



**DesignNews**

# Scratch Building Raspberry Pi RP2040 IoT Devices

**Day 4:**

Raspberry Pi Pico W Primer

Sponsored by



# Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

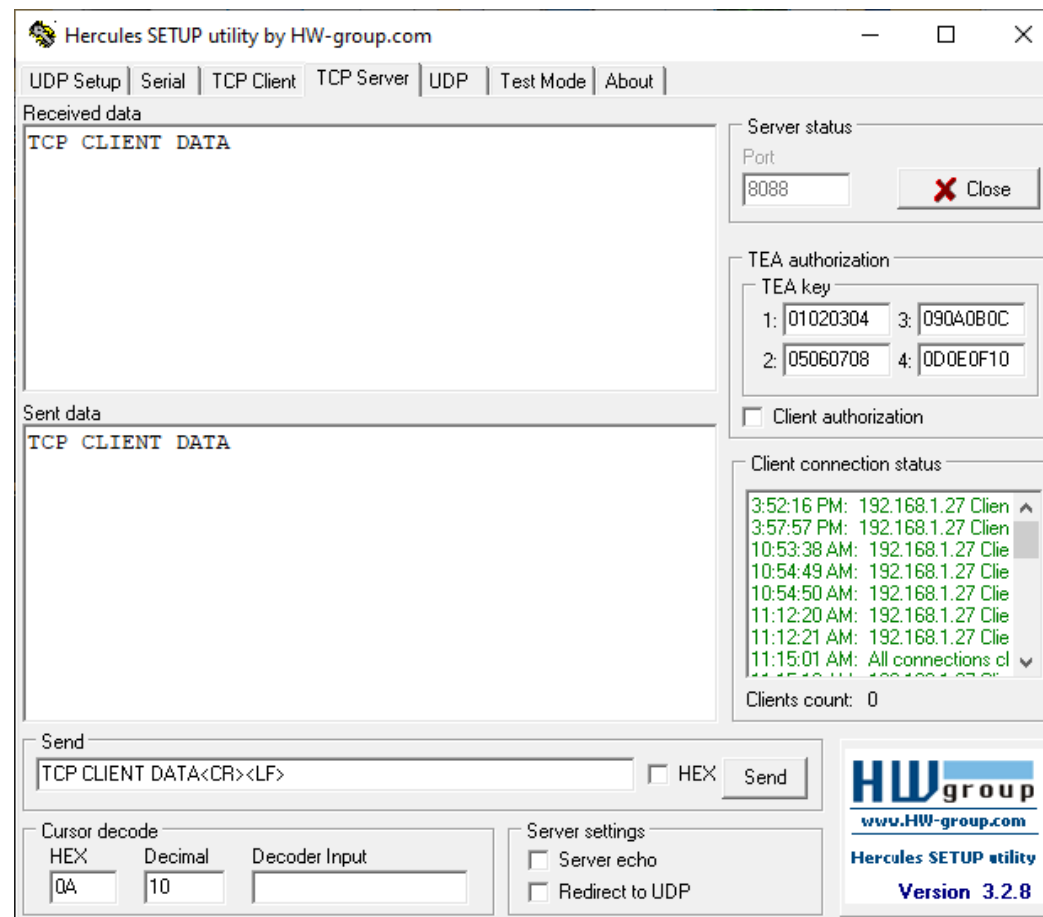


## Fred Eady

Visit 'Lecturer Profile' in your console for more details.

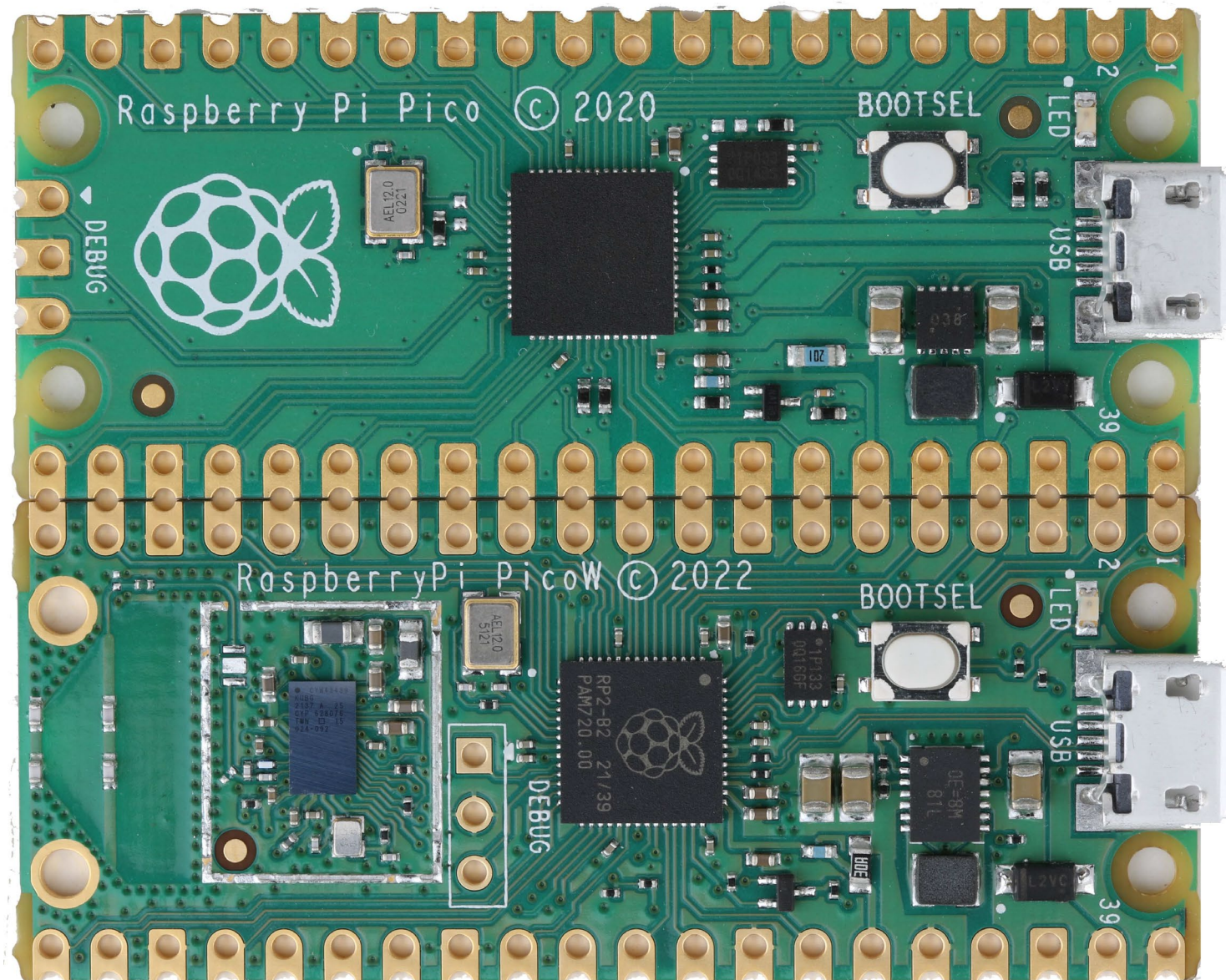
# AGENDA

- **Pico versus Pico W**
- **Connecting a Pico W**
- **Coding a Pico W TCP Client**



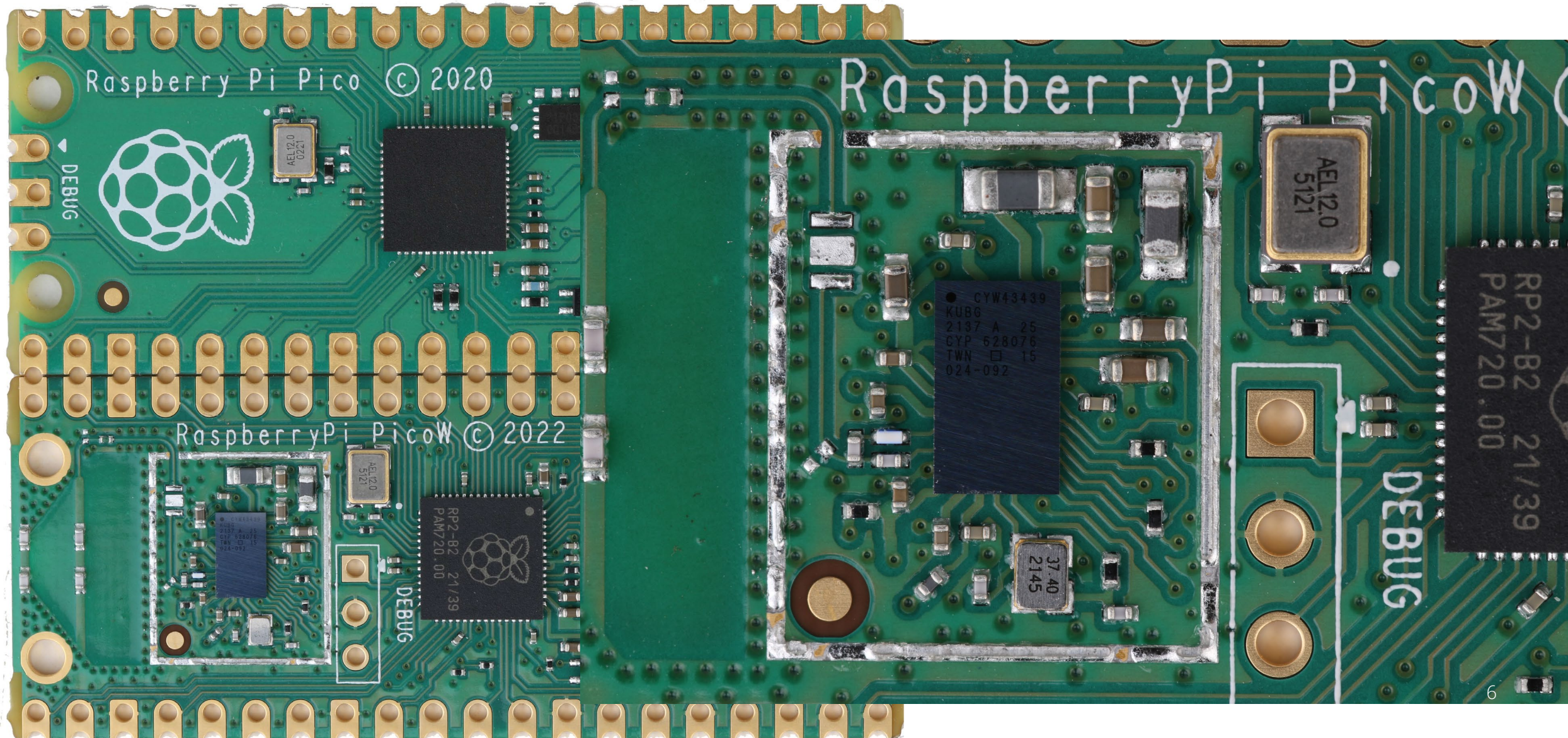


## Pair of Picos



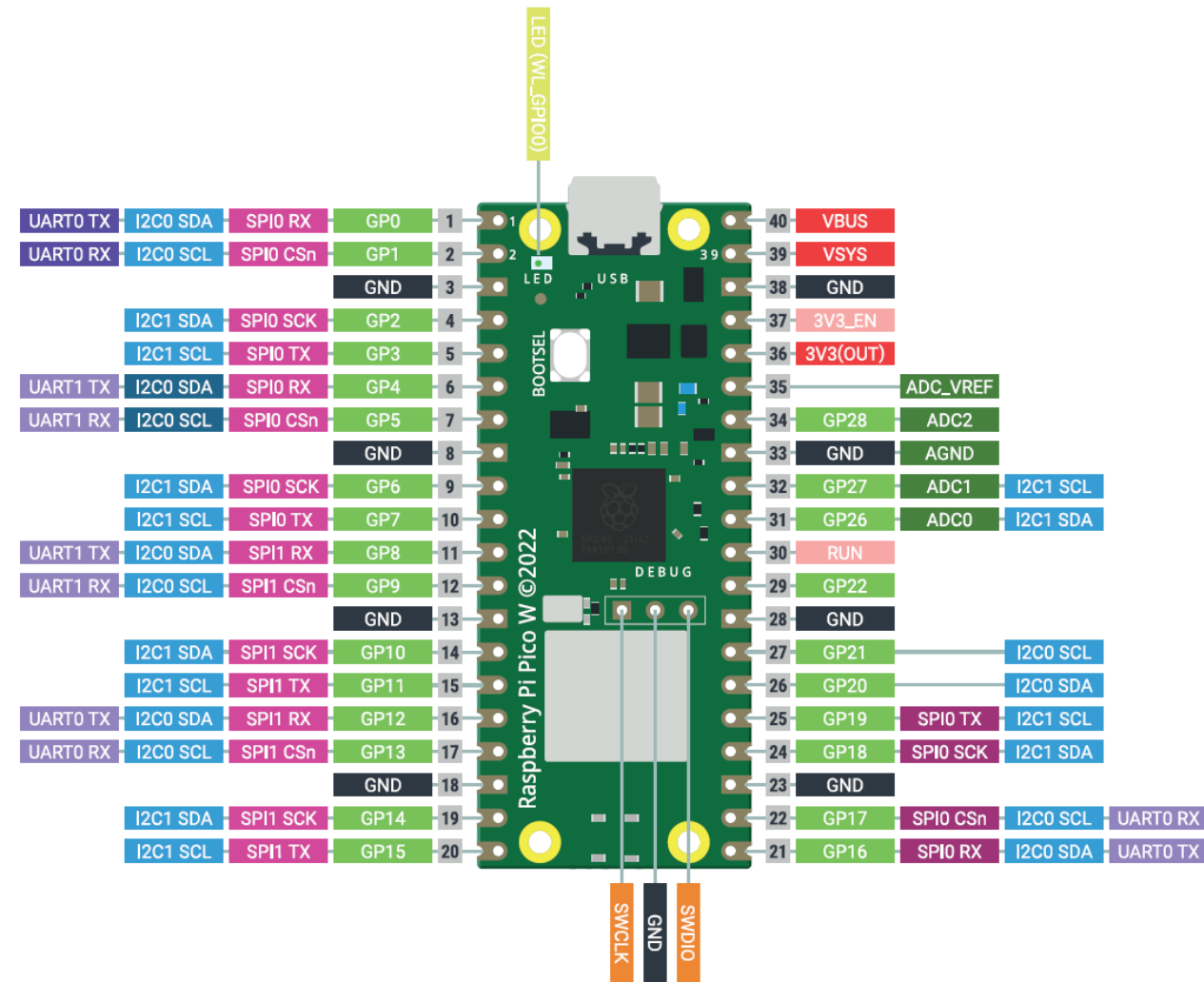
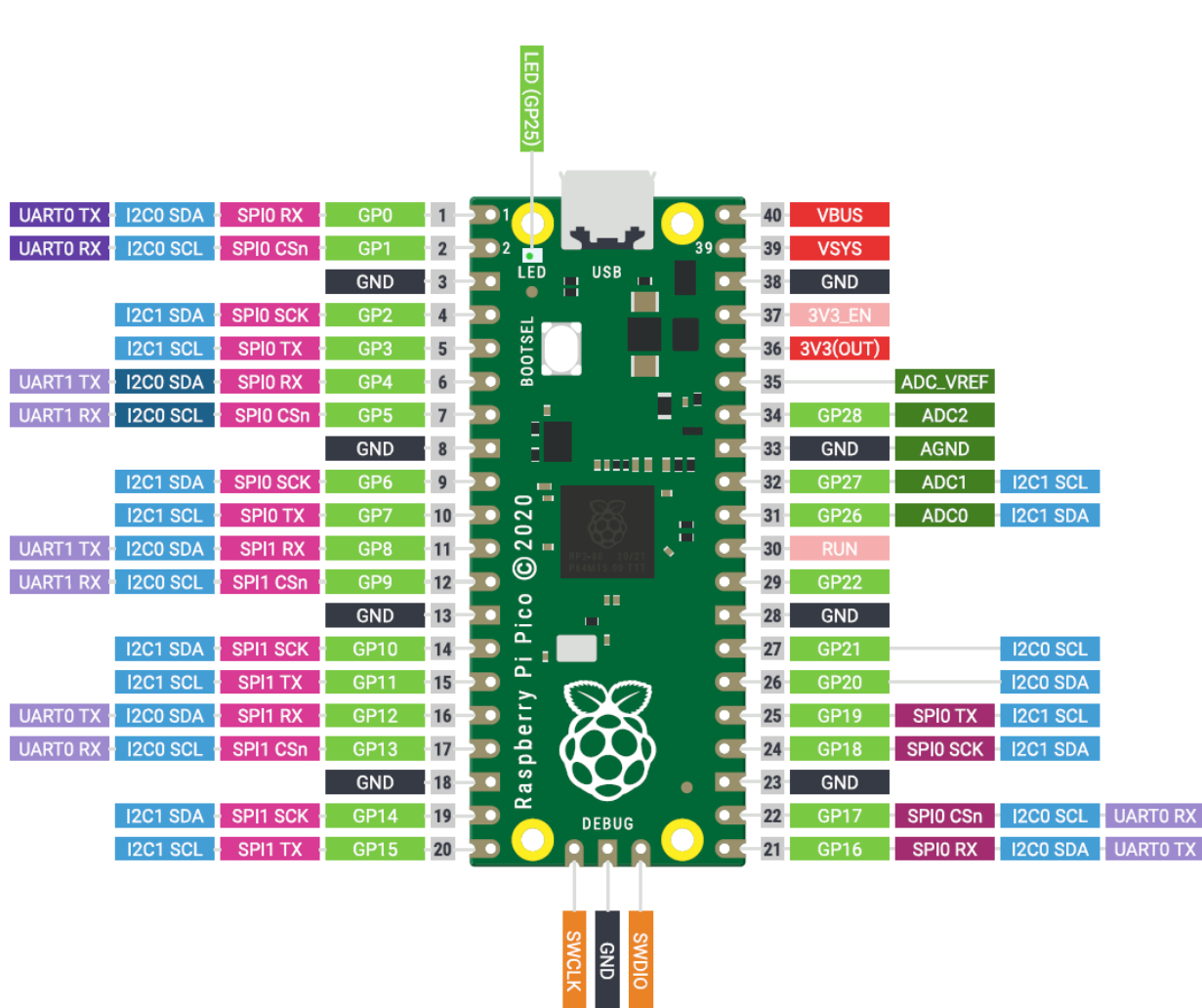


## Pair of Picos





## Pair of Picos



## Visual Studio Code Configuration

Settings

@ext:ms-vscode.cmake-tools

UserWorkspace

Extensions (56)  
CMake Tools (56)

☒ Build the target before running it.

**Cmake: Build Directory**  
The directory where CMake build files will go.  

{workspaceFolder}/build

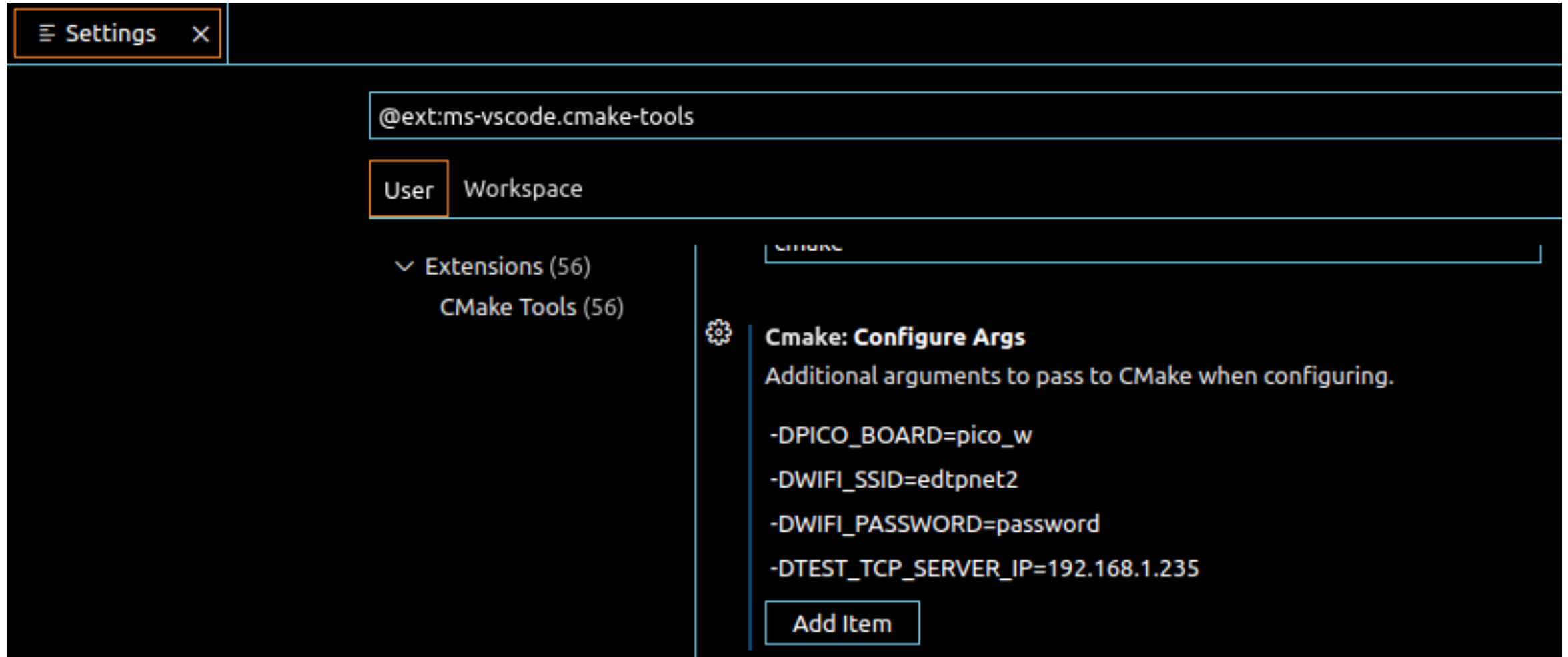
**Cmake: Build Environment**  
Environment variables to pass to CMake during build.

Item	Value
PICO_SDK_PATH	/home/fred/pico/pico-sdk

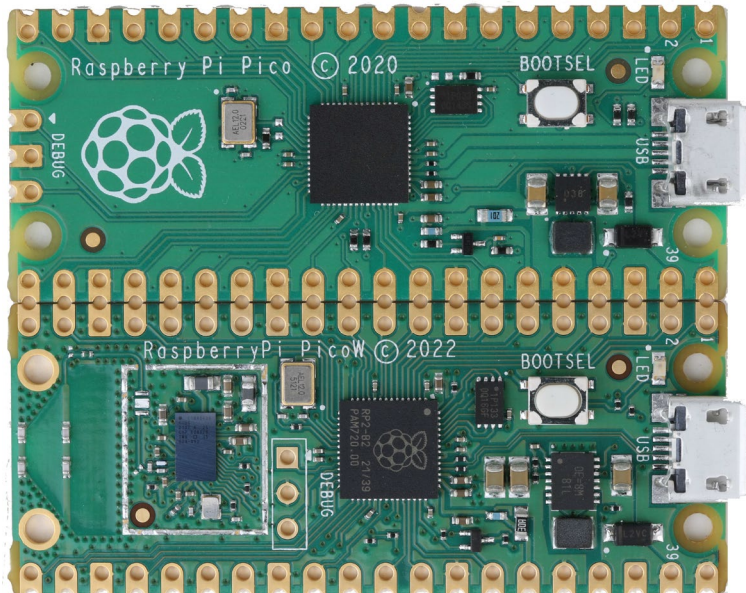
Add Item



## Visual Studio Code Configuration



## CMakeLists.txt



## CMakeLists.txt

```

1  # Set minimum required version of CMake
2  cmake_minimum_required(VERSION 3.12)
3
4  # Include build functions from Pico SDK
5  include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)
6
7  # Set name of project (as PROJECT_NAME) and C/C++ standards
8  project(connect2edtpnet C CXX ASM)
9  set(CMAKE_C_STANDARD 11)
10 set(CMAKE_CXX_STANDARD 17)
11
12 # Creates a pico-sdk subdirectory in our project for the libraries
13 pico_sdk_init()
14
15 # Tell CMake where to find the executable source file
16 add_executable(${PROJECT_NAME}
17     main.c
18 )
19 target_compile_definitions(${PROJECT_NAME} PRIVATE
20     WIFI_SSID="\${WIFI_SSID}\\"
21     WIFI_PASSWORD="\${WIFI_PASSWORD}\\"
22     TEST_TCP_SERVER_IP="\${TEST_TCP_SERVER_IP}\\"
23 )
24 target_include_directories(${PROJECT_NAME} PRIVATE
25     ${CMAKE_CURRENT_LIST_DIR}
26     ${CMAKE_CURRENT_LIST_DIR}/.. # for our common lwipopts
27 )
28
29 # Link to pico_stdlib (gpio, time, etc. functions)
30 target_link_libraries(${PROJECT_NAME}
31     pico_cyw43_arch_lwip_threadsafe_background
32     pico_stdlib
33 )
34
35 # Create map/bin/hex/uf2 files
36 pico_add_extra_outputs(${PROJECT_NAME})
37 # Enable usb output, disable uart output
38 pico_enable_stdio_usb(${PROJECT_NAME} 0)
39 pico_enable_stdio_uart(${PROJECT_NAME} 1)

```



## Cmake: Configure Args

Additional arguments to pass to CMake when configuring.

```

-DPICO_BOARD=pico_w
-DWIFI_SSID=edtpnet2
-DWIFI_PASSWORD=password
-DTEST_TCP_SERVER_IP=192.168.1.235

```



**main.c**

```
C main.c
1  #include <stdio.h>
2  #include "pico/stdlib.h"
3  #include "pico/cyw43_arch.h"
4
5  int main()
6  {
7      stdio_init_all();
8
9      if(cyw43_arch_init_with_country(CYW43_COUNTRY_USA))
10     {
11         printf("CYW43 INIT FAILED!!\r\n");
12         return 1;
13     }
14     printf("CYW43 INITIALIZED!!\r\n");
15
16     cyw43_arch_enable_sta_mode();
17
18     if(cyw43_arch_wifi_connect_timeout_ms(WIFI_SSID, WIFI_PASSWORD, CYW43_AUTH_WPA2_AES_PSK, 10000))
19     {
20         printf("CONNECT FAILED!!\r\n");
21         return 1;
22     }
23     printf("CYW43 CONNECTED!!\r\n");
24 }
```

Load – Run - Connect

<

>

Home

wconnect

build

Recent

Starred

Home

Desktop

Documents

Downloads

Music

Pictures

Videos

Trash

Other Locations

Name

CMakeFiles

elf2uf2

generated

pico-sdk

pioasm

CMakeCache.txt

cmake\_install.cmake

compile\_commands.json

connect2edtpnet.bin

connect2edtpnet.dis

connect2edtpnet.elf

connect2edtpnet.elf.map

connect2edtpnet.hex

connect2edtpnet.uf2

Makefile

File

Edit

Target

Options

View

Help

Project information

Setting

Value

General

Project name

Host connection

TIF

Type

Init. speed

Speed

Target

MCU

Core

Endian

Check core ID

Use target RAM

Flashbank No. 0

SEgger J-Flash V7.68b - [\*]

/home/fred/wconnect/build/connect2edtpnet.hex @ 10000000

Go To:

1

2

4

1000\_0000

00 B5 32 4B 21 20 58 60

98 68 02 21 88 43 98 60

μ2K!..X`.h.!.C.`

1000\_0010

08 60 18 61 58 61 2E 4B

00 21 99 60 02 21 59 61

0`..aXa.K.!.!.!Ya

1000\_0020

01 21 F0 22 99 50 2B 49

19 60 01 21 99 60 35 20

!ð".P+I`.!.!`5.

1000\_0030

00 F0 44 F8 02 22 90 42

14 D0 06 21 19 66 00 F0

..ððø."..B.ð.!.f.ð

1000\_0040

34 F8 19 6E 01 21 19 66

00 20 18 66 1A 66 00 F0

4ø.n.!.f...f.f.ð

1000\_0050

2C F8 19 6E 19 6E 19 6E

05 20 00 F0 2F F8 01 21

,ø.n.n.n...ð/ø.!

1000\_0060

08 42 F9 D1 00 21 99 60

1B 49 19 60 00 21 59 60

.BùN.!.!.I..!Y`

1000\_0070

1A 49 1B 48 01 60 01 21

99 60 EB 21 19 66 A0 21

.I.H.`.!.!`è!f.!

1000\_0080

19 66 00 F0 12 F8 00 21

99 60 16 49 14 48 01 60

.f.ð.ø.!.!.I.H.`

1000\_0090

01 21 99 60 01 BC 00 28

00 D0 00 47 12 48 13 49

!.`.¿.(!.ð.G.H.I

1000\_00A0

08 60 03 C8 80 F3 08 88

08 47 03 B5 99 6A 04 20

..È.ó...G.μ.j..

1000\_00B0

01 42 FB D0 01 20 01 42

F8 D1 03 BD 02 B5 18 66

.B0ð...BøN.¿.μ.f

1000\_00C0

18 66 FF F7 F2 FF 18 6E

18 6E 02 BD 00 00 02 40

.fÿ+öÿ.n.n.¿...@

1000\_00D0

00 00 00 18 00 00 07 00

00 03 5F 00 21 22 00 00

.....!"..

1000\_00E0

F4 00 00 18 22 20 00 A0

00 01 00 10 08 ED 00 E0

ò....".....í.à

1000\_00F0

00 00 00 00 00 00 00 00

00 00 00 00 74 B2 4E 7A

.....t²Nz

1000\_0100

00 20 04 20 F7 01 00 10

C3 01 00 10 C5 01 00 10

....÷...Ä...Ä...

1000\_0110

C1 01 00 10 C1 01 00 10

C1 01 00 10 C1 01 00 10

Ä...Ä...Ä...Ä...

1000\_0120

C1 01 00 10 C1 01 00 10

C1 01 00 10 C7 01 00 10

Ä...Ä...Ä...Ç...

1000\_0130

C1 01 00 10 C1 01 00 10

C9 01 00 10 CB 01 00 10

Ä...Ä...È...È...

1000\_0140

CD 01 00 10 CD 01 00 10

CD 01 00 10 CD 01 00 10

Í...Í...Í...Í...

1000\_0150

CD 01 00 10 CD 01 00 10

CD 01 00 10 CD 01 00 10

Í...Í...Í...Í...

1000\_0160

CD 01 00 10 CD 01 00 10

CD 01 00 10 CD 01 00 10

Í...Í...Í...Í...

1000\_0170

CD 01 00 10 CD 01 00 10

CD 01 00 10 CD 01 00 10

Í...Í...Í...Í...

1000\_0180

CD 01 00 10 CD 01 00 10

CD 01 00 10 CD 01 00 10

Í...Í...Í...Í...

Log

- ROMID[0] @ E00FF000

- [0][0]: E000E000 CID B105E00D PID 000BB008 SCS

- [0][1]: E0001000 CID B105E00D PID 000BB00A DWT

- [0][2]: E0002000 CID B105E00D PID 000BB00B FPB

- Reset: Halt core after reset via DEMCR.VC\_CORERESET.

- Reset: Reset device via AIRCR.SYSRESETREQ.

- AfterResetTarget() start

- AfterResetTarget() end

- Target application started

Ready

Not connected

12

"connect2edtpnet.hex" selected (857.0 kB)



Connect Flow

Serial Input/Output Monitor

File Edit View Configuration

ASCII HEX Line Status Clear Terminal Columns Display Data Graph Ribbon Classic  
ASCII Send HEX Send Input/Output Viewing Options Tools Menu Style

```
CYW43 INITIALIZED!!  
Version: 7.95.49 (2271bb6 CY) CRC: b7a28ef3 Date: Mon 2021-11-29 22:50:27 PST Ucode Ver: 1  
043.2162 FWID 01-c51d9400  
cyw43 loaded ok, mac 28:cd:c1:00:ae:f3  
API: 12.2  
Data: RaspberryPi.PicoW  
Compiler: 1.29.4  
ClimImport: 1.47.1  
Customization: v5 22/06/24  
Creation: 2022-06-24 06:55:08  
connect status: joining  
connect status: no ip  
connect status: link up  
CYW43 CONNECTED!!
```

63 6F 6E 6E 65 63 74 20 73 74 61 74 75 73 3A 20 6C 69 6E 6B 20 75 70 0D 0A  
43 59 57 34 33 20 43 4F 4E 4E 45 43 54 45 44 21 21 0D 0A

ASCII  Send  
HEX  Send  
COM6 8N1 115200 R 0 C 0 R16 C 1 Disconnect

## CMakeLists.txt

**M** CMakeLists.txt

```
1  # Set minimum required version of CMake
2  cmake_minimum_required(VERSION 3.12)
3
4  # Include build functions from Pico SDK
5  include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)
6
7  # Set name of project (as PROJECT_NAME) and C/C++ standards
8  project(ptcpclient C CXX ASM)
9  set(CMAKE_C_STANDARD 11)
10 set(CMAKE_CXX_STANDARD 17)
11
12 # Creates a pico-sdk subdirectory in our project for the libraries
13 pico_sdk_init()
14
15 if (NOT TEST_TCP_SERVER_IP)
16 |   message("Skipping tcp_client example as TEST_TCP_SERVER_IP is not defined")
17 else()
18 |   add_executable(${PROJECT_NAME}
19 |       picow_tcp_client.c
20 |   )
21 |   target_compile_definitions(${PROJECT_NAME} PRIVATE
22 |       WIFI_SSID=\"${WIFI_SSID}\"
23 |       WIFI_PASSWORD=\"${WIFI_PASSWORD}\"
24 |       TEST_TCP_SERVER_IP=\"${TEST_TCP_SERVER_IP}\"
25 |   )
26 |   target_include_directories(${PROJECT_NAME} PRIVATE
27 |       ${CMAKE_CURRENT_LIST_DIR}
28 |       ${CMAKE_CURRENT_LIST_DIR}/.. # for our common lwipopts
29 |   )
30 |   target_link_libraries(${PROJECT_NAME}
31 |       pico_cyw43_arch_lwip_poll
32 |       pico_stdlib
33 |   )
34 |   pico_add_extra_outputs(${PROJECT_NAME})
35 endif()
```



## DUMP\_BYTES Helper Function

```
19  #if 1
20  static void dump_bytes(const uint8_t *bptr, uint32_t len) {
21      unsigned int i = 0;
22
23      printf("dump_bytes %d", len);
24      for (i = 0; i < len;) {
25          if ((i & 0x0f) == 0) {
26              printf("\n");
27          } else if ((i & 0x07) == 0) {
28              printf(" ");
29          }
30          printf("%02x ", bptr[i++]);
31      }
32      printf("\n");
33  }
34  #define DUMP_BYTES dump_bytes
35  #else
36  #define DUMP_BYTES(A,B)
37  #endif
```

## picow\_tcp\_client.c

```
C picow_tcp_client.c
1
2
3
4  #include "pico/stdlib.h"
5  #include "pico/cyw43_arch.h"
6
7  #include "lwip/pbuf.h"
8  #include "lwip/tcp.h"
9
10 #if !defined(TEST_TCP_SERVER_IP)
11 #error TEST_TCP_SERVER_IP not defined
12 #endif
13
14 #define TCP_PORT 8088
15 #define DEBUG_printf printf
16 #define BUF_SIZE 17
17 #define POLL_TIME_S 5
18
19 typedef struct TCP_CLIENT_T {
20     struct tcp_pcb *tcp_pcb;
21     ip_addr_t remote_addr;
22     uint8_t buffer[BUF_SIZE];
23     int buffer_len;
24     int sent_len;
25     bool complete;
26     bool connected;
27 } TCP_CLIENT_T;
```



## picow\_tcp\_client.c – Connect Flow

```
C picow_tcp_client.c > main()
227
228
229 int main()
230 {
231     stdio_init_all();
232
233     if (cyw43_arch_init())
234     {
235         DEBUG_printf("failed to initialise\n");
236         return 1;
237     }
238     cyw43_arch_enable_sta_mode();
239
240     printf("Connecting to WiFi...\n");
241     if (cyw43_arch_wifi_connect_timeout_ms(WIFI_SSID, WIFI_PASSWORD, CYW43_AUTH_WPA2_AES_PSK, 30000))
242     {
243         printf("failed to connect.\n");
244         return 1;
245     } else
246     {
247         printf("Connected.\n");
248     }
```

## picow\_tcp\_client.c – TCP Client Flow

```
C picow_tcp_client.c > ...
```

```
209
210     TCP_CLIENT_T *state = tcp_client_init();
211     if (!state)
212     {
213         printf("state not created.\n");
214         while(1);
215     }
216     if (!tcp_client_open(state))
217     {
218         printf("client failed to open.\n");
219         while(1);
220     }
221     while(!state->complete)
222     {
223         cyw43_arch_poll();
224         sleep_ms(1);
225     }
226     cyw43_arch_deinit();
227     return 0;
228 }
```



## picow\_tcp\_client.c – Initialize and Open Client

C picow\_tcp\_client.c > ...

```
160
161 static TCP_CLIENT_T* tcp_client_init(void) {
162     TCP_CLIENT_T *state = calloc(1, sizeof(TCP_CLIENT_T));
163     if (!state) {
164         DEBUG_printf("failed to allocate state\n");
165         return NULL;
166     }
167     ip4addr_aton(TEST_TCP_SERVER_IP, &state->remote_addr);
168     return state;
169 }
170
171 static bool tcp_client_open(void *arg) {
172     TCP_CLIENT_T *state = (TCP_CLIENT_T*)arg;
173     DEBUG_printf("Connecting to %s port %u\n", ip4addr_ntoa(&state->remote_addr), TCP_PORT);
174     state->tcp_pcb = tcp_new_ip_type(IP_GET_TYPE(&state->remote_addr));
175     if (!state->tcp_pcb) {
176         DEBUG_printf("failed to create pcb\n");
177         return false;
178     }
179
180     tcp_arg(state->tcp_pcb, state);
181     tcp_poll(state->tcp_pcb, tcp_client_poll, POLL_TIME_S * 2);
182     tcp_sent(state->tcp_pcb, tcp_client_sent);
183     tcp_recv(state->tcp_pcb, tcp_client_recv);
184     tcp_err(state->tcp_pcb, tcp_client_err);
185
186     state->buffer_len = 0;
187     cyw43_arch_lwip_begin();
188     err_t err = tcp_connect(state->tcp_pcb, &state->remote_addr, TCP_PORT, tcp_client_connected);
189     cyw43_arch_lwip_end();
190
191     return err == ERR_OK;
192 }
```

## picow\_tcp\_client.c – Client TCP Receive Callback

```
C picow_tcp_client.c > ...
125
126 err_t tcp_client_recv(void *arg, struct tcp_pcb *tpcb, struct pbuf *p, err_t err) {
127     TCP_CLIENT_T *state = (TCP_CLIENT_T*)arg;
128     if (!p) {
129         printf("receive buffer error.\n");
130         while(1);
131     }
132     // this method is callback from lwIP, so cyw43_arch_lwip_begin is not required, however you
133     // can use this method to cause an assertion in debug mode, if this method is called when
134     // cyw43_arch_lwip_begin IS needed
135     cyw43_arch_lwip_check();
136     if (p->tot_len > 0) {
137         DEBUG_printf("recv %d err %d\n", p->tot_len, err);
138         for (struct pbuf *q = p; q != NULL; q = q->next) {
139             DUMP_BYTES(q->payload, q->len);
140         }
141         // Receive the buffer
142         const uint16_t buffer_left = BUF_SIZE - state->buffer_len;
143         state->buffer_len += pbuf_copy_partial(p, state->buffer + state->buffer_len,
144         | | | | | | | | | | p->tot_len > buffer_left ? buffer_left : p->tot_len, 0);
145         tcp_recved(tpcb, p->tot_len);
146     }
147     pbuf_free(p);
148
149     // If we have received the whole buffer, send it back to the server
150     if (state->buffer_len == BUF_SIZE) {
151         DEBUG_printf("Writing %d bytes to server\n", state->buffer_len);
152         err_t err = tcp_write(tpcb, state->buffer, state->buffer_len, TCP_WRITE_FLAG_COPY);
153         if (err != ERR_OK) {
154             DEBUG_printf("Failed to write data %d\n", err);
155             while(1);
156         }
157     }
158     return ERR_OK;
159 }
```

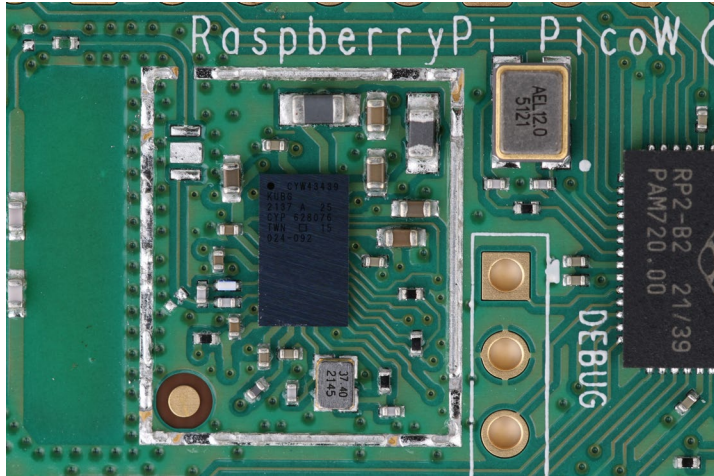


## picow\_tcp\_client.c – Supporting Callbacks

C picow\_tcp\_client.c > ...

```
82
83 static err_t tcp_client_sent(void *arg, struct tcp_pcb *tpcb, u16_t len) {
84     TCP_CLIENT_T *state = (TCP_CLIENT_T*)arg;
85     DEBUG_printf("tcp_client_sent %u\n", len);
86     return ERR_OK;
87 }
88
89 static err_t tcp_client_connected(void *arg, struct tcp_pcb *tpcb, err_t err) {
90     TCP_CLIENT_T *state = (TCP_CLIENT_T*)arg;
91     if (err != ERR_OK) {
92         printf("connect failed %d\n", err);
93         while(1);
94     }
95     state->connected = true;
96     DEBUG_printf("Waiting for buffer from server\n");
97     return ERR_OK;
98 }
99
100 static err_t tcp_client_poll(void *arg, struct tcp_pcb *tpcb) {
101     DEBUG_printf("tcp_client_poll\n");
102     return 0; //tcp_result(arg, 0); // no response is an error?
103 }
104
105 static void tcp_client_err(void *arg, err_t err) {
106     if (err != ERR_ABRT) {
107         DEBUG_printf("tcp_client_err %d\n", err);
108         while(1);
109     }
110 }
```

picow\_tcp\_client.c – Client/Server Data Exchange



Serial Input/Output Monitor

File

Edit

View

Configuration

ASCII

HEX

ASCII Send

HEX Send

Line Status

Clear Terminal

Columns

Input/Output

Viewing Op

```

Version: 7.95.49 (2271bb6 CY) CRC: b7a28ef3 Da
1043.2162 FWID 01-c51d9400
cyw43 loaded ok, mac 28:cd:c1:00:ae:f3
API: 12.2
Data: RaspberryPi.PicoW
Compiler: 1.29.4
ClimImport: 1.47.1
Customization: v5 22/06/24
Creation: 2022-06-24 06:55:08
Connecting to WiFi...
connect status: joining
connect status: no ip
connect status: link up
Connected.
Connecting to 192.168.1.235 port 8088
tcp_client_poll
tcp_client_poll
Waiting for buffer from server
tcp_client_poll
tcp_client_poll
recv 17 err 0
dump_bytes 17
54 43 50 20 43 4c 49 45 4e 54 20 44 41 54 41 0d
0a
Writing 17 bytes to server
tcp_client_sent 17
tcp_client_poll
  
```

Hercules SETUP utility by HW-group.com

UDP Setup

Serial

TCP Client

TCP Server

UDP

Test Mode

About

Received data

TCP CLIENT DATA

Sent data

TCP CLIENT DATA

Server status

Port: 8088

Close

TEA authorization

TEA key

1: 01020304 3: 090A0B0C

2: 05060708 4: 0D0E0F10

Client authorization

Client connection status

2:55:36 PM: 192.168.1.32 Client co

4:06:25 PM: All connections closed

4:08:07 PM: 192.168.1.32 Client co

Clients count: 0

Send

TCP CLIENT DATA<CR><LF>

HEX

Send

Cursor decode

HEX: 0A

Decimal: 10

Decoder Input

Server settings

Server echo

Redirect to UDP

HWgroup

www.HW-group.com

Hercules SETUP utility

Version 3.2.8



**MORE TO COME..**

# Thank you for attending!!!

Please consider the resources below:

- [raspberrypi.org](https://raspberrypi.org)
- **RP2040 Datasheet**
- **Raspberry Pi Pico C/C++ SDK**
- **SEGGER J-Link**
- **SEGGER Ozone Debugger**
- **lwIP API Documentation**





**DesignNews**

# Thank You

Sponsored by

