Enable ICMP PING

**Table of Contents**

[1. Getting started 2](#__RefHeading___Toc22995_4059769644)

[Update submodules 2](#__RefHeading___Toc22997_4059769644)

[Get the toolchain 2](#__RefHeading___Toc22999_4059769644)

[Build the firmware 2](#__RefHeading___Toc23001_4059769644)

[Generate a new upload file 2](#__RefHeading___Toc23003_4059769644)

[2. UNOR4USBBridge changes 2](#__RefHeading___Toc28494_4059769644)

[3. WiFiS3 changes 3](#__RefHeading___Toc1357_3203275137)

[Appendix A: Add to UNOR4USBBridge/cmds\_esp\_generic.h 4](#__RefHeading___Toc2481_3203275137)

[Appendix B: examples/WiFiPing.ino 5](#__RefHeading___Toc1363_3203275137)

# 1. Getting started

Download the latest version code : https://github.com/arduino/uno-r4-wifi-usb-bridge

## Update submodules

git submodule update --init --depth 1 --no-single-branch

## Get the toolchain

cd hardware/esp32-patched/esp32/tools

./get.py

## Build the firmware

./compile.sh

## Generate a new upload file

./export.sh

Can be found as UNOR4USBBridge/build/esp32-patched.esp32.arduino\_unor4wifi\_usb\_bridge/S3-ALL.bin. You can now rename that if you want.

# 2. UNOR4USBBridge changes

**UNOR4USBBridge/cmds\_esp\_generic.h**

1. in top of file add just below #include "at\_handler.h") :

#include "ESPping.h"

char rsl[5]; // ping time

2. Add appendix A to the bottom in this file, just above the latest ‘}’

**UNOR4USBBridge/commands.h**

add:

#define \_PINGNAME "+PINGNAME"

**in folder uno-r4-wifi-usb-bridge/libraries**

Perform action to add submodule :

git submodule add https://github.com/dvarrel/ESPping.git

***generate a new USB-WIFI-Bridge software and flash to ESPS3***

# 3. WiFiS3 changes

**Renesas\_uno/1.1.0/libraries/WiFiS3/src/WiFi.cpp**

add just above the last line: **CWifi WiFi;**

/\* -------------------------------------------------------------------------- \*/

float CWifi::ping(IPAddress ip, unsigned int count) {

/\* -------------------------------------------------------------------------- \*/

return ping(ip.toString().c\_str(), count);

}

/\* -------------------------------------------------------------------------- \*/

float CWifi::ping(const char\* host, unsigned int count) {

/\* -------------------------------------------------------------------------- \*/

modem.begin();

string res = "";

if (modem.write(string(PROMPT(\_PINGNAME)),res, "%s,%s,%d\r\n", CMD\_WRITE(\_PINGNAME), host, count)) {

String rsl = res.c\_str();

return rsl.toFloat();

}

return 0;

}

**Renesas\_uno/1.1.0/libraries/WiFiS3/src/WiFi.h**

add in public section

/\*

\* PING

\*/

float ping(IPAddress ip, unsigned int count = 5);

float ping(const char\* host, unsigned int count = 5);

**Renesas\_uno/1.1.0/libraries/WiFiS3/src/WiFiCommands.h**

add:

#define \_PINGNAME "+PINGNAME"

CAREFULL IN THE MAIN BRANCH THIS FILE IS A SYMBOLIC LINK !!!

In the final version this is a normal file.

**Renesas\_uno/1.1.0/libraries/WiFiS3/examples**

add WifiPing.ino example from Appendix B.

# Appendix A: Add to UNOR4USBBridge/cmds\_esp\_generic.h

/\* ....................................................................... \*/

command\_table[\_PINGNAME] = [this](auto & srv, auto & parser) {

/\* ....................................................................... \*/

switch (parser.cmd\_mode) {

case chAT::CommandMode::Write: {

if (parser.args.size() != 3) {

return chAT::CommandStatus::ERROR;

}

// get host name

auto &host = parser.args[1];

if (host.empty()) {

return chAT::CommandStatus::ERROR;

}

// get count

auto &cnt = parser.args[2];

if (cnt.empty()) {

return chAT::CommandStatus::ERROR;

}

unsigned int count = atoi(cnt.c\_str());

auto res = Ping.ping((const char\* ) host.c\_str(), count);

if (res) { // ping was succesfull

srv.write\_response\_prompt();

sprintf(rsl,"%.2f", Ping.averageTime());

srv.write\_cstr((const char \*) rsl);

srv.write\_line\_end();

return chAT::CommandStatus::OK;

}

srv.write\_response\_prompt();

srv.write\_error();

srv.write\_line\_end();

return chAT::CommandStatus::ERROR;

}

default:

return chAT::CommandStatus::ERROR;

}

};

# Appendix B: examples/WiFiPing.ino

/\*

Web ICMP Ping

This sketch pings a device based on the IP address or the hostname

using the WiFi module. By default the attempt is performed 5 times, but can

be changed to max. 255

It requires the latest USB Wifi bridge firmware level and WiFiS3 library.

This example is written for a network using WPA encryption. For

WEP or WPA, change the WiFi.begin() call accordingly.

created 14 February 2024

by paulvha

\*/

#include "WiFiS3.h"

#include "arduino\_secrets.h"

///////please enter your sensitive data in the Secret tab/arduino\_secrets.h

char ssid[] = SECRET\_SSID; // your network SSID (name)

char pass[] = SECRET\_PASS; // your network password (use for WPA, or use as key for WEP)

int keyIndex = 0; // your network key index number (needed only for WEP)

int status = WL\_IDLE\_STATUS;

/\* -------------------------------------------------------------------------- \*/

void setup() {

/\* -------------------------------------------------------------------------- \*/

//Initialize serial and wait for port to open:

Serial.begin(115200);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

// check for the WiFi module:

if (WiFi.status() == WL\_NO\_MODULE) {

Serial.println("Communication with WiFi module failed. freeze !");

// don't continue

while (true);

}

String fv = WiFi.firmwareVersion();

if (fv < WIFI\_FIRMWARE\_LATEST\_VERSION) {

Serial.println("Please upgrade to the WiFi USB bridge firmware. freeze !");

// don't continue

while (true);

}

// attempt to connect to WiFi network:

while (status != WL\_CONNECTED) {

Serial.print("Attempting to connect to SSID: ");

Serial.println(ssid);

// Connect to WPA/WPA2 network. Change this line if using open or WEP network:

status = WiFi.begin(ssid, pass);

// wait 10 seconds for connection:

delay(10000);

}

printWifiStatus();

}

/\* -------------------------------------------------------------------------- \*/

void loop() {

/\* -------------------------------------------------------------------------- \*/

// Ping IP

const IPAddress remote\_ip(140,82,121,4);

Serial.print("Trying to ping github.com on IP: ");

Serial.println(remote\_ip);

// using default ping count of 5

float res = WiFi.ping(remote\_ip);

if (res != 0) {

Serial.print("Pin average response time: ");

Serial.print(res);

Serial.println(" ms");

}

else {

Serial.println("Timeout on IP!");

}

// Ping Host

const char\* remote\_host = "www.google.com";

Serial.print("Trying to ping host: ");

Serial.println(remote\_host);

// setting ping count to 10 instead of default 5

float res1 = WiFi.ping(remote\_host,10);

if (res1 != 0) {

Serial.print("Pin average response time: ");

Serial.print(res1);

Serial.println(" ms");

}

else {

Serial.println("Timeout on host!");

}

Serial.println();

delay(1000);

}

/\* -------------------------------------------------------------------------- \*/

void printWifiStatus() {

/\* -------------------------------------------------------------------------- \*/

// print the SSID of the network you're attached to:

Serial.print("SSID: ");

Serial.println(WiFi.SSID());

// print your board's IP address:

IPAddress ip = WiFi.localIP();

Serial.print("IP Address: ");

Serial.println(ip);

// print the received signal strength:

long rssi = WiFi.RSSI();

Serial.print("signal strength (RSSI):");

Serial.print(rssi);

Serial.println(" dBm");

}

**arduino\_secrets.h (*tab in sketch)***

#define SECRET\_SSID ""

#define SECRET\_PASS ""