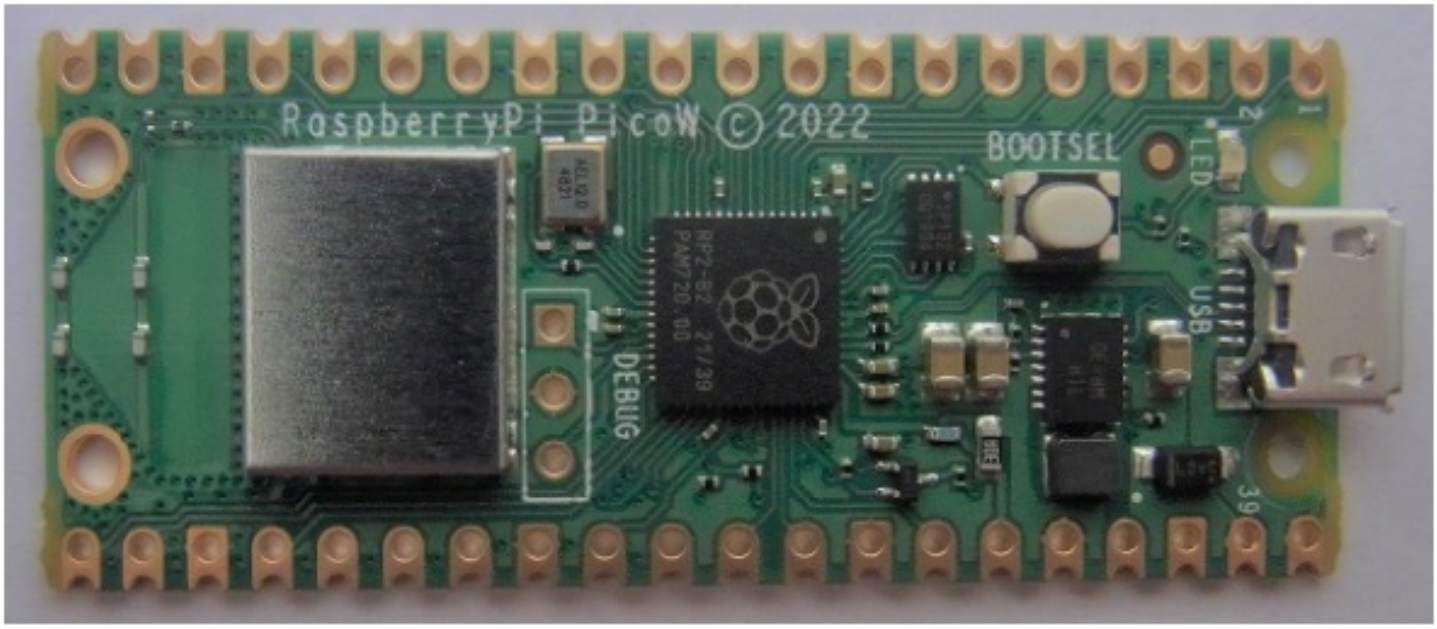
PICOWI: CONTROLADOR WIFI INDEPENDIENTE PARA PI PICO W

6 de diciembre de 2022, Categorías Pico, WiFi

INTRODUCCIÓN



Raspberry Pi Pico W

El objetivo de este proyecto es proporcionar un controlador ***WiFi*** rápido para el chip ***CYW43439*** en el módulo ***Pi Pico W***, con código C ejecutándose en el procesador ***RP2040***; También se puede utilizar con chips similares de ***Broadcom/Cypress/Infineon*** que tengan una interfaz ***SPI***, como el ***CYW4343W***.

Se basa en mi proyecto ***Zerowi*** (https://iosoft.blog/zerowi/) que realiza una función similar en el dispositivo ***Pi Zero CYW43438***, que tiene una interfaz ***SDIO***. Sin embargo, debido a las innumerables dificultades para ejecutar el código, el código se reestructuró y simplificó para enfatizar las diversas etapas en la configuración del chip y para proporcionar una gran cantidad de diagnósticos en tiempo de ejecución.

El enfoque estructurado de los controladores ***WiFi*** se refleja en los programas de ejemplo y las partes individuales de este blog; van desde un simple programa de ***flash LED*** hasta uno que proporciona alguna funcionalidad ***TCP/IP***.

Un problema importante con la depuración del código ***Pico-W*** es la dificultad de conectar cualquier herramienta de diagnóstico de hardware, como un osciloscopio o un analizador lógico; esto se solucionó admitiendo una placa adicional con un módulo ***Murata 1DX*** y un chip ***CYW4343W***; los detalles completos se dan en la parte 1 (https://iosoft.blog/picowi_part1) de este proyecto.

ENTORNO DE DESARROLLO

Para simplificar, use una Raspberry Pi 4 para crear el código y programar el Pico, con dos líneas de E/S conectadas a la interfaz Pico SWD. Esto es realmente fácil de configurar, usando un solo script que instala el SDK y todas las herramientas de software necesarias en el Pi 4:

```
wget https://raw.githubusercontent.com/raspberrypi/pico-setup/master/pico_setup.sh
chmod +x pico_setup.sh
./pico_setup.sh
```

Para la programación SWD, el Pico debe estar conectado a los pines de E/S en el Pi de la siguiente manera:

```
Pico SWCLK   Pi pin 22 (GPIO 25)
Pico GND     Pi pin 20
Pico SWDIO   Pi pin 18 (GPIO 24)
```

La interfaz serial se usa ampliamente para mostrar información de diagnóstico; El pin 1 del Pico es la salida serial y el pin 3 es tierra. Utilizo un adaptador serial ***USB FTDI*** de 3.3 voltios para mostrar la salida serial en una ***PC***, pero podría usar la entrada serial ***Pi 4*** en su lugar.

El código fuente de ***PicoWi*** está en github (https://github.com/jbentham/picowi):

```
cd ~
git clone http://github.com/jbentham/picowi
cd ~/picowi/build
chmod +x prog
chmod +x reset
```

Incluí el CMakeLists.txt necesario, por lo que el proyecto se puede construir usando los siguientes comandos:

```
cd ~/picowi/build
cmake ..       # create the makefiles
make picowi    # make the picowi library
make blink     # make the 'blink' application
./prog blink   # program the RP2040, and run the application
```

Al compilar el pico-SDK actual, hay algunas advertencias de "pase de parámetros" cuando se compila el ensamblador pio; estos pueden ser ignorados.

La compilación es razonablemente rápida en Pi 4; Una vez que se han creado las bibliotecas SDK, puede realizar una reconstrucción completa de la aplicación y la biblioteca PicoWi en 10 segundos, y reprogramar el RP2040 en menos de 3 segundos.

El comando 'restablecer' es útil cuando solo desea reiniciar el Pico, sin cargar ningún código nuevo.

Si ***OpenOCD*** informa "leer DLIPDR", entonces hay un problema incorrecto con el cableado. Establecí la velocidad SWD en 2 MHz, lo que debería funcionar sin errores, siempre que los cables sean lo suficientemente cortos (por ejemplo, menos de 6 pulgadas o 150 mm) y haya buenas conexiones de alimentación y tierra entre Pi y Pico. Utilizo un cable USB corto para alimentar el Pico desde el Pi, y esto generalmente no presenta problemas, aunque a veces el Pi no arranca con el Pico conectado; Esto parece ser un problema de comunicación USB.

CONFIGURACIÓN DE TIEMPO DE COMPILACIÓN

Actualmente solo hay una configuración en el archivo ***CMakeLists.txt***, para elegir entre el dispositivo ***CYW43439*** integrado o un módulo ***CYW4343W*** externo:

```
# Set to 0 for Pico-W CYW43439, 1 for Murata 1DX (CYW4343W)
set (CHIP_4343W 0)
```

La configuración específica de Pico está en ***picowi_pico.h***:

```
#define USE_GPIO_REGS   0       // Establezca un valor distinto de cero para el acceso
                                // directo al registro
                                // (aumenta SPI de 2 a 5,4 MHz de lectura, 7,8 MHz de
                                // escritura)
#define SD_CLK_DELAY    0       // Tiempo de retardo de encendido/apagado del reloj en uSeg
#define USE_PIO         1       // Establezca un valor distinto de cero para usar Pico PIO
                                // para SPI
#define PIO_SPI_FREQ    8000000 // Frecuencia SPI si se usa PIO
```

Estos surgen de la forma en que se maneja la interfaz SPI; el valor predeterminado es usar ***Pico PIO*** (E/S programable) con la frecuencia de reloj dada; 8 MHz es un valor conservador, lo ejecutó a 12 MHz, y posiblemente serían velocidades más altas con algunas configuraciones de la configuración de E/S.

Establecer ***USE_PIO*** en cero habilitará un controlador ***'bit-bashed'*** (o ***'bit-banged'***); esto puede funcionar a más de 7 MHz si se usan escrituras directas de registro, o 2 MHz si se usan llamadas de funciones normales.

Notará que no ha incluido un controlador para el periférico SPI dentro del RP2040; Esto hubiera sido más fácil de usar que el periférico PIO, pero el chip ***CYW43439*** integrado no está conectado a los pines de E/S adecuados. Los pines reales utilizados se definen en ***picowi_pico.h***:

```
#define SD_ON_PIN       23
#define SD_CMD_PIN      24
#define SD_DIN_PIN      24
#define SD_D0_PIN       24
#define SD_CS_PIN       25
#define SD_CLK_PIN      29
#define SD_IRQ_PIN      24
```

Verás que el pin 24 está realizando múltiples funciones; esta configuración de hardware se analiza en detalle en la siguiente parte de este blog. Si está usando un módulo externo, las definiciones de los pines se pueden modificar para usar cualquiera de los pines de E/S del RP2040.

CONFIGURACIÓN DE DIAGNÓSTICO

Mi código hace un uso extensivo de una consola en serie con fines de diagnóstico, y generalmente uso un adaptador en serie ***USB FTDI*** conectado al pin 1 del módulo Pico para monitorear esto. En vista de la gran cantidad de información que se puede producir, modifico la interfaz serial para que corra a ***460800 baudios***:

```
Edit pico-sdk/src/rp2_common/hardware_uart/include/hardware/uart.h
Change definition to:
  #define PICO_DEFAULT_UART_BAUD_RATE 460800
```

Originalmente intenté ejecutar la interfaz a ***921600 baudios***, pero esto resultó en la pérdida ocasional de caracteres, lo cual es un problema importante cuando se trata de entender qué es lo que está fallando en el código.

En su lugar, puede usar el enlace ***Pico USB***; debe habilitarse en ***CMakeLists.txt***, usando el nombre del archivo principal, por ejemplo, para habilitarlo para el programa de ejemplo ***'ping'***:

```
pico_enable_stdio_usb(ping 1)
```

Luego puede usar un programa de terminal como ***minicom en el Pi 4*** para ver la consola:

```
# Run minicom, press ctrl-A X to exit.
minicom -D /dev/ttyACM0 -b 460800
```

Una desventaja de este enfoque es que cuando el Pico se reprograma, su CPU se reinicia, lo que provoca una falla en el enlace USB. Después de unos segundos, el enlace se restablece, pero habrá un espacio en la pantalla de la consola, lo que puede ser engañoso. Además, existe la posibilidad de que la carga de trabajo adicional de mantener una conexión USB (potencialmente muy ocupada) pueda causar problemas de temporización, por lo que, si está haciendo un uso intensivo de los diagnósticos, podría ser necesario utilizar una interfaz serie cableada.

Puede controlar hasta qué punto se informan los datos de diagnóstico en la consola; esto se hace insertando llamadas a funciones, en lugar de usar definición en tiempo de compilación, para brindar un control detallado. Las opciones de visualización están en un campo de bits, por lo que se pueden habilitar o deshabilitar individualmente, por ejemplo:

```
// Mostrar detalles de tráfico SPI
set_display_mode(DISP_INFO|DISP_EVENT|DISP_SDPCM|
                 DISP_REG|DISP_JOIN|DISP_DATA);

// Mostrar nada
set_display_mode(DISP_NOTHING);

// Mostrar transferencias de datos ARP e ICMP
set_display_mode(DISP_ARP|DISP_IP|DISP_ICMP);
```

LA RED WIFI

Por el momento, el código no es compatible con la funcionalidad de punto de acceso dentro del chip WiFi. Solo se puede unir a una red que no esté encriptada, o con encriptación WPA1 o WPA2, según lo establecido en el archivo ***picowi_join.h***:

```
// Configuración de seguridad: 0 para ninguno, 1 para WPA_TKIP, 2 para WPA2
#define SECURITY            2
```

El nombre de la red (SSID) y la contraseña se definen en el archivo "principal" de cada aplicación, por ejemplo, ***join.c*** o ***ping.c***, lo que significa que no son seguros, ya que cualquiera que tenga acceso al código fuente o binario puede verlos. ejecutable:

```
// ¡Contraseña insegura solo con fines de prueba!
#define SSID                "testnet"
#define PASSWD              "testpass"
```

OTROS RECURSOS

Vale la pena leer las hojas de datos de ***CYW43439*** (https://www.infineon.com/cms/en/product/wireless-connectivity/airoc-wi-fi-plus-bluetooth-combos/wi-fi-4-802.11n/cyw43439),
y ***CYW4343W*** (https://www.infineon.com/cms/en/product/wireless-connectivity/airoc-wi-fi-plus-bluetooth-combos/wi-fi-4-802.11n/cyw4343w/), ya que contienen una buena descripción de la interfaz SPI de bajo nivel, pero no contienen nada sobre el funcionamiento interno de estos chips increíblemente complicados. El ***entorno de desarrollo WICED de Infineon*** (https://www.infineon.com/cms/en/design-support/tools/sdk/wireless-connectivity-wiced-software-sdk/) tiene una cobertura muy completa de los chips WiFi, aunque llevaría a cabo algo de trabajo para transferir este código al RP2040. ***El***

SDK de Pi Pico (https://github.com/raspberrypi/pico-sdk) contiene el código fuente completo para controlar el CYW43439, con la pila TCP/IP de código abierto lwIP (IP ligera).

Estoy usando un enfoque diferente, con un controlador de bajo nivel completamente nuevo y una pila de IP integrada para maximizar el rendimiento, como se describe en las siguientes partes:

Introducción (https://iosoft.blog/picowi)
Parte 1: interfaz de bajo nivel (https://iosoft.blog/picowi_part1)
Parte 2: inicialización (https://iosoft.blog/picowi_part2)
Parte 3 : IOCTL y eventos (https://iosoft.blog/picowi_part3)
Parte 4 : escanear y unirse a una red (https://iosoft.blog/picowi_part4)
Parte 5 (https://iosoft.blog/picowi_part5***):*** código fuente ARP, IP e ICMP (https://github.com/jbentham/picowi)

Lanzaré actualizaciones con más funciones de TCP/IP.