Pico-WiFi-Module

Firmware Version 1.01
User Guide
Updated May 16th, 2025

Add-on C-language module to merge with your existing (C-language) project, giving it access to your Wi-Fi network.

IMPORTANT :

This User Guide is about Pico-WiFi-Module Firmware Version 1.01 from Andre St. Louys.
To get the full potential of this library, you must NOT simply clone the GitHub repository.
Read this User Guide and carefully follow the indicated procedure.

Join our Pico-WiFi-Module discussion group on:
https://github.com/astlouys/Pico-WiFi-Module/discussions

Index

Pico-Wi-Fi-Module User Guide

Introduction

The Pico-WiFi-Module is a C-Language add-on module that will allow your existing C-language Pico program or project to access your Wi-Fi network. This could open new opportunities to your projects: access to Network-Time-Protocol (NTP) servers, Message Queuing Telemetry Transport (MQTT) protocol and / or other Internet of things (IoT) features...

To help you figure out the details on how to use the Pico-WiFi-Module, an example is included in the repository: « Pico-WiFi-Example ». This is a basic C-Language program making use of the Pico-WiFi-Module. This simple program is also described later in this User Guide.

To get the full potential of the Pico-WiFi-Module, make sure you carefully follow the instructions given in this User Guide.

1) The « Setup part 1 » covers the steps required to install the Pico-WiFi-Example to your system. Make sure everything is installed as required and that you can build and run the program without problem. You can them go on with the following step to merge the add-on module to your own program.

2) The « Setup part 2 » covers the steps to follow when you want to add the Pico-WiFi-Module to one of your own existing C-Language program.

NOTE: « Pico » is used throughout this User Guide to describe the microcontroller used but of course, since we are using WiFi, a PicoW is required (the « plain Pico » does not have the circuitry to allow Wi-Fi communications). I didn't have the chance to work with the Pico2W so far, but I assume there should not be major changes, if not for the CMakeLists.txt where I assume you must specify which microcontroller is used and which cores. I'm interested in hearing from your experience in adapting this Firmware on the new Pico2W...

You can join other users of the Pico-WiFi-Module on the discussion group on:
https://github.com/astlouys/Pico-WiFi-Module/discussions
I can also be reached with my personal email at:
astlouys@gmail.com

Pico-WiFi-Module setup

Setup part 1: Installing / copying the Pico-WiFi-Module to your system.

NOTE: The instructions below assume that your development system is a Raspberry Pi running the Raspberry Pi O.S. (previously called "Raspbian") and that you're using Visual Studio Code. It also assumes that you followed the recommended naming and structure conventions for the directory structure on your development system. If this is not the case, you will have to adapt the instructions accordingly.

1) Create the project directory on your system:
   ```
   mkdir  /home/pi/pico/Pico-WiFi-Module
   ```

2) Clone / copy all files from the GitHub repository to the « /home/pi/pico/Pico-WiFi-Module » project directory created in step 1 above.

3) While in the « /home/pi/pico/Pico-WiFi-Module » project directory create a symbolic link pointing to the pico_sdk_import.cmake file. This way, if / when you update the Pico SDK, your project will always use the latest version when it is built:
   ```
   ln  -s  /home/pi/pico/pico-sdk/external/pico_sdk_import.cmake
   ```

4) Create a new « UTILITIES » directory in the pico directory. I use to put in this directory bash scripts and other utilities used while developing project for the Pico:
   ```
   mkdir  /home/pi/pico/UTILITIES
   ```

5) Move the file « baseline.h » from the Pico-WiFi-Module project directory to the UTILITIES directory just created:
   ```
   mv  baseline.h  /home/pi/pico/UTILITIES
   ```

6) Create a symbolic link from the project directory to the « baseline.h » file:
   ```
   ln  -s  /home/pi/pico/UTILITIES/baseline.h
   ```

7) The baseline.h file contains basic definitions that I almost always use in most projects. You may want to take a look at it and add your own pieces of code that you use often. When you update the « baseline.h » file in the UTILITIES directory, it is automatically updated in all your projects through the symbolic link.

8) Use your text editor to modify the .bashrc file...
   ```
   sudo  nano  /home/pi/.bashrc
   ```

9) ...and add the following lines:
   ```
   export WIFI_SSID=your network name here
   export WIFI_PASSWORD=your network password here
   ```

This will create the environment variables containing your network name and network password (read next paragraph for more explanations).

10) The CMakeLists.txt file of the Pico-WiFi-Example could be used as an example. It reads the environment variables for your network SSID (network name) and password, and makes them available to the program being built. This way, even if you copy and / or give away your source code, your network name and password are never included in the files (as long as you don't give the executable file where they will be put during the build process). NOTE: The environment variables will be created / generated only the next time you open a session (or next time you reboot to make things simple).

11) This completes the first part of the setup. At this point, make sure you are able to build and use the Pico-WiFi-Example application. Refer to the section of this User Guide on how to use it.

Setup part 2: Integrating the Pico-WiFi-Module to one of your projects.

NOTE: The instructions below assume that you development system is a Raspberry Pi running the Raspberry Pi O.S. (previously called « Raspbian ») and that you're using Visual Studio Code. It also assumes that you followed the recommended naming and structure conventions for the directory structure on your development system. If this is not the case, you will have to adapt the instructions accordingly.

Make sure you carefully followed the instructions in the « Setup part 1 » above, since they are a prerequisite for this section.

1) From your project directory: « /home/pi/pico/Pico-My-Project », create a symbolic link pointing to both the « .c » and « .h » files of the Pico-WiFi-Module:
```
ln  -s  /home/pi/pico/Pico-WiFi-Module/Pico-WiFi-Module.c
ln  -s  /home/pi/pico/Pico-WiFi-Module/Pico-WiFi-Module.h
```

2) From your project directory: « /home/pi/pico/Pico-My-Project », create a symbolic link pointing to the « baseline.h » file:
```
ln  -s  /home/pi/pico/UTILITIES/baseline.h
```

3) In your project file: « /home/pi/pico/Pico-My-Project.c », add the include file « Pico-WiFi-Module.h » file:
```
#include "Pico-WiFi-Module.h"
```

4) In your project file: « /home/pi/pico/Pico-My-Project.c », declare the structure defined in the Pico-WiFi-Module.h file:
```
struct  struct_wifi  StructWiFi;
```

5) As indicated in the Pico-WiFi-Module.h file, some members of the structure must be completed / specified by the user. If you properly followed the instructions in this User Guide, the Wi-Fi network name and network password should have already been extracted from environment variables. Adjust the country code according to your country. You may refer to the CYW43 documentation for your specific country. This allows the PicoW to use the Wi-Fi frequencies used in your country.

6) Copy-paste the function « log_info() » to your program since it is referenced in Pico-WiFi-Module.

7) Remember that your CMakeLists.txt file must be properly configured / adapted to read the environment variables for the network name and password. It must also make reference to all required libraries. You may want to copy the CMakeLists.txt file from the Pico-WiFi-Module directory to your project directory and modify it accordingly.

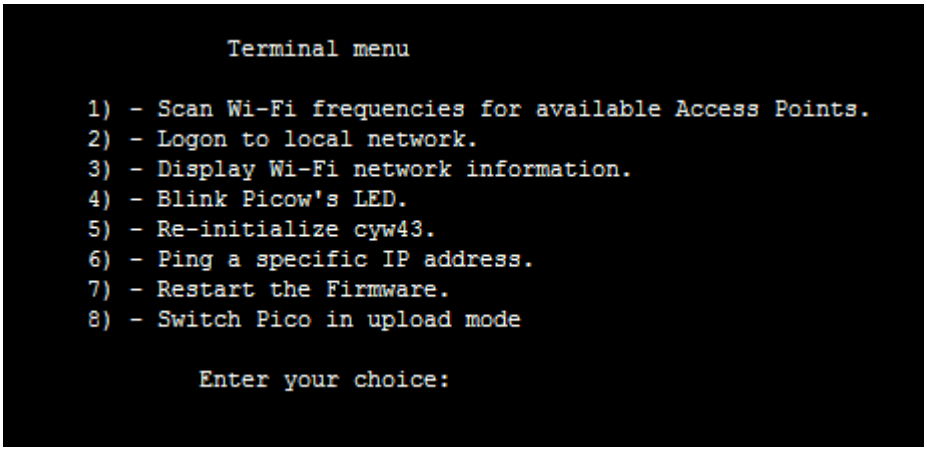
Pico-WiFi-Example

The « Pico-WiFi-Example » application doesn't do much... Its main purpose is to show how to integrate the « Pico-WiFi-Module » to your current project. Nonetheless, here is how to use it.

Build the Pico-WiFi-Example application and download it to your Pico. The application will automatically be launched when uploaded to the Pico.

Once started, the application will blink the Pico's LED while it waits for a USB CDC connection. So, start you preferred terminal emulator program while connected to the Pico. You may have to setup the serial port of the terminal emulator program to enable the communication between the Pico and your terminal emulator program. For those not familiar with this, I strongly recommend version 5 of the TeraTerm terminal emulator program. Its features make it an excellent choice. If you use it frequently, consider sending a contribution to the developer.

When the USB CDC connection has been established, you will see a short menu on the screen. Most items are self-explanatory.



NOTE: As a general guideline, menu choices should be executed in sequence, beginning with choice number 1 and going down with choices number 2, 3, and so on and so forth.

1) « Scan Wi-Fi frequencies for available Access Points »: This menu choice will scan the frequencies reserved for Wi-Fi communications in your country (this assumes that you properly defined your Country Code in the StructWiFi structure as explain above in the setup section). It will the list the Access Points that have been found. The display first shows the Access Point as they are discovered. They are then list in channel order and then in MAC address order. The scan must be done just after the cyw43 has been initialized. It is then a good idea to execute it just after the application is started. If you want to execute it later, you should re-initialize the cyw43 before proceeding (menu choice 5).

2) « Logon to local network »: This menu choice will logon to your local Wi-Fi network using the network name and network password that you specified in your « .bashrc » file in the setup procedure.

3) « Display Wi-Fi network information »: This menu choice will display information related to the Wi-Fi network. Be aware that option 2 (Logon to local network) must be done prior to option 3. Otherwise, some information will be wrong / incomplete.

4) « Blink PicoW's LED »: This option will blink the PicoW's LED a few times. Since the LED on the PicoW is addressed in a different way as the « plain Pico », this gives an example on how to do it.

5) « Re-initialize cyw43 »: This option must be used if you want to use option 1 (scan Wi-Fi frequencies) after using other options do that you have a « freshly reset CYW43 ».

6) « Ping a specific IP address »: Will ping the specified IP address until you press a key to restart the application. CYW43 must have been initialized and you must be logged on the local wi-fi network for the ping procedure to work.

7) « Restart the firmware »: Simply reboot the Pico and restart the application (menu choice number 88).

8) « Switch Pico in upload mode »: Allows the user to upload a new firmware to the Pico without having to make a power cycle while holding down the « BootSel » Pico button (menu choice number 99).