CEC Continuing Education Center

DesignNews

Scratch Building Raspberry Pi RP2040 IoT Devices

Day 3:

RP2040 Firmware Generation Using CMake, Visual Studio Code and the C/C++ SDK

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

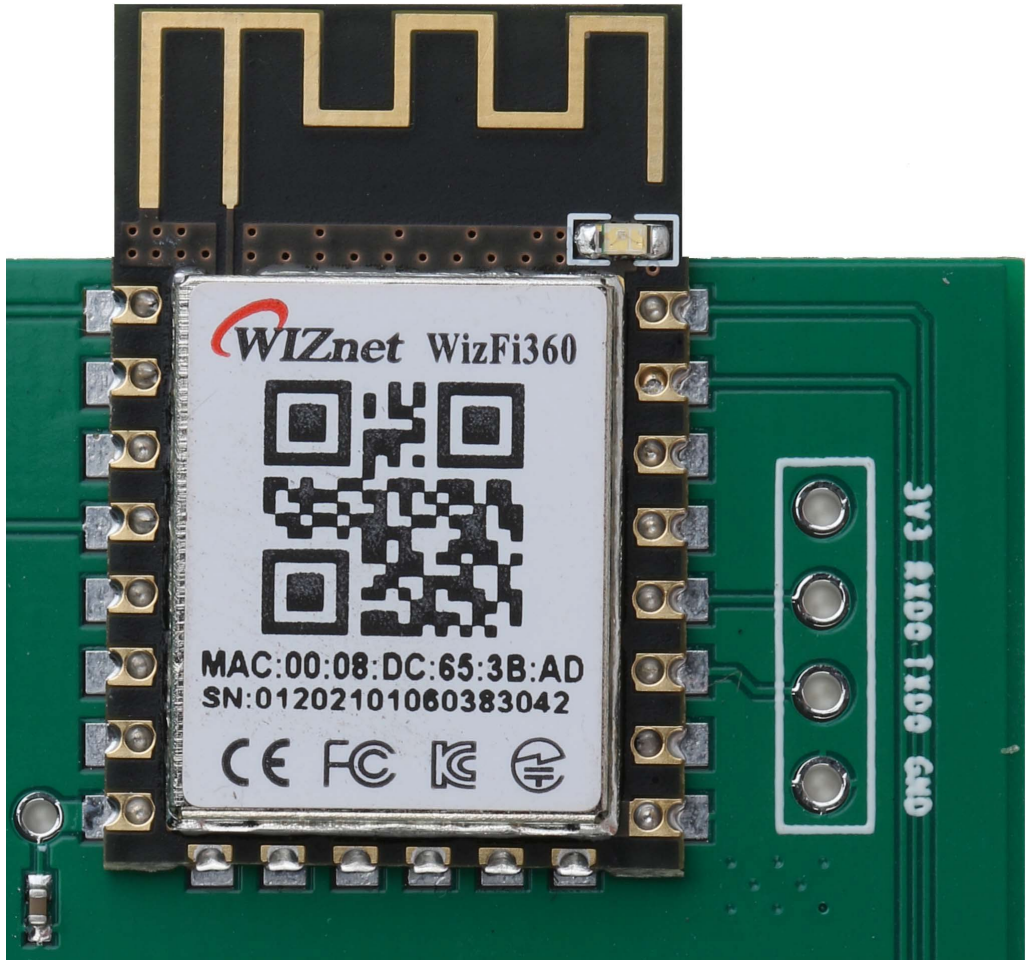
Visit ‘Lecturer Profile’ in your console for more details.

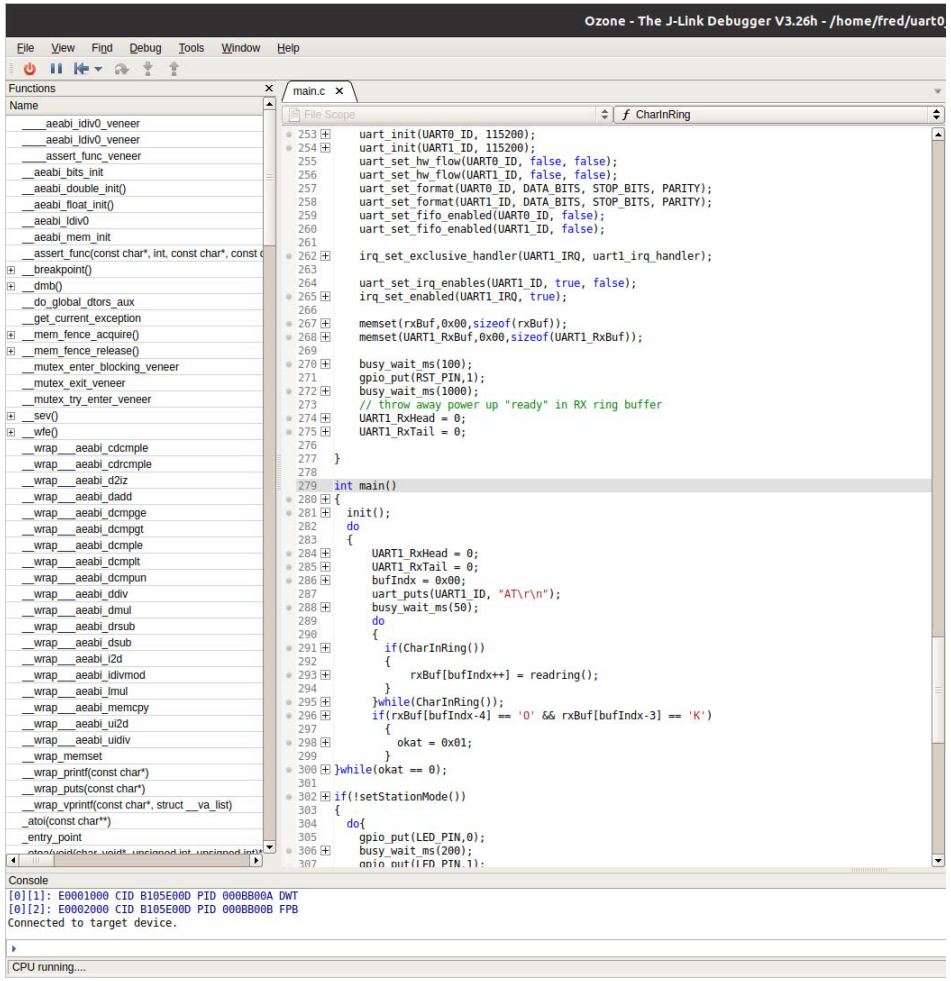
CEC Continuing Education Center

Sponsored By Digi-Key Corporation

AGENDA

- **Code an RP2040 WizFi360 Driver**
- **A Quick and Nasty Look at the SEGGER Ozone Debugger**





Generate CMakeLists.txt

M CMakeLists.txt

```
# Set name of project (as PROJECT_NAME) and C/C++ standards
project(rp2040_uart0 C CXX ASM)

set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)

# Creates a pico-sdk subdirectory in our project for the libraries
pico_sdk_init()

# Tell CMake where to find the executable source file
add_executable(${PROJECT_NAME}
    main.c
)

# Create map/bin/hex/uf2 files
pico_add_extra_outputs(${PROJECT_NAME})

# Link to pico_stdlib (gpio, time, etc. functions)
target_link_libraries(${PROJECT_NAME}
    pico_stdlib
)
```

Generate main.h

main.h > ...

```c
//*****************************************************************
//* RP2040 WizFi360 DRIVER
//* VERSION 1.00 A
//* WRITTEN BY FRED EADY
//* LAST UPDATED 07/23/2021
//* CHANGES/ADDITIONS
//*****************************************************************

//* Define to prevent recursive inclusion
#ifndef __main_H
#define __main_H

#ifdef __cplusplus
extern "C" {
#endif

#include "pico/stdlib.h"
#include "hardware/uart.h"
#include "hardware/irq.h"
#include <string.h>

void init(void);
uint8_t CharInRing(void);
uint8_t readring(void);
void uart1_irq_handler(void);
uint8_t chk_atok(void);
uint8_t setStationMode(void);
uint8_t setSingleConnectionMode(void);
uint8_t setDhcpEnable(void);
uint8_t connect2AP(void);
uint8_t connect2Server(void);
uint8_t send2Server(void);

#ifdef __cplusplus
}
#endif
#endif /* __main_H */
```

Generate main.c – Definitions and Aliases

C main.c > ...

```c
//*****************************************************************************
//* PICO RP2040 WizFi360 DRIVER
//* VERSION 1.00 A TCP CLIENT
//* COMPILED USING GCC
//* WRITTEN BY FRED EADY
//* LAST UPDATED 07/24/2022
//* CHANGES/ADDITIONS
//*****************************************************************************
//*****************************************************************************
//* INCLUDES
//*****************************************************************************
#include "main.h"
//*****************************************************************************
//* DEFINITIONS AND ALIASES
//*****************************************************************************
#define UART0_ID  uart0
#define UART1_ID  uart1
#define BAUD_RATE 115200
#define DATA_BITS 8
#define STOP_BITS 1
#define PARITY    UART_PARITY_NONE

// define uart pins
#define UART0_TX_PIN 28
#define UART0_RX_PIN 29
#define UART1_TX_PIN 4
#define UART1_RX_PIN 5
#define WP_PIN       10
#define RST_PIN      11
#define LED_PIN      24
```

Generate main.c – Variables

C main.c > ...

```c
//*****************************************************************************
//* VARIABLES
//*****************************************************************************
uint8_t  scratch8;
uint8_t  rxBuf[256];
uint16_t bufIndx;
uint8_t  okat;
uint8_t  rc;
//*****************************************************************************
//* RCVR BUFFER VARIABLES
//* USART RECEIVE BUFFERS SETUP
//* 1,2,4,8,16,32,64,128 or 256 BYTES
//*****************************************************************************
#define UART1_RX_BUFFER_SIZE  256
#define UART1_RX_BUFFER_MASK ( UART1_RX_BUFFER_SIZE - 1 )

uint8_t   UART1_RxHead;
uint8_t   UART1_RxTail;
uint8_t   UART1_RxBuf[UART1_RX_BUFFER_SIZE];
uint8_t   data;
uint8_t   tmphead;
uint8_t   tmptail;
uint32_t  uart1Error;
```

Generate main.c – UART1 RX Interrupt Handler

main.c > ...

```c


//*****************************************************************************
//* UART1 RX INTERRUPT HANDLER
//*****************************************************************************
void uart1_irq_handler(void)
{
    while (uart_is_readable(UART1_ID))
    {
        data = uart_getc(UART1_ID);
        // calculate buffer index
        tmphead = ( UART1_RxHead + 1 ) & UART1_RX_BUFFER_MASK;
        // store new index
        UART1_RxHead = tmphead;
        // store received data in ring buffer
        UART1_RxBuf[tmphead] = data;//(uint8_t)(data & 0x000000FF);
        // uncomment for debugging
        uart_putc(UART0_ID,UART1_RxBuf[tmphead]);
    }
}
```

Generate main.c – UART1 Ring Buffer Functions

main.c > ...

```c
//*****************************************************************************
//*   CHECK FOR CHARACTER IN RING
//*****************************************************************************
uint8_t CharInRing(void)
{
  return(UART1_RxHead != UART1_RxTail);
}
//*****************************************************************************
//*   GET BYTE FROM RX RING BUFFER
//*****************************************************************************
uint8_t readring(void)
{
  // calculate buffer index
  tmptail = ( UART1_RxTail + 1 ) & UART1_RX_BUFFER_MASK;
  // store new index
  UART1_RxTail = tmptail;
  return UART1_RxBuf[tmptail];
}
```

Generate main.c – Check for OK Function

C main.c > ...

```c

//*****************************************************************************
//* CHECK FOR OK
//*****************************************************************************
uint8_t chk_atok(void)
{
  rc = 0;
  gpio_put(LED_PIN,0);
  busy_wait_ms(1000);
  do{
    if(CharInRing())
    {
      rxBuf[bufIndx++] = readring();
    }
  }while(CharInRing());
  if(rxBuf[bufIndx-4] == 'O' && rxBuf[bufIndx-3] == 'K')
  {
    rc = 1;
  }
  return rc;
}
```

Generate main.c – Initialize Function

main.c > ...

```c
//*****************************************************************************
//* INITIALIZE FUNCTION
//*****************************************************************************
void init(void)
{
    gpio_init(LED_PIN);
    gpio_init(WP_PIN);
    gpio_init(RST_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);
    gpio_set_dir(WP_PIN, GPIO_OUT);
    gpio_set_dir(RST_PIN,GPIO_OUT);
    gpio_put(LED_PIN,0);
    gpio_put(RST_PIN,0);
    gpio_put(WP_PIN,0);

    gpio_set_function(UART0_TX_PIN, GPIO_FUNC_UART);
    gpio_set_function(UART0_RX_PIN, GPIO_FUNC_UART);
    gpio_set_function(UART1_TX_PIN, GPIO_FUNC_UART);
    gpio_set_function(UART1_RX_PIN, GPIO_FUNC_UART);

    uart_init(UART0_ID, 115200);
    uart_init(UART1_ID, 115200);
    uart_set_hw_flow(UART0_ID, false, false);
    uart_set_hw_flow(UART1_ID, false, false);
    uart_set_format(UART0_ID, DATA_BITS, STOP_BITS, PARITY);
    uart_set_format(UART1_ID, DATA_BITS, STOP_BITS, PARITY);
    uart_set_fifo_enabled(UART0_ID, false);
    uart_set_fifo_enabled(UART1_ID, false);

    irq_set_exclusive_handler(UART1_IRQ, uart1_irq_handler);

    uart_set_irq_enables(UART1_ID, true, false);
    irq_set_enabled(UART1_IRQ, true);

    memset(rxBuf,0x00,sizeof(rxBuf));
    memset(UART1_RxBuf,0x00,sizeof(UART1_RxBuf));

    busy_wait_ms(100);
    gpio_put(RST_PIN,1);
    busy_wait_ms(1000);
    // throw away power up "ready" in RX ring buffer
    UART1_RxHead = 0;
    UART1_RxTail = 0;

}
```

Generate main.c – Typical Command Function Call

main.c > main()

```c
if(!setStationMode())
{
    do{
        gpio_put(LED_PIN,0);
        busy_wait_ms(200);
        gpio_put(LED_PIN,1);
        busy_wait_ms(200);
    }while(1);
}
```

WizFi360 Driver Command Flow – WizFi360 Wake Up

Serial Input/Output Monitor

```
ready
AT

OK
AT+CWMODE_CUR=1

OK
AT+CIPMUX=0

OK
AT+CWDHCP_CUR=1,1

OK
AT+CWJAP_CUR="edtpnet2","    password    "
WIFI CONNECTED
WIFI GOT IP

OK
AT+CIPSTART="TCP","192.168.1.235",8088
CONNECT

OK
AT+CIPSEND=10

OK
>
Recv 10 bytes

SEND OK

+IPD,10:0123456789
```

```
53 45 4E 44 20 4F 4B 0D 0A
0D 0A
2B 49 50 44 2C 31 30 3A 30 31 32 33 34 35 36 37 38 39
```

ASCII Send

HEX Send

COM6 8N1 115200 R 0 C 0 R34 C19 Disconnect

main.c > main()

```c
int main()
{
  init();
  do
  {
      UART1_RxHead = 0;
      UART1_RxTail = 0;
      bufIndx = 0x00;
      uart_puts(UART1_ID, "AT\r\n");
      busy_wait_ms(50);
      do
      {
        if(CharInRing())
        {
          rxBuf[bufIndx++] = readring();
        }
      }while(CharInRing());
      if(rxBuf[bufIndx-4] == 'O' && rxBuf[bufIndx-3] == 'K')
        {
          okat = 0x01;
        }
  }while(okat == 0);
```

WizFi360 Driver Command Flow – Set Station Mode

Serial Input/Output Monitor

File Edit View Configuration

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

```
ready
AT

OK
AT+CWMODE_CUR=1

OK
AT+CIPMUX=0

OK
AT+CWDHCP_CUR=1,1

OK
AT+CWJAP_CUR="edtpnet2","password"
WIFI CONNECTED
WIFI GOT IP

OK
AT+CIPSTART="TCP","192.168.1.235",8088
CONNECT

OK
AT+CIPSEND=10

OK
>
Recv 10 bytes

SEND OK

+IPD,10:0123456789
```

```
53 45 4E 44 20 4F 4B 0D 0A
0D 0A
2B 49 50 44 2C 31 30 3A 30 31 32 33 34 35 36 37 38 39
```

ASCII [ ] Send
HEX [ ] Send
COM6 8N1 115200 | R 0 C 0 | R34 C19 | Disconnect

C main.c > ...

```c
//*****************************************************************************
//* SET STATION MODE
//*****************************************************************************
uint8_t setStationMode(void)
{
  rc = 0;
  UART1_RxHead = 0;
  UART1_RxTail = 0;
  uart_puts(UART1_ID,"AT+CWMODE_CUR=1\r\n");
  if(chk_atok())
  {
    rc = 1;
  }
  return rc;
}
```

WizFi360 Driver Command Flow – Set Connection Mode

Serial Input/Output Monitor

```
ready
AT

OK
AT+CWMODE_CUR=1

OK
AT+CIPMUX=0

OK
AT+CWDHCP_CUR=1,1

OK
AT+CWJAP_CUR="edtpnet2","  password  "
WIFI CONNECTED
WIFI GOT IP

OK
AT+CIPSTART="TCP","192.168.1.235",8088
CONNECT

OK
AT+CIPSEND=10

OK
>
Recv 10 bytes

SEND OK

+IPD,10:0123456789
```

```
53 45 4E 44 20 4F 4B 0D 0A
0D 0A
2B 49 50 44 2C 31 30 3A 30 31 32 33 34 35 36 37 38 39
```

ASCII Send
HEX Send
COM6 8N1 115200 R 0 C 0 R34 C19 Disconnect

C main.c > ...

```c
//*****************************************************************************
//* SET SINGLE CONNECTION MODE
//*****************************************************************************
uint8_t setSingleConnectionMode(void)
{
  rc = 0;
  UART1_RxHead = 0;
  UART1_RxTail = 0;
  uart_puts(UART1_ID,"AT+CIPMUX=0\r\n");
  if(chk_atok())
  {
    rc = 1;
  }
  return rc;
}
```

WizFi360 Driver Command Flow – Enable DHCP

Serial Input/Output Monitor

File Edit View Configuration

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

```
ready
AT

OK
AT+CWMODE_CUR=1

OK
AT+CIPMUX=0

OK
AT+CWDHCP_CUR=1,1

OK
AT+CWJAP_CUR="edtpnet2","  password  "
WIFI CONNECTED
WIFI GOT IP

OK
AT+CIPSTART="TCP","192.168.1.235",8088
CONNECT

OK
AT+CIPSEND=10

OK
>
Recv 10 bytes

SEND OK

+IPD,10:0123456789
```

```
53 45 4E 44 20 4F 4B 0D 0A
0D 0A
2B 49 50 44 2C 31 30 3A 30 31 32 33 34 35 36 37 38 39
```

ASCII [ ] Send
HEX [ ] Send
COM6 8N1 115200 | R 0 C 0 | R34 C19 | Disconnect

C main.c > ...

```c
//*****************************************************************************
//* SET DHCP ENABLE
//*****************************************************************************
uint8_t setDhcpEnable(void)
{
  rc = 0;
  UART1_RxHead = 0;
  UART1_RxTail = 0;
  uart_puts(UART1_ID,"AT+CWDHCP_CUR=1,1\r\n");
  if(chk_atok())
  {
    rc = 1;
  }
  return rc;
}
```

WizFi360 Driver Command Flow – Connect to the AP

Serial Input/Output Monitor

File Edit View Configuration

ASCII | HEX | ASCII Send | HEX Send | Line Status | Clear Terminal | Columns | Display | Data Graph | Ribbon | Classic

Input/Output | Viewing Options | Tools | Menu Style

```
ready
AT

OK
AT+CWMODE_CUR=1

OK
AT+CIPMUX=0

OK
AT+CWDHCP_CUR=1,1

OK
AT+CWJAP_CUR="edtpnet2","password"
WIFI CONNECTED
WIFI GOT IP

OK
AT+CIPSTART="TCP","192.168.1.235",8088
CONNECT

OK
AT+CIPSEND=10

OK
>
Recv 10 bytes

SEND OK

+IPD,10:0123456789
```

```
53 45 4E 44 20 4F 4B 0D 0A
0D 0A
2B 49 50 44 2C 31 30 3A 30 31 32 33 34 35 36 37 38 39
```

ASCII Send

HEX Send

COM6 8N1 115200 R 0 C 0 R34 C19 Disconnect

C main.c > ...

```c
//*****************************************************************************
//* CONNECT TO THE AP
//*****************************************************************************
uint8_t connect2AP(void)
{
  rc = 0;
  UART1_RxHead = 0;
  UART1_RxTail = 0;
  uart_puts(UART1_ID,"AT+CWJAP_CUR=\"edtpnet2\",\"password\"\r\n");
  busy_wait_ms(2000);
  if(chk_atok())
  {
    rc = 1;
  }
  return rc;
}
```

WizFi360 Driver Command Flow – Connect to the Server

Serial Input/Output Monitor

```
ready
AT

OK
AT+CWMODE_CUR=1

OK
AT+CIPMUX=0

OK
AT+CWDHCP_CUR=1,1

OK
AT+CWJAP_CUR="edtpnet2","  password  "
WIFI CONNECTED
WIFI GOT IP

OK
AT+CIPSTART="TCP","192.168.1.235",8088
CONNECT

OK
AT+CIPSEND=10

OK
>
Recv 10 bytes

SEND OK

+IPD,10:0123456789
```

```
53 45 4E 44 20 4F 4B 0D 0A
0D 0A
2B 49 50 44 2C 31 30 3A 30 31 32 33 34 35 36 37 38 39
```

ASCII Send

HEX Send

COM6 8N1 115200 R 0 C 0 R34 C19 Disconnect

C main.c > ...

```c
//*****************************************************************************
//* CONNECT TO THE SERVER
//*****************************************************************************
uint8_t connect2Server(void)
{
  rc = 0;
  UART1_RxHead = 0;
  UART1_RxTail = 0;
  uart_puts(UART1_ID,"AT+CIPSTART=\"TCP\",\"192.168.1.235\",8088\r\n");
  busy_wait_ms(2000);
  if(chk_atok())
  {
    rc = 1;
  }
  return rc;
}
```

WizFi360 Driver Command Flow – Send Data to the Server

Serial Input/Output Monitor

File Edit View Configuration

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

```
ready
AT

OK
AT+CWMODE_CUR=1

OK
AT+CIPMUX=0

OK
AT+CWDHCP_CUR=1,1

OK
AT+CWJAP_CUR="edtpnet2","  password  "
WIFI CONNECTED
WIFI GOT IP

OK
AT+CIPSTART="TCP","192.168.1.235",8088
CONNECT

OK
AT+CIPSEND=10

OK
>
Recv 10 bytes

SEND OK

+IPD,10:0123456789
```

```
53 45 4E 44 20 4F 4B 0D 0A
0D 0A
2B 49 50 44 2C 31 30 3A 30 31 32 33 34 35 36 37 38 39
```

ASCII [ ] Send
HEX [ ] Send
COM6 8N1 115200 | R 0 C 0 | R34 C19 | Disconnect

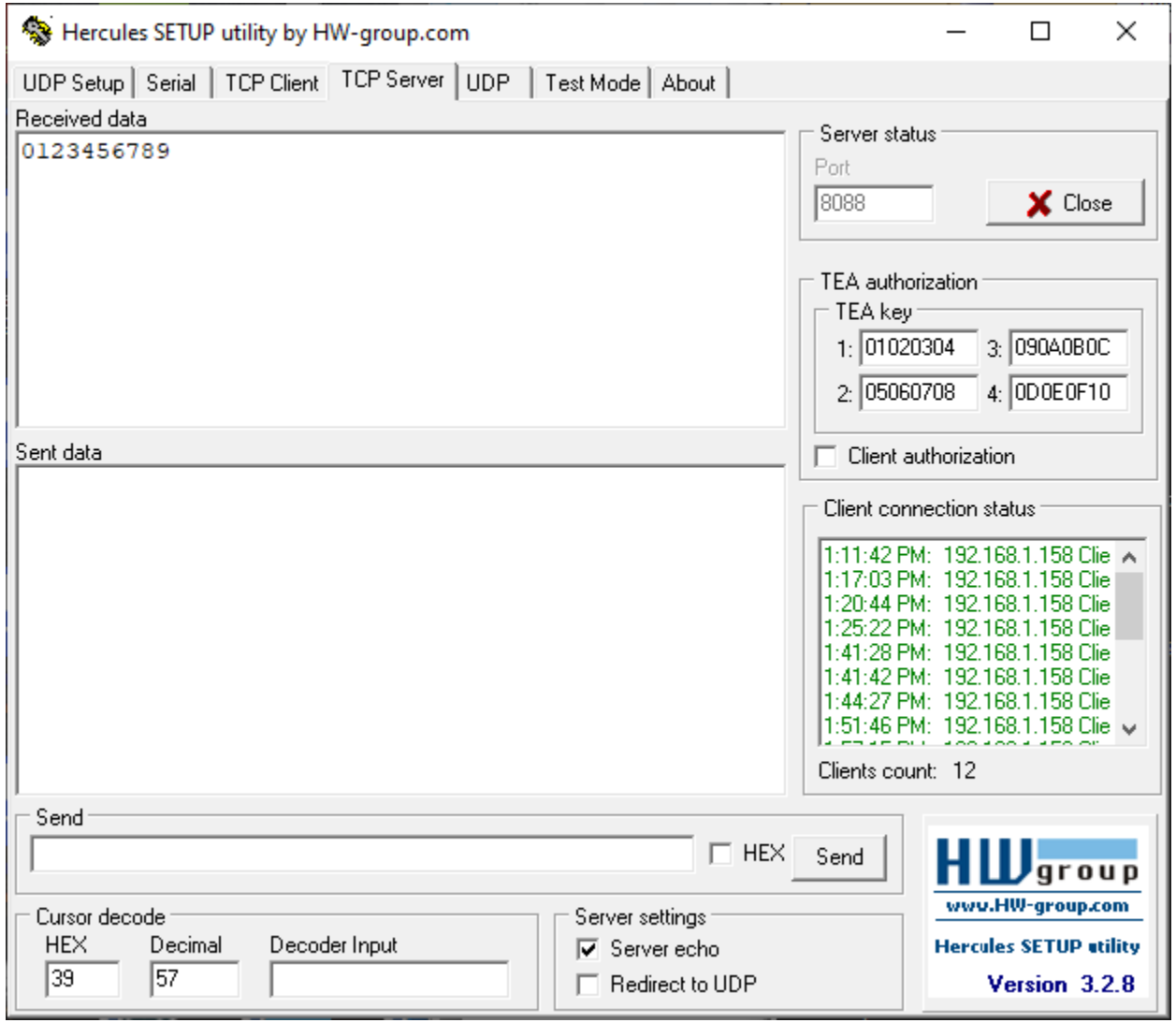
Macro List

main.c > ...

```c
//*****************************************************************************
//* SEND DATA TO THE SERVER
//*****************************************************************************
uint8_t send2Server(void)
{
  uint8_t bite = 0x30;
  rc = 0;
  UART1_RxHead = 0;
  UART1_RxTail = 0;
  bufIndx = 0;
  uart_puts(UART1_ID,"AT+CIPSEND=10\r\n");
  busy_wait_ms(1000);
  do{
    if(CharInRing())
    {
      rxBuf[bufIndx++] = readring();
    }
  }while(CharInRing());
  if(rxBuf[bufIndx-2] == '>')
  {
    UART1_RxHead = 0;
    UART1_RxTail = 0;
    bufIndx = 0;
    for(scratch8=0;scratch8<10;scratch8++)
    {
      uart_putc(UART1_ID,bite++);
    }
    busy_wait_ms(1000);
    do{
      if(CharInRing())
      {
        rxBuf[bufIndx++] = readring();
      }
    }while(CharInRing());
    if(rxBuf[bufIndx-4] == 'O' && rxBuf[bufIndx-3] == 'K')
    {
      rc = 1;
    }
    return rc;
  }

}
```

WizFi360 Driver Command Flow – Send Data to the Server

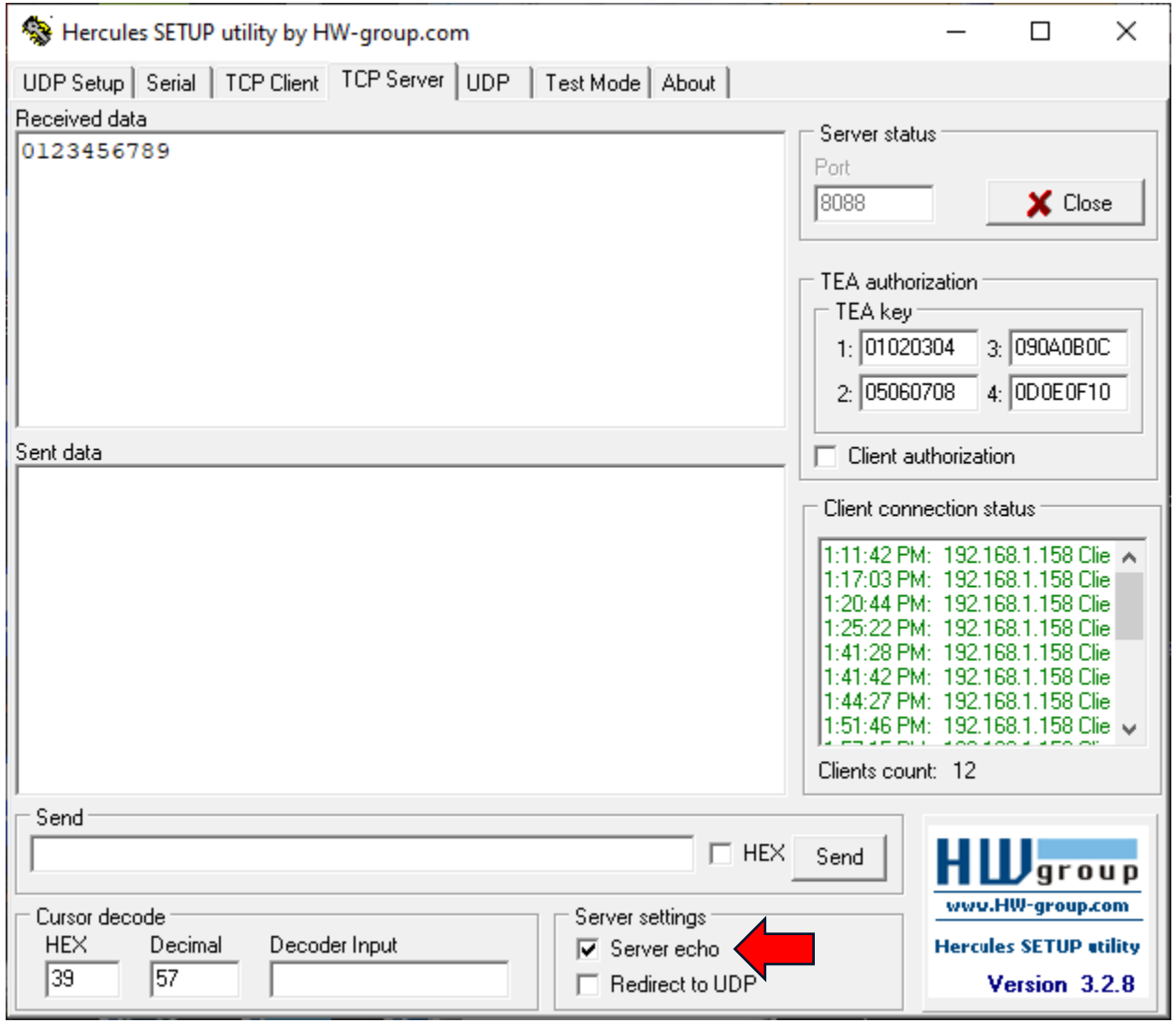


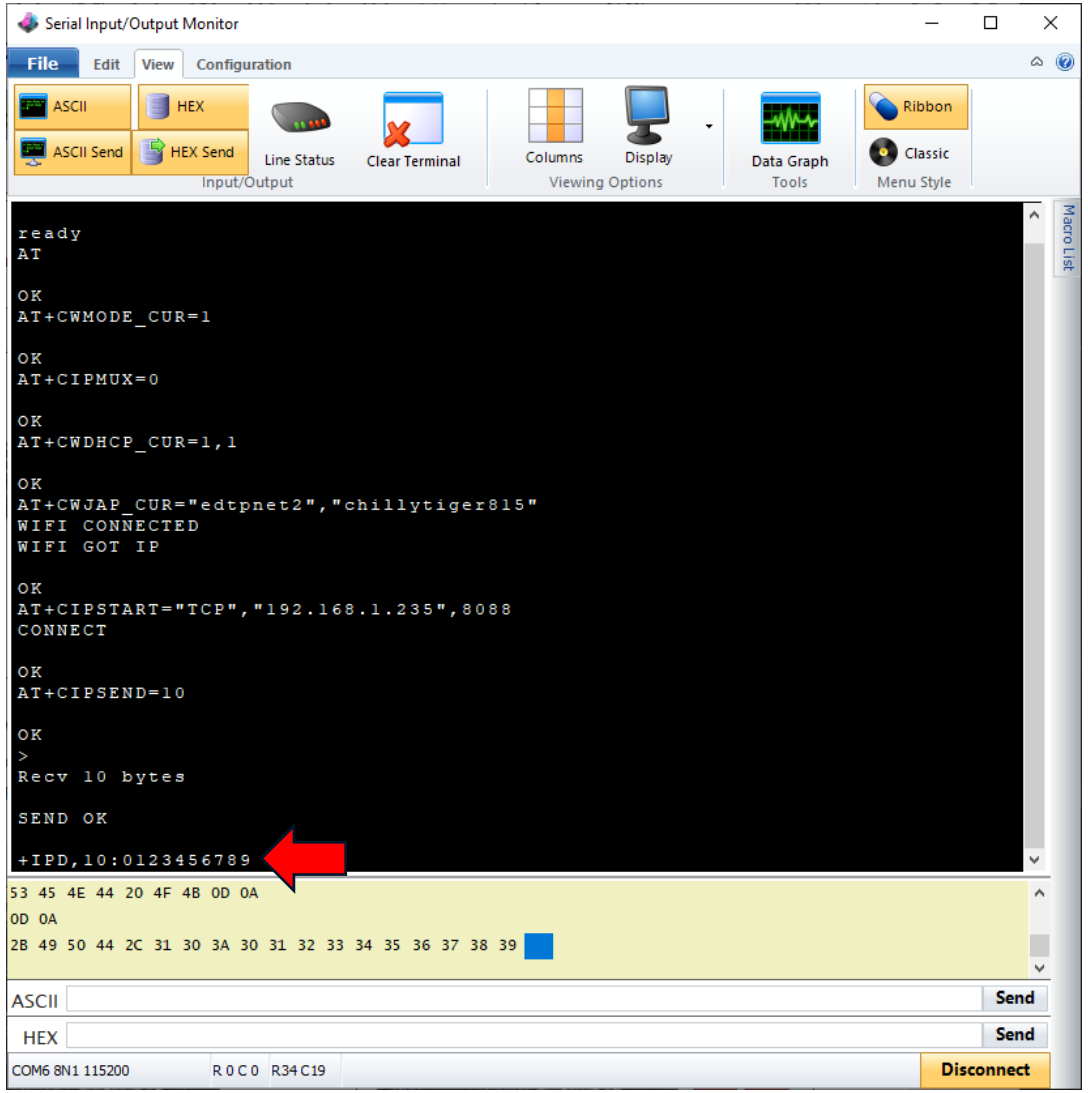

```c
main.c > ...

//*****************************************************************************
//* SEND DATA TO THE SERVER
//*****************************************************************************
uint8_t send2Server(void)
{
  uint8_t bite = 0x30;
  rc = 0;
  UART1_RxHead = 0;
  UART1_RxTail = 0;
  bufIndx = 0;
  uart_puts(UART1_ID,"AT+CIPSEND=10\r\n");
  busy_wait_ms(1000);
  do{
    if(CharInRing())
    {
      rxBuf[bufIndx++] = readring();
    }
  }while(CharInRing());
  if(rxBuf[bufIndx-2] == '>')
  {
    UART1_RxHead = 0;
    UART1_RxTail = 0;
    bufIndx = 0;
    for(scratch8=0;scratch8<10;scratch8++)
    {
      uart_putc(UART1_ID,bite++);
    }
    busy_wait_ms(1000);
    do{
      if(CharInRing())
      {
        rxBuf[bufIndx++] = readring();
      }
    }while(CharInRing());
    if(rxBuf[bufIndx-4] == 'O' && rxBuf[bufIndx-3] == 'K')
    {
      rc = 1;
    }
    return rc;
  }

}
```

WizFi360 Driver Command Flow – Send Data to the Server





Ozone – The J-Link Debugger

Ozone - The J-Link Debugger V3.26h - /home/fred/uart0_interrupt/build/rp2040_uart0.elf

File View Find Debug Tools Window Help

Functions

Name

```
____aeabi_idiv0_veneer
____aeabi_ldiv0_veneer
____assert_func_veneer
__aeabi_bits_init
__aeabi_double_init()
__aeabi_float_init()
__aeabi_ldiv0
__aeabi_mem_init
__assert_func(const char*, int, const char*, const c
+ __breakpoint()
+ __dmb()
__do_global_dtors_aux
__get_current_exception
+ __mem_fence_acquire()
+ __mem_fence_release()
__mutex_enter_blocking_veneer
__mutex_exit_veneer
__mutex_try_enter_veneer
+ __sev()
+ __wfe()
__wrap___aeabi_cdcmple
__wrap___aeabi_cdrcmple
__wrap___aeabi_d2iz
__wrap___aeabi_dadd
__wrap___aeabi_dcmpge
__wrap___aeabi_dcmpgt
__wrap___aeabi_dcmple
__wrap___aeabi_dcmplt
__wrap___aeabi_dcmpun
__wrap___aeabi_ddiv
__wrap___aeabi_dmul
__wrap___aeabi_drsub
__wrap___aeabi_dsub
__wrap___aeabi_i2d
__wrap___aeabi_idivmod
__wrap___aeabi_lmul
__wrap___aeabi_memcpy
__wrap___aeabi_ui2d
__wrap___aeabi_uidiv
__wrap_memset
__wrap_printf(const char*)
__wrap_puts(const char*)
__wrap_vprintf(const char*, struct __va_list)
_atoi(const char**)
_entry_point
```

main.c — File Scope — CharInRing

```
253     uart_init(UART0_ID, 115200);
254     uart_init(UART1_ID, 115200);
255     uart_set_hw_flow(UART0_ID, false, false);
256     uart_set_hw_flow(UART1_ID, false, false);
257     uart_set_format(UART0_ID, DATA_BITS, STOP_BITS, PARITY);
258     uart_set_format(UART1_ID, DATA_BITS, STOP_BITS, PARITY);
259     uart_set_fifo_enabled(UART0_ID, false);
260     uart_set_fifo_enabled(UART1_ID, false);
261
262     irq_set_exclusive_handler(UART1_IRQ, uart1_irq_handler);
263
264     uart_set_irq_enables(UART1_ID, true, false);
265     irq_set_enabled(UART1_IRQ, true);
266
267     memset(rxBuf,0x00,sizeof(rxBuf));
268     memset(UART1_RxBuf,0x00,sizeof(UART1_RxBuf));
269
270     busy_wait_ms(100);
271     gpio_put(RST_PIN,1);
272     busy_wait_ms(1000);
273     // throw away power up "ready" in RX ring buffer
274     UART1_RxHead = 0;
275     UART1_RxTail = 0;
276
277   }
278
279   int main()
280   {
281     init();
282     do
283     {
284         UART1_RxHead = 0;
285         UART1_RxTail = 0;
286         bufIndx = 0x00;
287         uart_puts(UART1_ID, "AT\r\n");
288         busy_wait_ms(50);
289         do
290         {
291           if(CharInRing())
292           {
293               rxBuf[bufIndx++] = readring();
294           }
295         }while(CharInRing());
296         if(rxBuf[bufIndx-4] == 'O' && rxBuf[bufIndx-3] == 'K')
297           {
298             okat = 0x01;
299           }
300   }while(okat == 0);
301
302   if(!setStationMode())
303   {
304     do{
305       gpio_put(LED_PIN,0);
306       busy_wait_ms(200);
307       gpio_put(LED_PIN,1);
```

Disassembly

```
main
$Thumb
{
 10000BE4   PUSH    {R4-R5, LR}
 10000BE6   SUB     SP, SP, #12
init();
 10000BE8   BL      init                    ; 0x10000AA0
 10000BEC   B       0x10000CB4              ; <main>+0xD0
uart_putc_raw(uart, *s);
 10000BEE   MOV     R3, SP
 10000BF0   ADDS    R2, R3, #7
 10000BF2   STRB    R4, [R2]
static inline void uart_putc_raw(uart_inst_t *uart, char c) {
```

Watched Data 1

Expression	Value	Location	Refresh
UART1_RxBuf		2000 0444	Off
[0]	0 ('\0')	2000 0444	Off
[1]	65 ('A')	2000 0445	Off
[2]	84 ('T')	2000 0446	Off
[3]	43 ('+')	2000 0447	Off
[4]	67 ('C')	2000 0448	Off
[5]	87 ('W')	2000 0449	Off
[6]	68 ('D')	2000 044A	Off
[7]	72 ('H')	2000 044B	Off
[8]	67 ('C')	2000 044C	Off
[9]	80 ('P')	2000 044D	Off
[10]	95 ('_')	2000 044E	Off
[11]	67 ('C')	2000 044F	Off
[12]	85 ('U')	2000 0450	Off
[13]	82 ('R')	2000 0451	Off
[14]	61 ('=')	2000 0452	Off
[15]	49 ('1')	2000 0453	Off
[16]	44 (',')	2000 0454	Off
[17]	49 ('1')	2000 0455	Off
[18]	13 ('\r')	2000 0456	Off
[19]	10 ('\n')	2000 0457	Off
[20]	13 ('\r')	2000 0458	Off
[21]	10 ('\n')	2000 0459	Off
[22]	79 ('O')	2000 045A	Off
[23]	75 ('K')	2000 045B	Off
[24]	13 ('\r')	2000 045C	Off
[25]	10 ('\n')	2000 045D	Off
[26]	0 ('\0')	2000 045E	Off
[27]	0 ('\0')	2000 045F	Off
[28]	0 ('\0')	2000 0460	Off
[29]	0 ('\0')	2000 0461	Off
[30]	0 ('\0')	2000 0462	Off
[31]	0 ('\0')	2000 0463	Off
[32]	0 ('\0')	2000 0464	Off

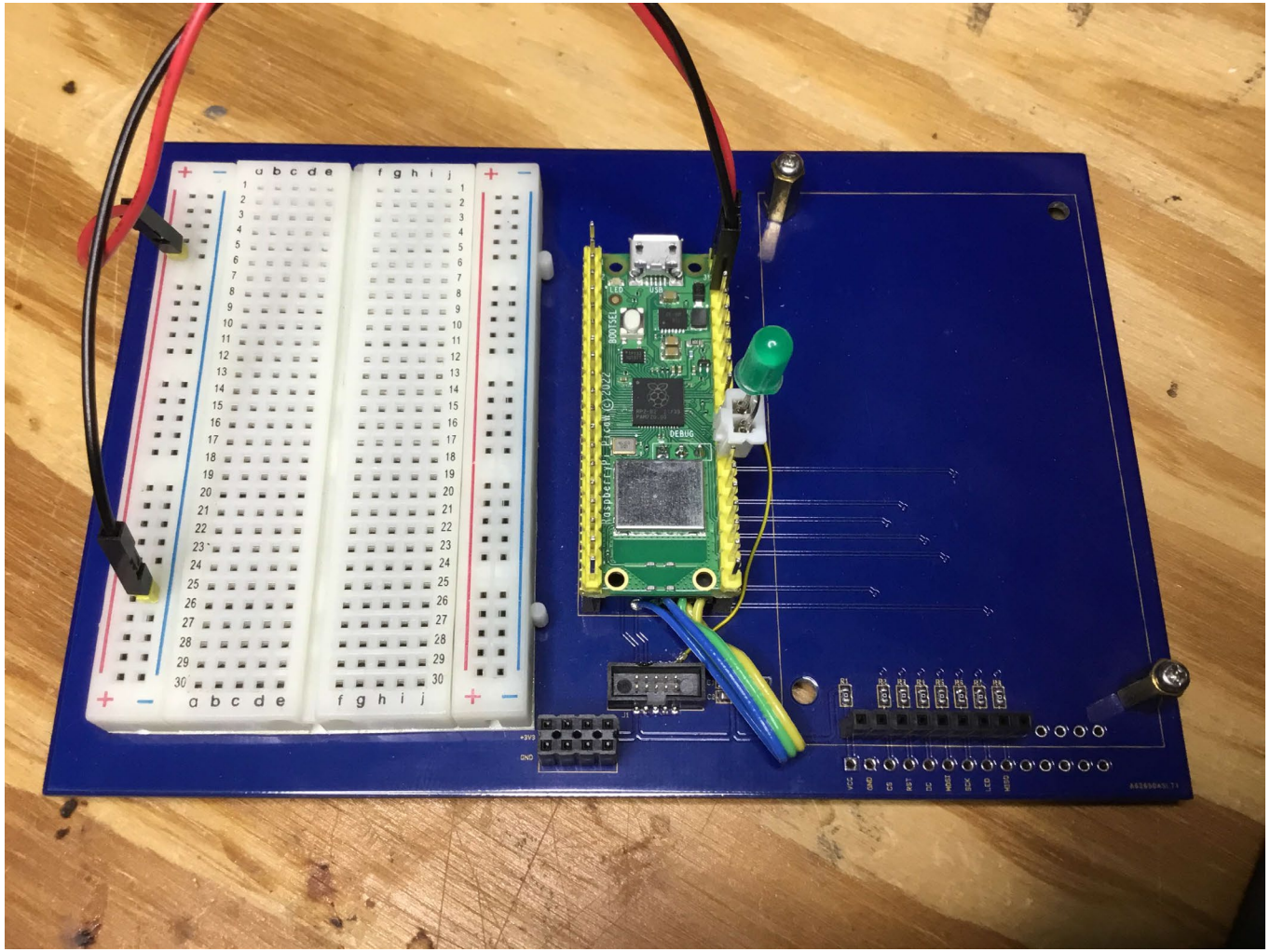
Console

```
[0][1]: E0001000 CID B105E00D PID 000BB00A DWT
[0][2]: E0002000 CID B105E00D PID 000BB00B FPB
Connected to target device.
```

CPU running.... | Ln 279 Ch 6 | Connected @ 4 MHz

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

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

informa markets