



DesignNews

Raspberry Pi Pico/Pico2 Development Using Visual Studio Code

Day 3:

Raspberry Pi Pico 2 C Programming 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.

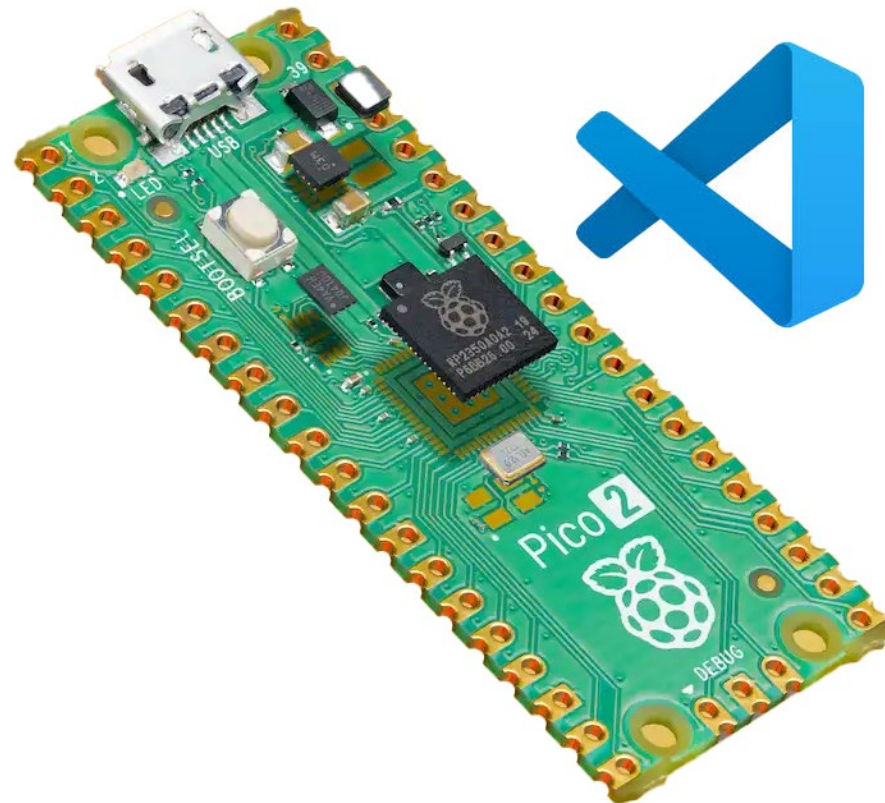


Fred Eady

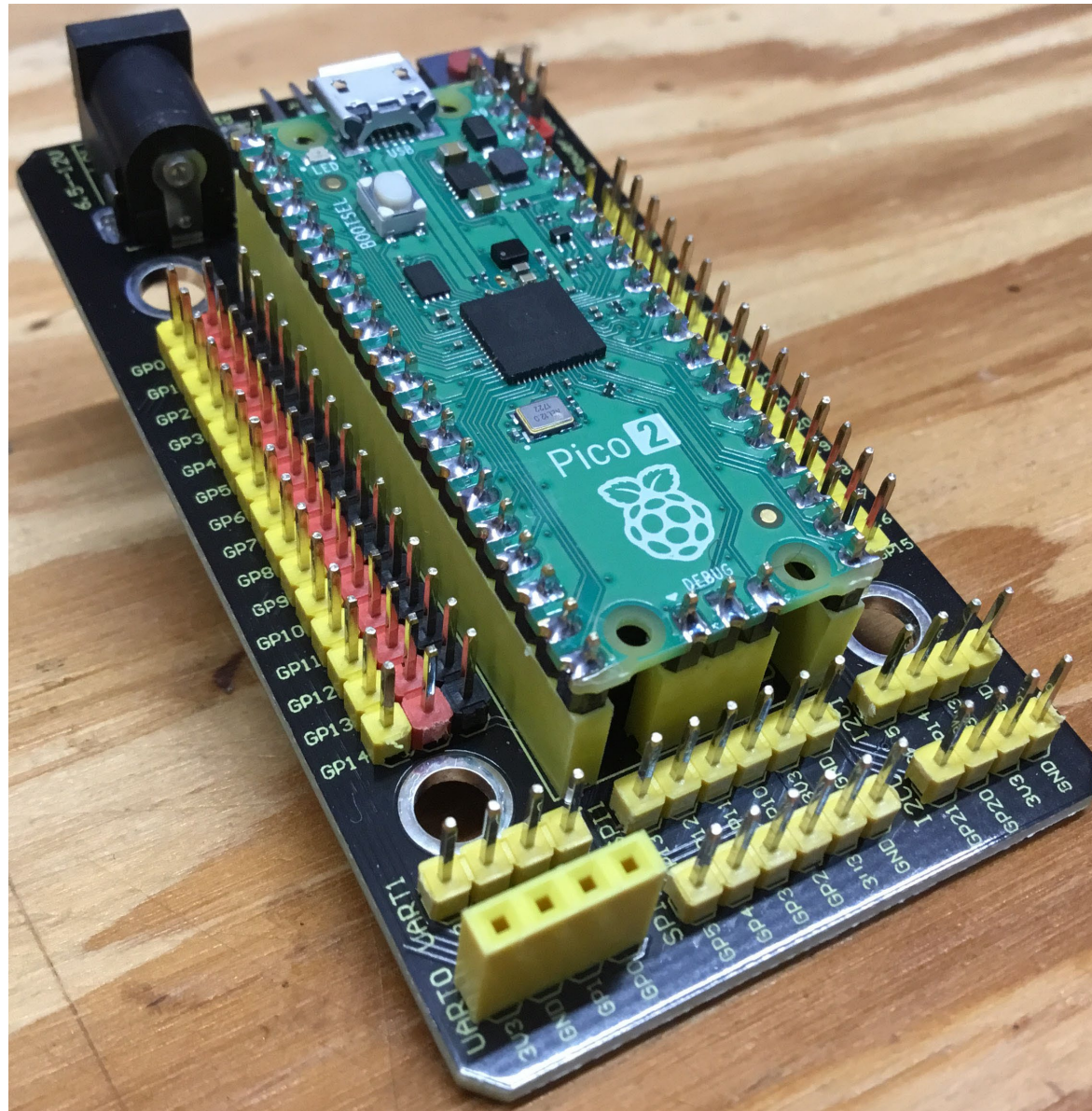
Visit 'Lecturer Profile' in your console for more details.

AGENDA

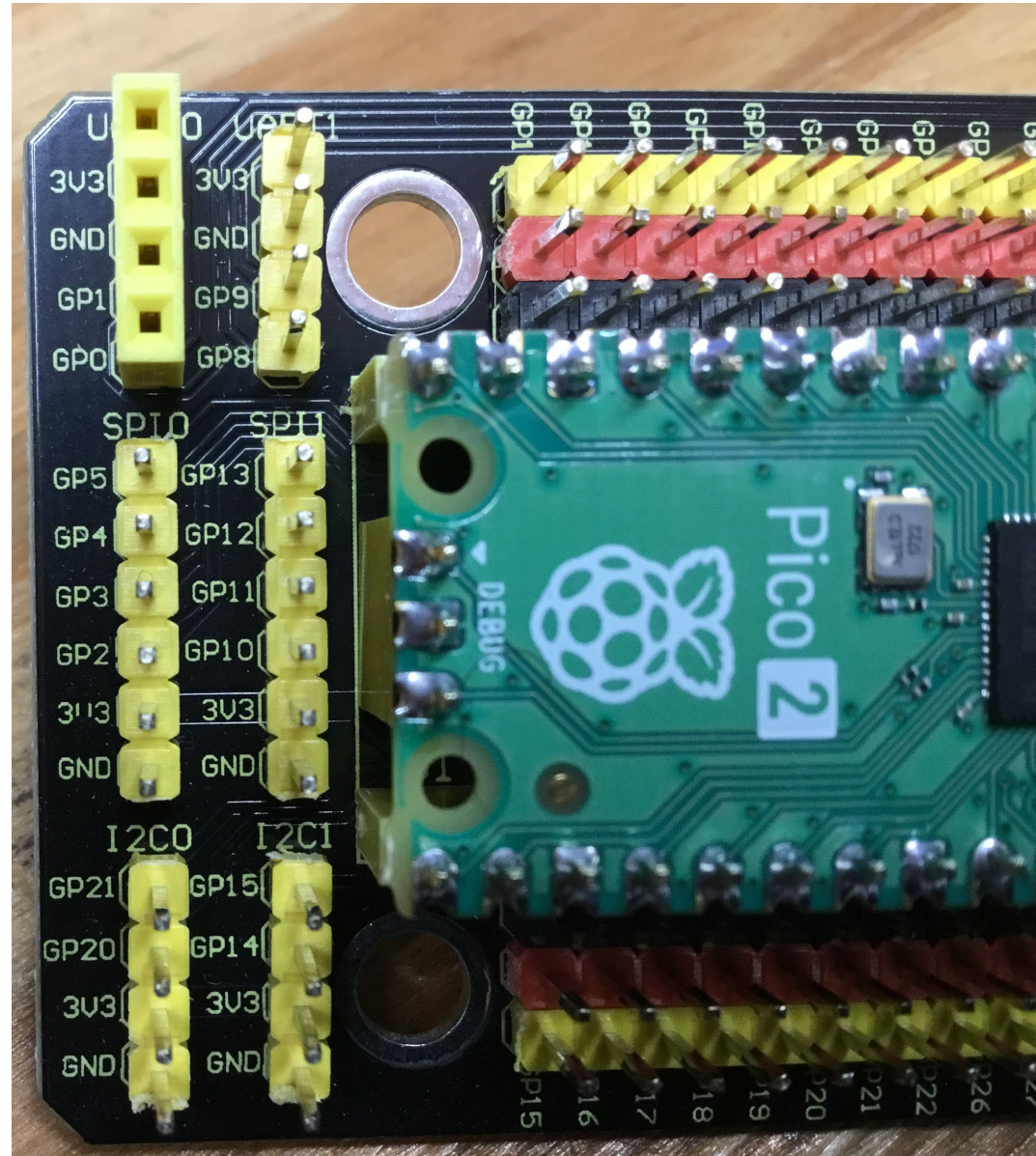
- **Pico 2 Development Hardware**
- **Raspberry Pi Debug Probe**
 - **The Hook Up**
- **Code a Pico 2 Ring Buffer Application**



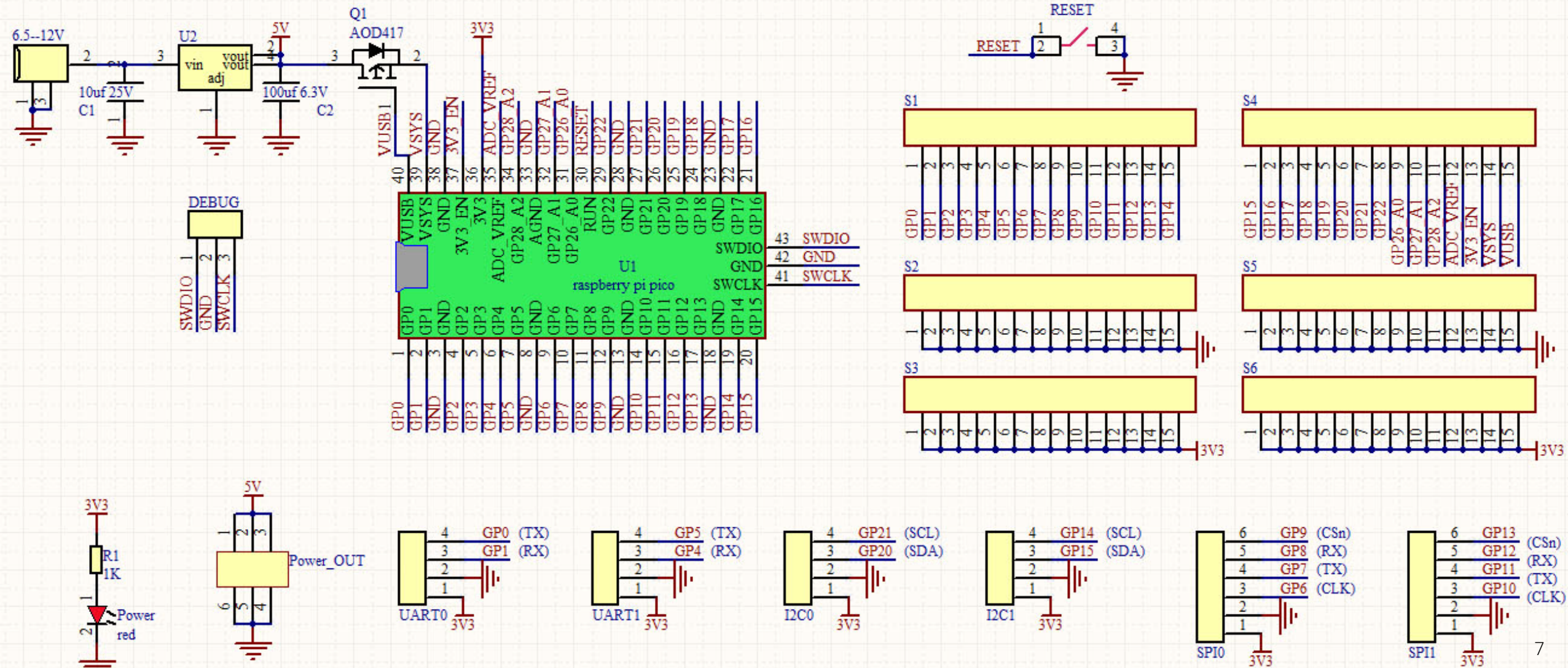
Pico 2 Carrier Board



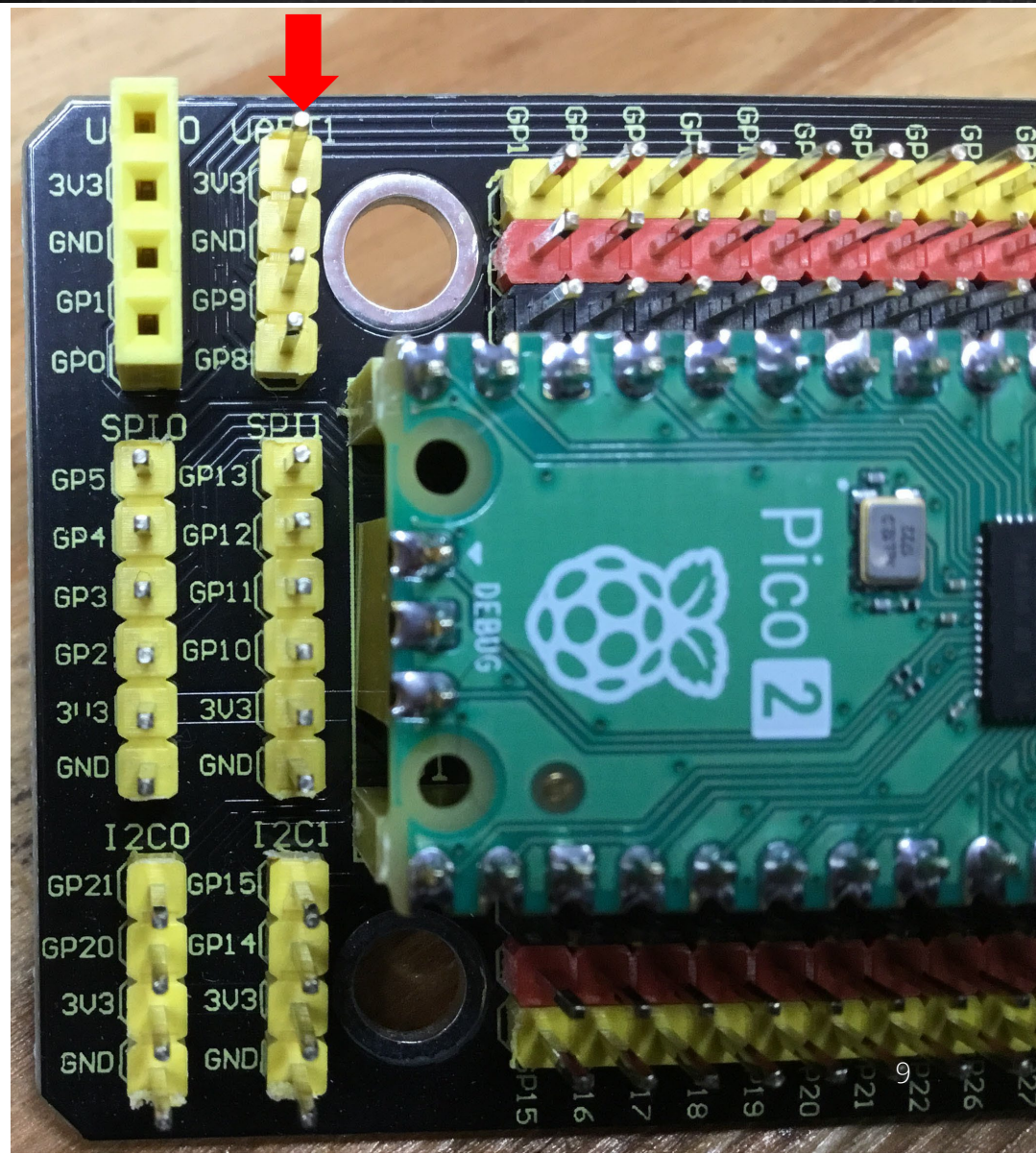
Pico 2 Carrier Board



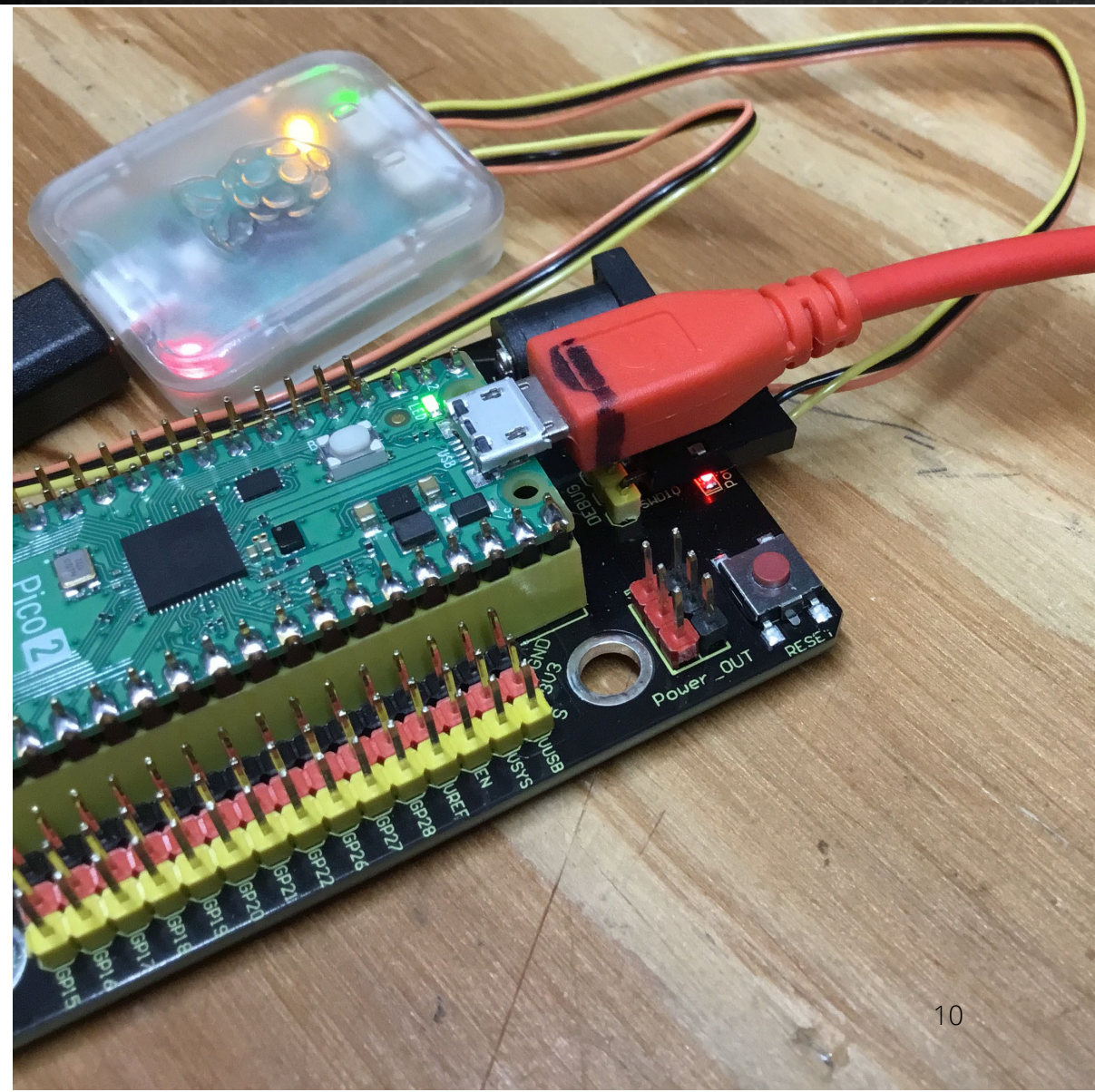
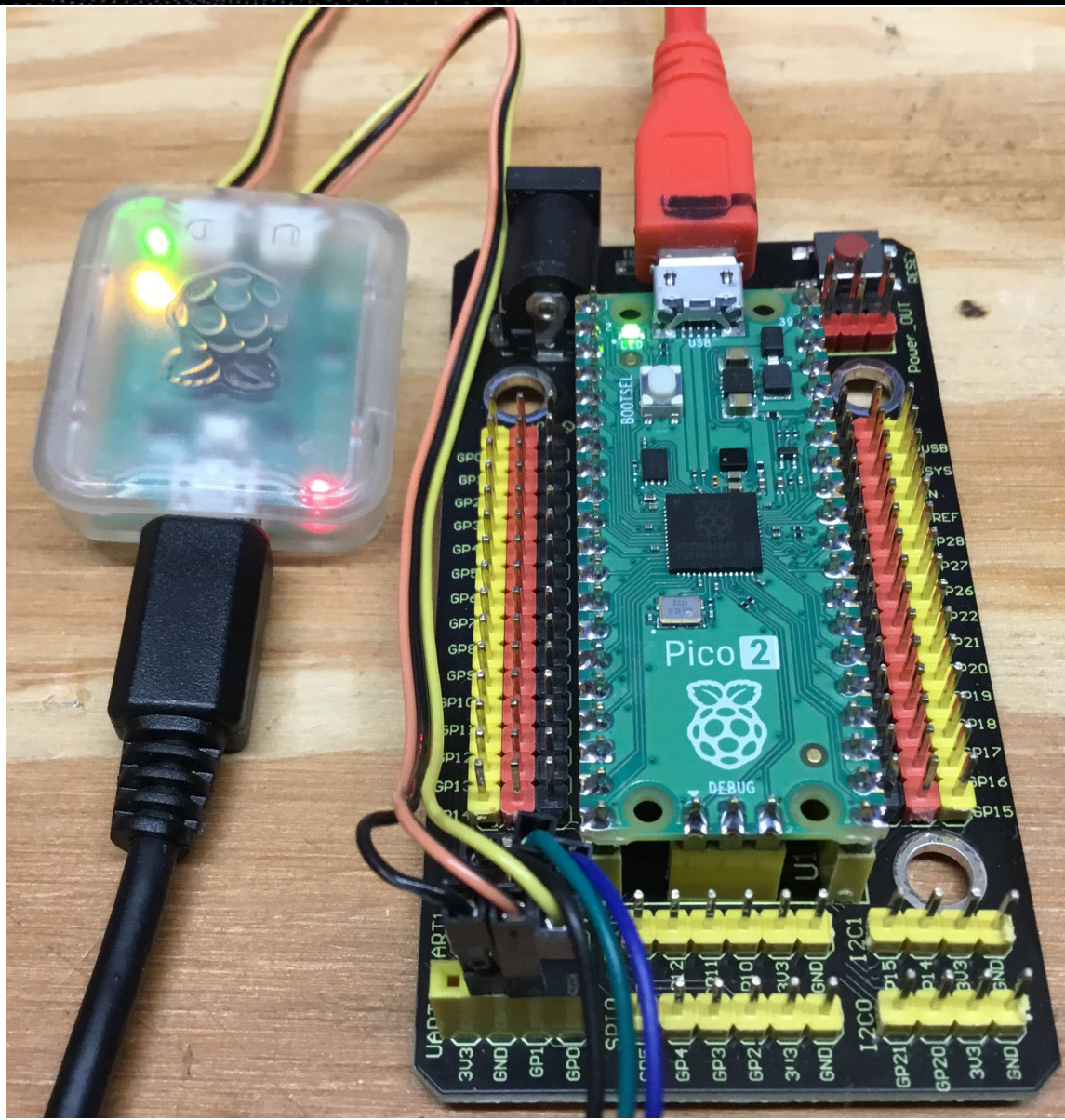
Pico 2 Carrier Board

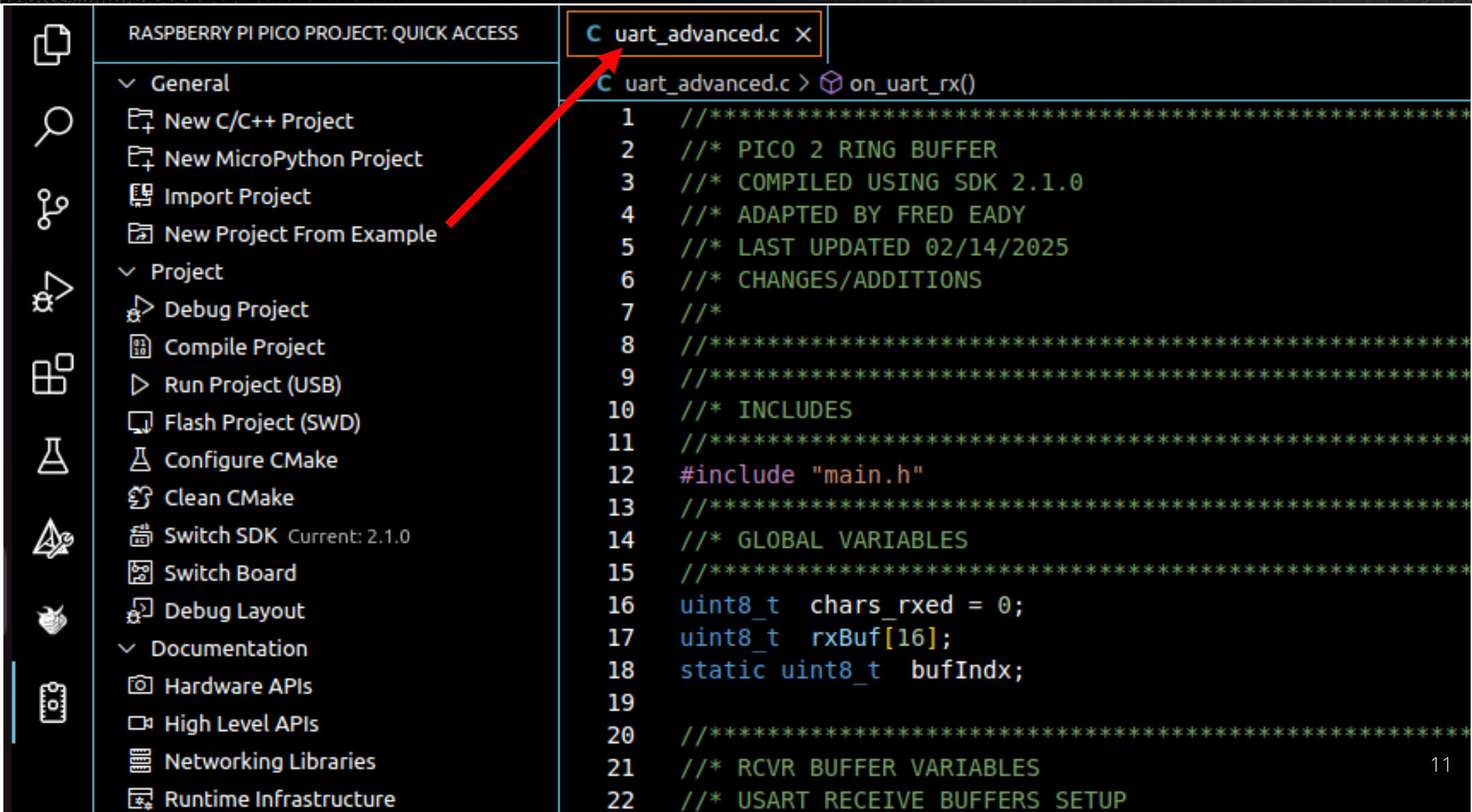


The Hook Up



The Hook Up



Modify the *uart_advanced.c* Example

RASPBERRY PI PICO PROJECT: QUICK ACCESS

- 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: 2.1.0
 - Switch Board
 - Debug Layout
- Documentation
 - Hardware APIs
 - High Level APIs
 - Networking Libraries
 - Runtime Infrastructure

C uart_advanced.c X

```
C uart_advanced.c > on_uart_rx()
1 //*****
2 /* PICO 2 RING BUFFER
3 /* COMPILED USING SDK 2.1.0
4 /* ADAPTED BY FRED EADY
5 /* LAST UPDATED 02/14/2025
6 /* CHANGES/ADDITIONS
7 /*
8 //*****
9 //*****
10 /* INCLUDES
11 //*****
12 #include "main.h"
13 //*****
14 /* GLOBAL VARIABLES
15 //*****
16 uint8_t chars_rxed = 0;
17 uint8_t rxBuf[16];
18 static uint8_t bufIndx;
19
20 //*****
21 /* RCVR BUFFER VARIABLES
22 /* USART RECEIVE BUFFERS SETUP
```

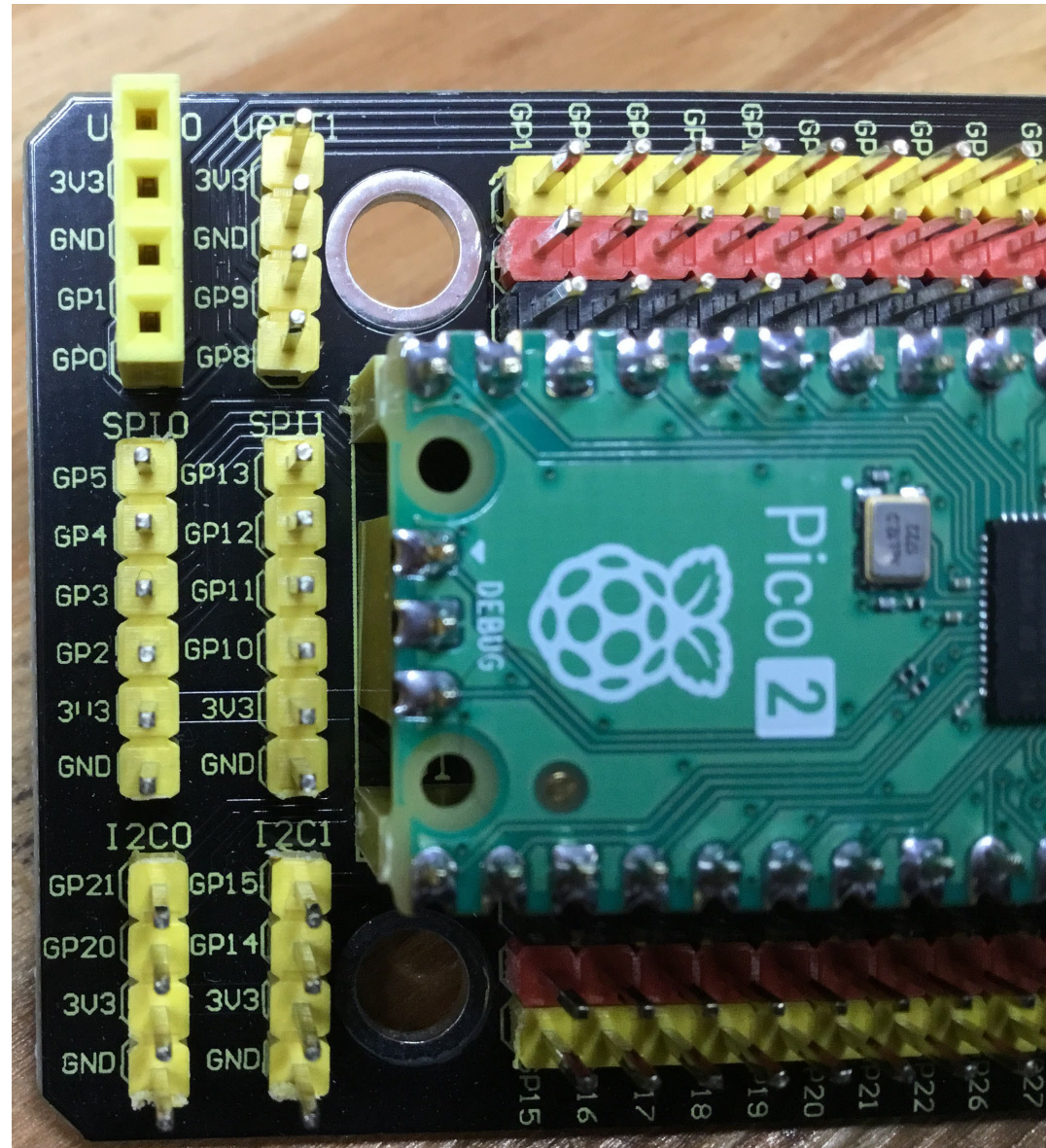
Modify the *uart_advanced.c* Example

```

//*****
/* PICO 2 RING BUFFER
/* COMPILED USING SDK 2.1.0
/* WRITTEN BY FRED EADY
/* LAST UPDATED 02/14/2025
/* CHANGES/ADDITIONS
/*
//*****
//*****
/* INCLUDES
//*****
#include "pico/stdlib.h"
#include "hardware/uart.h"
#include "hardware/irq.h"
//*****
/* FUNCTION PROTOTYPES
//*****
uint8_t readring(void);
void on_uart_rx(void);
//*****
/* UART DEFINITIONS
//*****
#define UART_ID uart1
#define BAUD_RATE 115200
#define DATA_BITS 8
#define STOP_BITS 1
#define PARITY    UART_PARITY_NONE

#define UART_TX_PIN 8
#define UART_RX_PIN 9
#define UART_RX_BUFFER_SIZE 8
#define UART_RX_BUFFER_MASK ( UART_RX_BUFFER_SIZE - 1 )

```



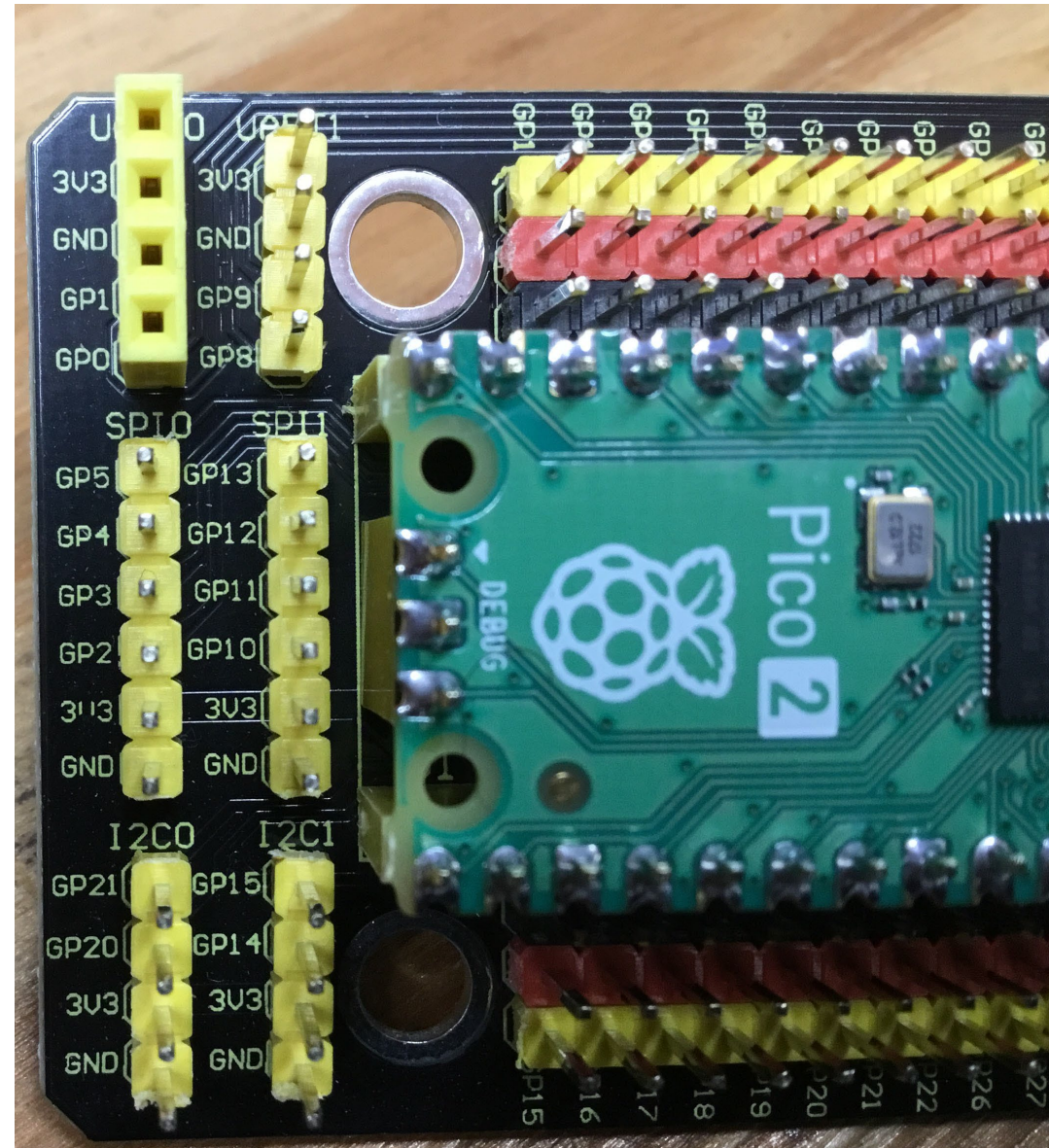
Modify the *uart_advanced.c* Example

```

//*****
/* RCVR BUFFER VARIABLES
/* USART RECEIVE BUFFERS SETUP
/* 1,2,4,8,16,32,64,128 or 256 BYTES
//*****
uint8_t  UART_RxHead;
uint8_t  UART_RxTail;
uint8_t  UART_RxBuf[UART_RX_BUFFER_SIZE];
uint8_t  data;
uint8_t  tmphead;
uint8_t  tmptail;

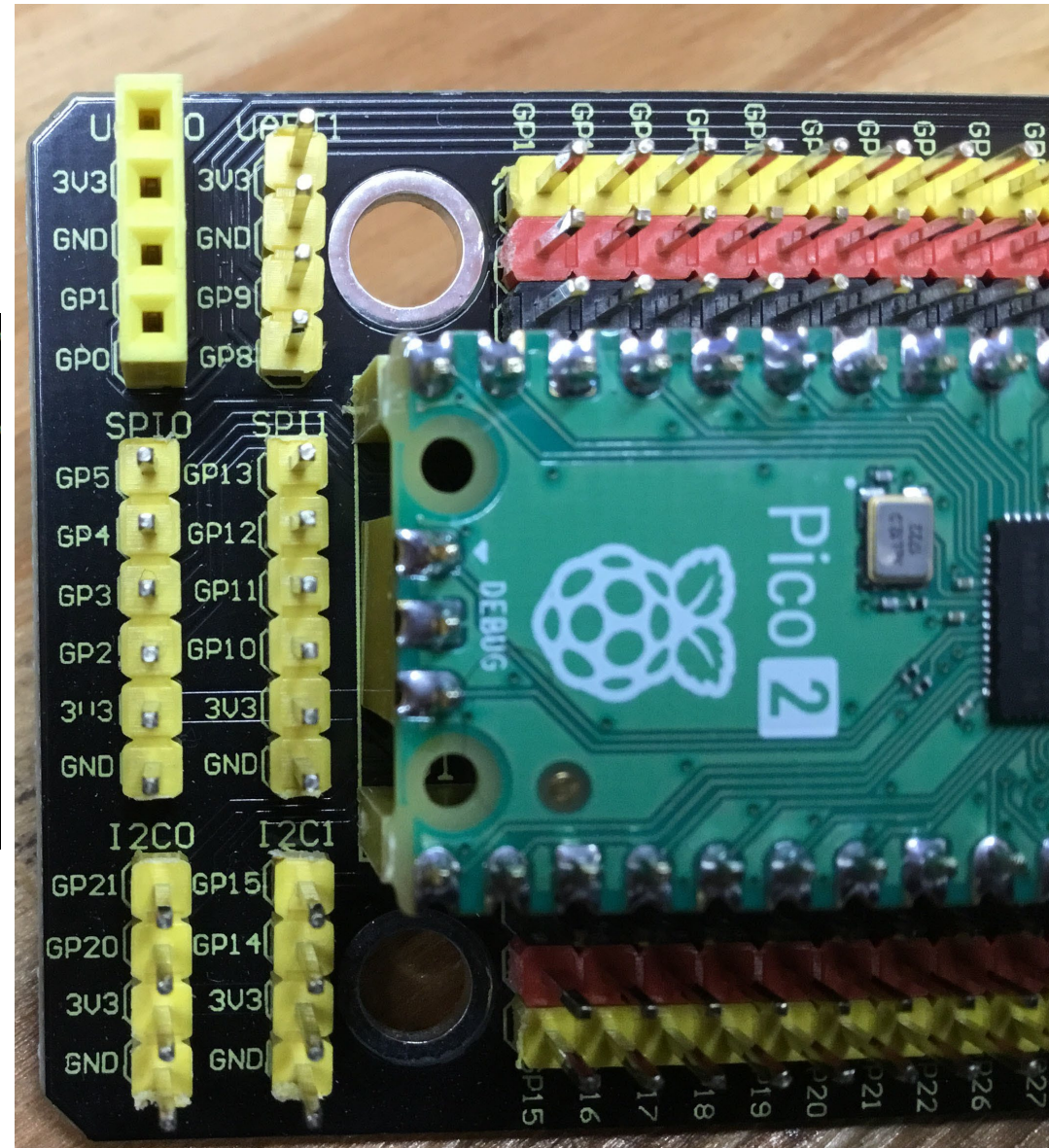
void on_uart_rx()
{
    while (uart_is_readable(UART_ID))
    {
        data = uart_getc(UART_ID);
        // calculate buffer index
        tmphead = ( UART_RxHead + 1 ) & UART_RX_BUFFER_MASK;
        // store new index
        UART_RxHead = tmphead;
        // store received data in ring buffer
        UART_RxBuf[tmphead] = data; //(uint8_t)(data & 0x000000FF);
        chars_rxed++;
        //uart_putc(UART_ID, UART_RxBuf[tmphead]);
    }
}

```



Modify the *uart_advanced.c* Example

```
/**
 * GET BYTE FROM RX RING BUFFER
 */
uint8_t readring(void)
{
    // calculate buffer index
    tmptail = ( UART_RxTail + 1 ) & UART_RX_BUFFER_MASK;
    // store new index
    UART_RxTail = tmptail;
    return UART_RxBuf[tmptail];
}
```



Modify the *uart_advanced.c* Example

```

int main() {
    // Set up our UART with a basic baud rate.
    uart_init(UART_ID, 2400);

    // Set the TX and RX pins by using the function select on the GPIO
    // Set datasheet for more information on function select
    gpio_set_function(UART_TX_PIN, UART_FUNCSEL_NUM(UART_ID, UART_TX_PIN));
    gpio_set_function(UART_RX_PIN, UART_FUNCSEL_NUM(UART_ID, UART_RX_PIN));

    // Actually, we want a different speed
    // The call will return the actual baud rate selected, which will be as close
    // possible to that requested
    int __unused actual = uart_set_baudrate(UART_ID, BAUD_RATE);

    // Set UART flow control CTS/RTS, we don't want these, so turn them off
    uart_set_hw_flow(UART_ID, false, false);

    // Set our data format
    uart_set_format(UART_ID, DATA_BITS, STOP_BITS, PARITY);

    // Turn off FIFO's - we want to do this character by character
    uart_set_fifo_enabled(UART_ID, false);

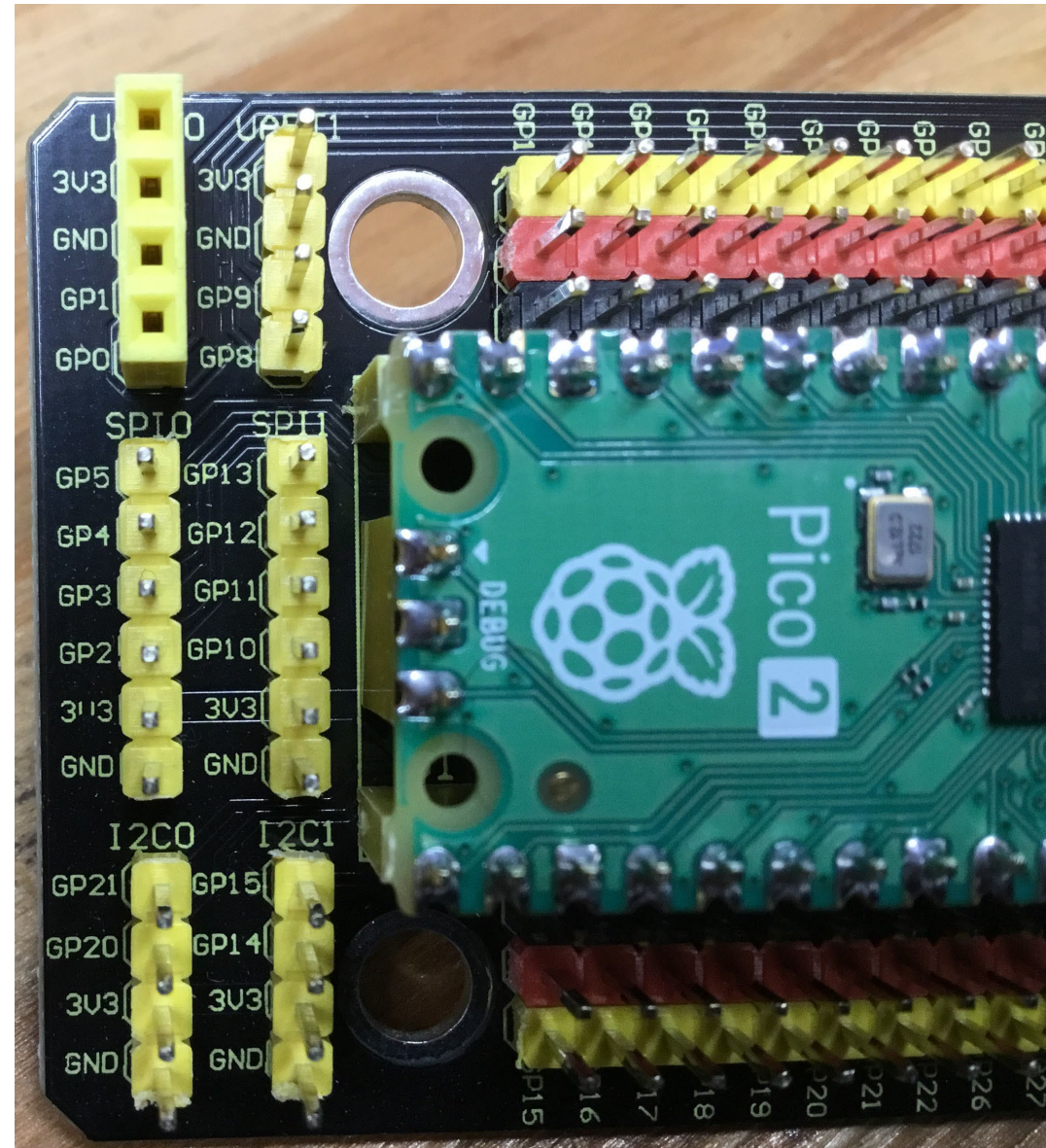
    // Set up a RX interrupt
    // We need to set up the handler first
    // Select correct interrupt for the UART we are using
    int UART_IRQ = UART_ID == uart0 ? UART0_IRQ : UART1_IRQ;

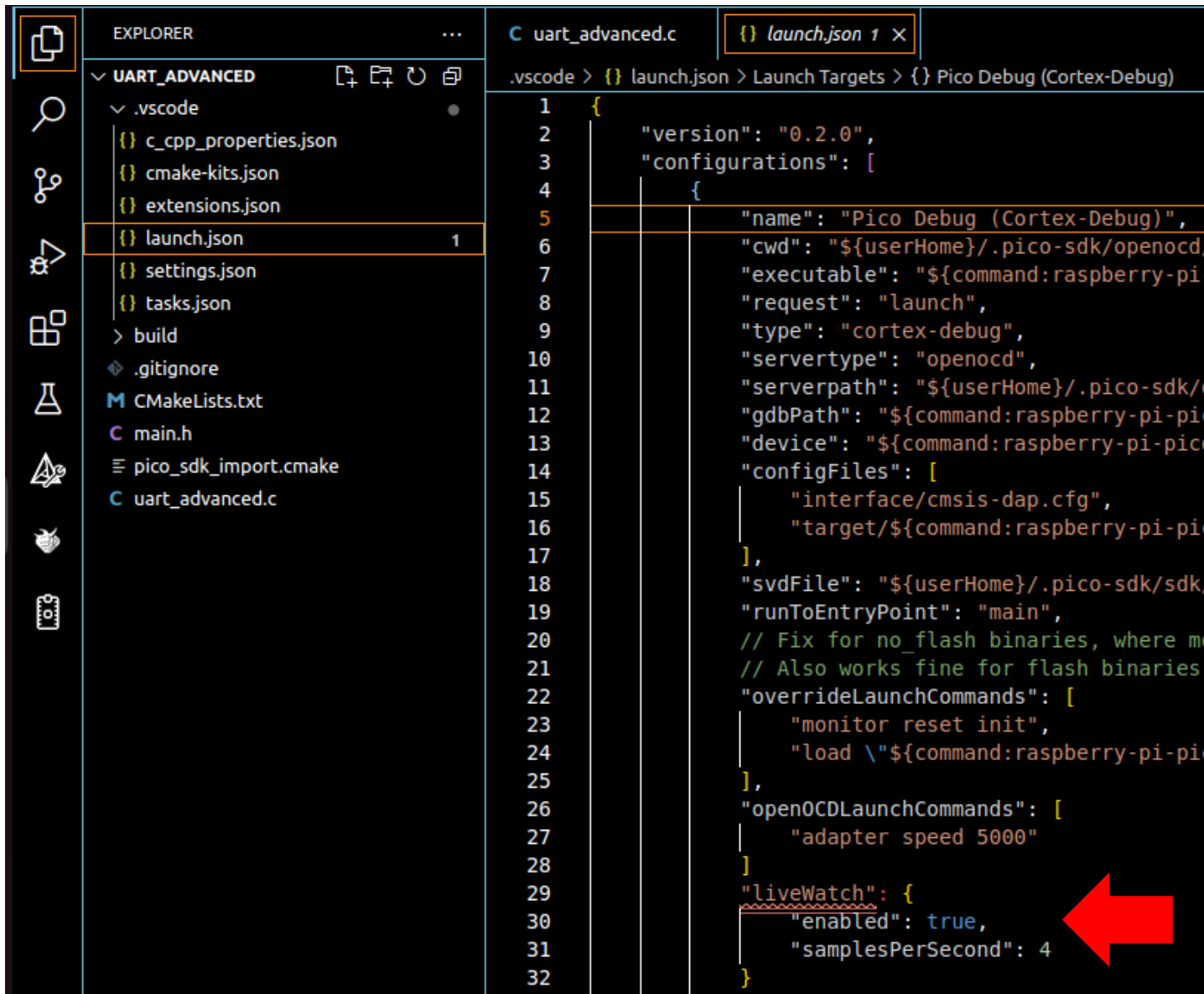
    // And set up and enable the interrupt handlers
    irq_set_exclusive_handler(UART_IRQ, on_uart_rx);
    irq_set_enabled(UART_IRQ, true);

    // Now enable the UART to send interrupts - RX only
    uart_set_irq_enables(UART_ID, true, false);

    // OK, all set up.
    uart_puts(UART_ID, "\nREADY\n");
}

```

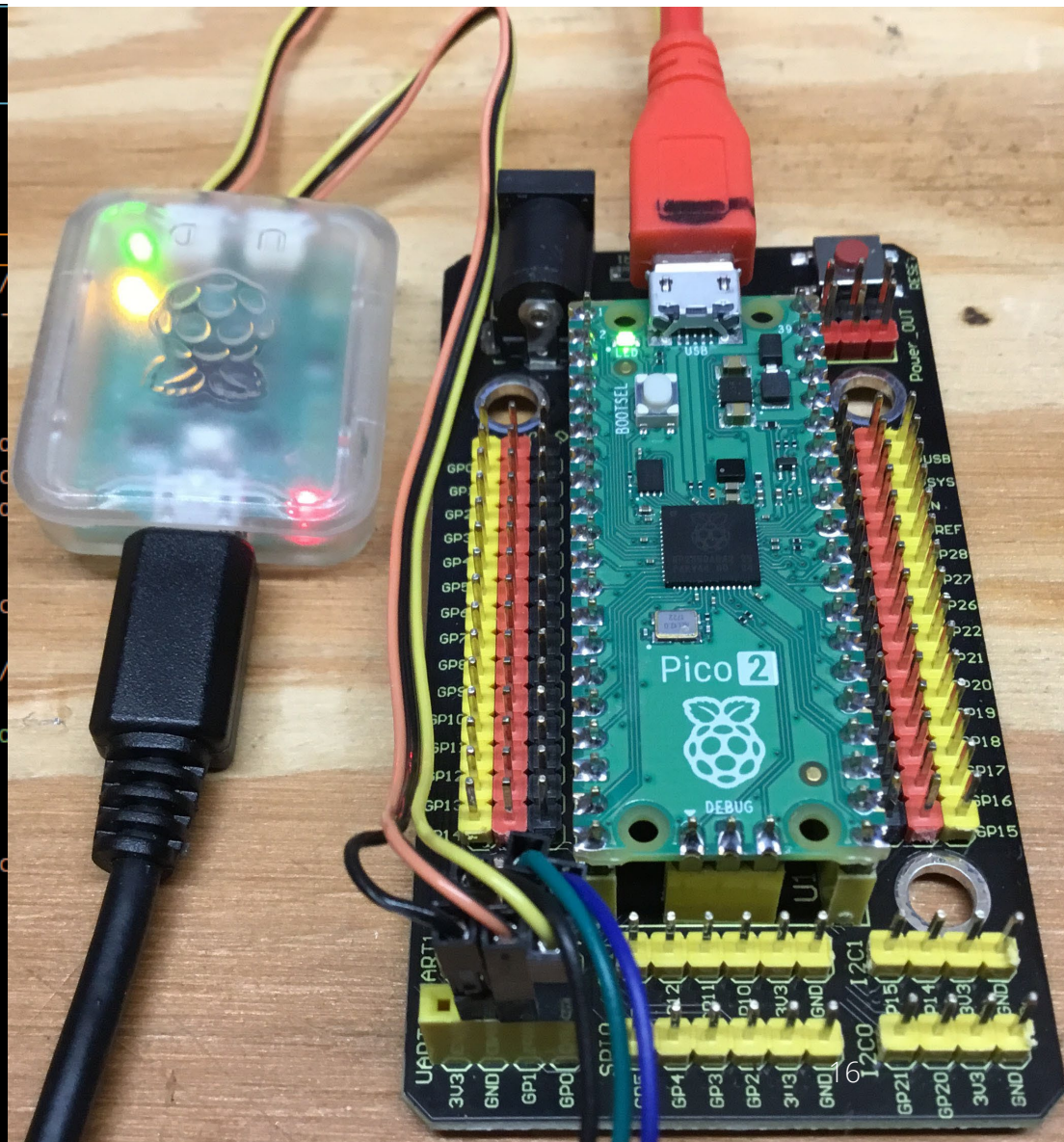


Enable *liveWatch*

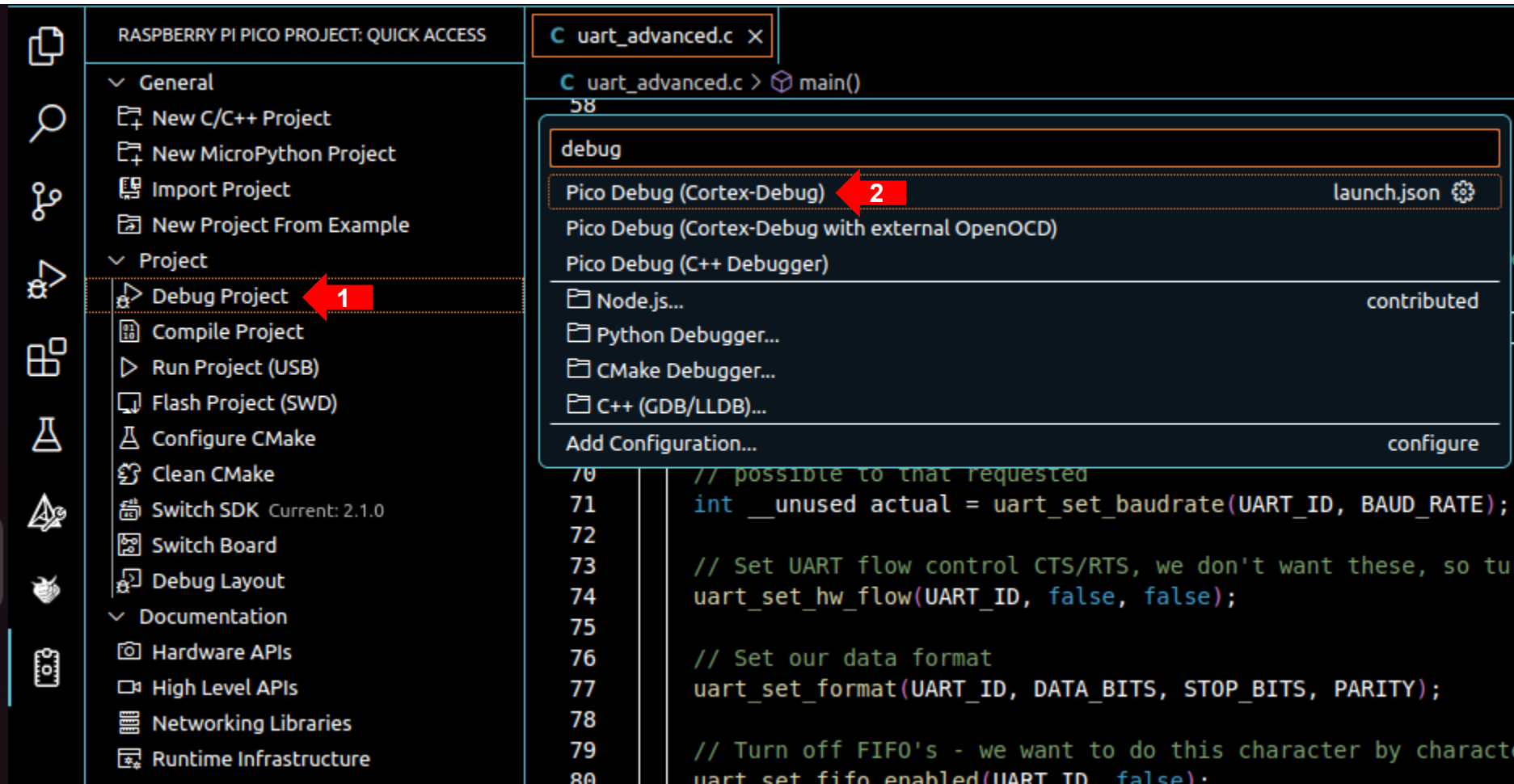
```
EXPLORER
├── .vscode
│   ├── c_cpp_properties.json
│   ├── cmake-kits.json
│   ├── extensions.json
│   ├── launch.json
│   ├── settings.json
│   └── tasks.json
├── build
├── .gitignore
├── CMakeLists.txt
├── main.h
├── pico_sdk_import.cmake
└── uart_advanced.c

C uart_advanced.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": "${userHome}/.pico-sdk/openocd/",
7              "executable": "${command:raspberry-pi-...",
8              "request": "launch",
9              "type": "cortex-debug",
10             "servertype": "openocd",
11             "serverpath": "${userHome}/.pico-sdk/o...",
12             "gdbPath": "${command:raspberry-pi-pi...",
13             "device": "${command:raspberry-pi-pi...",
14             "configFiles": [
15                 "interface/cmsis-dap.cfg",
16                 "target/${command:raspberry-pi-pi...",
17             ],
18             "svdFile": "${userHome}/.pico-sdk/sdk/...",
19             "runToEntryPoint": "main",
20             // Fix for no_flash binaries, where mo...
21             // Also works fine for flash binaries
22             "overrideLaunchCommands": [
23                 "monitor reset init",
24                 "load \\\"${command:raspberry-pi-pi...",
25             ],
26             "openOCDLaunchCommands": [
27                 "adapter speed 5000"
28             ]
29             "liveWatch": {
30                 "enabled": true,
31                 "samplesPerSecond": 4
32             }
33         }
34     ]
35 }
```

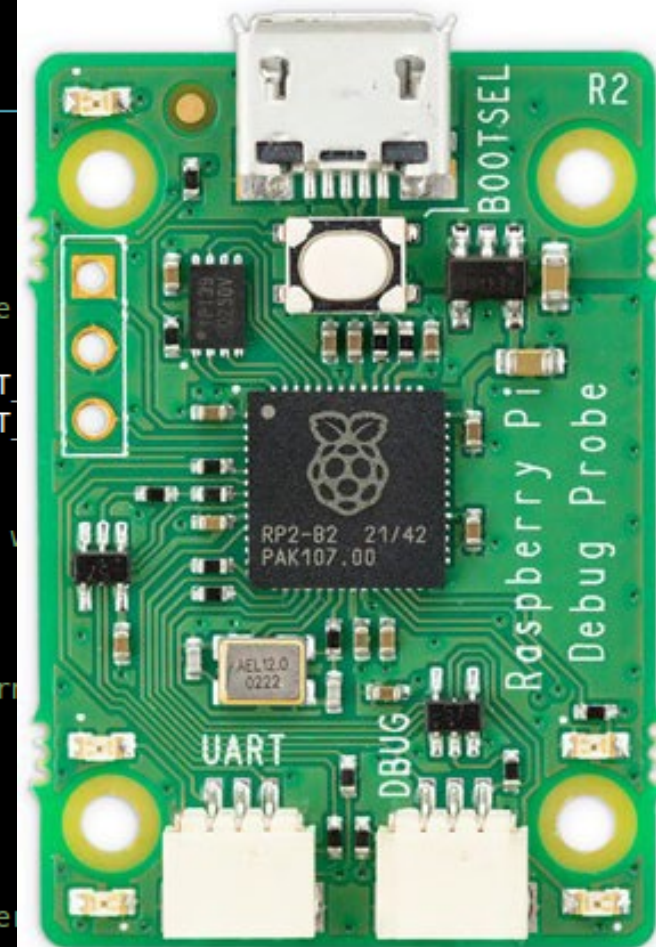


Debug the Pico 2 Ring Buffer Application



The screenshot shows the Visual Studio Code interface for a Raspberry Pi Pico project. The left sidebar displays the 'Project' menu with 'Debug Project' highlighted by a red arrow labeled '1'. The main editor area shows the 'uart_advanced.c' file with a dropdown menu for debug configurations. 'Pico Debug (Cortex-Debug)' is selected, highlighted by a red arrow labeled '2'. The code in the editor includes comments and function calls for setting UART parameters.

```
C uart_advanced.c > main()
58
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
70 // possible to that requested
71 int __unused actual = uart_set_baudrate(UART_ID, BAUD_RATE);
72
73 // Set UART flow control CTS/RTS, we don't want these, so turn
74 uart_set_hw_flow(UART_ID, false, false);
75
76 // Set our data format
77 uart_set_format(UART_ID, DATA_BITS, STOP_BITS, PARITY);
78
79 // Turn off FIFO's - we want to do this character by character
80 uart_set_fifo_enabled(UART_ID, false);
```



Debug the Pico 2 Ring Buffer Application

Visual Studio Code interface showing the debug console for a Raspberry Pi Pico 2. The left sidebar displays the **CORTX LIVE WATCH** window, showing memory addresses and values for `UART_RxBuf` and `rxBuf`. The main editor displays the C code for `uart_advanced.c`, showing the `readring` function. The bottom panel shows the **SERIAL MONITOR** window, which is currently empty, indicating that the serial port has been opened but no data has been received yet.

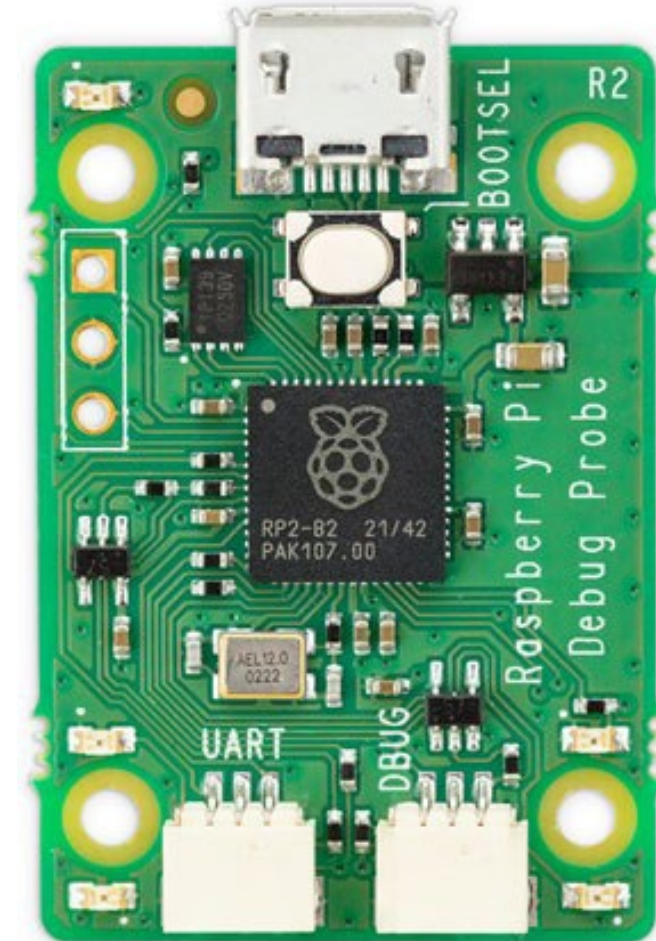
```
C uart_advanced.c > main()
32 void on_uart_rx()
44     | | | //uart_putc(UART_ID, UART_RxBuf[tmphead]);
45 }
46 }
47 //*****
48 /* GET BYTE FROM RX RING BUFFER
49 //*****
50 uint8_t readring(void)
51 {
52     // calculate buffer index
53     tmptail = ( UART_RxTail + 1 ) & UART_RX_BUFFER_MASK;
54     // store new index
55     UART_RxTail = tmptail;
56     return UART_RxBuf[tmptail];
57 }
58
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY SERIAL MONITOR

+ Open an additional monitor

Monitor Mode View Mode Port Baud rate

---- Opened the serial port /dev/ttyUSB0 ----



Debug the Pico 2 Ring Buffer Application

Visual Studio Code interface showing the debug console for a Raspberry Pi Pico 2. The code in the background is 'uart_advanced.c' with a ring buffer implementation. The SERIAL MONITOR shows the output 'READY' and '---- Sent utf8 encoded message: "12345678" ----'.

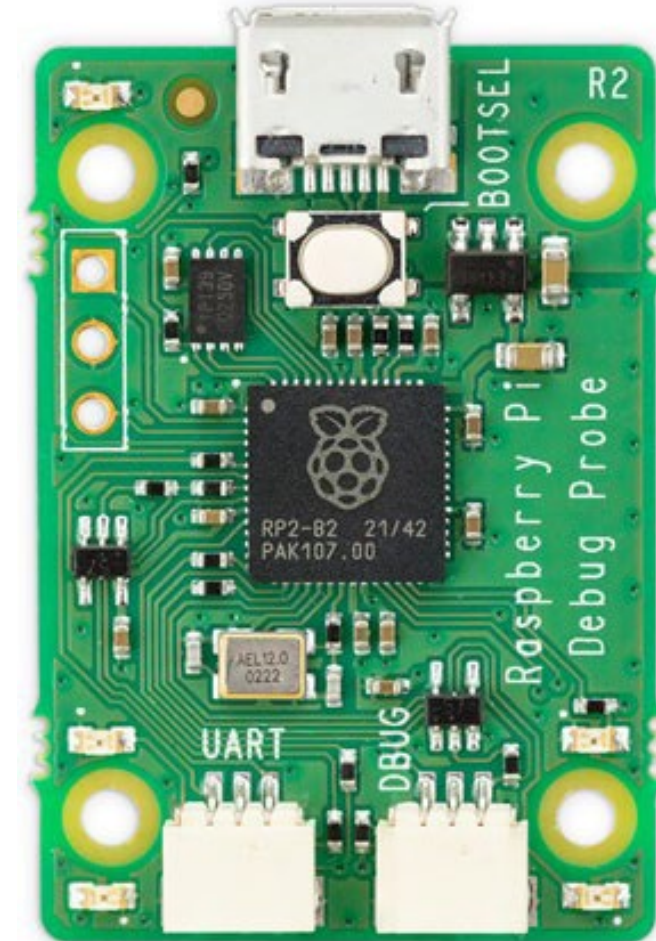
```
C uart_advanced.c > main()
50 uint8_t readring(void)
54 // store new index
55 UART_RxTail = tmptail;
56 return UART_RxBuf[tmptail];
57 }
58
59 int main() {
60 // Set up our UART with a basic baud rate.
61 uart_init(UART_ID, 2400);
62
63 // Set the TX and RX pins by using the function select on the GPIO
64 // Set datasheet for more information on function select
65 gpio_set_function(UART_TX_PIN, UART_FUNCSEL_NUM(UART_ID, UART_TX_PIN));
66 gpio_set_function(UART_RX_PIN, UART_FUNCSEL_NUM(UART_ID, UART_RX_PIN));
67
68 // Actually, we want a different speed
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY SERIAL MONITOR

+ Open an additional monitor

Monitor Mode Serial View Mode Text Port /dev/ttyUSB0 - FTDI Baud rate 115200

READY
---- Sent utf8 encoded message: "12345678" ----

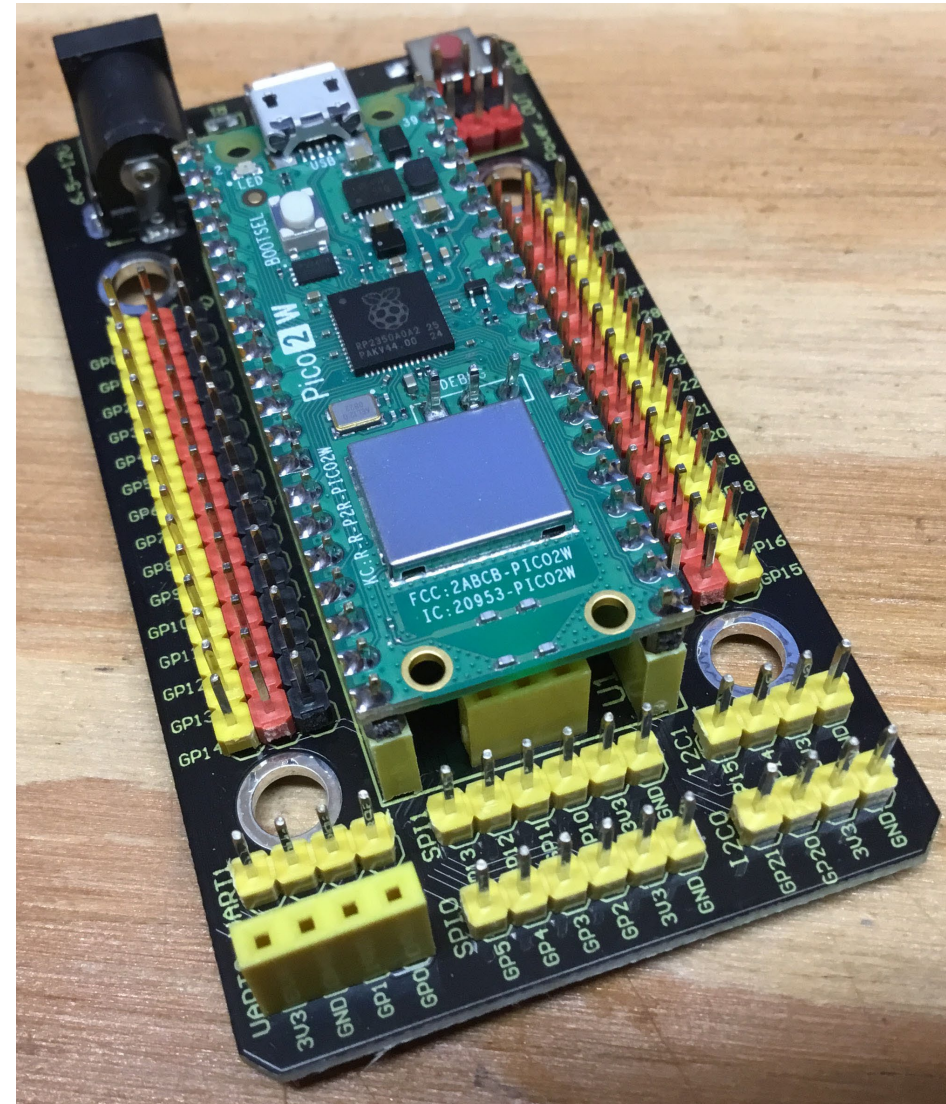


Next Time...**MORE TO COME..**

Thank you for attending!!!

Please consider the resources below:

- [Today's Download Package](#)
- raspberrypi.org





DesignNews

Thank You

Sponsored by

DigiKey

