CEC Continuing Education Center

DesignNews

Scratch Building Raspberry Pi RP2040 IoT Devices

Day 4:

Raspberry Pi Pico W Primer

Sponsored by

Digi-Key ELECTRONICS

informa markets

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

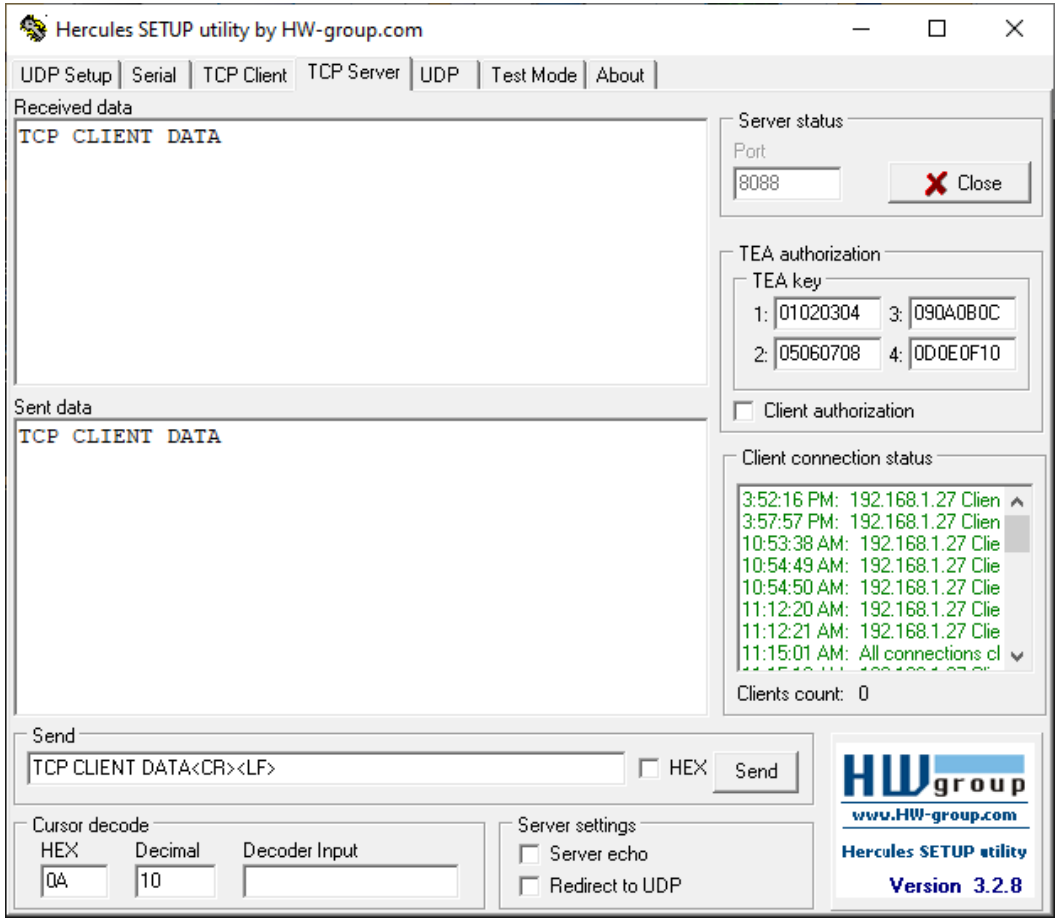
Visit ‘Lecturer Profile’ in your console for more details.

CEC Continuing Education Center

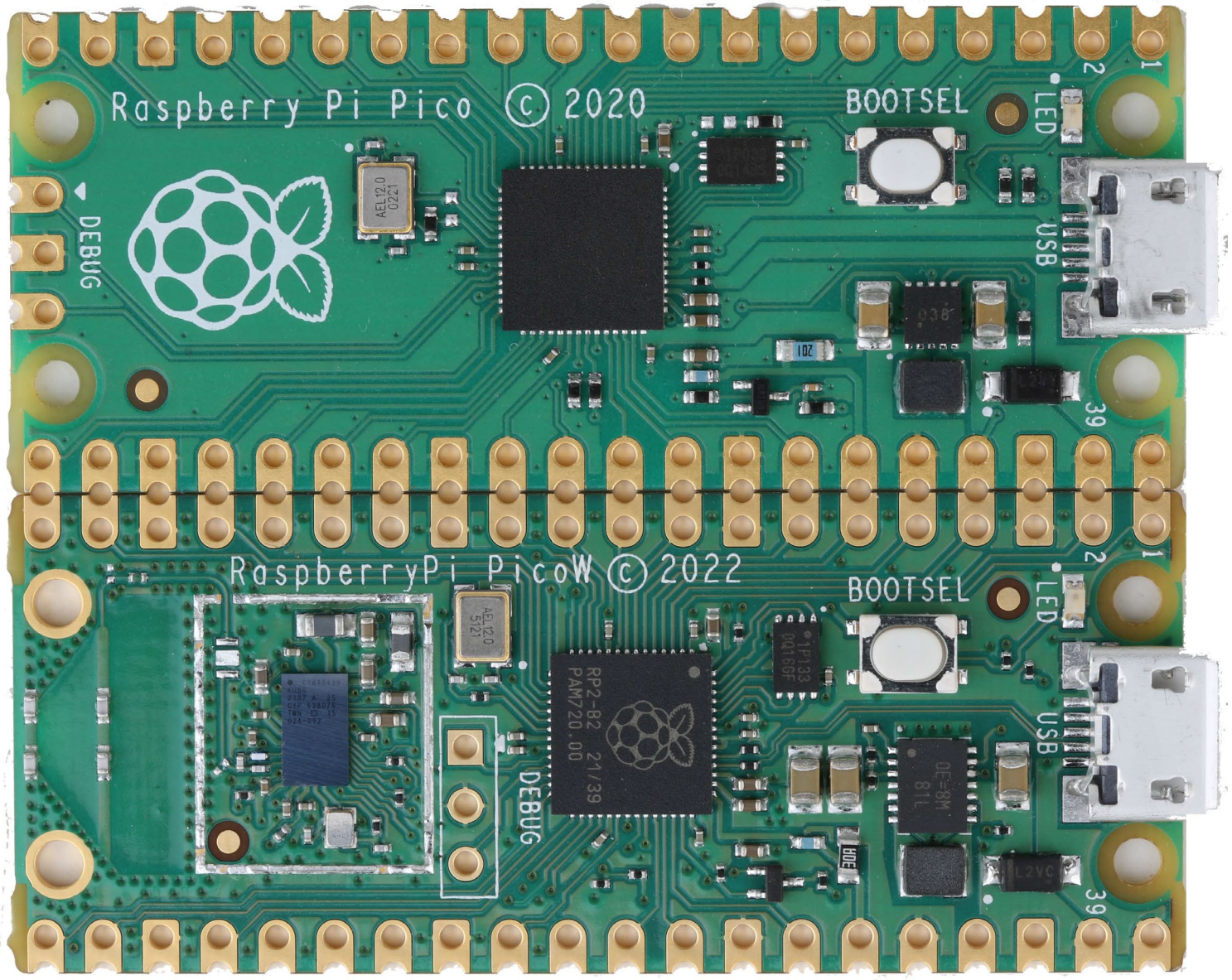
Sponsored By Digi-Key Corporation

AGENDA

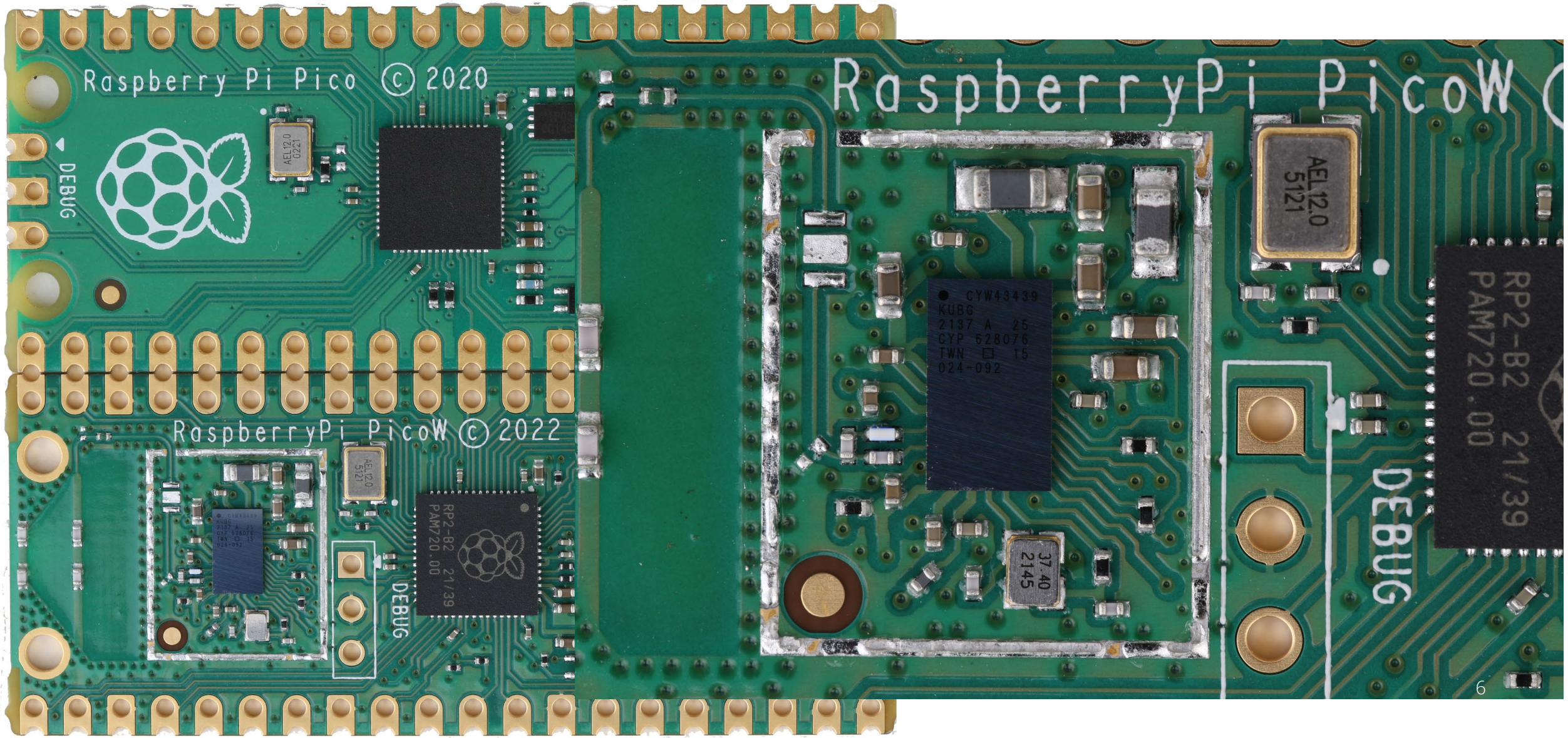
- Pico versus Pico W
- Connecting a Pico W
- Coding a Pico W TCP Client



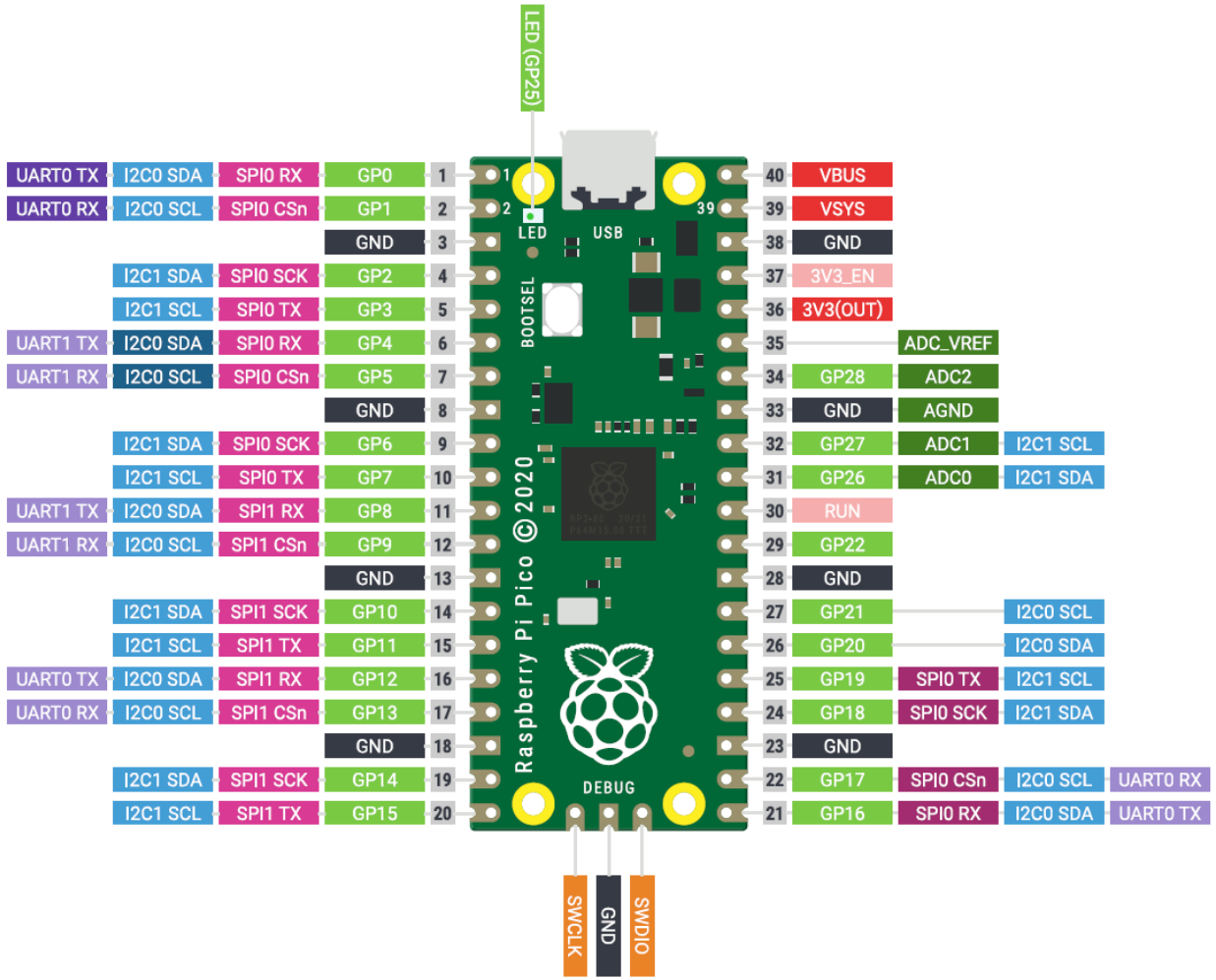
Pair of Picos

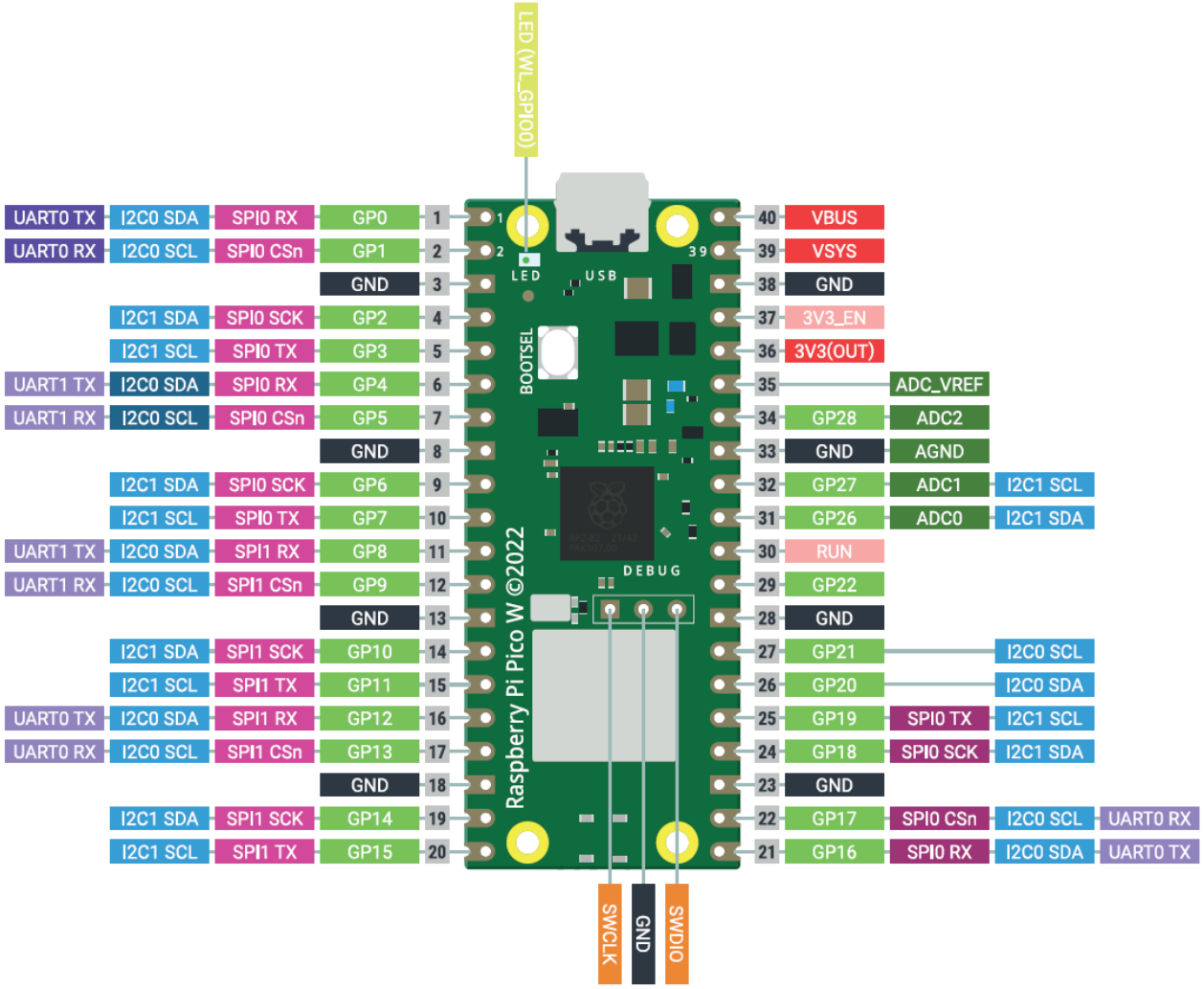


Pair of Picos

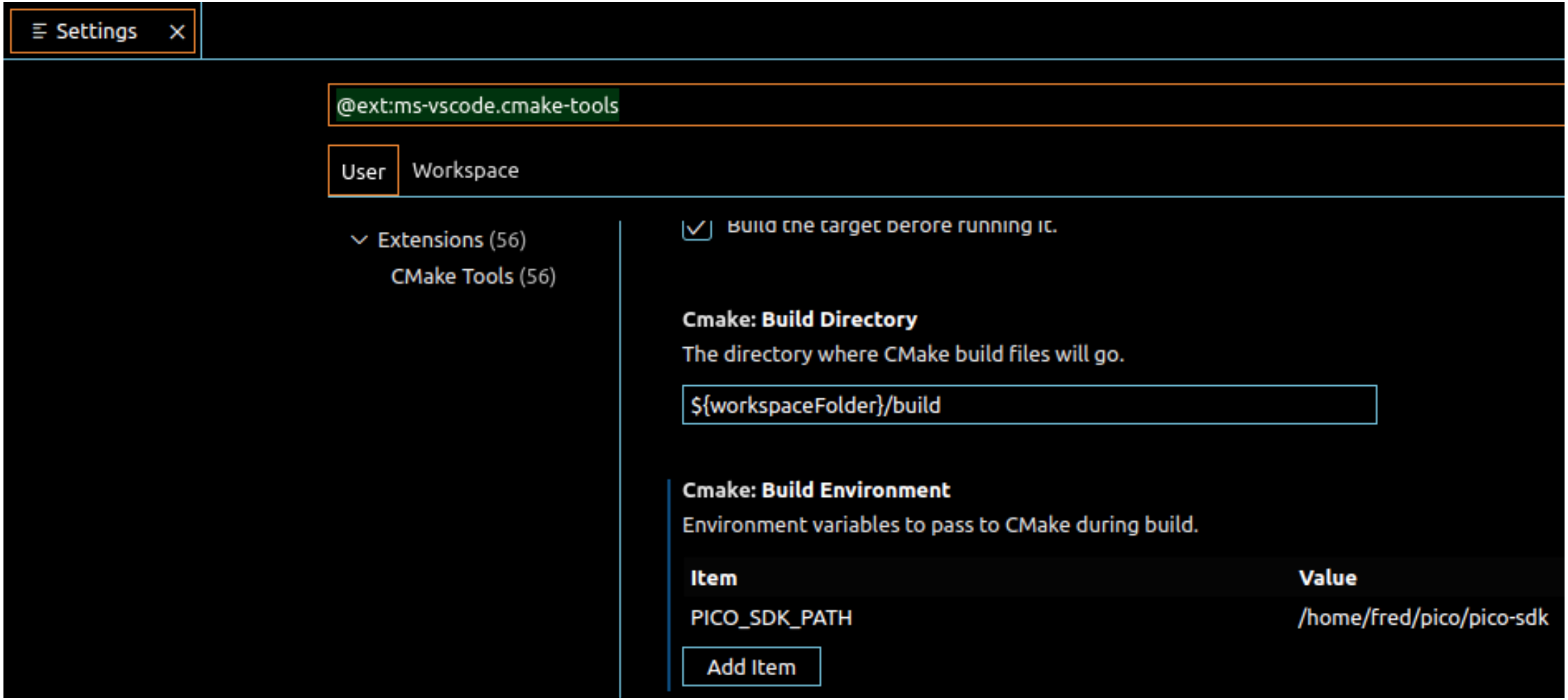


Pair of Picos

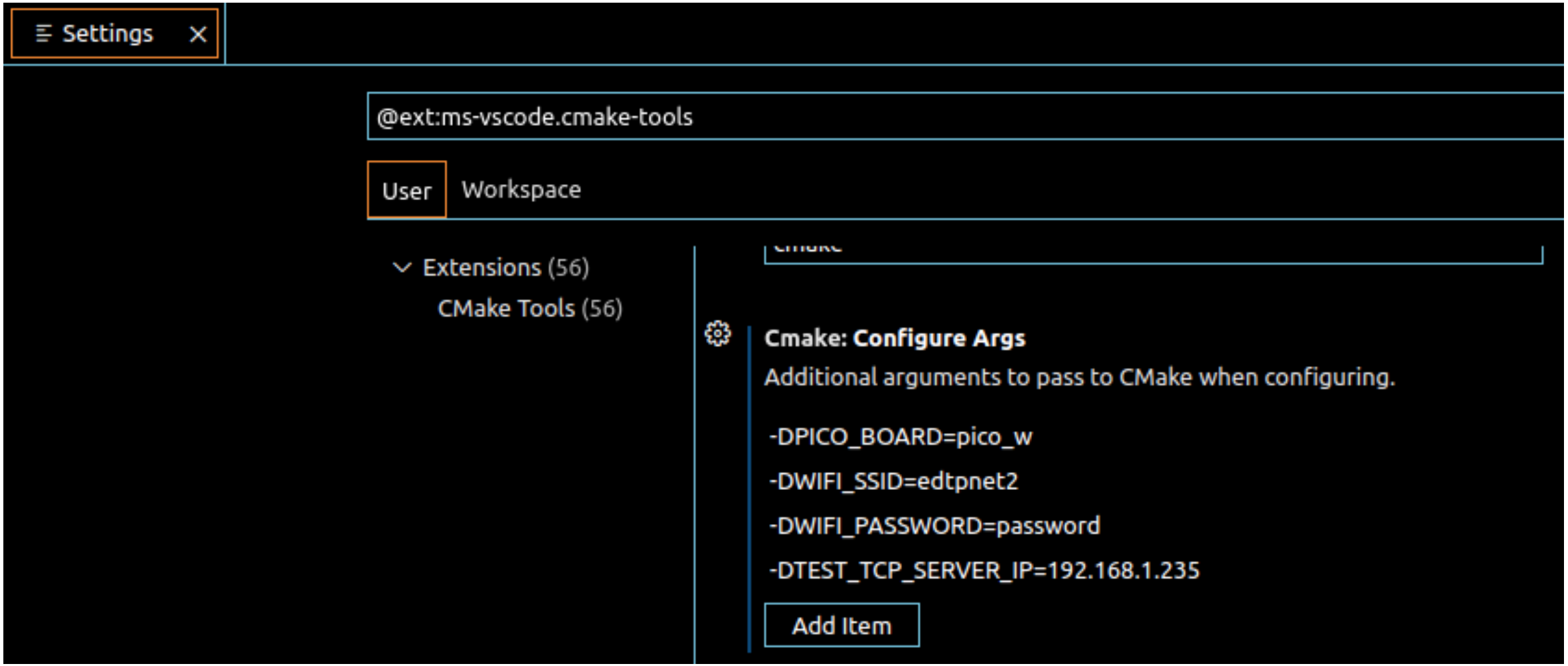




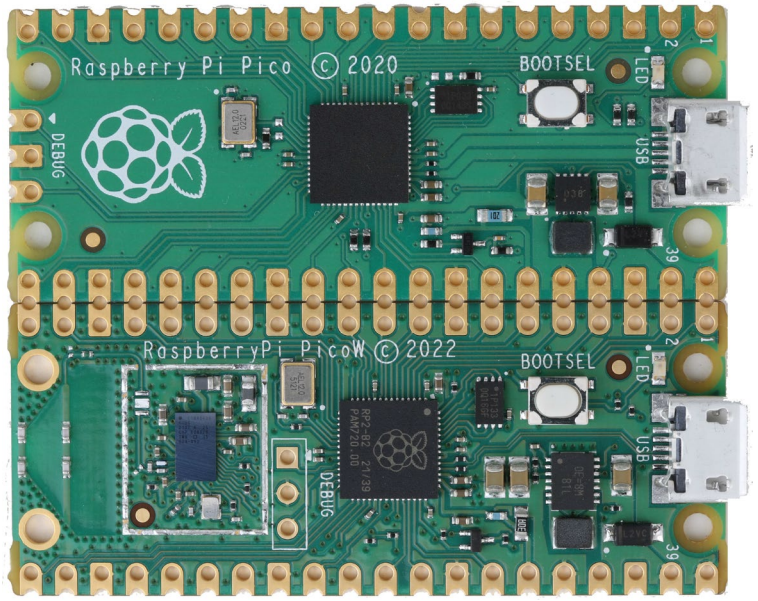
Visual Studio Code Configuration



Visual Studio Code Configuration



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($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)
6
7  # Set name of project (as PROJECT_NAME) and C/C++ standards
8  project(connect2edtpnet C CXX ASM)
9  set(CMAKE_C_STANDARD 11)
10 set(CMAKE_CXX_STANDARD 17)
11
12 # Creates a pico-sdk subdirectory in our project for the libraries
13 pico_sdk_init()
14
15 # Tell CMake where to find the executable source file
16 add_executable(${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
25 target_include_directories(${PROJECT_NAME} PRIVATE
26         ${CMAKE_CURRENT_LIST_DIR}
27         ${CMAKE_CURRENT_LIST_DIR}/.. # for our common lwipopts
28         )
29 # Link to pico_stdlib (gpio, time, etc. functions)
30 target_link_libraries(${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

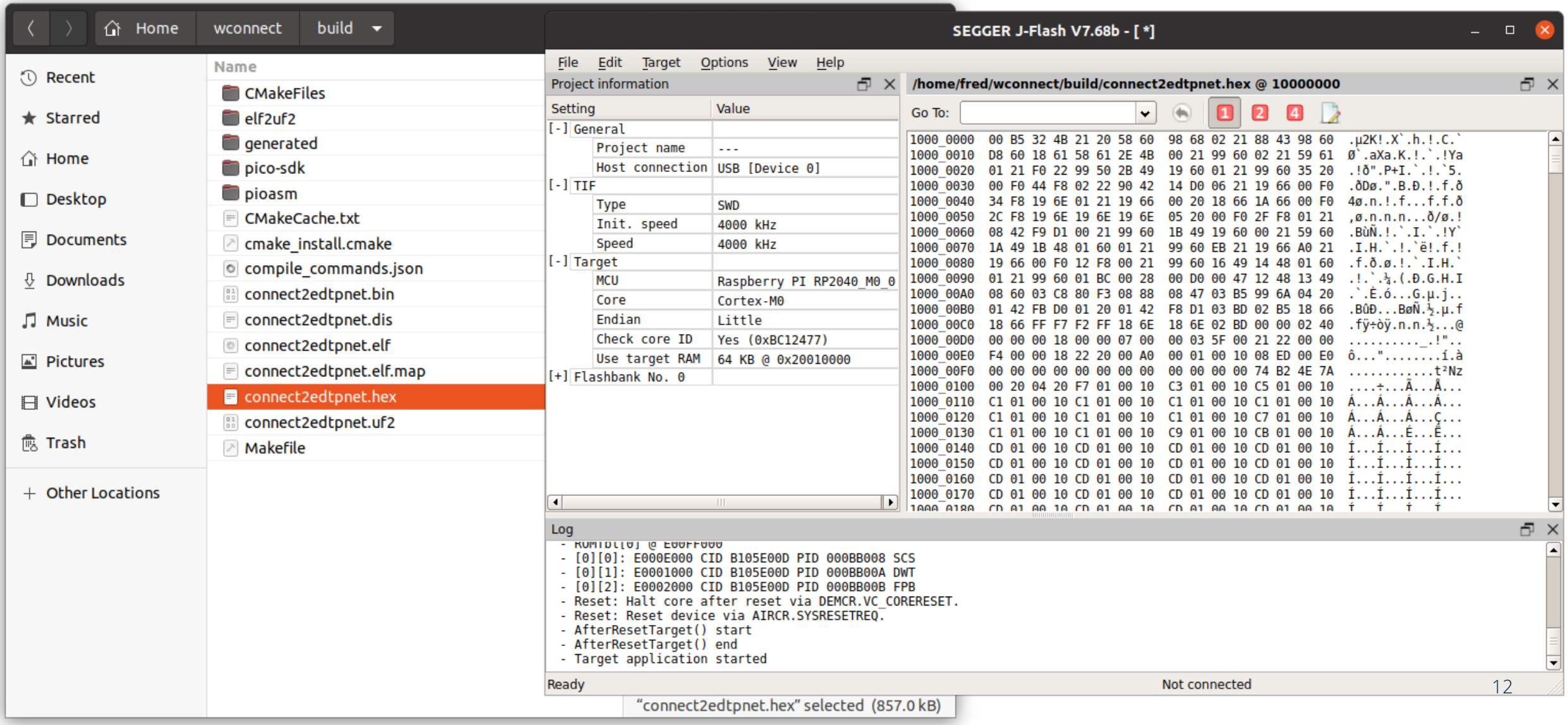
Add Item

main.c

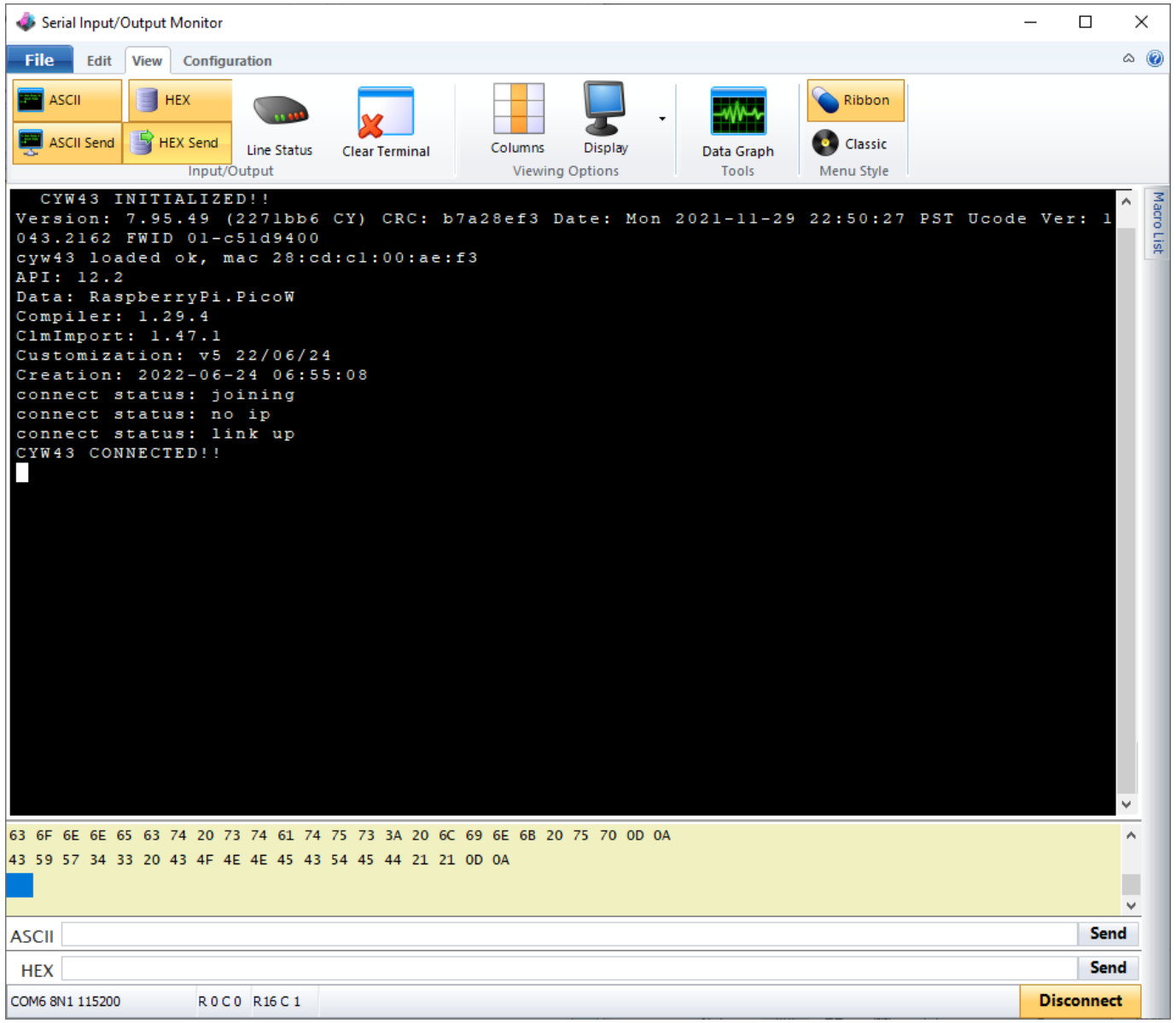
C main.c

```c
#include <stdio.h>
#include "pico/stdlib.h"
#include "pico/cyw43_arch.h"

int main()
{
    stdio_init_all();

    if(cyw43_arch_init_with_country(CYW43_COUNTRY_USA))
    {
        printf("CYW43 INIT FAILED!!\r\n");
        return 1;
    }
    printf("CYW43 INITIALIZED!!\r\n");

    cyw43_arch_enable_sta_mode();

    if(cyw43_arch_wifi_connect_timeout_ms(WIFI_SSID, WIFI_PASSWORD, CYW43_AUTH_WPA2_AES_PSK, 10000))
    {
        printf("CONNECT FAILED!!\r\n");
        return 1;
    }
    printf("CYW43 CONNECTED!!\r\n");
}
```

Load – Run - Connect



Connect Flow



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

```c
#include "pico/stdlib.h"
#include "pico/cyw43_arch.h"

#include "lwip/pbuf.h"
#include "lwip/tcp.h"

#if !defined(TEST_TCP_SERVER_IP)
#error TEST_TCP_SERVER_IP not defined
#endif

#define TCP_PORT 8088
#define DEBUG_printf printf
#define BUF_SIZE 17
#define POLL_TIME_S 5

typedef struct TCP_CLIENT_T_ {
    struct tcp_pcb *tcp_pcb;
    ip_addr_t remote_addr;
    uint8_t buffer[BUF_SIZE];
    int buffer_len;
    int sent_len;
    bool complete;
    bool connected;
} 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

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 > ...

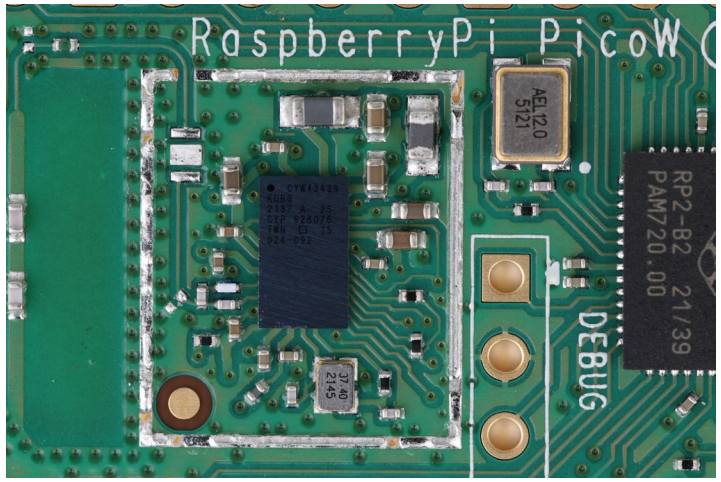
```
126  err_t tcp_client_recv(void *arg, struct tcp_pcb *tpcb, struct pbuf *p, err_t err) {
127      TCP_CLIENT_T *state = (TCP_CLIENT_T*)arg;
128      if (!p) {
129          printf("receive buffer error.\n");
130          while(1);
131      }
132      // this method is callback from lwIP, so cyw43_arch_lwip_begin 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 > ...

```c

static err_t tcp_client_sent(void *arg, struct tcp_pcb *tpcb, u16_t len) {
    TCP_CLIENT_T *state = (TCP_CLIENT_T*)arg;
    DEBUG_printf("tcp_client_sent %u\n", len);
    return ERR_OK;
}

static err_t tcp_client_connected(void *arg, struct tcp_pcb *tpcb, err_t err) {
    TCP_CLIENT_T *state = (TCP_CLIENT_T*)arg;
    if (err != ERR_OK) {
        printf("connect failed %d\n", err);
        while(1);
    }
    state->connected = true;
    DEBUG_printf("Waiting for buffer from server\n");
    return ERR_OK;
}

static err_t tcp_client_poll(void *arg, struct tcp_pcb *tpcb) {
    DEBUG_printf("tcp_client_poll\n");
    return 0; //tcp_result(arg, 0); // no response is an error?
}

static void tcp_client_err(void *arg, err_t err) {
    if (err != ERR_ABRT) {
        DEBUG_printf("tcp_client_err %d\n", err);
        while(1);
    }
}
```

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
ClmImport: 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

HW group 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
- RP2040 Datasheet
- Raspberry Pi Pico C/C++ SDK
- SEGGER J-Link
- SEGGER Ozone Debugger
- lwIP API Documentation

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

informa markets