

Raspberry Pi Pico/Pico2 Development Using Visual Studio Code

Day 4:

Coding Pico 2 Networking With VS Code

Sponsored by

DigiKey

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.

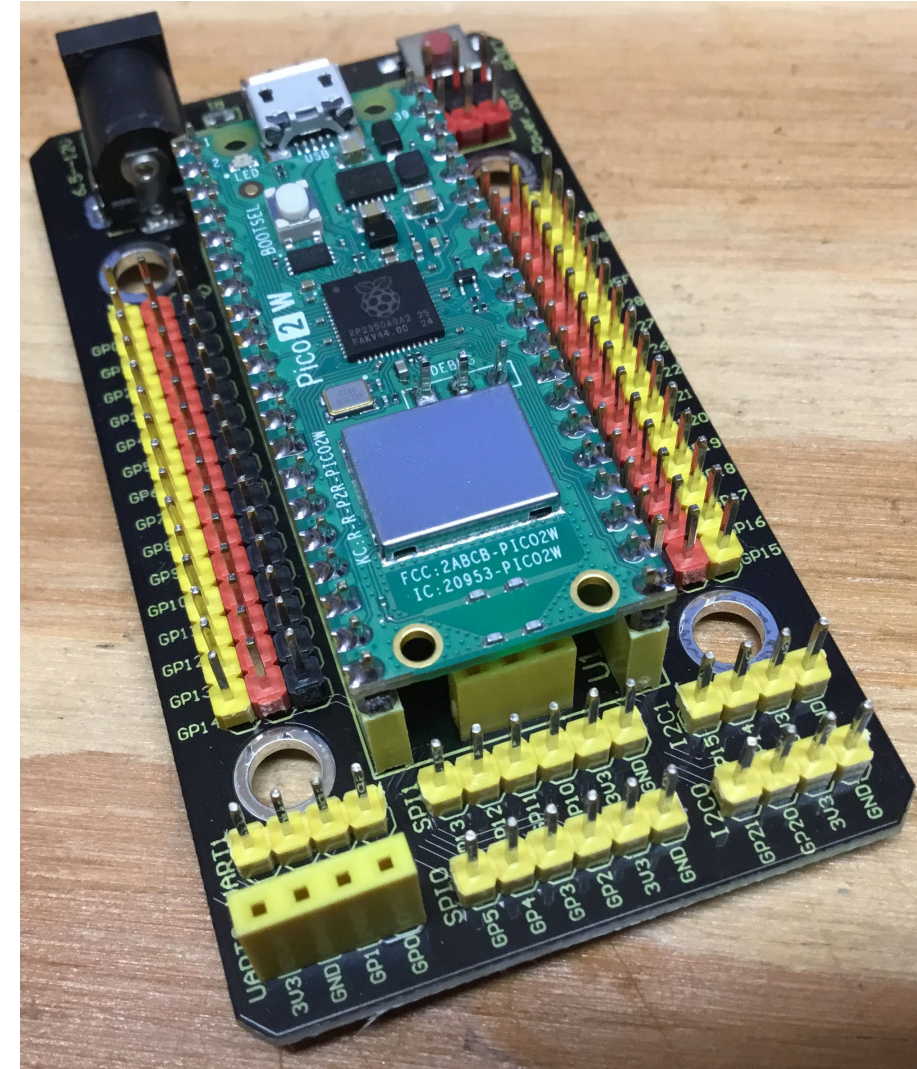


Fred Eady

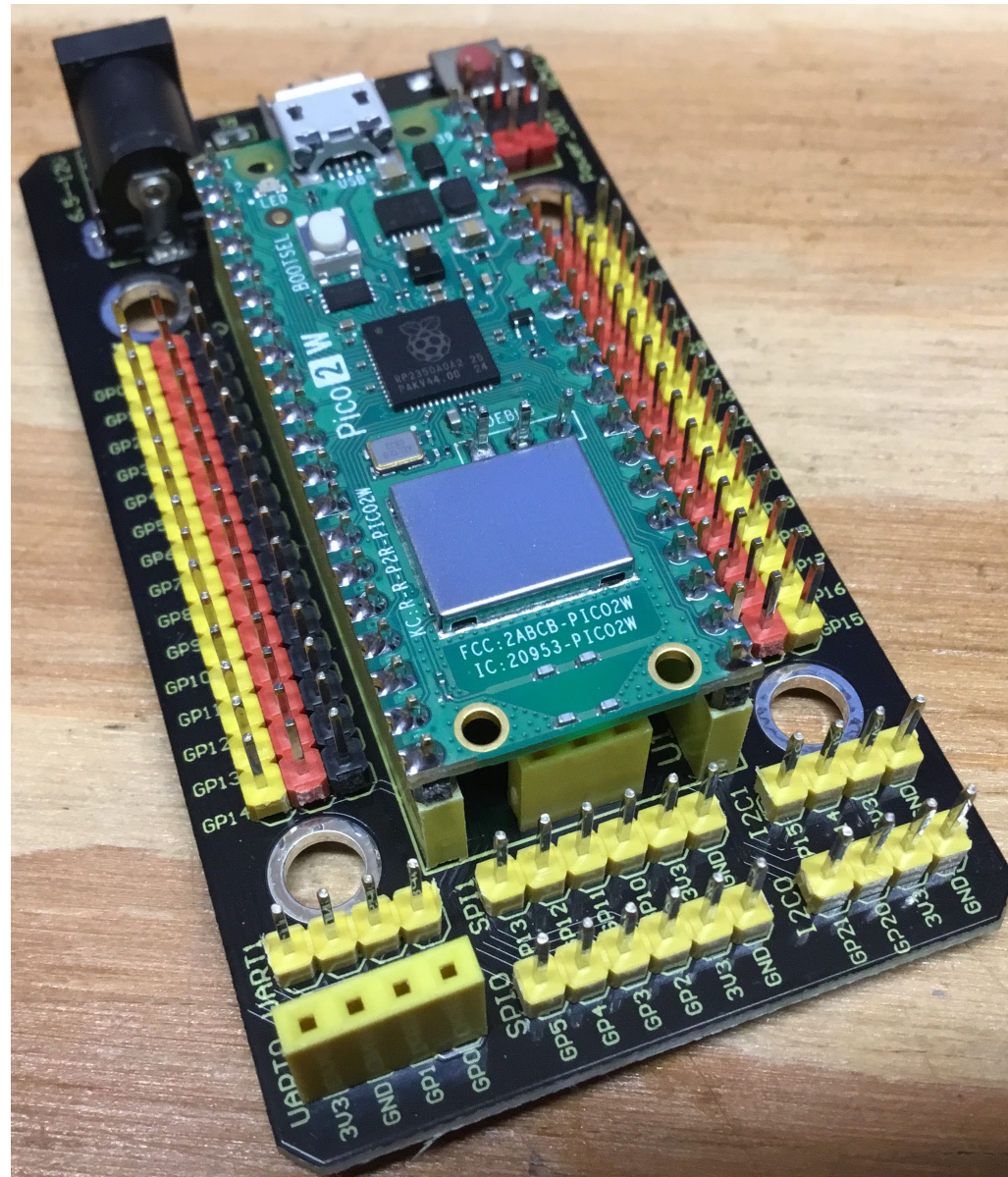
Visit 'Lecturer Profile' in your console for more details.

AGENDA

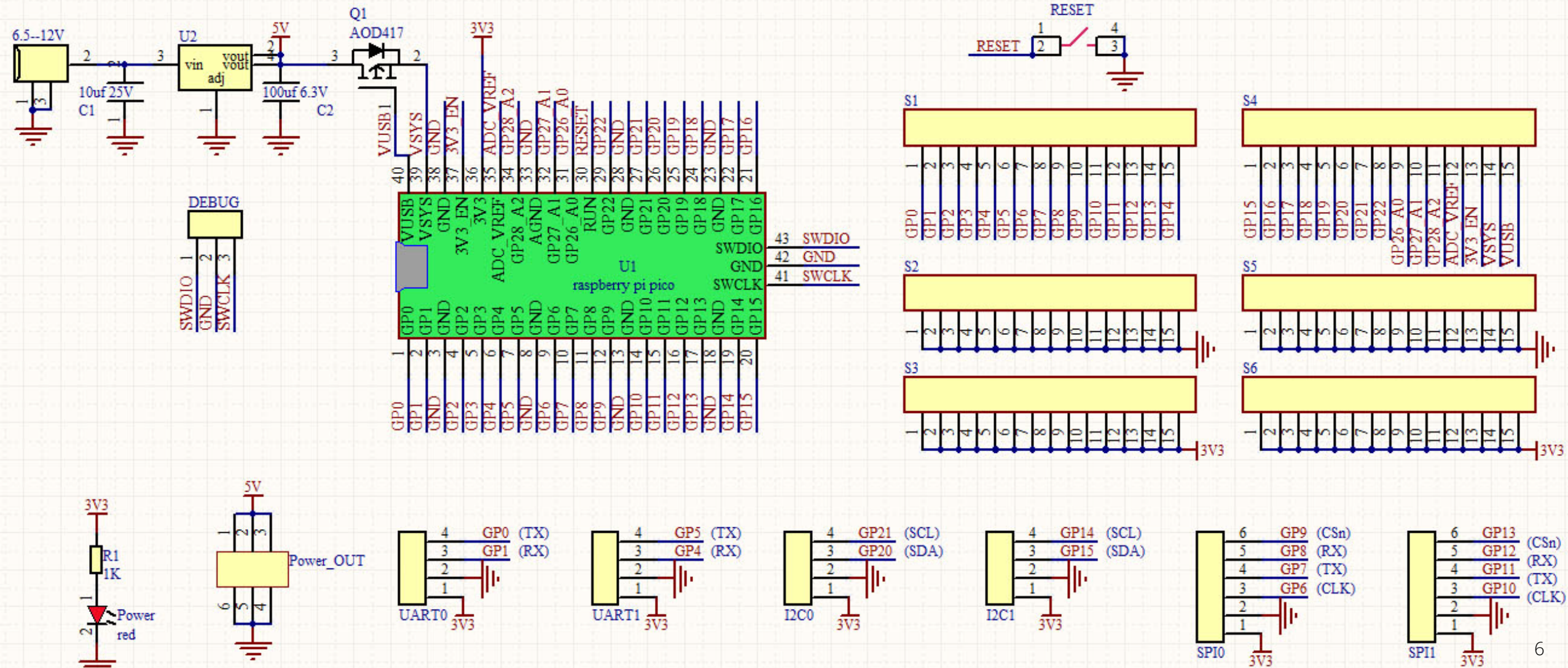
- **Pico 2 W Development Hardware**
- **Raspberry Pi Debug Probe**
 - **The Hook Up**
- **Code a Pico 2 W Wi-Fi Application**
 - **Connect to the EDTP LAN Using Wi-Fi**
 - **Code a Pico 2 UDP Application**



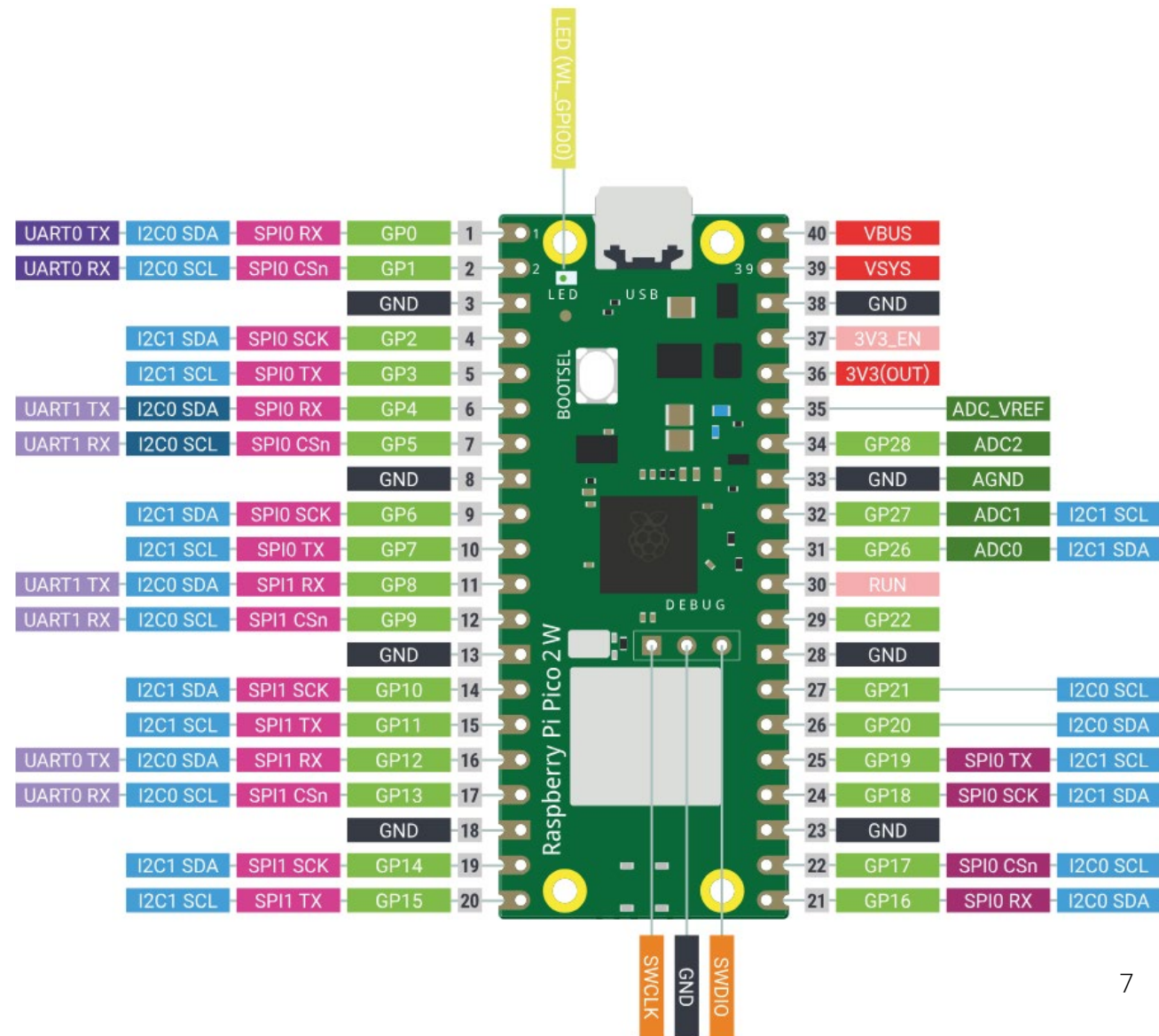
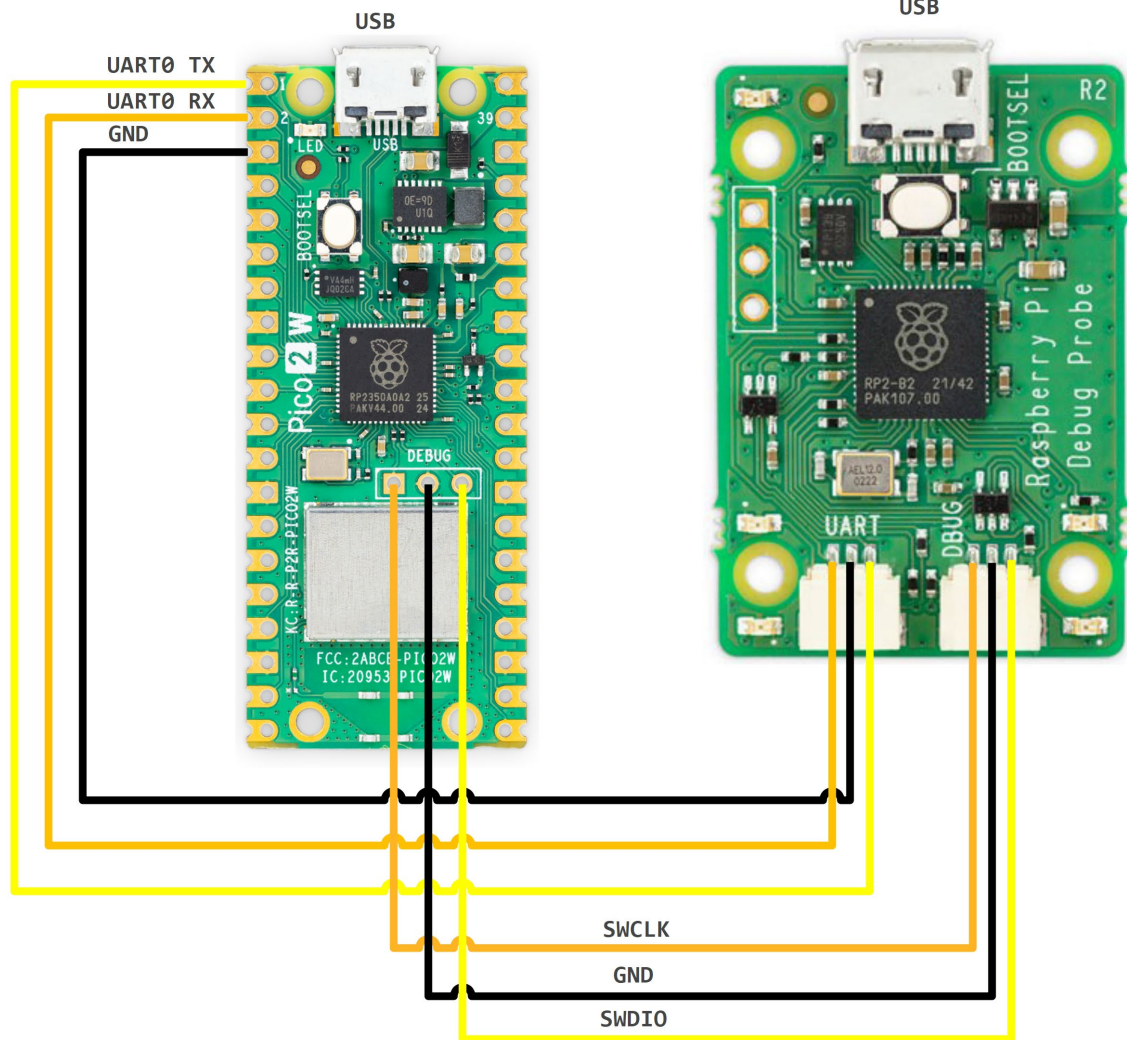
Pico 2 W Carrier Board



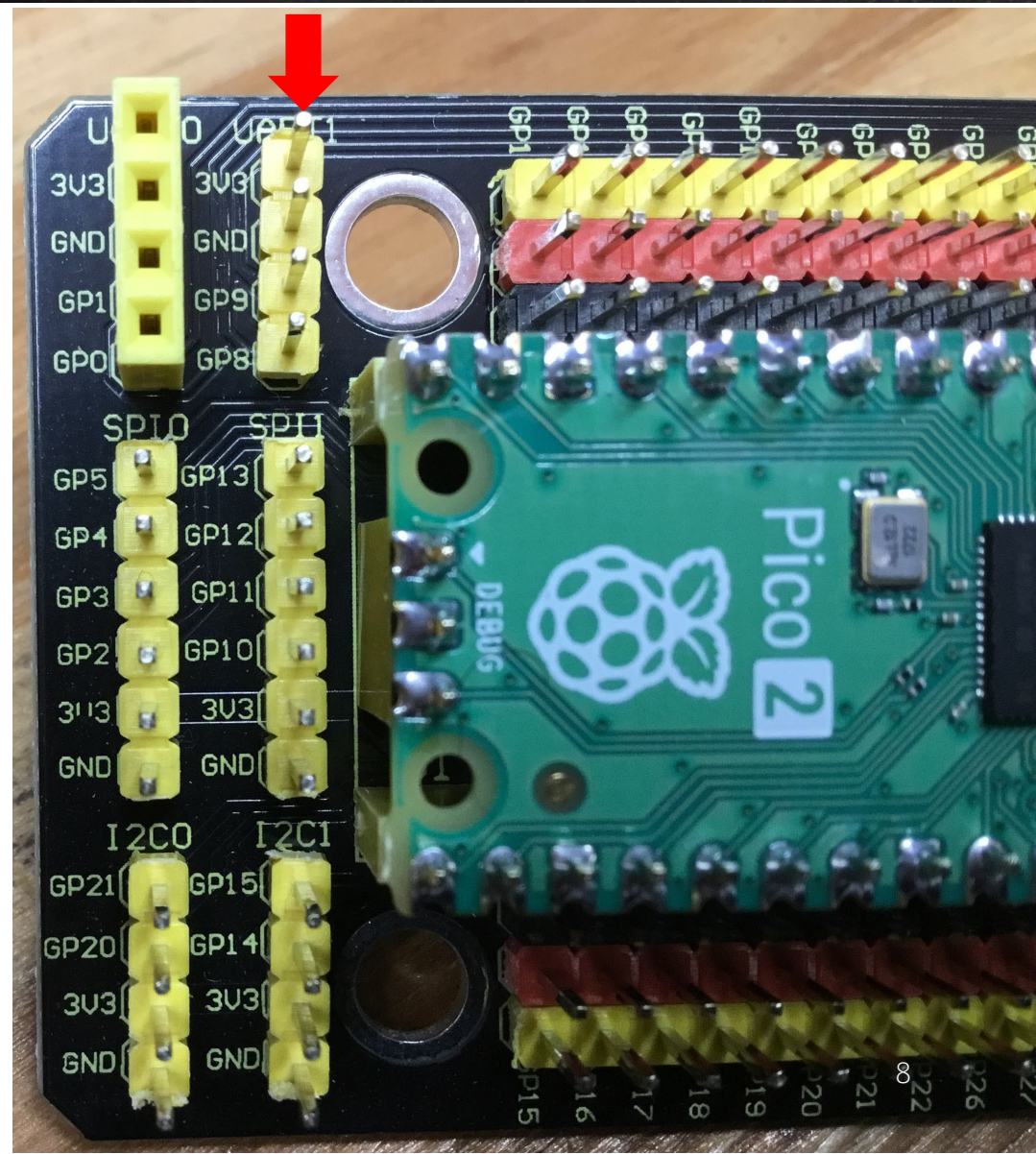
Pico 2 Carrier Board



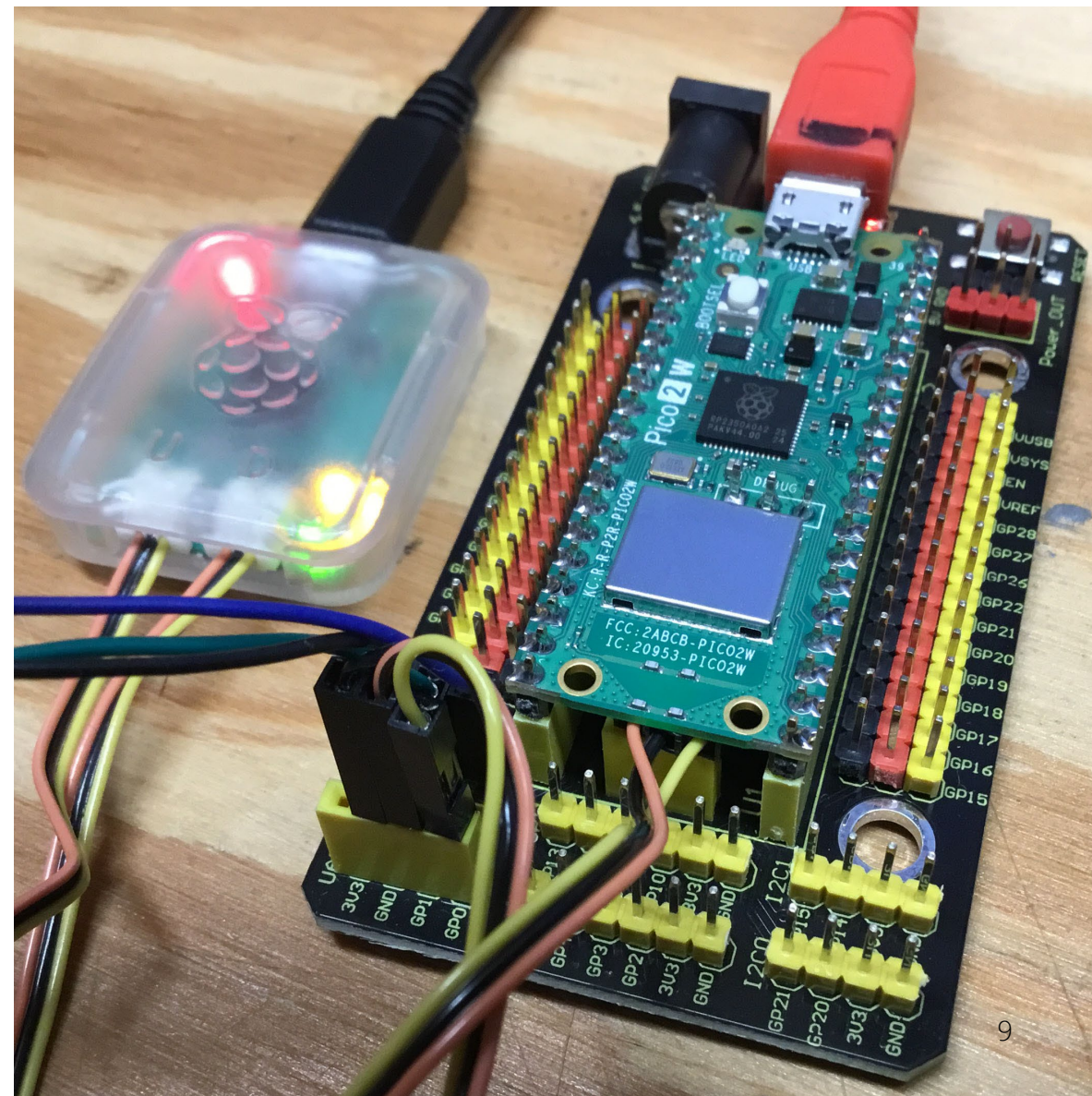
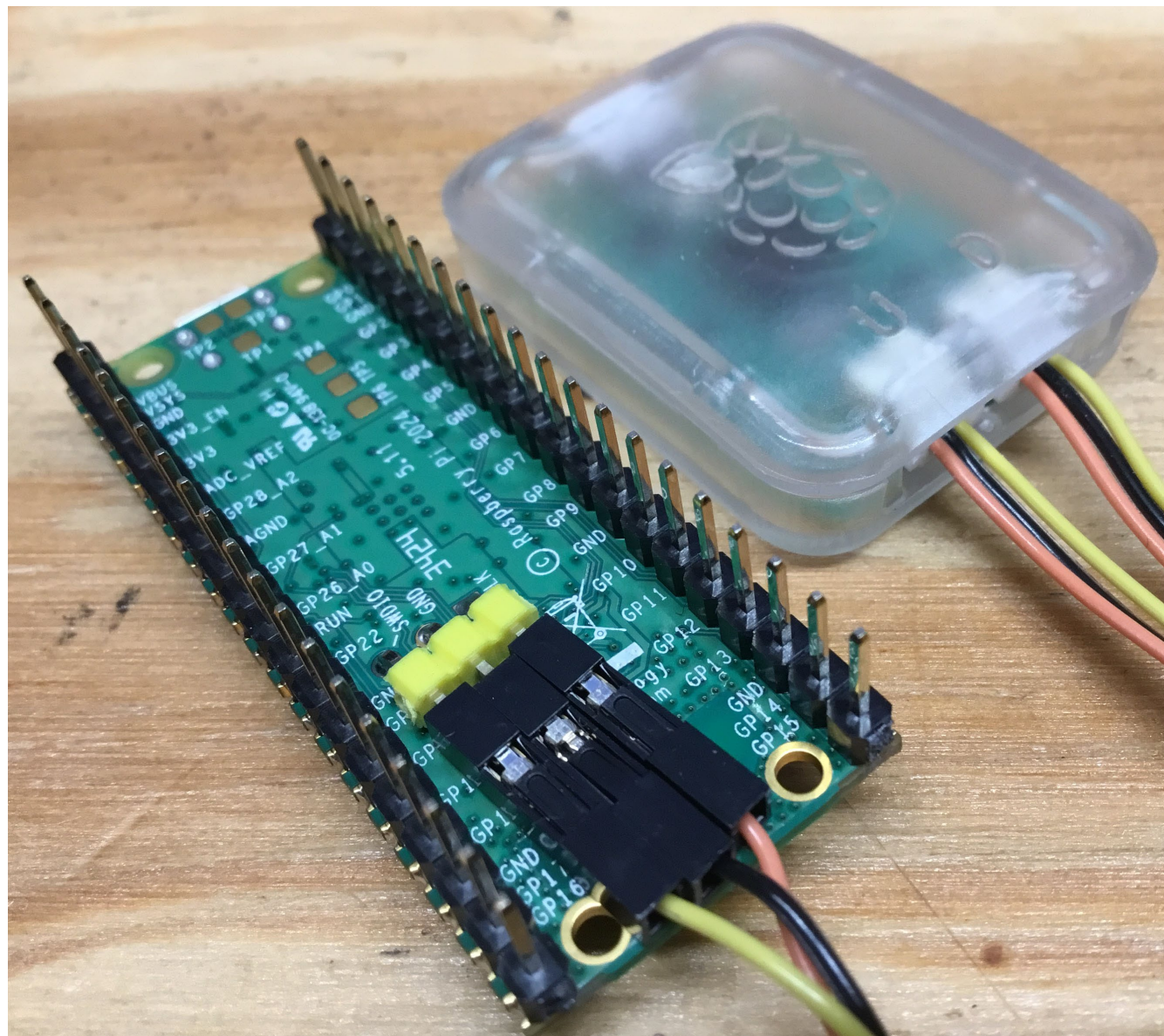
The Hook Up



The Hook Up



The Hook Up



Connect to the EDTP Shop LAN

The screenshot shows the Visual Studio Code interface for creating a new Raspberry Pi Pico project. The left sidebar contains a 'RASPBERY PI PICO PROJECT: QUICK ACCESS' menu with options like 'New C/C++ Project', 'New MicroPython Project', 'Import Project', and 'New Project From Example'. A red arrow points to 'New C/C++ Project'. The main workspace shows the 'New Pico Project' dialog with the following settings:

- Name:** pico2_wifi
- Example:** Example
- Board type:** Pico 2 W
- Architecture (Pico 2):** RISC-V
- Location:** /home/fred/pico2_Projects
- Select Pico SDK version:** v2.1.0

The 'Features' section is also visible at the bottom of the dialog.

Connect to the EDTP Shop LAN

RASPBERRY PI PICO PROJECT: QUICK ACCESS

Welcome | New Pico Project X

General

- New C/C++ Project
- New MicroPython Project
- Import Project
- New Project From Example

Project

- Debug Project
- Compile Project
- Run Project (USB)
- Flash Project (SWD)
- Configure CMake
- Clean CMake
- Switch SDK Current: N/A
- Switch Board
- Debug Layout

Documentation

- Hardware APIs
- High Level APIs
- Networking Libraries
- Runtime Infrastructure

Raspberry Pi Pico

Basic Settings | Features | Stdio support | Pico wireless options | Code generation options | Debugger

Features

Add example code snippets to demonstrate use of these features

<input type="checkbox"/> SPI	<input type="checkbox"/> I2C interface	<input type="checkbox"/> UART
<input type="checkbox"/> PIO interface	<input type="checkbox"/> DMA support	<input type="checkbox"/> HW interpolation
<input type="checkbox"/> HW watchdog	<input type="checkbox"/> HW timer	<input type="checkbox"/> HW clocks

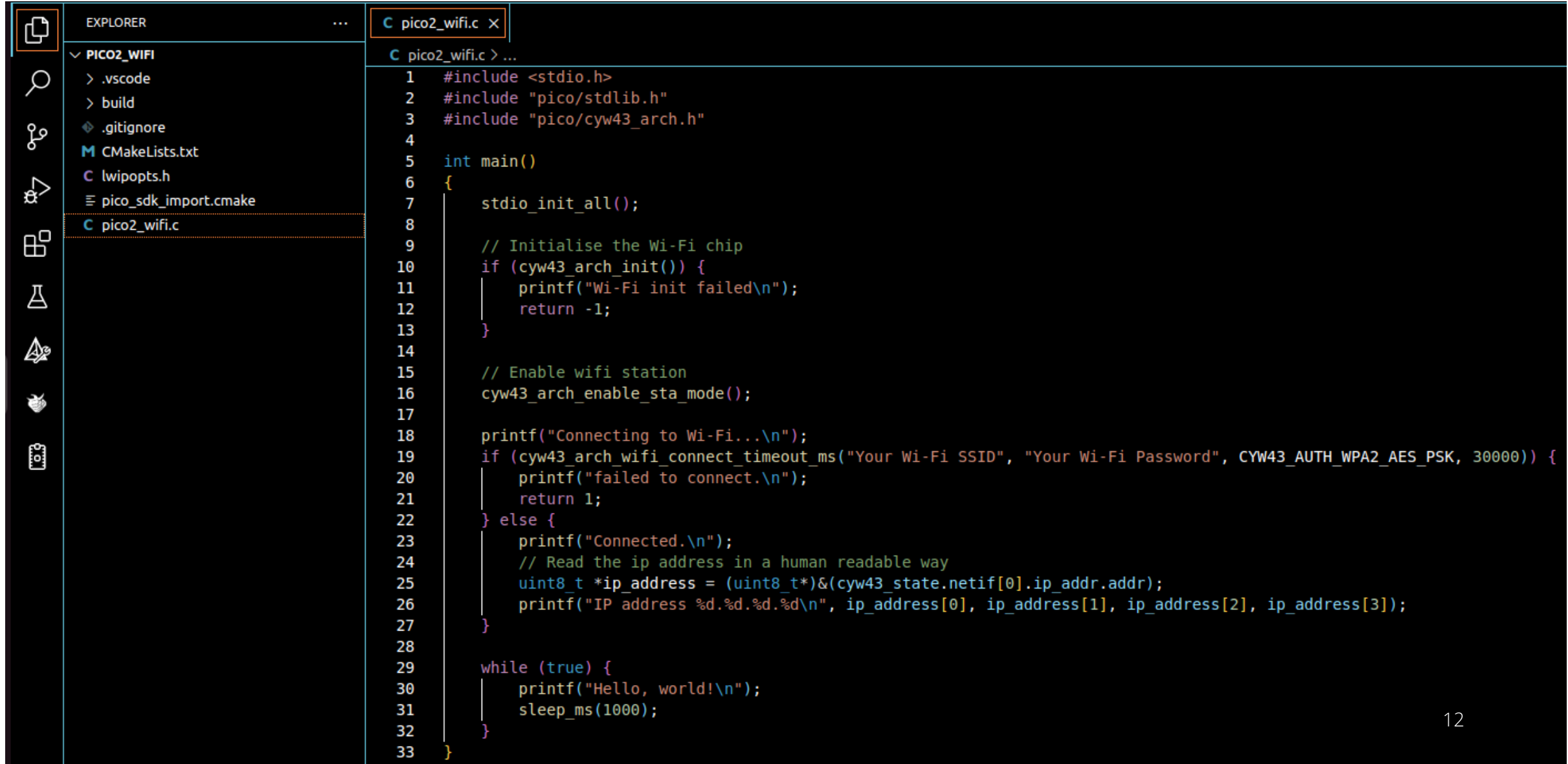
Stdio support

<input type="checkbox"/> Console over UART	<input type="checkbox"/> Console over USB (disables other USB use)
--------------------------------------------	--------------------------------------------------------------------

Pico wireless options

<input type="radio"/> None	<input type="radio"/> Pico W onboard LED	<input checked="" type="radio"/> Polled lwIP	<input type="radio"/> Background lwIP
----------------------------	------------------------------------------	----------------------------------------------	---------------------------------------

Connect to the EDTP Shop LAN



```
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     // Initialise the Wi-Fi chip
10    if (cyw43_arch_init()) {
11        printf("Wi-Fi init failed\n");
12        return -1;
13    }
14
15    // Enable wifi station
16    cyw43_arch_enable_sta_mode();
17
18    printf("Connecting to Wi-Fi...\n");
19    if (cyw43_arch_wifi_connect_timeout_ms("Your Wi-Fi SSID", "Your Wi-Fi Password", CYW43_AUTH_WPA2_AES_PSK, 30000)) {
20        printf("failed to connect.\n");
21        return 1;
22    } else {
23        printf("Connected.\n");
24        // Read the ip address in a human readable way
25        uint8_t *ip_address = (uint8_t*)&(cyw43_state.netif[0].ip_addr.addr);
26        printf("IP address %d.%d.%d.%d\n", ip_address[0], ip_address[1], ip_address[2], ip_address[3]);
27    }
28
29    while (true) {
30        printf("Hello, world!\n");
31        sleep_ms(1000);
32    }
33 }
```

Connect to the EDTP Shop LAN

```
C pico2_wifi.c > main()
1  #include <stdio.h>
2  #include "pico/stdlib.h"
3  #include "pico/cyw43_arch.h"
4  #include "hardware/uart.h"
5
6  #define UART1_ID uart1
7  #define BAUD_RATE 115200
8  #define DATA_BITS 8
9  #define STOP_BITS 1
10 #define PARITY    UART_PARITY_NONE
11
12 #define UART1_TX_PIN 8
13 #define UART1_RX_PIN 9
```



Connect to the EDTP Shop LAN

```
C pico2_wifi.c > main()
1  #include <stdio.h>
2  #include "pico/stdlib.h"
3  #include "pico/cyw43_arch.h"
4  #include "hardware/uart.h"
5
6  #define UART1_ID uart1
7  #define BAUD_RATE 115200
8  #define DATA_BITS 8
9  #define STOP_BITS 1
10 #define PARITY    UART_PARITY_NONE
11
12 #define UART1_TX_PIN 8
13 #define UART1_RX_PIN 9
14
15 int main()
16 {
17     gpio_set_function(UART1_TX_PIN, UART_FUNCSEL_NUM(UART1_ID, UART1_TX_PIN));
18     gpio_set_function(UART1_RX_PIN, UART_FUNCSEL_NUM(UART1_ID, UART1_RX_PIN));
19
20     //uart_init(UART0_ID, 115200);
21     uart_init(UART1_ID, 115200);
22     //uart_set_hw_flow(UART0_ID, false, false);
23     uart_set_hw_flow(UART1_ID, false, false);
24     //uart_set_format(UART0_ID, DATA_BITS, STOP_BITS, PARITY);
25     uart_set_format(UART1_ID, DATA_BITS, STOP_BITS, PARITY);
26     //uart_set_fifo_enabled(UART0_ID, false);
27     uart_set_fifo_enabled(UART1_ID, false);
```



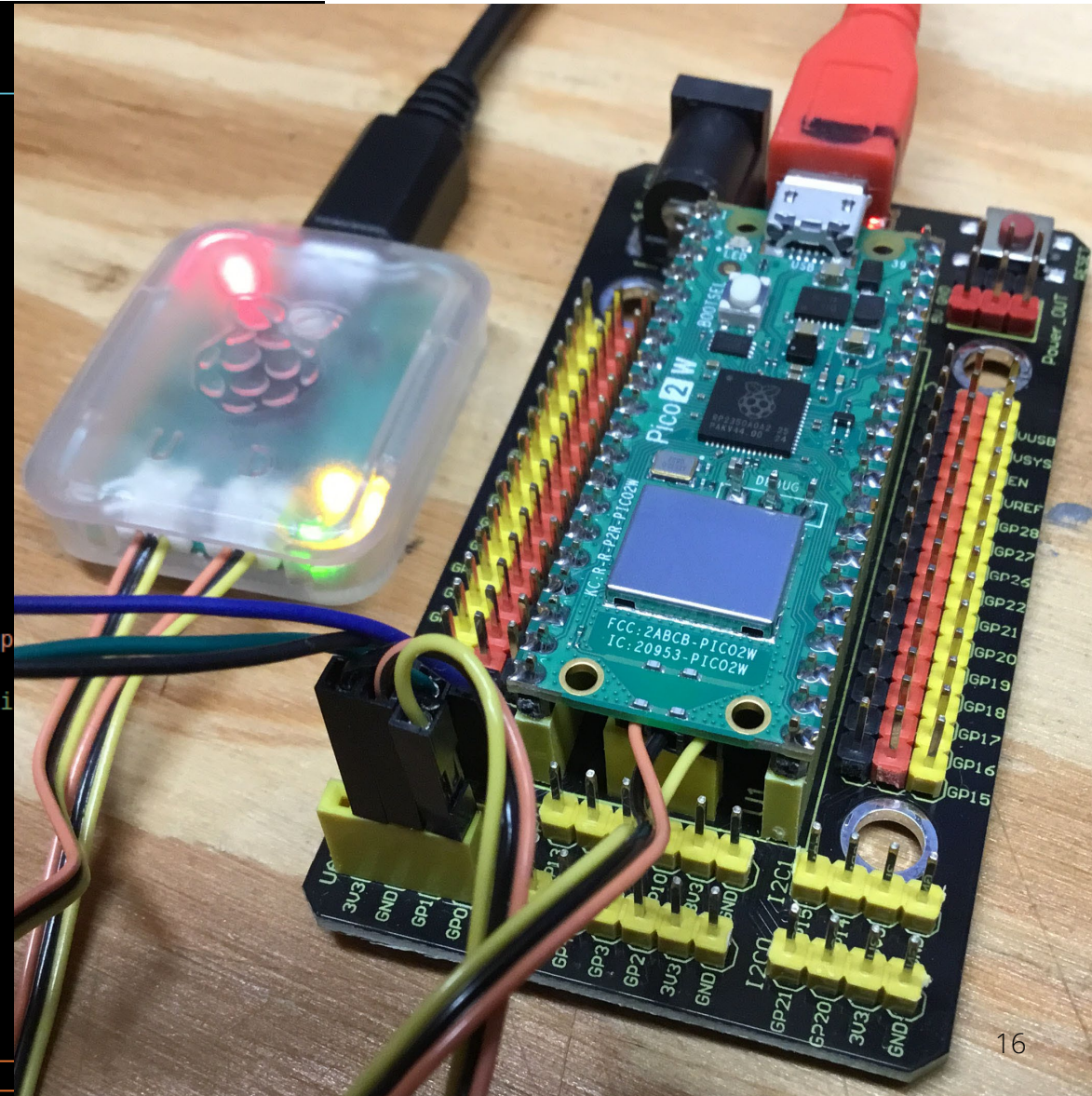
Connect to the EDTP Shop LAN

```
29 // Initialise the Wi-Fi chip
30 if (cyw43_arch_init()) {
31     uart_puts(UART1_ID, "Wi-Fi init failed\n");
32     return -1;
33 }
34
35 // Enable wifi station
36 cyw43_arch_enable_sta_mode();
37
38 uart_puts(UART1_ID, "Connecting to Wi-Fi...\n");
39 if (cyw43_arch_wifi_connect_timeout_ms("edtpnet2", "Your Password", CYW43_AUTH_WPA2_AES_PSK, 30000)) {
40     uart_puts(UART1_ID, "failed to connect.\n");
41     return 1;
42 } else {
43     uart_puts(UART1_ID, "Connected.\n");
44 }
45
46 while (true) {
47     uart_puts(UART1_ID, "Connected to edtpnet2!\n");
48     sleep_ms(1000);
49 }
50 }
```

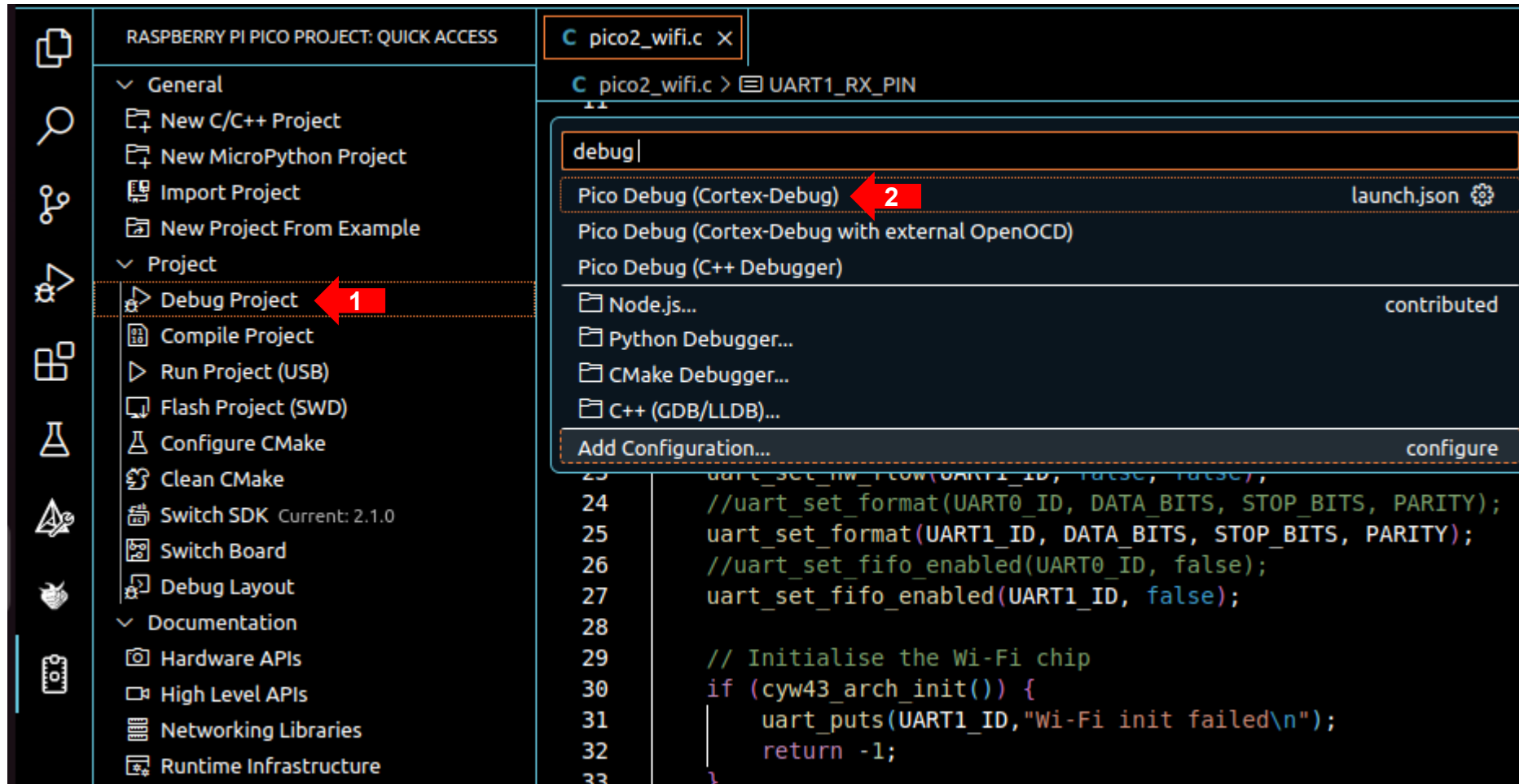


Enable *liveWatch*

```
C pico2_wifi.c {} launch.json 1 X
.vscode > {} launch.json > Launch Targets > {} Pico Debug (Cortex-Debug)
1  {
2  "version": "0.2.0",
3  "configurations": [
4  {
5  "name": "Pico Debug (Cortex-Debug)",
6  "cwd": "${userHome}/.pico-sdk/openocd/0.12.0+dev/scripts",
7  "executable": "${command:raspberry-pi-pico.launchTargetPath}",
8  "request": "launch",
9  "type": "cortex-debug",
10 "serverType": "openocd",
11 "serverPath": "${userHome}/.pico-sdk/openocd/0.12.0+dev/openocd.exe",
12 "gdbPath": "${command:raspberry-pi-pico.getGDBPath}",
13 "device": "${command:raspberry-pi-pico.getChipUppercase}",
14 "configFiles": [
15   "interface/cmsis-dap.cfg",
16   "target/${command:raspberry-pi-pico.getTarget}.cfg"
17 ],
18 "svdFile": "${userHome}/.pico-sdk/sdk/2.1.0/src/${command:raspberry-pi-pico.getChipUppercase}.svd",
19 "runToEntryPoint": "main",
20 // Fix for no_flash binaries, where monitor reset halt doesn't do what it should
21 // Also works fine for flash binaries
22 "overrideLaunchCommands": [
23   "monitor reset init",
24   "load \\\"${command:raspberry-pi-pico.launchTargetPath}\\\""
25 ],
26 "openOCDLaunchCommands": [
27   "adapter speed 5000"
28 ]
29 "liveWatch": {
30   "enabled": true,
31   "samplesPerSecond": 4
32 }
```



Debug the Pico 2 W Wi-Fi Connect Application

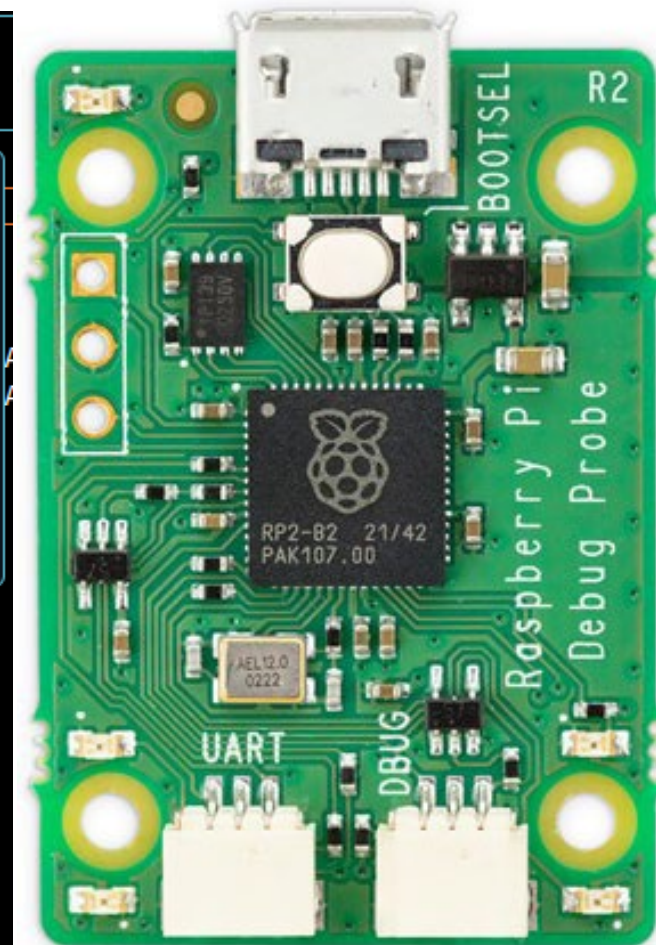


The screenshot shows the Visual Studio Code interface. On the left, the 'Project' menu is open, and 'Debug Project' is highlighted with a red arrow labeled '1'. In the center, a search box contains the text 'debug', and the search results show 'Pico Debug (Cortex-Debug)' highlighted with a red arrow labeled '2'. The code editor on the right shows the following C code:

```
C pico2_wifi.c x
C pico2_wifi.c > UART1_RX_PIN

debug|
Pico Debug (Cortex-Debug) ← 2 launch.json ⚙
Pico Debug (Cortex-Debug with external OpenOCD)
Pico Debug (C++ Debugger)
Node.js... contributed
Python Debugger...
CMake Debugger...
C++ (GDB/LLDB)...
Add Configuration... configure

23  uart_set_hw_flow(UART1_ID, false, false);
24  //uart_set_format(UART0_ID, DATA_BITS, STOP_BITS, PARITY);
25  uart_set_format(UART1_ID, DATA_BITS, STOP_BITS, PARITY);
26  //uart_set_fifo_enabled(UART0_ID, false);
27  uart_set_fifo_enabled(UART1_ID, false);
28
29  // Initialise the Wi-Fi chip
30  if (cyw43_arch_init()) {
31      uart_puts(UART1_ID, "Wi-Fi init failed\n");
32      return -1;
33  }
```



Debug the Pico 2 W Wi-Fi Connect Application

RUN AND D... Pico Debu... ...

VARIABLES

- Local
 - hi = 0
 - > timer = 0x400b0000
- Global
- Static: .././././sdk/2.1.0/src/rp2_common/hardw...
- Registers

WATCH

Code Editor:

```
C pico2_wifi.c
home > fred > .pico-sdk > sdk > 2.1.0 > src > rp2_common > hardware_timer > include > hardware > C timer.h > tim
7  #ifndef _HARDWARE_TIMER_H
15  extern "C" {
309  /*
310  void busy_wait_until(absolute_time_t t);
311
312  /*! \brief Check if the specified timestamp has been reached on the given timer
313  * \ingroup hardware_timer
```

Serial Monitor:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY SERIAL MONITOR

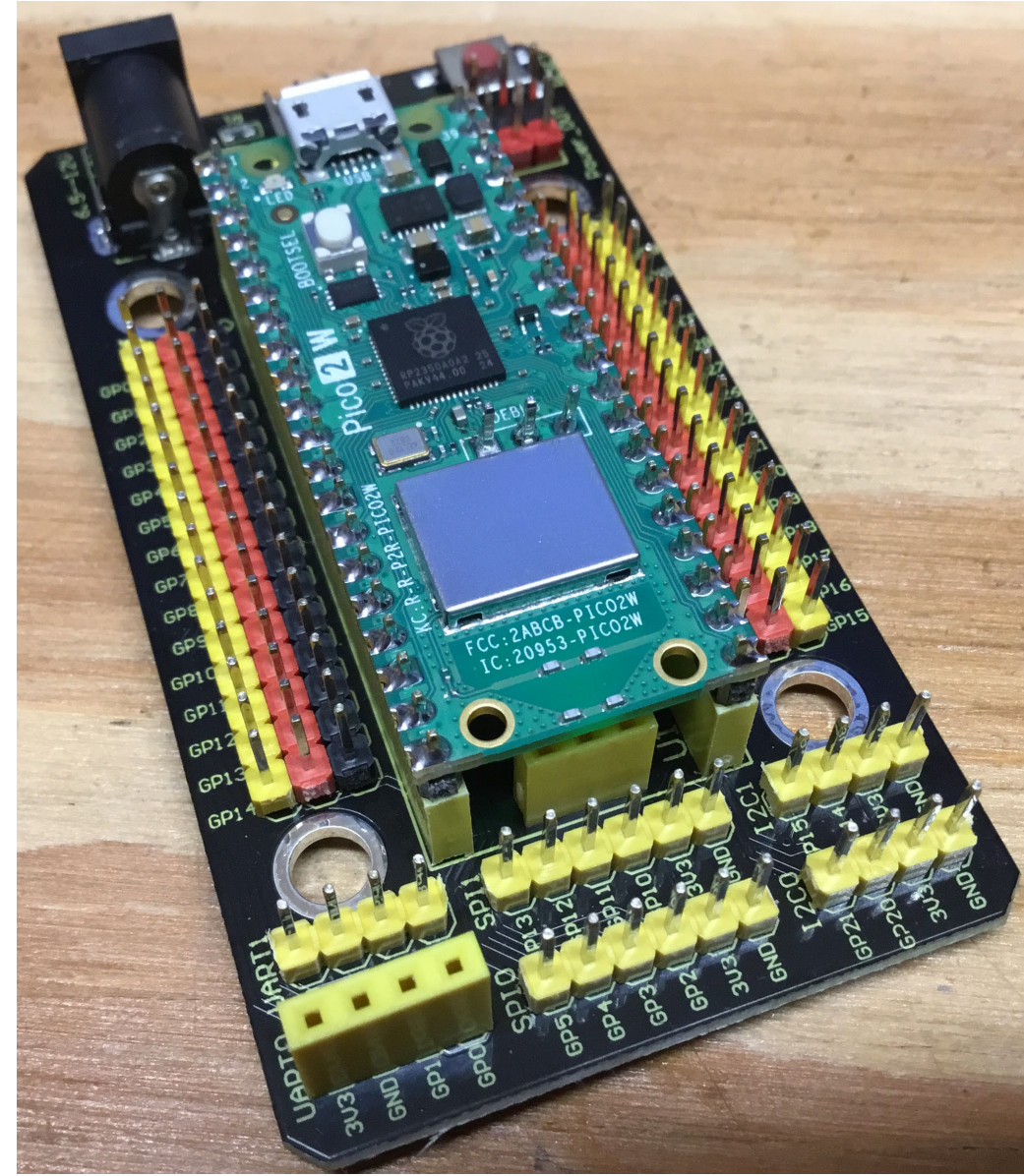
+ Open an additional monitor

Monitor Mode View Mode Port Baud rate

Connecting to Wi-Fi...
Connected.
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!
Connected to edtpnet2!

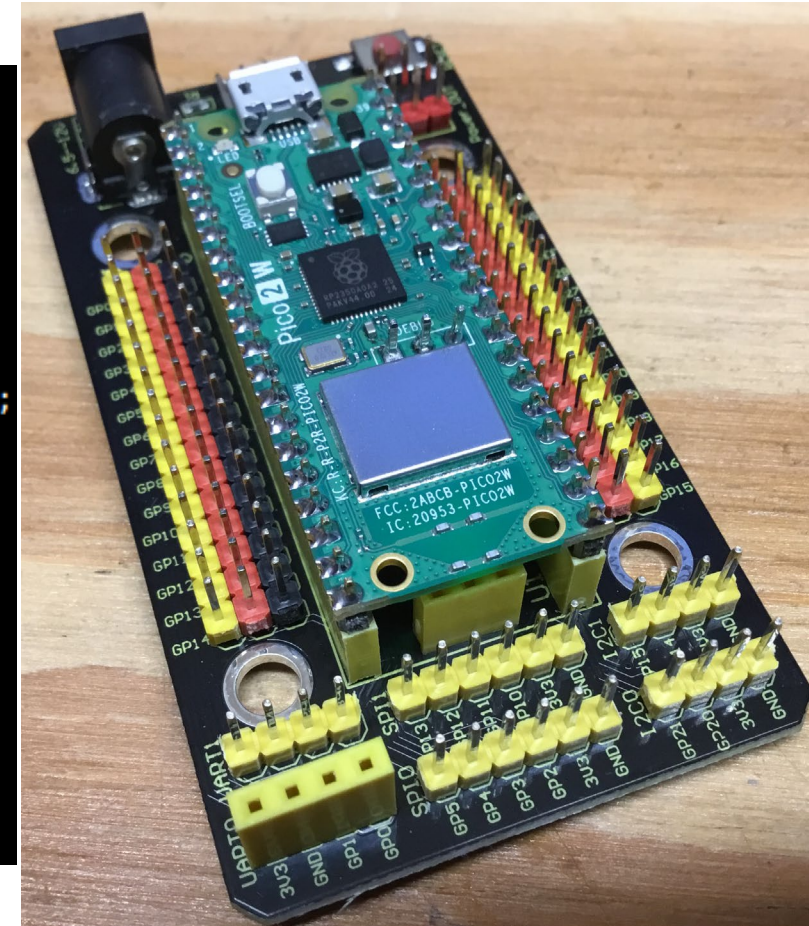
Define the UDP Application Ports and IP Addresses

```
C pico2w_udp.c X
C pico2w_udp.c > main()
1  #include <stdio.h>
3  #include "pico/cyw43_arch.h"
4
5  #include "lwip/pbuf.h"
6  #include "lwip/udp.h"
7  #include "hardware/uart.h"
8
9  #define UART1_ID uart1
10 #define BAUD_RATE 115200
11 #define DATA_BITS 8
12 #define STOP_BITS 1
13 #define PARITY    UART_PARITY_NONE
14
15 #define UART1_TX_PIN 8
16 #define UART1_RX_PIN 9
17
18 #define UDP_PORT 4444
19 #define UDP_MSG_LEN_MAX 127
20 #define UDP_TARGET "255.255.255.255"
21 #define UDP_TX_INTERVAL 1000
22
23 uint16_t sent_pkt_cnt = 0;
```



Code the UDP *send_udp_packet* Function

```
25 void send_udp_packet() {
26     struct udp_pcb* pcb = udp_new();
27
28     ip_addr_t addr;
29     ipaddr_aton(UDP_TARGET, &addr);
30
31     while (true) {
32         struct pbuf *p = pbuf_alloc(PBUF_TRANSPORT, UDP_MSG_LEN_MAX+1, PBUF_RAM);
33         char *pkt = (char *)p->payload;
34         memset(pkt, 0, UDP_MSG_LEN_MAX+1);
35         snprintf(pkt, UDP_MSG_LEN_MAX, "%d\n", sent_pkt_cnt);
36         err_t er = udp_sendto(pcb, p, &addr, UDP_PORT);
37         pbuf_free(p);
38         sent_pkt_cnt++;
39         cyw43_arch_poll();
40         sleep_ms(UDP_TX_INTERVAL);
41     }
42 }
```



Add the *send_udp_packet* Function Call

```
58 // Initialise the Wi-Fi chip
59 if (cyw43_arch_init()) {
60     uart_puts(UART1_ID, "Wi-Fi init failed\n");
61     return -1;
62 }
63
64 // Enable wifi station
65 cyw43_arch_enable_sta_mode();
66
67 uart_puts(UART1_ID, "Connecting to Wi-Fi...\n");
68 if (cyw43_arch_wifi_connect_timeout_ms("edtpnet2", "Your Password", CYW43_AUTH_WPA2_AES_PSK, 30000)) {
69     uart_puts(UART1_ID, "failed to connect.\n");
70     return 1;
71 } else {
72     uart_puts(UART1_ID, "Connected.\n");
73 }
74
75 send_udp_packet();
76 return 0;
77 }
```



Debug the Pico 2 W UDP Application

The image shows a dual-screen view of a development environment. On the left is Visual Studio Code, and on the right is the Hercules SETUP utility.

Visual Studio Code (Left):

- File Explorer: Shows the project structure with `pico2w_udp.c` selected.
- Debugger: Shows the execution of `main()` in `pico2w_udp.c`. The code includes:

```
25 void send_udp_packet() {  
43  
44 int main()  
45 {  
46     gpio_set_function(UART1_TX,  
47     gpio_set_function(UART1_RX,  
48  
49     //uart_init(UART0_ID, 115200);
```
- WATCH: Shows `sent_pkt_cnt: 9`.
- CORTEX LIVE WATCH: Shows `sent_pkt_cnt: 9`.
- Terminal: Shows the output of the debugger: `Connecting to Wi-Fi... Connected.`

Hercules SETUP utility (Right):

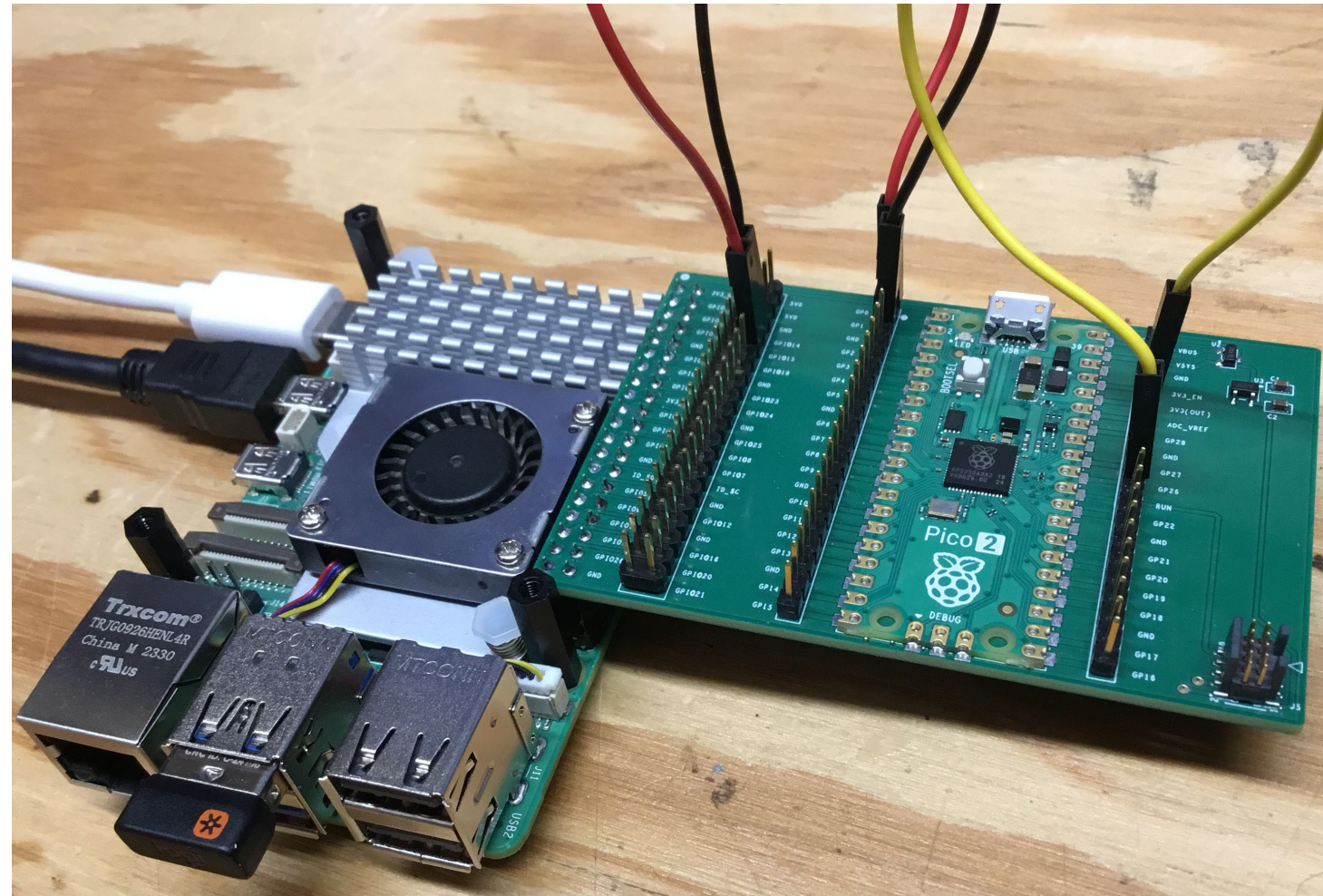
- Window Title: Hercules SETUP utility by HW-group.com
- Navigation: UDP Setup | Serial | TCP Client | TCP Server | UDP | Test Mode | About
- Received data: A list of 9 empty lines (0-9).
- Sent data: A large empty text area.
- UDP Settings: Module IP (192.168.1.235), Port (4044), Local port (4444), and a Close button.
- Server settings: Server echo, Redirect to TCP Server, Redirect to TCP Client.
- UDP broadcast: File name: No file, Load file, Send.
- Send area: Three input fields, each with a HEX checkbox and a Send button.
- Footer: HWgroup logo, www.HW-group.com, Hercules SETUP utility, Version 3.2.8.

Next Time...**MORE TO COME..**

Thank you for attending!!!

Please consider the resources below:

- **Today's Download Package**
- **[raspberrypi.org](https://www.raspberrypi.org)**





DesignNews

Thank You

Sponsored by

DigiKey

