión o diodos. También debes tener cuidado de que la potencia consumida en las líneas de E/S sea la soportada. No conectes múltiples salidas a un pin.
Wixel no soporta el protocolo RS232 debido al voltaje de sus pins.

9.d I/O Repeater App

Introducción

Esta aplicación inalámbrica permite ampliar el alcance de las líneas de su microcontrolador hasta 50 pies, unos 15 metros usando dos o más Wixels. Una pin de entrada en un Wixel se asigna a un pin de salida en otro Wixel. Cuando la conexión de entrada está alta, la conexión de salida estará también en alta (3.3V) y al contrario Cada Wixel puede llegar a tener hasta 15 pins de entrada, 15 pins de salida o una mezcla de ambos. Cada conexión de entrada puede asignarse a uno o más de salida en uno o más Wixels.

Instrucciones de instalación

Descarga: **`I/O Repeater App v1.1 (20k wxl)`**. Abrela con *Wixel Configuration Utility*, escoge los ajustes y escríbela en dos o más Wixels. Mira sección 4 para más información de cómo hacerlo.

Descripción

Los 15 pins siguientes de cada Wixel pueden usarse como entradas o salidas (o deshabilitarse):

- Todos los pins en el puerto 0: P0_0, P0_1, P0_2, P0_3, P0_4, P0_5.
- Todos los pins en el puerto 1: P1_0, P1_1, P1_2, P1_3, P1_4, P1_5, P1_6, P1_7.
- Pin P2_1 (el pin del led rojo).

El comportamiento de cada pin está determinada por su *link ID*, que es un parámetro que se puede establecer individualmente para cada pin en cada Wixel mediante la Utilidad de configuración Wixel. Un *link ID* de 0 significa que se desactivará (que será una entrada, pero su valor de entrada no tendrá ningún efecto). Un *link ID* negativo entre -1 y -127 significa que el pin será una entrada digital y su valor será transmitido por radio. Un *link ID* positivo entre 1 y 127 significa que el pin será una salida digital y su valor de salida será determinado por el valor de entrada del pin enlazado contrario (negativo) ID del otro Wixel. Por ejemplo, si el pasador P1_3 en un Wixel tiene un *link*

ID de -13, entonces será una entrada digital y su valor se verá reflejado en todos los pines de salida de los que tengan un identificador de enlace en 13 en todos los otros Wixels. Los pins de entrada no tienen ningún efecto sobre los pines de salida que están en la misma Wixel. Si un Wixel esta ejecutando esta aplicación y tiene uno o más pins configurados como entradas entonces transmitirá un paquete de radio único cada 10 ms. (aproximadamente) que contiene los valores de entrada y de link IDs de todas sus entradas. Cualquier Wixel que recibe correctamente este paquete lo procesarla y lo utilizara para actualizar el estado de sus pins de salida.
Esta aplicación funciona con más de dos Wixels en el mismo canal de radio. En ese caso, asegúrese de que no tienes múltiples pins de entrada en Wixels diferentes y con el mismo link ID: de lo contrario, el pin de salida correspondiente cambiará el estado de forma impredecible cada vez que haya un conflicto entre pins de entrada diferentes. Está bien tener múltiples pins de salida con el mismo enlace ID. Cada pin configurado como entrada tiene una resistencia interna de pull-up de 20k , excepto P1_0 y P1_1, que flotan cuando son entradas. Esto significa que si dejas el pin de entrada desconectado, estará en alto por defecto al arrancar.
Cada pin de salida está en bajo (0 V) por defecto antes de que los paquetes de radio se reciben para que cambien su estado.
Después de haber cargado esta aplicación en un Wixel, aparecerá en el equipo como puerto COM virtual (con USB 0x2200 ID del producto). Si está utilizando Windows, puedes verlo en el Administrador de dispositivos en "Ports (COM & LPT)", mientras se esté ejecutando. No se pueden enviar o recibir datos en este puerto COM. Su único propósito es permitir que la utilidad de configuración Wixel pueda poner el Wixel en modo bootloader..

Parámetros

radio_channel: Número de canal entre 0 y 255 que determina la banda de frecuencia a utilizar. Por defecto es **128**.

P*m_n*_link: El link ID del pin P*m_n* en donde *m* es el número de puerto (0–2) y *n* es el numero de pin (0–7).

Ajustes por defecto

Pin	Link ID	Función
P0_0	-1	Entrada con resistencia pull-up.
P2_1 (led **rojo**)	1	Salida enlace a P0_0 del otro Wixel.

Por lo tanto, si cargas esta aplicación en dos Wixels con la configuración predeterminada, debe comportarse como sigue: si nada está conectado a la línea de P0_0 de ambos Wixel, el led **rojo** en ambos Wixels estará encendido. Si se conectas la línea de P0_0 de un Wixel a GND mediante un cable, deberías ver el led **rojo** en la otra Wixel desactivarse. Esto demuestra el funcionamiento básico de la aplicación.

Ejemplos de uso

El pin de salida puede usarse para el control de un led. Asegurando que el consumo se limita con una resistencia entre ambos de 1kΩ.
El pin de entrada puede usarse para la lectura del estado de un pulsador o interruptor. Conectando el mismo a la entrada y a GND, al pulsar o abrir el interruptor estará en alto y al cerrar se pondrá bajo.
Un pin de salida a la entrada de otro microcontrolador.
Un pin de entrada conectado a un pin de salida de otro microcontrolador. La salida no debe dirigirse a voltajes mayores de 3.3V. Acuérdate de los límites en los voltajes si conectas con dispositivos y microcontroladores que trabajen con voltajes de 5V.

Advertencias

El cambio en un pin de entrada se refleja en su correspondiente de salida en un tiempo aproximado de entre 10–100ms, pero los paquetes de radio tiene posibilidad de perderse por lo que no hay ninguna latencia garantizada.
Por lo tanto, esta aplicación sólo es apta para señales digitales de muy baja velocidad, como la señal de un pulsador o la señal utilizada para controlar un led. No es adecuada para las señales de servos PWM o RC. Esta aplicación utiliza E/S digital, lo que significa que cada lectura se transmite como un 0 o 1 por lo que no admite valores analógicos.

Precaución: Recuerda nivelar los voltajes en el caso de sean distintos entre dispositivos. Wixel no tolera 5V. (Mira la discusión en P1_0 y P1_1 en sección 1.a).

Versiones

`I/O Repeater App v1.1` (20k wxl), actualizada 2011-07-26: Fija el bug en v1.0 que prevé P1_6 y P1_7 para ser usados como salidas.

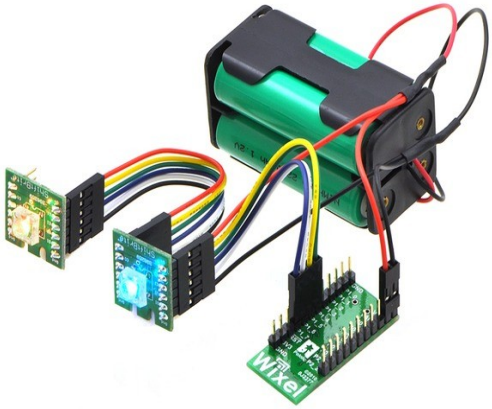
`I/O Repeater App v1.0` (18k wxl), actualizada 2011-03-25: Inicial.

9.e ShiftBrite App

Introducción

Esta aplicación se utiliza para el control inalámbrico de uno o más módulos ShiftBrite (leds RGB) desde una distancia de 50 pies (16m) desde un PC.

Necesitarás dos Wixels para establecer un enlace inalámbrico entre ambos. Uno irá conectado a los módulos ShiftBrite, ejecutando `Shiftbrite.app` y otro conectado al USB del PC ejecutando `Wireless Serial App` (sección 9.b).



Usando un programa terminal o tu propio software y desde el puerto virtual COM enviarás series de caracteres HEX indicando el color deseado para cada módulo.

Los caracteres transmitidos se reciben en el Wixel remoto, son decodificados y enviados a la cadena de ShiftBrite provocando la iluminación de cada módulo con el color enviado. Aproximadamente puedes enviar 1000 comandos de color por segundo produciendo grandes pantallas de iluminación y suaves animaciones de color.

La ShiftBrite App es compatible con el módulo ShiftBar, ya que usa el mismo control electrónico.

Instrucciones de instalación

Sigue las instrucciones para el Wireless Serial App en la sección 9.b para ajustar y testear el enlace básico wireless entre dos Wixels, usando la última versión de la aplicación.

Descarga: **`ShiftBrite App v1.1 (20k wxl)`**. Abre con Wixel Configuration Utility, escoge los ajustes y escribe dentro de uno de los Wixels. Mira la sección 4 para más información.

Conexión del Wixel a la cadena de ShiftBrite

Las siguientes conexiones deben ser realizadas entre el Wixel que ejecuta la aplicación y el primer ShiftBrite de la cadena. Además, el Wixel y ShiftBrites pueden compartir el mismo VIN, siempre y cuando los requisitos de consumo de ambos módulos estén satisfechos. Para la prueba inicial, se puede utilizar la salida VALT para alimentar los ShiftBrites desde USB (consulte la Sección 5.a).

Uso de ShiftBrite App

Wixel	ShiftBrite	Función
P1_4	EI	Enable
P1_5	CI	Clock
P1_6	DI	Datos
P1_7	LI	Latch
GND	GND	Masa

Después de hacer las conexiones correctas y su alimentación, abre un programa de terminal y conecta con el puerto COM creado por el Wixel para ejecutar `Wireless Serial App`. Escribe `ffffff`. A medida que escribes los caracteres, el eco regresa a la consola. Pulsa Intro y el primer ShiftBrite en su cadena se iluminará en color blanco. Ahora escribe `ff0000` y pulsa Enter, ahora el primer ShiftBrite será de color rojo y el segundo de color blanco (si existe).

Para cambiar el color con múltiples comandos a la vez, por ejemplo cuando se desea establecer los colores para toda la cadena, escribe una serie de comandos de un solo color seguidos y a continuación pulsa Enter para aplicarlos todos de golpe.

Parámetros

radio_channel: El número de canal entre 0 y 255 que determina la frecuencia a usar en la banda. Por defecto es 128. Los dos Wixels deben tener el mismo canal para comunicarse entre sí.

input_bits: El número de bits por canal que quieres usar para enviar la información. Los valores permitidos están entre 1 y 16. Por defecto es de 8. Corresponden a los colores representados por

valores de 6 dígitos hexadecimales. Sin embargo, el ShiftBrite soporta 10 bits de resolución, así que puedes elegir un valor de 10 para hacer uso de una mayor gama. En este caso, debes enviar 9 dígitos para definir un color. Por ejemplo, 3ff3ff3ff es el blanco más brillante posible y 001000000 es el rojo más oscuro posible.
Otros valores de este parámetro puede ser útiles en situaciones especiales (por ejemplo, 4 bits de resolución) permite especificar un color en tres bytes, como el f00 para el rojo, con lo que obtenemos una velocidad de actualización más alta.

Formato de datos

Los datos consisten en series de valores para rojo, verde y azul (RGB) en cadenas ASCII hexadecimales. Cada valor contiene de 1 a 4 caracteres, dependiendo del valor de *input_bits* especificando un número entre 0 y 2^{input_bits}-1.
Cuando un juego completo de valores para R, G y B se ha recibido los valores se multiplican o dividen con el factor apropiado generando un paquete de datos de 10 bits usado por el ShiftBrite y que se irá desplazando por la cadena de módulos.
Un carácter de ENTER (ASCII 10 o 13) causará que el pin LATCH se invierta e instantáneamente ajuste cada ShiftBrite a su nuevo color.

Consejos

El led amarillo está normalmente encendido y parpadea cuando se reciben los datos por radio, puede ser útil para la depuración de la comunicación inalámbrica. Puedes utilizar grupos de cables con terminales para crear las conexiones entre el Wixel y la cadena de ShiftBrite.

Versiones

`Shiftbrite App v1.1` (20k wxl), actualizada 2011-04-06: Cambio en el led amarillo que normalmente estará encendido, pero parpadeara siempre que se reciban datos por radiofrecuencia. Esta versión es solo compatible con la Wireless Serial App v1.2 y posteriores.
`Shiftbrite App v1.0` (18k wxl), actualizada 2011-03-28: Inicial. En esta versión el led amarillo está apagado al inicio y se enciende después de recibir el primer Enter. Esta versión es solo compatible con Wireless Serial App versiones 1.0 y 1.1.

9.f Wireless Tilt Mouse App

Introducción

Esta aplicación te permite hacer un ratón inalámbrico de inclinación para tu equipo. Según la inclinación del ratón, podrás controlar la posición del cursor.
El ratón también es compatible con dos pulsadores para los clicks.
Se requieren dos Wixels: un emisor y un receptor.
El Wixel transmisor detecta una inclinación desde un acelerómetro y transmite de forma inalámbrica esa información al Wixel receptor que los reenvía al ordenador como movimientos de ratón.
El receptor aparece en el ordenador como un dispositivo de interfaz USB estándar humana (HID) y después de configurar el Wixels no hace falta instalación de drivers para utilizar ese ratón.

Partes necesarias

Hay varias formas de desarrollar este proyecto. Usaremos el siguiente material:

- 2 Wixels (ensamblados) + USB Cable
- 1 MMA7361L 3-Axis Acelerómetro ±1.5/6g
- 1 Porta pilas para 3 baterías AAA, cerrado y con interruptor
- 1 Placa de prototipos
- 2 Pulsadores

Sugerimos los kits de cables grimpados de Pololu con sus terminales a punto para las conexiones.
Los pulsadores son opcionales, y cualquier pulsador o conmutador nos puede servir si permanecen abiertos.
Si deseas tener el botón izquierdo y el botón derecho del ratón debes usar dos pulsadores. Nota en la imagen anterior se muestra sólo un botón (en P1_2).

Wixel configuración

Descarga: **`Wireless Tilt Mouse App (26k wxl)`**. Abrela con *Wixel Configuration Utility*, elige los ajustes y escríbela en el **transmisor** Wixel. El transmisor puede aparecer en el ordenador por el puerto virtual COM (como USB producto ID 0x2200). No puedes enviar o recibir datos por ese puerto. Su único objetivo es dejar que la Wixel Configuration Utility ponga el transmisor en modo bootloader cuando se conecte al ordenador a través de USB.

Descarga: **`Wireless Tilt Mouse Receiver App`** (15k wxl). Abrela con *Wixel Configuration Utility*, ajústala y escríbela dentro del **receptor** Wixel. El receptor puede aparecer como (HID) Human Interface Device. Si estas usando Windows puedes ver una nueva entrada en el Administrador de dispositivos llamada "HID-compliant mouse" en la sección "Mouse y otros dispositivos de puntero". El receptor también aparecerá como un teclado, pero no utiliza dicha interfaz. La *Wixel Configuration Utility* no puede aún detectar Wixels con interfaces USB HID. Por lo tanto, si necesitas volver a configurar el receptor Wixel, tendrás que hacerlo manualmente en modo bootloader (configurar pin P2_2 alto y hacer un reset, aunque esta aplicación supervisa P2_2 todo el tiempo por lo que no es muy necesario).

Ensamblado

Con las baterías desconectadas, conectar la alimentación para el transmisor Wixel. El cable negro de la batería va a GND en el Wixel. El cable rojo se conecta VIN. Para probar, enciende la batería: el transmisor encenderá el led amarillo del Wixel. Desactiva la batería mientras realizas conexiones.

Pin	Función de los pins de conexión del Transmisor
P0_1	Entrada analógica para velocidad vertical del ratón
P0_2	Entrada analógica para velocidad horizontal del ratón
P1_2	Entrada botón izquierdo con pull-up en alto
P1_7	Entrada botón derecho con pull-up en alto

Conecte el GND de la Wixel al pin GND (masa) del acelerómetro. Conecta la alimentación para el acelerómetro. El acelerómetro MMA7361L trabaja a 3,3V y el consumo de corriente es bastante bajo por lo que lo conectamos a la línea de 3V3 del Wixel desde VDD del acelerómetro.

Haz las conexiones necesarias para activar el acelerómetro MMA7361L, conectando SLP a VDD.

Elige el eje del acelerómetro que controlara el movimiento horizontal del cursor del ratón y conecta su salida al pin de P0_2 del Wixel transmisor. Si tu acelerómetro está orientado como se muestra en la imagen anterior, debes utilizar el eje Y.

Elige el eje del acelerómetro que controlara el movimiento vertical del cursor del ratón y conectar su salida al pin de P0_1 del Wixel transmisor. Si tu acelerómetro está orientado como se muestra en la imagen anterior, debes utilizar el eje X.

Conectar un pulsador entre el pin P1_2 y el GND del transmisor. Será el botón izquierdo del ratón. La línea de entrada P1_2 tiene una resistencia interna de pull-up, por lo que la tensión de la línea estará alta (3.3V) cuando no se presiona el botón. Ira a baja (0V) al pulsar. De la misma manera, conecta un pulsador entre el pin P1_7 y el GND del transmisor.

Este será el botón derecho del ratón.

Parámetros

Ambas aplicaciones, transmisión y recepción tienen el parámetro **radio_channel**. El número de canal estará entre 0 y 255 y determina la frecuencia de funcionamiento. Por defecto es 128. Deben tener el mismo canal ambos. Si tienes interferencia cambia las frecuencias en ambos.

La aplicación de transmisión tiene parámetros adicionales para configurarse:

invert_x: ajusta a 1 para invertir el movimiento horizontal del ratón. Por defecto es **0**.

invert_y: ajusta a 1 para invertir el movimiento vertical del ratón. Por defecto es **0**.

speed: Un número positivo determina la velocidad del ratón. Por defecto es 100. La velocidad es proporcional al número ajustado.

Partes alternativas

Puedes reemplazar el pack de baterías por una de 9V y un regulador que saque un voltaje adecuado para el Wixel de 2.7–6.5V, como el step-down voltage regulator D24V3ALV, o puedes usar 1 o 2 células que con un regulador apropiado suministre 5V, como el boost regulator NCP1402.

Puedes reemplazar el acelerómetro con otro cuyas salidas sean iguales a la mitad del voltaje de entrada cuando la aceleración en su eje correspondiente sea de 0.

Cálculo de la velocidad del ratón

La velocidad del ratón se calcula con un factor de aceleración media del dispositivo. El vector de aceleración apunta normalmente hacia arriba y tiene un valor de 1g por la gravedad de la Tierra. El Wixel mide los resultados X e Y del acelerómetro que le indican los componentes del vector de aceleración del dispositivo en el plano del acelerómetro. La resultante de dos dimensiones se multiplica por su propia magnitud para lograr un mayor control a bajas velocidades. El vector se multiplica por el parámetro de velocidad para convertirlo en una velocidad del ratón recibida por el ordenador de forma inalámbrica.

9.g Serial-to-I²C App

Introducción

Esta aplicación convierte el Wixel en un puente serie-a-I²C, actuando como master en un bus I²C.
Para realizar las operaciones I²C, otro dispositivo le puede enviar comandos serie ASCII al Wixel por radio, UART o interfaz USB.
I²C es una interface de dos hilos muy común en comunicaciones entre periféricos como el compás LSM303DLH y el sensor de presión BMP085. Las características del protocolo I²C han sido publicadas por NXP (UM10204.pdf de 371k).

Instrucciones de instalación

Descarga: **`Serial_to_I2C App 1.0 (26k wxl)`**. Abrela en *Wixel Configuration Utility*, ajusta sus parámetros y escríbela en el Wixel. Mira la sección 4 para más información.

Pin	Nombre	Función de los pins de conexión
P1_0	SCL	I²C clock (0–3.3V)
P1_1	SDA	I²C data (0–3.3V)
P1_6	TX	transmisión datos serie (0–3.3V)
P1_7	RX	Recepción datos serie (0–3.3V)

Descripción

Este dispositivo aparecerá como USB host en el puerto Virtual COM (USB producto ID 0x2200). Si usas Windows tendrás una entrada “Wixel” en el Administrador de dispositivos

Hay tres modos de crear un puente que puedes seleccionar:

`Radio-a-I²C: Los comandos enviados por radio se usan en el control de las transacciones I²C. Los datos leídos por un esclavo I²C slave se devuelven vía radio. Otro Wixel deberá ejecutar la aplicación Standard Wireless Serial app para poder comunicarse.`
`UART-a-I²C: Los comandos serie por la línea RX de la UART se usan para el control de transacciones del bus I²C. Los datos leídos desde el esclavo se devuelven a la línea TX de la UART.`
`USB-a-I²C: Los comandos serie por el USB virtual COM port se usan para el control de transferencias I²C. Los datos leídos del esclavo I²C se devuelven al puerto virtual COM.`

Debes seleccionar que tipo de puente vas a utilizar mediante el parámetro **bridge_mode** con el número que corresponda (ver listado). Por defecto es 0 que se corresponde al puente Radio-a-I²C.
El pin RX de UART tiene una resistencia de pull-up de 20 kΩ .
Las líneas I²C (SCL y SDA) están por defecto en P1_0 y P1_1 respectivamente pero estas pueden cambiarse con los parámetros **I2C_SCL_pin** y el **I2C_SDA_pin**. No tienen pull-ups activados, por lo que deben agregarse pull-ups externos para que el bus se ajuste a las especificaciones I²C.
(Algunas placas para dispositivos I²C incluyen ya, las resistencias de pull-up para estas líneas).
La frecuencia del bus I²C bus es de 100 kHz por defecto pero puede modificarse con el parámetro **I2C_freq_kHz**. La aplicación no soporta I²C multimaestro, por lo que no puede usarse en un bus I²C que trabaje con varios maestros.

Comandos serie

Esta aplicación escucha los comandos en la interfaz serie seleccionada y realiza las transacciones I²C que correspondan. El formato del comando es similar al utilizado por el controlador de bus I²C

NXP SC18IM700 maestro con interfaz UART. Tres comandos, en forma de caracteres ASCII, se pueden reconocer.

El formato general para una secuencia de comandos comienza con un START ('S'), seguido de la dirección del dispositivo esclavo, el número de bytes de datos que se escribirán o leerán, los datos a escribir (si está en escritura) y al final un comando STOP ('P'). Si se hace una lectura, se devolverán los datos leídos desde el dispositivo esclavo. El bit menos significativo de la dirección del dispositivo esclavo (el bit de dirección de datos) indica si la operación es de escritura (0) o de lectura (1). Podemos general un START repetido mediante la emisión de otro comando START sin enviar el comando STOP.

Comando	Descripción
'S'	I²C START
'P'	I²C STOP
'E'	Get Errors

Por ejemplo, para escribir el valor `0x2E` al registro `0xF4` en el dispositivo esclavo basta con escribir la dirección `0xEE`, seguida de la secuencia de bytes que serán enviados por el Wixel:

`'S', 0xEE, 2, 0xF4, 0x2E, 'P'`

Para leer el valor de dos bytes del registro `0xF6` del mismo dispositivo, cuya dirección más un bit tiene que ser `0xEF`, irá seguida de la secuencia de repetición (el segundo 'S' es el repite START):

`'S', 0xEE, 1, 0xF6, 'S', 0xEF, 2, 'P'`

El Wixel respondería con dos bytes de datos procedentes de la lectura del esclavo. Cualquier comando no válido o no reconocido se ignora y hace que el bit de Invalid Command error se active.

Para evitar una secuencia de comandos sin terminar salidos del bus I²C en un estado inusual, esta aplicación tiene un tiempo de espera y resetea el bus si el byte no se recibe en la interfaz en menos de 500 ms. desde el último byte. Después de esto, el bit del Command Timeout error se activa y una nueva secuencia de comandos se puede iniciar con un START. El retraso de tiempo de comandos se puede modificar con el parámetro *cmd_timeout_ms* y un valor de 0 lo desactiva.

Si el Wixel recibe un comando *Get Errors ('E')*, responderá con un solo byte que contiene el estado de diversas condiciones de error. El comando puede emitirse en cualquier momento y la aplicación no espera la dirección de esclavo ni los bytes de datos.

Cuando ocurre un error, el bit correspondiente en el byte de error se activa y permanece hasta que los errores se leen con el comando *Get Errors*, después de lo cual se limpia el byte. La siguiente tabla muestra la condición de error que corresponde a cada bit:

Bit	Error	Descripción
0 (LSB)	NACK en dirección	Recibe NACK en el bus I²C después de transmitir la dirección del esclavo.
1	NACK en datos	Recibe NACK en el bus I²C después de transmitir un byte de datos.
2	I²C Timeout	La línea SCL ha estado baja mucho tiempo (sin respuestas) y el bus I²C se ha reseteado. El retraso permitido es de 10 ms. por defecto y puede modificarse con el parámetro **I2C_timeout_ms**.
3	Comando no válido	De ha recibido un comando no válido.
4	Comando Timeout	Retraso entre comandos. El timeout es de 500 ms. por defecto y puede cambiarse o desactivarse con **cmd_timeout_ms**.
5	UART Overflow	El buffer de recepción de la UART está lleno.
6	UART Error	Un error de paquetes ha ocurrido en la UART.

Indicadores leds

El led verde parpadea cuando hay los datos se transfieren a través de USB.

El led amarillo se enciende cuando se detecta alimentación en VIN poder.

El led rojo indica una condición de error cuando está encendido, se puede restablecer mediante la emisión de un comando *Get Errors* (descrito anteriormente).

Parámetros generales

bridge_mode: Selecciona el modo de puente (0–2, ver abajo). Por defecto es **0**.

baud_rate: Velocidad de la UART, en bits/segundo. Por defecto es **9600**. Recomendamos no exceder de 115200. Este parámetro no afecta a la comunicación el puerto virtual COM (USB).

I2C_freq_kHz: Frecuencia de reloj para el bus I²C, en kilohertz. Por defecto es **100**. Las más comunes son 10 kHz (baja), 100 kHz (estándar) y 400 kHz (alta).
El rango de frecuencias posibles es de 2-500kHz; a causa redondeos e inexactitudes y restricciones de tiempo, la frecuencia real puede ser menor que la frecuencia seleccionada, pero nunca será mayor de la ajustada.
I2C_timeout_ms: Retardo en milisegundos, antes de que la línea SCL cause un I²C bus timeout. Esto resetea el bus y activa el bit de I²C Timeout error. Por defecto es **10**.
cmd_timeout_ms: El retraso permitido, en milisegundos, por no recibir el siguiente byte de una secuencia en tiempo de espera. Esto resetea el bus I²C y activa el bit de Command Timeout error.
El valor por defecto es 500. Un valor de 0 lo desactiva.
radio_channel**:** El número de canal es de 0 a 255 y determina que la frecuencia de radio. El valor por defecto es 128. Los Wixels deben estar en el mismo canal para comunicarse entre sí.
Parámetros para asignación de pins
Los siguientes parámetros se utilizan para reasignar las líneas I²C a diferentes pins en el Wixel. El valor de cada parámetro debe ser el número de un pin no utilizado en el Wixel.
El número puede calcularse multiplicando el primer dígito en el nombre de pin por 10 y agregando la segunda cifra del nombre de pin. Por ejemplo, si desea asignar el pin SCL a P1_2, establecer **I2C_SCL_pin** a 12.
I2C_SCL_pin: Pin de asignación para la línea I²C SCL (reloj). Por defecto es 10 (P1_0).
I2C_SDA_pin: Pin de asignación para la línea I²C SDA (datos). Por defecto es 11 (P1_1).
Advertencias
Si el Wixel recibe comandos en la línea RX de la UART más rápido de lo que puedan procesarse en el bus I²C, se perderán datos.
Si tienes problemas, intenta reducir la cantidad de datos enviados por la línea RX disminuyendo la velocidad o agregando retrasos en el código del microcontrolador.
El comando *Get errors* comprueba el overflow de la UART y permite detectar estos problemas.
Precaución: Recuerda nivelar los voltajes en caso de que entre dispositivos sean diferentes.
El Wixel no tolera líneas a 5V. (Mira la discusión en P1_0 y P1_1 en sección 1.a).
El Wixel no soporta voltajes usados por los puertos RS-232 o DB9. Las líneas de E/S incluyendo las líneas RX y TX del Wixel operan con voltajes de entre 0 y 3.3V. Recuérdalo.
Versiones: `Serial-to-I2C App 1.0` (26k wxl), actualizada 2011-05-03: Inicial.

10. Escribiendo tu aplicación para Wixel

10.a Aplicación desde Windows

Para empezar a desarrollar tus propias aplicaciones Wixel utilizando Windows como plataforma de desarrollo, te recomendamos que descargues e instales el **Pololu Wixel Development Bundle**: `wixel_dev_bundle_110415.exe (10MB exe).`
Este paquete contiene cuatro aplicaciones:

1. *Wixel SDK*: Código fuente (librerías y app) que te ayudará en el desarrollo de la aplicación.
2. *SDCC – Small Device C Compiler*: Un compilador libre que debes usar con el código programado en Wixel SDK.
3. *Pololu GNU Build Utilities*: Versiones en Windows de algunas utilidades de código abierto requeridas por el SDK.
4. *Notepad++*: Un editor de texto gratuito muy bueno para editar las líneas de código para Wixel.

10.b Compilar un ejemplo de App

Después de haber instalado el software necesario (ya sea desde el paquete de desarrollo de Wixel o desde otras fuentes), debes tratar de compilar una aplicación de ejemplo y cargarlo en un Wixel para asegurarse de que todo funciona correctamente.

Compilación de Apps

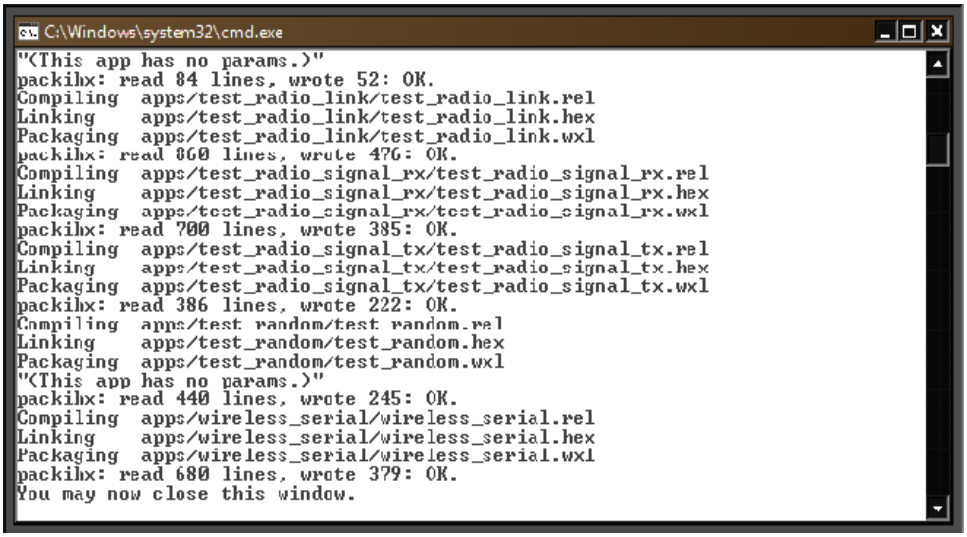
Vete a la carpeta donde se instaló el SDK de Wixel (`C:\wixel-sdk` es el valor predeterminado) y ejecuta `make_all.bat`. Si todo se ha instalado de forma correcta, la salida de ese archivo por lotes debe ser algo parecido a la imagen, se han construido todas las aplicaciones en el SDK de Wixel y todas las bibliotecas que dependen de ellos.

Cada subcarpeta de la carpeta **apps** ahora debe tener un archivo WXL en ella. Se trata de una aplicación compilada que puede ser cargada dentro del Wixels mediante *Wixel Configuration Utility* o la utilidad de línea de comandos Wixel (*WixelCmd*).

Podrías obtener el siguiente mensaje de error de hacerlo en Windows:

```
make: Interrupt/Exception caught (code = 0xc00000fd, addr = 0x425073)
```

Si te sale este error, ejecuta "`make -v`" desde la línea de comandos y comprueba que estas ejecutando GNU Make 3.82-pololu1. Esta versión de trabajo esta incluida en la última del Wixel Development Bundle (ver sección 10.a).

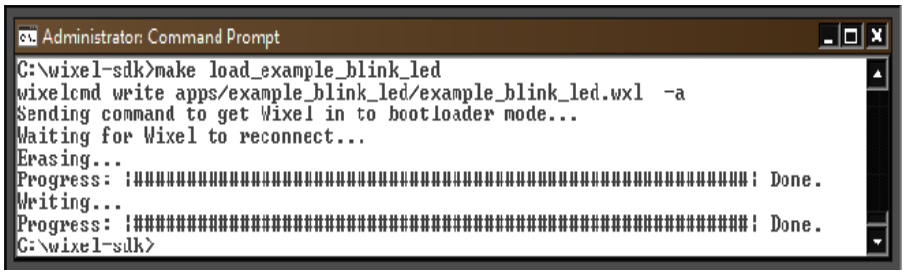


Si la salida de "make -v" muestra alguna otra versión después de instalar Wixel Development Bundle, tendrás que desinstalar desde el PATH o redireccionar a la versión 3.82-pololu1 que es la que debes usar. Puedes editar el PATH desde el Panel de Control. Mira la pololu/make wiki on GitHub para más información sobre este problema.

Carga de una aplicación en Wixels desde la línea de comandos

Cuando estés desarrollando una aplicación puede ser útil el disponer de una forma de carga de una aplicación en el Wixel desde la línea de comandos.

Puedes hacer esto con Makefile: conecta un Wixel a tu equipo, abre una ventana de símbolo del sistema, vas a la carpeta `wixel-sdk` y escribe `make load_example_blink_led`.



Esto hará que se compile la aplicación (si esta actualizada) y a la vez llamara a Wixel Command-Line Utility (WixelCmd) para grabar la misma en el Wixel conectado al ordenador. El resultado será parecido a la imagen y la aplicación example blink LED deberá ejecutarse ahora. El led amarillo del Wixel estará apagado y el rojo parpadeando. Bien, has conseguido compilar y grabar el programa en el Wixel

También puedes especificar la configuración desde la línea de comandos. Escribe:

```
make load_example_blink_led S="blink_period_ms=100"
```

Ahora verás que el led parpadea 5 veces más rápido que antes. Para especificar varias ajustes, debes ponerlos para todas las configuraciones, dentro de las comillas y separados por espacios.

Apertura de una aplicación con la utilidad de configuración de Wixel

También puedes abrir la aplicación en la *Wixel Configuration Utility* si lo deseas. Ejecuta:

```
make open_example_blink_led
```

La *Wixel Configuration Utility* arrancará y abrirá la aplicación: example_blink_led app.

Crear tus propias aplicaciones

Ahora que sabes cómo compilar aplicaciones y cargarlas rápidamente en el Wixel para probarlas, estás listo para desarrollar tus propias aplicaciones. Puedes modificar una de las existentes en el SDK o crear una propia. Para crear una aplicación simplemente copia una de las carpetas de la aplicación existente y cámbiale el nombre. No es necesario modificar el archivo Make cuando se crea una nueva aplicación el archivo Make detectará automáticamente la aplicación mientras se encuentra en la carpeta de aplicaciones. Cuando estás desarrollando tu aplicación, puedes utilizar todos los comandos de arriba pero recuerda reemplazar "example_blink_led" con el nombre de tu aplicación (el nombre de la subcarpeta en la carpeta de aplicaciones).

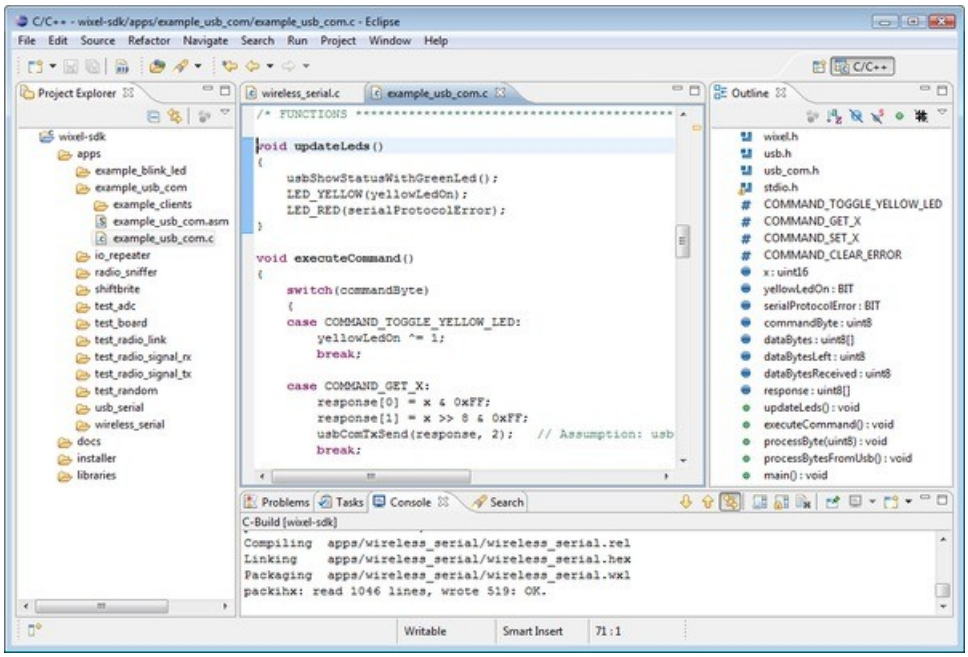
10.c Uso de Eclipse IDE

Puedes desarrollar aplicaciones usando un editor de texto. Si usas uno libre como Eclipse IDE podrás aprovecharte de las características avanzadas de C/C++. El siguiente tutorial te mostrará como ajustar Eclipse para usarlo con el Wixel SDK.

1.- Instala el Wixel SDK en el PC y compila un ejemplo siguiendo las instrucciones de las secciones anteriores.

2.- Descarga e instala el `Eclipse IDE for C/C++ Developers`. Si eres usuario de Windows puedes descargar la versión de **32-bit** de Eclipse, a no ser que tengas el de *64-bit*. Necesitas instalar el `Java Runtime Environment (JRE)` y después de correr Eclipse.

3.- Ejecuta Eclipse. Te pedirá una carpeta para el área de trabajo. Eclipse no permite carpetas de trabajo y de proyectos iguales. Por lo tanto, se recomienda hacer una nueva carpeta llamada algo así como `eclipse_workspace`, mover la carpeta `wixel-sdk` dentro de ella y eligir esa nueva carpeta como la de ubicación del área de trabajo. De esta forma, puedes evitar tener un gran número de archivos en el espacio de trabajo. Más tarde puedes cambiar la ubicación del área de trabajo si es necesario.



4.- Después de elegir la carpeta de área de trabajo, verás una pantalla de bienvenida que dice *"Bienvenido a la IDE de Eclipse para desarrolladores en C/C ++"* y tiene varios iconos circulares sobre ella. Haga clic en el icono de la flecha amarilla para ir a la mesa de trabajo.

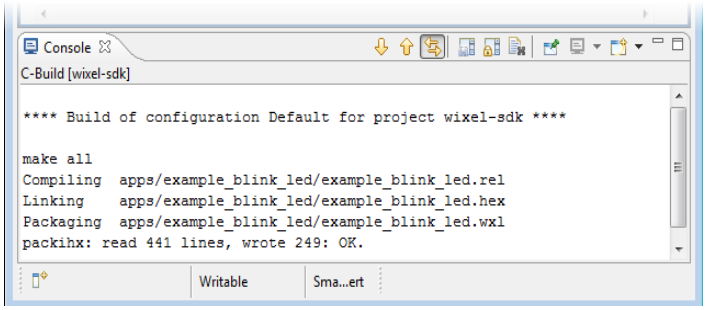
5.- Desde menú **File**, selecciona **New** > **C proyecto**. Se abre un cuadro de diálogo titulado *"Proyecto en C"*.

6.- Pon nombre al proyecto, `wixel-sdk`. El cuadro **Location** debe mostrar ahora la ubicación correcta de la carpeta `wixel-sdk`. Eclipse mostrará una advertencia en la parte superior de la ventana que dice *"Directory with specified name already exists."* (Vamos bien!).

7.- Bajo **Project type**, selecciona **Makefile project** > **Empty Project**. No importa que *toolchain* elijas.

8.- Clic en **Finish**. Después de crear el proyecto, Eclipse ejecutará automáticamente `make` en la carpeta `wixel-sdk`. Puedes ver los resultados generados seleccionando la pestaña **Console** en la parte inferior o ir a **Window** > **Show View** > **Console**.

9.- En el lado izquierdo del panel puedes ver **“C/C++ Projects”** o **“Project Explorer”**. Debe mostrar una entrada titulada `wixel-sdk` que representa el proyecto que acabas de crear. Clic en el triángulo situado a la izquierda del nombre del proyecto para expandir y ver todas las carpetas y archivos en su interior.



10.- Desplázate hasta la carpeta `apps/example_blink_led` y haz doble clic en `example_blink_led.c`. Se abrirá el archivo en el panel central.

11.- Haz un pequeño cambio en la misma como la eliminación de una línea en blanco o añadir un comentario para luego guardarlo pulsando **Ctrl + S** o seleccionando **Archivo > Guardar**.

12.- Presiona **Ctrl + B** o selecciona **Proyecto > Construir todo** para generar el proyecto Wixel SDK. Al hacer esto, Eclipse ejecuta `make` en la carpeta `wixel-sdk` y lo construye con todas las aplicaciones y bibliotecas dependientes. Una vez más, puedes ver los resultados generados desde la pestaña **Console** en la parte inferior o en **Window > Show View > Console**. El Wixel SDK’s Makefile detecta que `example_blink_led.c` ha cambiado, por lo que debe reconstruir la aplicación. La salida en la vista de consola debe ser algo parecida a la imagen.

13.- Felicitaciones! Has hecho el primer cambio en la aplicación y su compilación con Eclipse IDE.

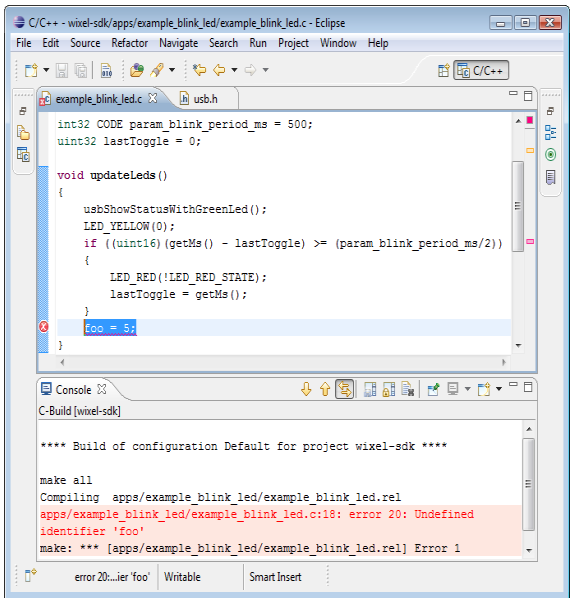
14.- Cuando estés listo para grabar una aplicación en el Wixel abre el Command Prompt busca en el directorio `wixel-sdk` y ejecuta `make load_APPNAME` donde *APPNAME* es el nombre de la aplicación. No tenemos otra forma integrada en Eclipse para carga de aplicaciones en el Wixel .

Características avanzadas de Eclipse

El atractivo de utilizar Eclipse es el gran número de características avanzadas para la edición en C/C++ que posee. Como:

- Eclipse admite resaltado de sintaxis C.
- Si se produce un error durante la compilación, puede hacer doble clic en la consola de error para saltar directamente a la línea de código que tiene el problema.

Si se sitúa el cursor de texto en un nombre de función, nombre de variable, macro preprocesador o directiva include, puedes pasar rápidamente al lugar donde ese elemento está definido pulsando F3 o con el botón derecho y seleccionando **Open Declaration.**



- Pulsando **Ctrl+Tab** sobre el fichero fuente (.c) abrirás el fichero cabecera correspondiente (.h) y viceversa. Esto es muy útil si escribes librerías para el Wixel.
- La vista esquema muestra todas las variables y funciones definidas en el archivo actual.
- Puedes hacer clic en uno de ellos para ir rápidamente a su definición. Incluso puedes arrastrar en la vista esquema para cambiar su orden en el archivo.
- Si pones el cursor sobre el nombre de función, puedes ver todos los sitios en donde se llama a esta función pulsando **Ctrl+Alt+H** o botón derecho y seleccionando **Open Call Heirarchy**.
- Eclipse hace muy fácil el renombrado de objetos (variables, métodos/funciones, etc.) propagando automáticamente los cambios a los otros ficheros del proyecto. Puedes acceder a esta ventaja pulsando el cursor sobre el nombre, con **Ctrl+Shift+R** o botón derecho y Refactor > Rename…

- Eclipse busca en todos los ficheros y comentarios que contengan “TODO”. Estos comentarios se agregan en la vista de tareas y también fácilmente se puede encontrar en el archivo actual haciendo clic en los rectángulos azules un poco a la derecha de la barra de desplazamiento vertical. Cada rectángulo azul representa la ubicación de un elemento “TODO” en el archivo.

Ocultar funciones que no se utilizan.

Por defecto Eclipse tiene un gran número de botones en la barra de herramientas, vistas (paneles) y elementos de menú que pueden no ser necesarios para desarrollar aplicaciones para Wixel. Puedes simplificar enormemente la interfaz de usuario ocultando esos elementos. Para ocultar selecciona **Window > Customize Perspective…**. Aparece la ventana "**Customize Perspective C/C++".**
Selecciona "Command Groups Availability". Puedes desmarcar un grupo de comandos y estos desaparecerán de forma automática de la barra de herramientas y de los menús. Los grupos de comandos útiles para desarrollar aplicaciones de Wixel son:
`C/C++ Coding, C/C++ Editor Presentation, C/C++ Element Creation, C/C++ Navigation, C/C++ Open Actions, C/C++ Search, Editor Navigation, Editor Presentation, Keyboard Shortcuts, Open Files, Search`
Si quieres personalizar más, usa el **“Tool Bar Visibility”** y **“Menu Visibility”** para mostrar u ocultar comandos individuales

10.d Compartir tu App con la comunidad Wixel.

Preparar la aplicación para la comunidad.

Asegúrate de que estas secciones están en tu fichero WXL:
description – Debe describir totalmente lo que hace tu aplicación o enlace a un sitio Web con su descripción completa. Como otras personas pueden usar esto en sus Wixels, asegúrate de poner tu atención en cualquier cosa que pudiera causar daños o alguna incompatibilidad. Importante, los puertos de E/S que se usan como salidas deben aparecer explícitamente listados en la descripción.
license – especifica las condiciones bajo las cuales otros pueden distribuir copias o modificaciones de tu aplicación. Te recomendamos utilizar la licencia MIT, que es simple y permite el uso generalizado del código.

```
Copyright (c) 2011 <YOUR NAME>.
Documentation for this app is available at: http://<YOUR SITE>/
Copyright (c) 2011 Pololu Corporation.
For more information, see
http://www.pololu.com/
http://forum.pololu.com/
Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
```

Ten en cuenta que si has usado SDK de Wixel para construir tu aplicación, deberás incluir nuestra licencia, pero puedes agregar restricciones propias si es necesario. Utiliza la plantilla, hará fácil para todos el compartir tu aplicación para su uso o para incluirla en el SDK de Wixel.

Anuncio de tu aplicación en el forum Wixel

Cuando tengas lista la aplicación para que la comunidad pueda probarla, puedes anunciarla en el Foro Wixel y adjuntar el archivo WXL. Es útil pegar una copia de la descripción en el mensaje para que se pueda ver lo que hace la aplicación, antes de descargarla.

Contribuyendo a su código fuente

Compartir el código fuente aumenta enormemente el valor de la aplicación, ya que otros miembros de la Comunidad pueden ser capaces de adaptarla a sus propios proyectos o aprender del código.

También puedes realizar cambios que te gustaría incorporar en la sección común del SDK de Wixel o haber descubierto correcciones que deseas compartir. Por esa razón mantenemos el repositorio de SDK de Wixel Pololu en GitHub como ubicación central de código para Wixel.
Puedes utilizar fácilmente este sitio para mostrar lo que estás trabajando, presentar cambios y otros.
GitHub usa el Git versión Control System para el seguimiento de versiones y ramificaciones de código y recomendamos aprender el uso de Git si quieres compartir con la comunidad Wixel o para ti para comprobar las últimas versiones. De Teaching Git hay tutoriales disponibles en la Web.
El Git tutorial proporcionado por GitHub puede ayudarte a ello.
Para el resto de esta sección, supondremos que estás cómodo con Git.
Si has estado siguiendo las instrucciones de este documento, el código se encuentra dentro de la carpeta `wixel-sdk` junto con el resto de los SDK de Wixel. Esta carpeta es realmente un *clon* de nuestro repositorio Git, lo que significa que incluye historia e información sobre la versión, así como un vínculo de vuelta al repositorio (a distancia), dentro de la carpeta `.git`. Como no has modificado el contenido de `.git`, debes poder utilizar Git para confirmar los cambios, extraer las actualizaciones y enviar los cambios al repositorio público. Para asegurarse de que Git está trabajando correctamente, intenta escribir el siguiente comando dentro de la carpeta `wixel-sdk` :

```
git log
```

Te mostrará una lista reciente de confirmaciones en el repositorio. Alternativamente puedes usar el programa `gitk` o puedes usar el comando `"Show Log"` de TortoiseGit para ver más detalle y la representación gráfica de log. Después de recibir el log puedes añadir y comentar todo sobre tus nuevos ficheros:

```
git add <list of new files...>
git commit -a
```

También puedes usar el comando `"Commit…"` de TortoiseGit cada vez que quieras mostrar la lista de ficheros nuevos, modificados o confirmados. Revisa atentamente y escribe una buena descripción de los cambios porque es difícil modificar las confirmaciones una vez se han hecho públicas.
En cualquier caso si deseas arrancar desde una copia fresca del Wixel SDK, puedes hacer:

```
git clone -o pololu git://github.com/pololu/wixel-sdk.git
```

dentro de la ventana de comandos. Y ese momento descargaras la última actualización y podrás mezclarlas con tu código escribiendo el siguiente comando:

```
git pull pololu
```

Inicialmente, se comprometerán los cambios del repositorio local, pero cuando estén listo para publicarlos, debes configurar tu propio repositorio en GitHub para hacerlos accesibles a la comunidad. Para ello, regístrate para obtener una cuenta de GitHub y pincha en nuestro repositorio al de tu cuenta. Para hacerlo pulsando el botón "**fork**" en la parte superior de nuestro repositorio para que se conecte dentro del sistema de GitHub: de esta manera verás los cambios y los puedes incorporar fácilmente en el SDK. Después de pinchar, puede insertar todos los cambios del repositorio local a GitHub con

```
git remote add myrepo <GitHub URL>
git push myrepo
```

dende `<GitHub URL>` es el SSH o HTTP URL mostrado en tu página GitHub.
Por último, quizás desees enviar un enlace a tu repositorio de GitHub en el Foro Wixel.
Esto ayudará a otros para saber del mismo y hacer comentarios sobre tu contribución.

10.e Configuraciones USB reconocidas por el Wixel Configuration Utility

La aplicación Wixel Configuration Utility y el Wixel Command-Line Utility pueden reconocer solo ciertas aplicaciones Wixel. Si tu aplicación no se reconoce, los usuarios no podrán reconfigurarla después de haberla integrado en el Wixel (deberán poner el Wixel en bootloader de modo manual).
Para que se reconozca tu aplicación debes implementar un interface USB particular.
Debes utilizar un puerto USB CDC ACM virtual COM con vendedor ID 0x1FFB y producto ID 0x2200. Cuando el host USB envía el comando a tu aplicación ajusta el baud rate a 333, tu aplicación debe entrar en modo bootloader. Esta debe trabajar con el controlador `wixel_serial.inf` que a sido instalado el Windows software (ver sección 3.a). El camino más

fácil para escribir una aplicación que cumpla estos requerimientos es usar `usb_cdc_acm.lib` librería en el Wixel SDK. En el futuro añadiremos soporte para otros tipos de interfaces USB si es necesario para ello. Ponte en contacto si deseas utilizar otros tipos de interfaces en tus aplicaciones.

10.f Formato del fichero de aplicación para Wixel

A efectos de carga de un programa compilado para Wixel, la *Wixel Configuration Utility* y el *Wixel Command-Line* (WixelCmd) aceptan dos formatos de archivo: Intel Hex (.hex) y Wixel App (.wxl). El formato **Intel (.hex)** de archivo es un estándar y puede ser creado con múltiples cadenas.

El formato de archivo **Wixel (.wxl)** es un formato sencillo creado por Pololu que incluye un archivo HEX de Intel en su interior junto con metadatos importantes, como nombres y direcciones de los parámetros configurables por el usuario. El formato de archivo de la aplicación Wixel está diseñado para ser extensible y fácil de generar. Los archivos Wixel son de texto sin formato: cada byte es un carácter ASCII. Los códigos ASCII que se permiten van desde el `0x20` al `0x7F` y el `0x09` (tab), `0xA` (nueva línea, "\ n") y `0xD` (retorno de carro, "\ r"). Los bytes del archivo se dividen en líneas con finales de línea con los de bytes siguientes: `"\r \n" (0xD, 0xA), "\r" (0xD), o "\n" (0xA).`

La primera línea debe ser `"Pololu aplicación Wixel - www.pololu.com"`. Esta línea explica el propósito de la aplicación Wixel. La segunda línea debe ser `"1.0"`. Especifica la versión del formato de archivo utilizado por el archivo. La tercera línea y todas las líneas siguientes se dividen en secciones. Cada sección tiene una línea de cabecera en la parte superior que especifica el nombre de sección. La línea de cabecera se compone de 6 signos de igual (=) seguidos de un espacio y seguido del nombre de la Sección. El orden de la sección no importa.

Las secciones requeridas por el software Wixel son **cdb** y **hex**.

Sección HEX

Contiene el archivo de Intel Hex que especifica los bytes a escribir en la memoria del Wixel. Ver la página de Intel Hex en Wikipedia para más información. La región de memoria flash en el CC2511F32 que está disponible para las aplicaciones va desde `0x400 a 0x77FF` (inclusive).

Cualquier línea del archivo HEX que escriba fuera de esta región será ignorado.

Sección de licencia

Es una sección con texto legible que debe especificar los términos bajo los cuales otros pueden distribuir copias o modificaciones de la aplicación. El SDK Wixel inserta automáticamente la licencia del MIT en todos los archivos de la aplicación, porque así se requiere.

Sección descripción

La descripción es también legible que describe lo que hace o el enlace Web acerca de la aplicación.

Sección cdb

Contiene los nombres y direcciones de los parámetros configurables por el usuario. Para cada parámetro, debe haber dos líneas en esta sección (no necesariamente consecutivos) como:

Una línea que contiene la dirección del parámetro y debe ser de la forma:

L:G$param_NAME$0$0:ADDRESS donde *NAME* es el nombre y *ADDRESS* especifica la dirección del parámetro en la memoria con 4 dígitos hexadecimales (por ejemplo, "11C3").

Otra línea contiene el tipo de parámetro y debe ser de la forma:

S:G$param_NAME$0$0(TYPE),D,0,0 donde *NAME* es el nombre y *TYPE* es el tipo de parámetro. Actualmente, el único permitido es entero de 32 bits con signo, que se codifica como *"{4} SL: S"*.

El SDK Wixel genera la sección cdb para ejecutar grep en el fichero CDB producido por SDCC durante el paso de enlace, seleccionando todas las líneas que contengan la cadena *"param"*. Esto significa que puede definir un parámetro simplemente escribiendo el siguiente código en su aplicación Wixel, fuera de cualquier función:

int32 CODE param_NAME = DEFAULT_VALUE;

donde *NAME* es el nombre de su parámetro y *DEFAULT_VALUE* es el valor por defecto que tendrá si el usuario no altera su valor. Este valor por defecto se almacena en la sección hex no en la cdb.

Se puede poner todo el fichero CDB en el archivo de WXL si quieres, el software hace caso omiso de las líneas no reconocidas. Sin embargo, el archivo CDB no debería ser muy largo.

La aplicación de Wixel tienen una línea non-Windows finalizando la sección cdb para verse correctamente en el Notepad o para otros editores de texto que solo reconocen líneas Windows finales.

11. El USB Bootloader para Wixel

El Wixel viene con un gestor de arranque desde USB que puede utilizarse junto con la Wixel Configuration Utility o el WixelCmd para cargar aplicaciones en el Wixel (no se requiere programador externo). Esta sección documenta algunos detalles técnicos del bootloader y está pensada para usuarios avanzados.

Flash Memory

El gestor de arranque ocupa el primer 1 KB (1024 bytes) y los últimos 2 KB de la memoria flash del CC2511F32's. Estas secciones de memoria están protegidas de posibles corrupciones del cargador de arranque. El resto de los 29 KB están destinados a las secciones de la aplicación. El bootloader no pone restricciones en los datos que se pueden escribir en esas secciones.

Sin embargo, considera que la aplicación es inválida y no permite ejecutar el su código si el primer byte de la aplicación (dirección `0x400`) es `0xFF`, lo que nunca debe ser el caso para una aplicación compilada correctamente. El vector de entrada estándar del CC2511 y los vectores de interrupción se reasignan por el gestor de arranque para el inicio de la aplicación. El vector de entrada de la aplicación se debe colocar en `0x400`. El vector de interrupción primero debe estar en `0x403` y los vectores de interrupción siguientes estarán a 8 bytes de distancia. La aplicación se puede poner en modo de arranque al saltar a la dirección `0x6` mediante una instrucción `ljmp`.

El bloque de memoria flash desde `0x3CC` a `0x3ff` en el gestor de arranque se utiliza para almacenar información que puede ser leída por la aplicación:

Las direcciones entre `0x3CC-0x3DD` contienen el descriptor de dispositivo USB del gestor de arranque, tal como se define en la especificación de USB 2.0.

Las direcciones entre `0x3E0-0x3E3` contienen el número serie del Wixel en 32 bits entero sin signo.

Las direcciones `0x3E6-0x3FD` contienen el número serie del Wixel en formato USB descriptor.

Los bytes entre `0x3CC-0x3ff` no mencionados antes están reservados y sólo contienen ceros.

Procedimiento de arranque

Cada vez que se alimenta el Wixel, se ejecuta primeramente el código del bootloader. Este código decide si se debe iniciar el gestor de arranque o la aplicación mediante el procedimiento siguiente:

En primer lugar, configura las tres líneas de leds para que tengan resistencia interna de pull-up y desactiva las resistencias de pull-up y pull-down de los pins del Port 2 . En segundo lugar, si la solicitud no es válida (primer byte es 0xFF), entra en modo bootloader. En tercer lugar, si hay alimentación en USB y el led amarillo (P2_2) esta en alto, entra en modo bootloader.

Por último, si ninguna de estas pruebas lo ha puesto en bootloader, se ejecuta la aplicación.