

# Bare Metal C Programming for STM32 Devices

**Day 3:**

**Bare Metal NUCLEO-F207ZG :**

**Mongoose Network Library API Primer**

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.



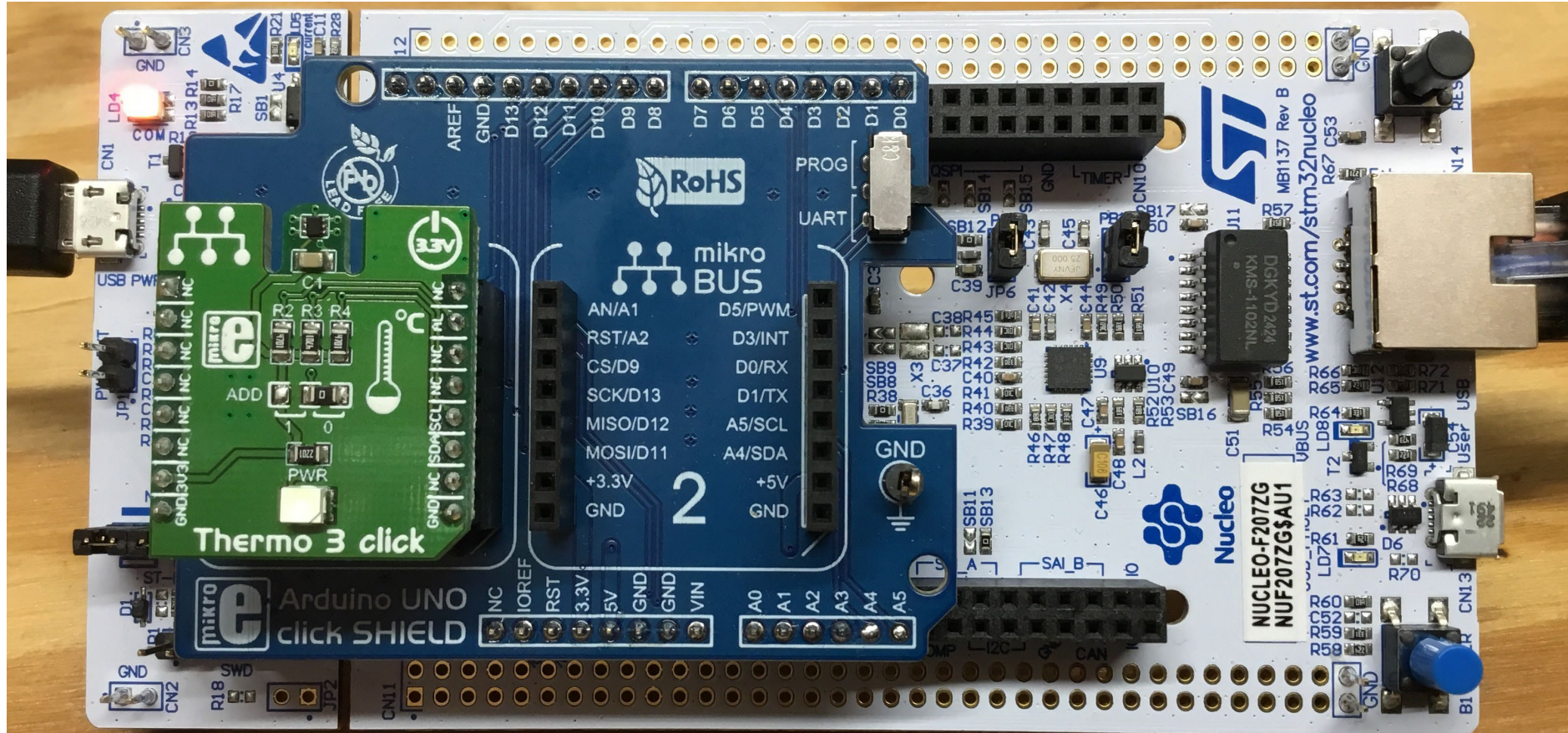
## Fred Eady

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



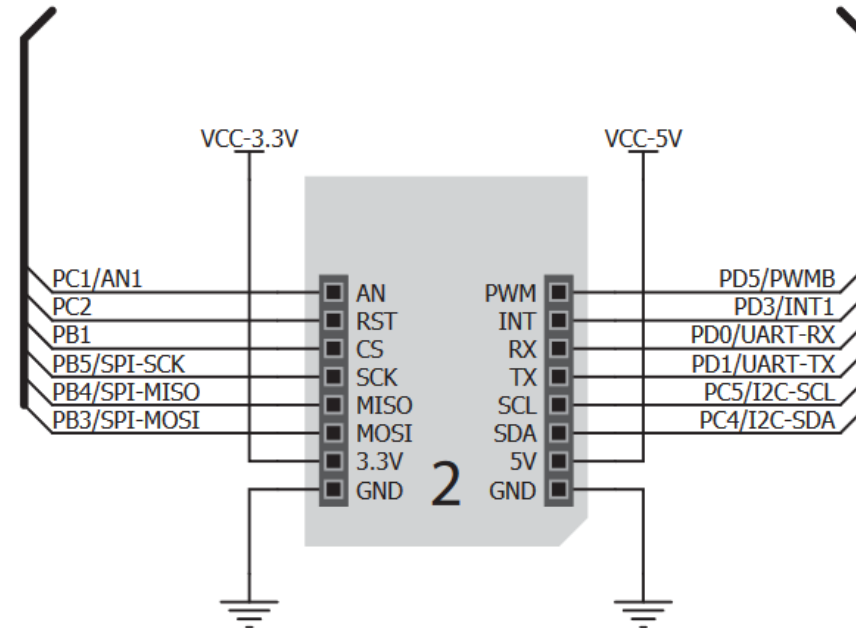
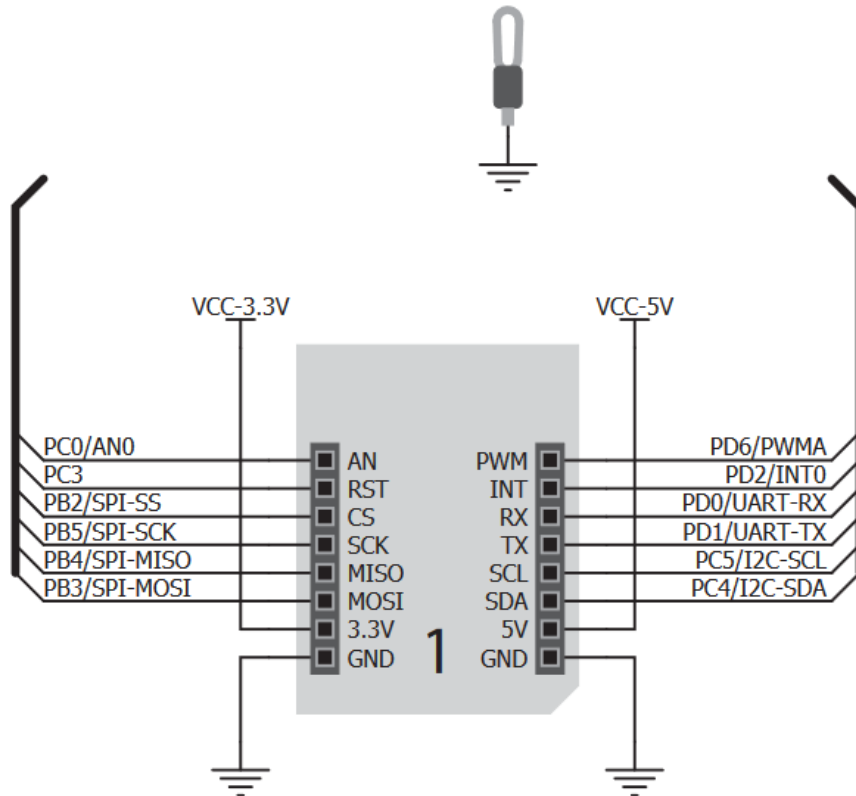
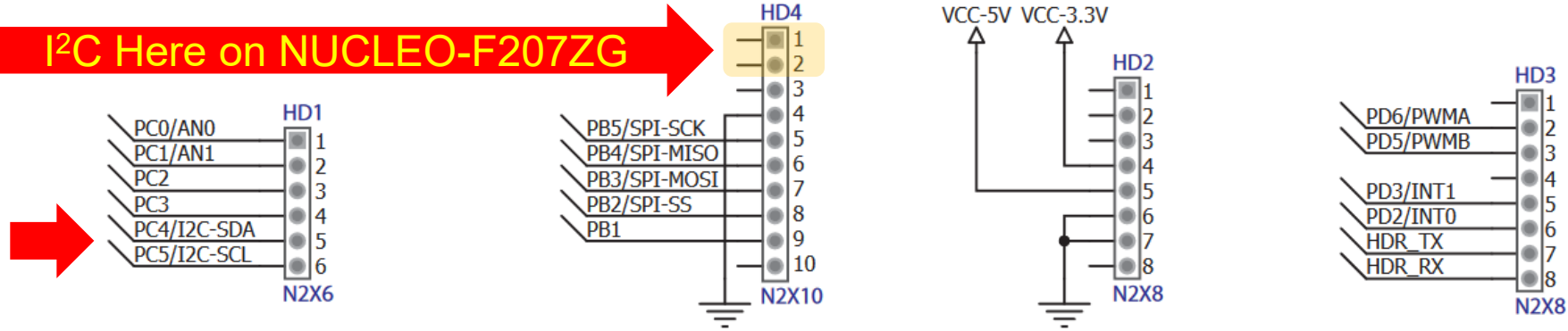
# AGENDA

- **Modify the Arduino UNO click SHIELD**
- **Thermo 3 click Driver**
- **Integrate Mongoose Using the STM32 Mongoose Package**

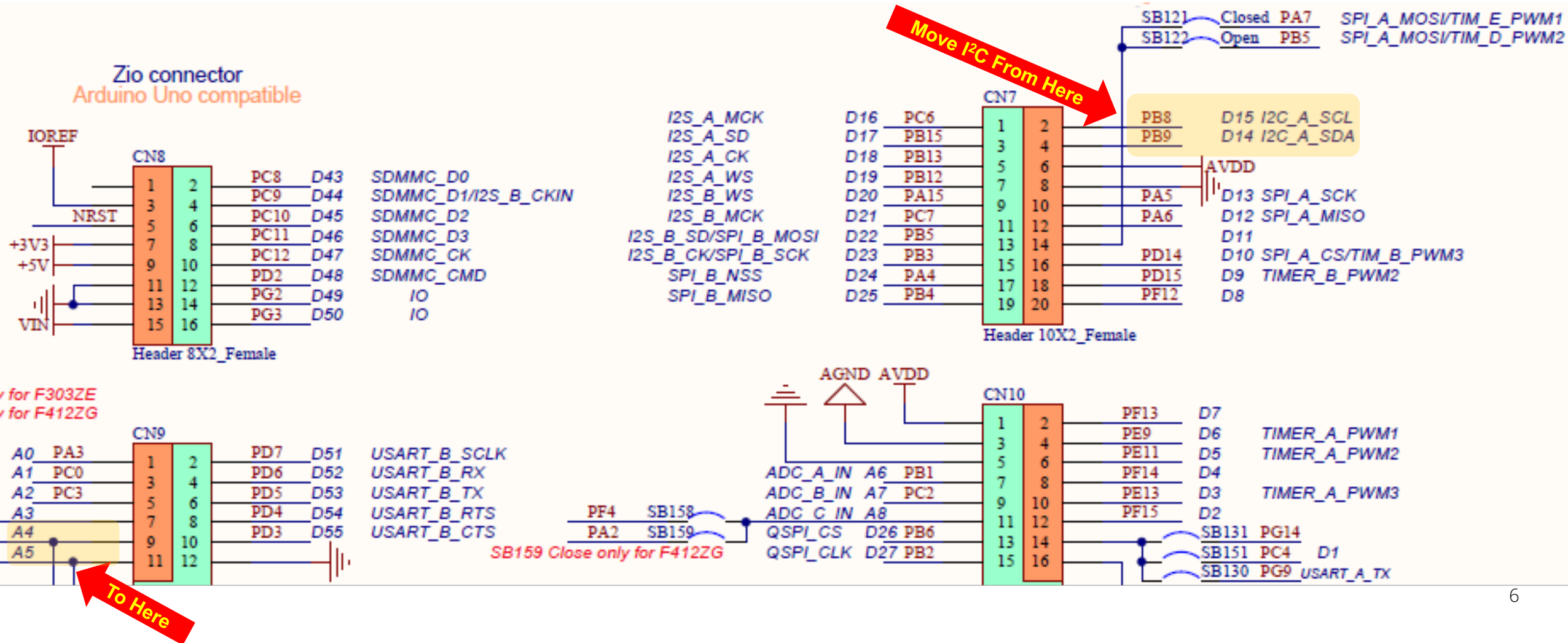


## Arduino UNO click SHIELD I<sup>2</sup>C Hardware Modification

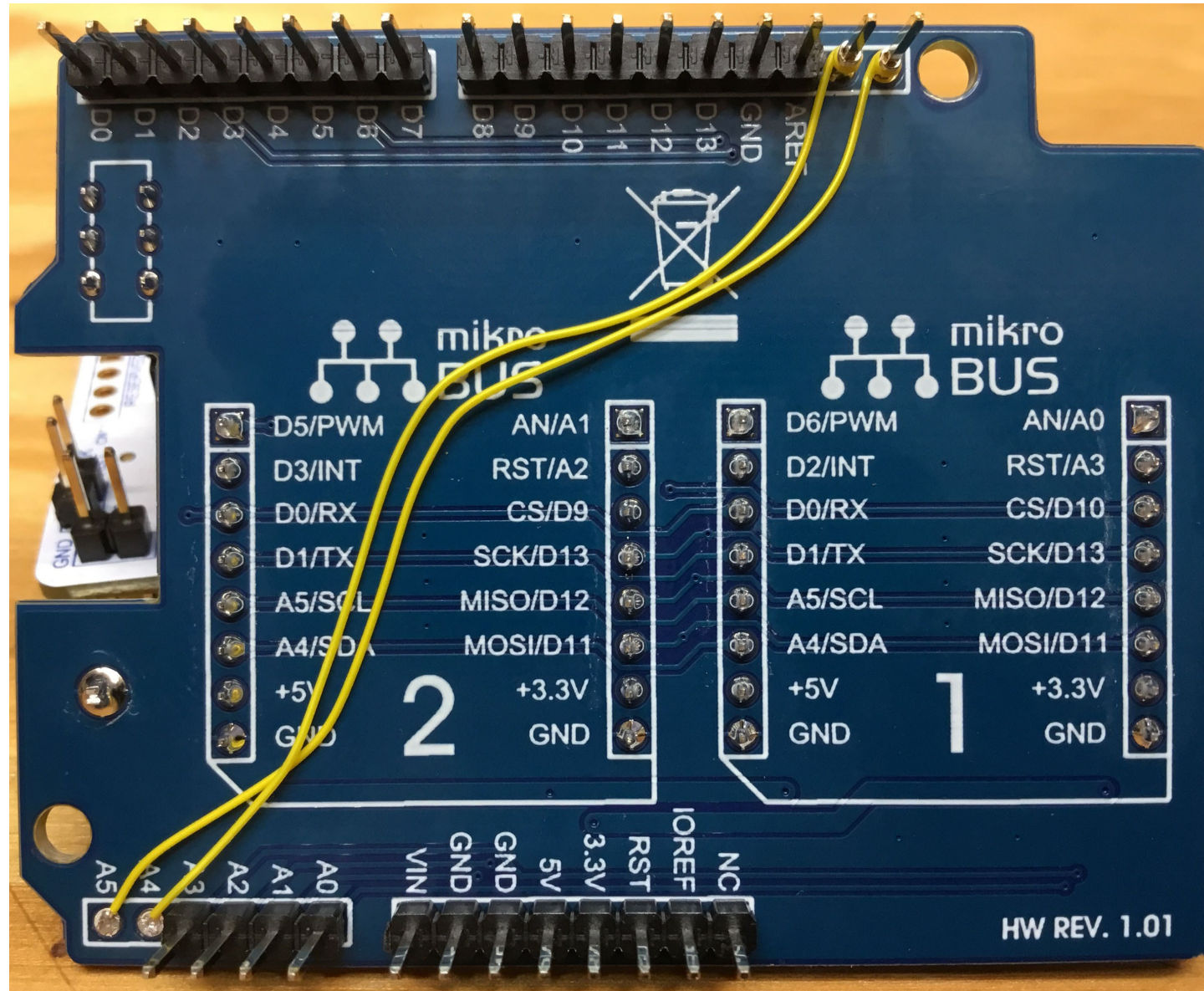
I<sup>2</sup>C Here on NUCLEO-F207ZG



## Arduino UNO click SHIELD I<sup>2</sup>C Hardware Modification



# Arduino UNO click SHIELD I<sup>2</sup>C Hardware Modification



## Configure and Activate I2C1

The screenshot displays the STM32CubeMX configuration tool. The left sidebar shows the 'Pinout & Configuration' tab with a search bar and a list of categories. Under 'Connectivity', I2C1 is selected and highlighted in blue. The main area shows the 'Clock Configuration' tab, specifically the 'I2C1 Mode and Configuration' section. The 'Mode' dropdown is set to 'I2C'. Below this, the 'Configuration' section includes a 'Reset Configuration' button and several checked settings: 'NVIC Settings', 'DMA Settings', 'GPIO Settings', 'Parameter Settings', and 'User Constants'. A search bar is present above the parameter list. The parameters are organized into 'Master Features' and 'Slave Features' sections.

I2C1 Mode and Configuration		
Mode		
I2C	I2C	
Configuration		
Reset Configuration		
<input checked="" type="checkbox"/> NVIC Settings	<input checked="" type="checkbox"/> DMA Settings	<input checked="" type="checkbox"/> GPIO Settings
<input checked="" type="checkbox"/> Parameter Settings		<input checked="" type="checkbox"/> User Constants
Configure the below parameters :		
Search (Ctrl+F)		
Master Features		
I2C Speed Mode	Fast Mode	
I2C Clock Speed (Hz)	400000	
Fast Mode Duty Cycle	Duty cycle Tlow/Thigh = 2	
Slave Features		
Clock No Stretch Mode	Disabled	
Primary Address Length selection	7-bit	
Dual Address Acknowledged	Disabled	
Primary slave address	0	
General Call address detection	Disabled	

## Configure and Activate I2C1

Pinout & Configuration      Clock Configuration      Project Manager

Software Packs

GPIO Mode an

M

Config

Group By Peripherals

RCC     SYS     US

GPIO     Single Mapped S

Search Signals

Search (Ctrl+F)

Show only Modified Pins

Pin Na...	Signal on ...	GPIO outp...	GPIO mode	GPIO Pull-u...	Maximum ...	User Label	Modified
PB8	I2C1_SCL	n/a	Alternate ...	No pull-up ...	High	A5_SCL	<input checked="" type="checkbox"/>
PB9	I2C1_SDA	n/a	Alternate ...	No pull-up ...	High	A4_SDA	<input checked="" type="checkbox"/>

DMA

GPIO

IWDG

NVIC

RCC

SYS

WWDG

Analog >

Timers >

Connectivity >

Multimedia >

NVIC

I2C

## Assemble tmp102.c

```
C tmp102.c X
Core > Src > C tmp102.c > ...
20  /*******
21  /*** INCLUDES
22  /*******
23  #include "tmp102.h"
24  /*******
25  /*** TYPEDEFS
26  /*******
27  extern I2C_HandleTypeDef hi2c1;
28  /*******
29  /*** TMP102 VARIABLES
30  /*******
31  uint8_t txBuf[8];      // I2C transmit buffer
32  uint8_t rxBuf[8];     // I2C receive buffer
33  uint16_t tempBuf;     // TMP102 temperature buffer
34  float tempC;         // temperature in C
35  char registerName[18]; // holding array for TMP102 register name
36  /*******
37  /*** TMP102 and I2C CONSTANTS
38  /*** 7-bit address = 0x48
39  /*** 8-bit address = 0x90 write
40  /*** 8-bit address = 0x91 read
41  /*******
42  const unsigned char tmp102_addr = 0x90; // TMP102 I2C 7-bit address = 0x48
```

## Assemble tmp102.c

C tmp102.c X

Core &gt; Src &gt; C tmp102.c &gt; ...

```
44 //*****
45 // Select TMP102 Register
46 //*****
47 void tmp102_select_reg(uint8_t reg)
48 {
49     //*****
50     // Write Pointer Register with location of the temperature register
51     // P7 P6 P5 P4 P3 P2 P1 P0
52     // 0 0 0 0 0 0 0 0 Temperature Register
53     // 0 0 0 0 0 0 0 1 Configuration Register
54     // 0 0 0 0 0 0 1 0 Temperature Low Register
55     // 0 0 0 0 0 0 1 1 Temperature High Register
56     //*****
57     txBuf[0] = reg;
58     HAL_I2C_Master_Transmit(hi2c: &hi2c1, DevAddress: tmp102_addr, pData: txBuf, Size: 0x01, Timeout: HAL_MAX_DELAY);
59     switch (txBuf[0])
60     {
61     case 0x00:
62         |   sprintf(registerName, "Temperature");
63         break;
64     case 0x01:
65         |   sprintf(registerName, "Configuration");
66         break;
67     case 0x02:
68         |   sprintf(registerName, "Temperature Low");
69         break;
70     case 0x03:
71         |   sprintf(registerName, "Temperature High");
72         break;
73     }
74 }
```

## Assemble tmp102.c

C tmp102.c X

Core &gt; Src &gt; C tmp102.c &gt; tmp102\_select\_reg

```
76 //*****
77 // Read the temperature
78 //*****
79 float tmp102_read_temperature(void)
80 {
81     HAL_I2C_Master_Receive(hi2c:&hi2c1, DevAddress: tmp102_addr, pData: rxBuf, Size: 2, Timeout: HAL_MAX_DELAY);
82     MG_INFO((fmt: "INFO: 2 bytes were read-> rxBuf[0] 0x%02X -- rxBuf[1] 0x%02X",rxBuf[0],rxBuf[1]));
83     //*****
84     // Convert temperature bytes to 16-bit value
85     //*****
86     tempBuf = (rxBuf[0] << 4) | (rxBuf[1] >> 4);
87     MG_INFO((fmt: "INFO: 16-bit value = 0x%04X",tempBuf));
88     //*****
89     // Compute and print the temperature
90     //*****
91     if(tempBuf & (1 << 11))
92     {
93         tempBuf |= 0xF800;
94     }
95     tempC = (float)tempBuf * 0.0625;
96     return tempC;
97 }
```

## Assemble tmp102.h

```
C tmp102.h x
Core > Inc > C tmp102.h > ...
17  #include "main.h"
18  #include "i2c.h"
19  #include "stm32f2xx_hal_def.h"
20  #include "stm32f2xx_hal_i2c.h"
21  #include "usart.h"
22  #include "gpio.h"
23  #include <stdio.h>
24  #include <stdint.h>
25  #include <string.h>
26  #include "mongoose.h"
27
28  void tmp102_select_reg(uint8_t reg);
29  float tmp102_read_temperature(void);
30
31  /*******
32  // Write Pointer Register with location of the temperature register
33  // P7 P6 P5 P4 P3 P2 P1 P0
34  // 0 0 0 0 0 0 0 0  Temperature Register
35  // 0 0 0 0 0 0 0 1  Configuration Register
36  // 0 0 0 0 0 0 1 0  Temperature Low Register
37  // 0 0 0 0 0 0 1 1  Temperature High Register
38  /*******
39  #define tempreg      0x00
40  #define confreg      0x01
41  #define tempreglo    0x02
42  #define tempreghi    0x03
```

## Assemble main.c – Main Loop

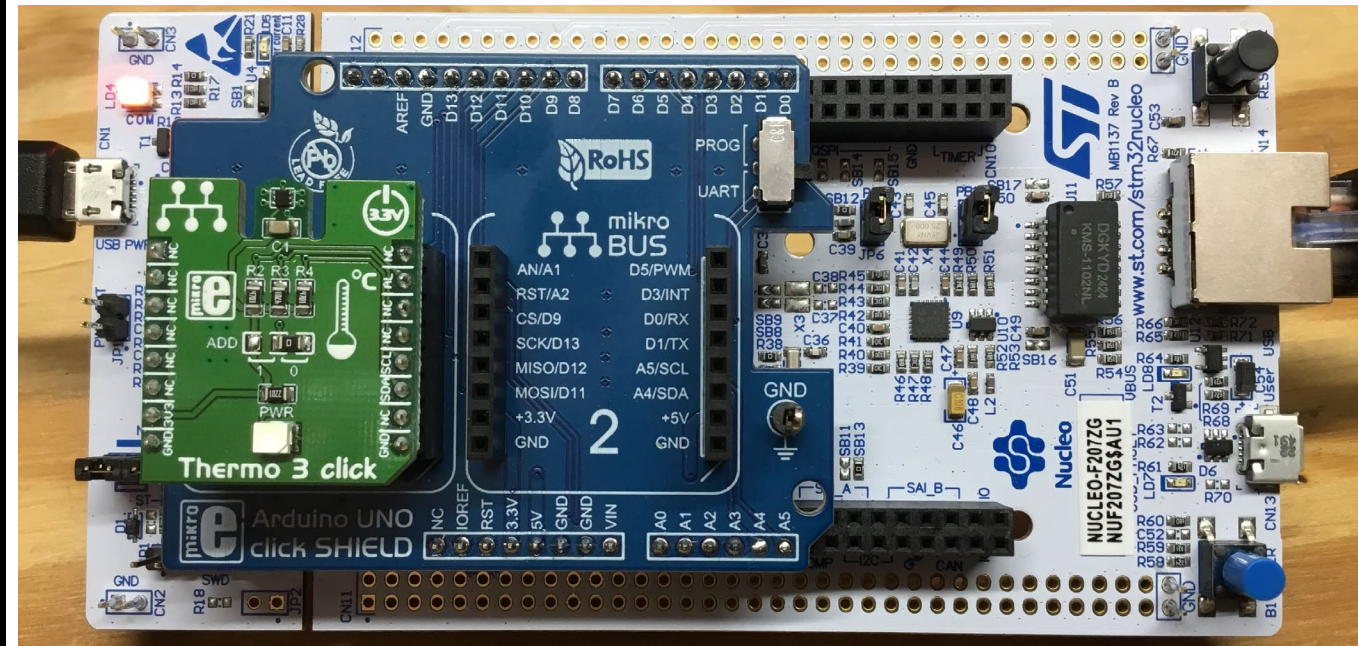
C main.c 9+ X

Core &gt; Src &gt; C main.c &gt; main

```

213  /* Initialize all configured peripherals */
214  MX_GPIO_Init();
215  MX_ETH_Init();
216  MX_USART3_UART_Init();
217  MX_USB_OTG_FS_PCD_Init();
218  MX_I2C1_Init();
219  /* USER CODE BEGIN 2 */
220  NVIC_EnableIRQ(IRQn: ETH_IRQn);
221  struct mg_mgr mgr;           // Mongoose event manager
222  mg_log_set(MG_LL_DEBUG);    // Set log level to debug
223  mg_mgr_init(&mgr);         // Initialise event manager
224  /* Kick off the server
225  mg_listen(&mgr, url: s_lsn,fn: server_fn,fn_data: NULL);
226  /* USER CODE END 2 */
227
228  /* Infinite loop */
229  /* USER CODE BEGIN WHILE */
230  while (1)
231  {
232      mg_mgr_poll(&mgr,ms: 0);
233      /* USER CODE END WHILE */
234
235      /* USER CODE BEGIN 3 */
236  }
237  /* USER CODE END 3 */
238  }

```



## Assemble main.c – Redirect printf

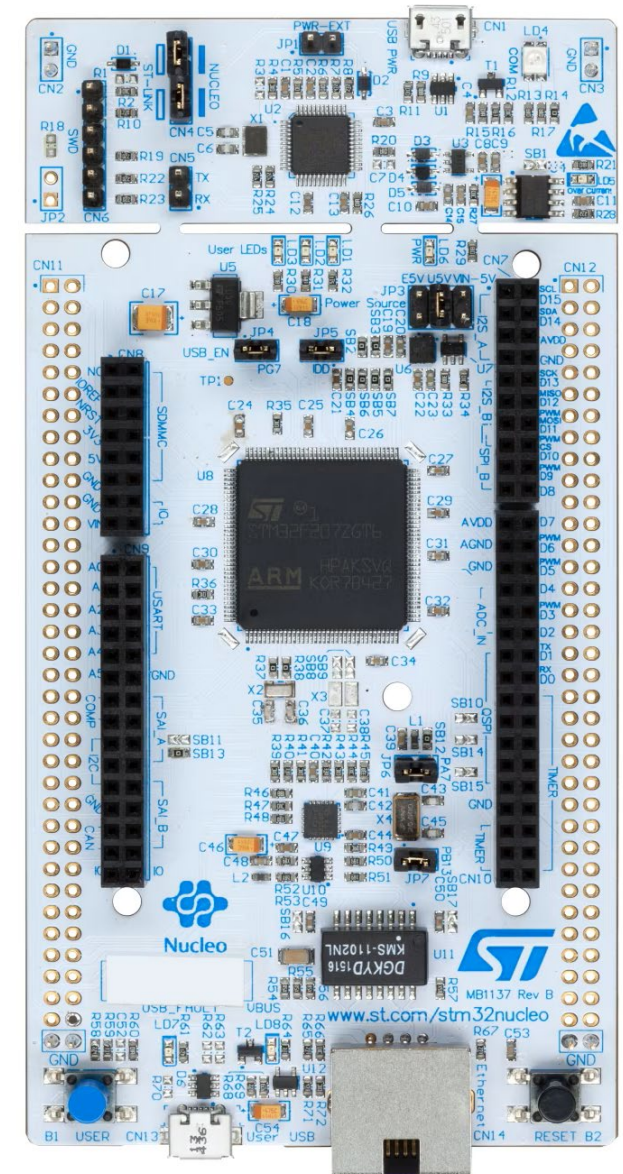
C main.c 9+ X

Core &gt; Src &gt; C main.c &gt; server\_fn

```

50  /* USER CODE BEGIN PV */
51  float temptosend;
52  /** The tcp://0.0.0.0:8088 indicates that the server listens for connections
53  /** on all available network interfaces. 8088 is the destination port.
54  static const char *s_lsn = "tcp://0.0.0.0:8088";
55  /* USER CODE END PV */
56
57  /* Private function prototypes -----*/
58  void SystemClock_Config(void);
59  /* USER CODE BEGIN PFP */
60  /* USER CODE END PFP */
61
62  /* Private user code -----*/
63  /* USER CODE BEGIN 0 */
64  /**-----*/
65  /** REDIRECT printf TO USART3
66  /**-----*/
67  #ifdef __GNUC__
68  #define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
69  #else
70  #define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
71  #endif
72  PUTCHAR_PROTOTYPE
73  {
74  // Place your implementation of fputc here
75  // e.g., write a character to the USART3 and loop until the end of transmission
76  HAL_UART_Transmit(&huart3, pData: (uint8_t *)&ch, Size: 1, Timeout: HAL_MAX_DELAY);
77  return ch;
78  }

```



## Assemble main.c – Build the Server Event Handler

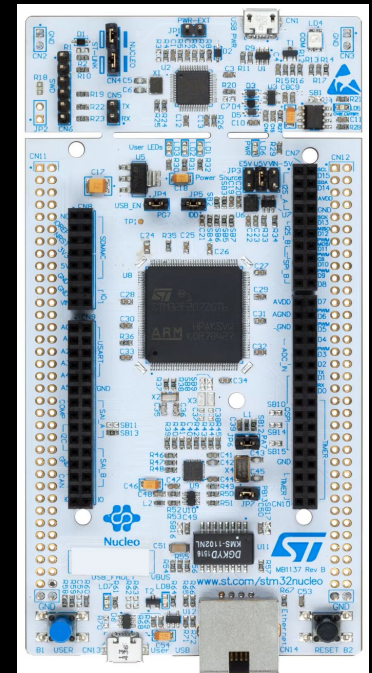
C main.c 9+ X

Core &gt; Src &gt; C main.c &gt; server\_fn

```

80  //*****
81  /* SERVER EVENT HANDLER
82  //*****
83  static void server_fn(struct mg_connection *c, int ev, void *ev_data)
84  {
85      if (ev == MG_EV_OPEN && c->is_listening == 1)
86      {
87          MG_INFO((fmt: "INFO: SERVER IS LISTENING"));
88      }
89      else if (ev == MG_EV_ACCEPT)
90      {
91          MG_INFO((fmt: "INFO: SERVER ACCEPTED A CONNECTION"));
92      }
93      else if (ev == MG_EV_READ)
94      {
95          MG_INFO((fmt: "INFO: SERVER RECEIVED DATA: %.*s", c->recv.len, c->recv.buf));

```



## Assemble main.c – Build the Server Event Handler

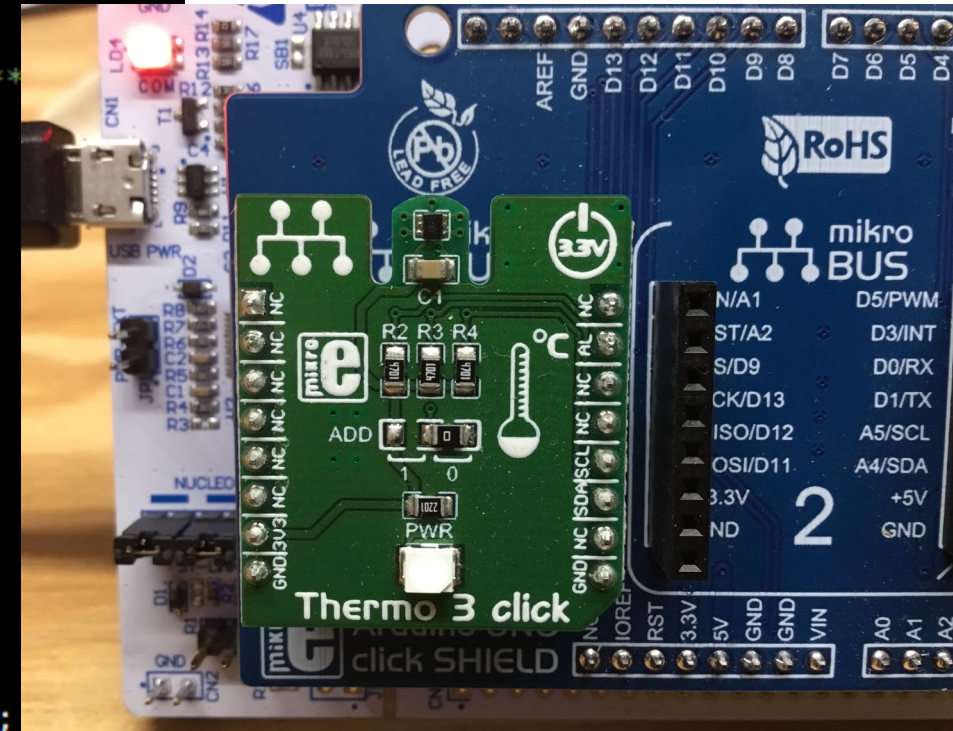
C main.c 9+ X

Core &gt; Src &gt; C main.c &gt; server\_fn

```

96 //*****
97 /* SERVER EVENT HANDLER - SELECT TMP102 WORKING REGISTER
98 //*****
99 if(strncmp(c->recv.buf,"SELECT REGISTER TEMP",21) == 0)
100 {
101     tmp102_select_reg(reg:0);
102     MG_INFO((fmt: "INFO: TMP102 TEMPERATURE REGISTER SELECTED!"));
103 }
104 else if(strncmp(c->recv.buf,"SELECT REGISTER CONFIG",23) == 0)
105 {
106     tmp102_select_reg(reg:1);
107     MG_INFO((fmt: "INFO: TMP102 CONFIGURATION REGISTER SELECTED!"));
108 }
109 else if(strncmp(c->recv.buf,"SELECT REGISTER TEMPL0",23) == 0)
110 {
111     tmp102_select_reg(reg:2);
112     MG_INFO((fmt: "INFO: TMP102 TEMPERATURE LO REGISTER SELECTED!"));
113 }
114 else if(strncmp(c->recv.buf,"SELECT REGISTER TEMPHI",23) == 0)
115 {
116     tmp102_select_reg(reg:3);
117     MG_INFO((fmt: "INFO: TMP102 TEMPERATURE HI REGISTER SELECTED!"));
118 }

```



## Assemble main.c – Build the Server Event Handler

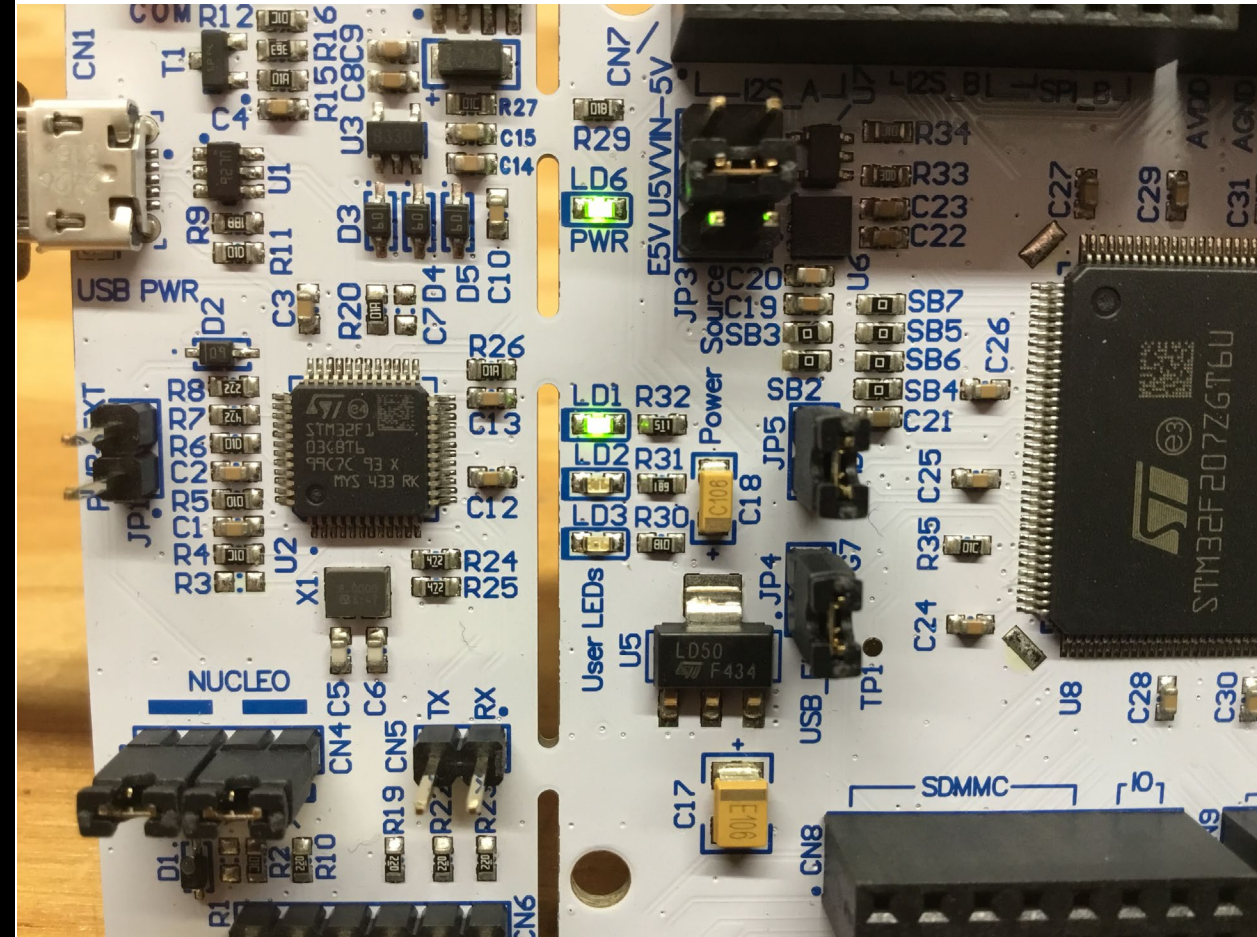
C main.c 9+ X

Core &gt; Src &gt; C main.c &gt; server\_fn

```

119 //*****
120 /* SERVER EVENT HANDLER - RED LED CONTROL
121 //*****
122     if(strncmp(c->recv.buf,"RED LED ON",11) == 0)
123     {
124         HAL_GPIO_WritePin(GPIOx: LED3R_GPIO_Port, GPIO_Pin: LED3R_Pin, PinState: GPIO_PIN_SET);
125         MG_INFO((fmt: "INFO: RED LED IS ON!"));
126     }
127     else if(strncmp(c->recv.buf,"RED LED OFF",12) == 0)
128     {
129         HAL_GPIO_WritePin(GPIOx: LED3R_GPIO_Port, GPIO_Pin: LED3R_Pin, PinState: GPIO_PIN_RESET);
130         MG_INFO((fmt: "INFO: RED LED IS OFF!"));
131     }
132 //*****
133 /* SERVER EVENT HANDLER - GREEN LED CONTROL
134 //*****
135     if(strncmp(c->recv.buf,"GREEN LED ON",13) == 0)
136     {
137         HAL_GPIO_WritePin(GPIOx: LED1G_GPIO_Port, GPIO_Pin: LED1G_Pin, PinState: GPIO_PIN_SET);
138         MG_INFO((fmt: "INFO: GREEN LED IS ON!"));
139     }
140     else if(strncmp(c->recv.buf,"GREEN LED OFF",14) == 0)
141     {
142         HAL_GPIO_WritePin(GPIOx: LED1G_GPIO_Port, GPIO_Pin: LED1G_Pin, PinState: GPIO_PIN_RESET);
143         MG_INFO((fmt: "INFO: GREEN LED IS OFF!"));
144     }
145 //*****
146 /* SERVER EVENT HANDLER - BLUE LED CONTROL
147 //*****
148     if(strncmp(c->recv.buf,"BLUE LED ON",12) == 0)
149     {
150         HAL_GPIO_WritePin(GPIOx: LED2B_GPIO_Port, GPIO_Pin: LED2B_Pin, PinState: GPIO_PIN_SET);
151         MG_INFO((fmt: "INFO: BLUE LED IS ON!"));
152     }
153     else if(strncmp(c->recv.buf,"BLUE LED OFF",13) == 0)
154     {
155         HAL_GPIO_WritePin(GPIOx: LED2B_GPIO_Port, GPIO_Pin: LED2B_Pin, PinState: GPIO_PIN_RESET);
156         MG_INFO((fmt: "INFO: BLUE LED IS OFF!"));
157     }

```

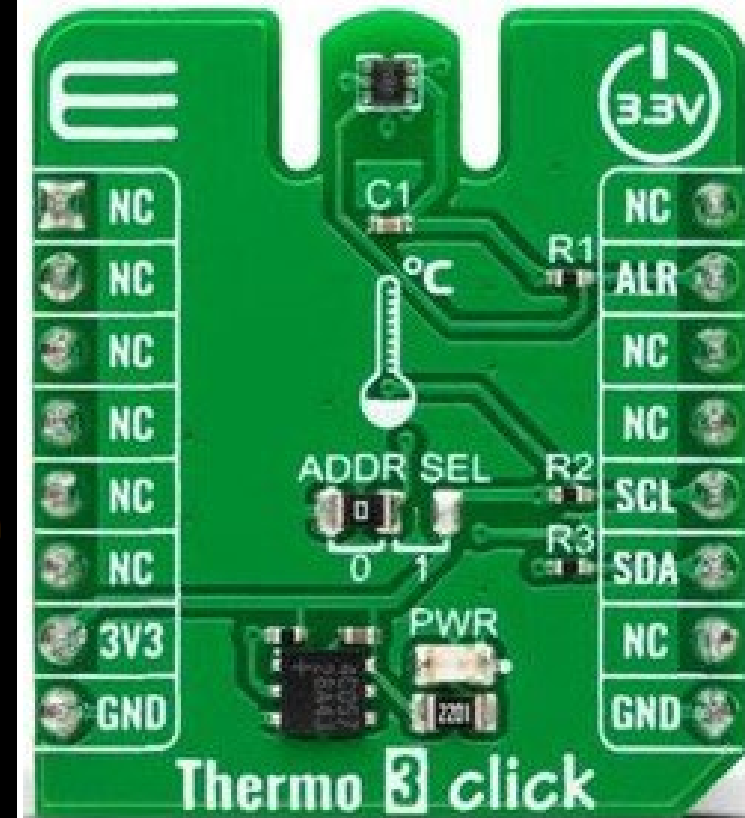


## Assemble main.c – Build the Server Event Handler

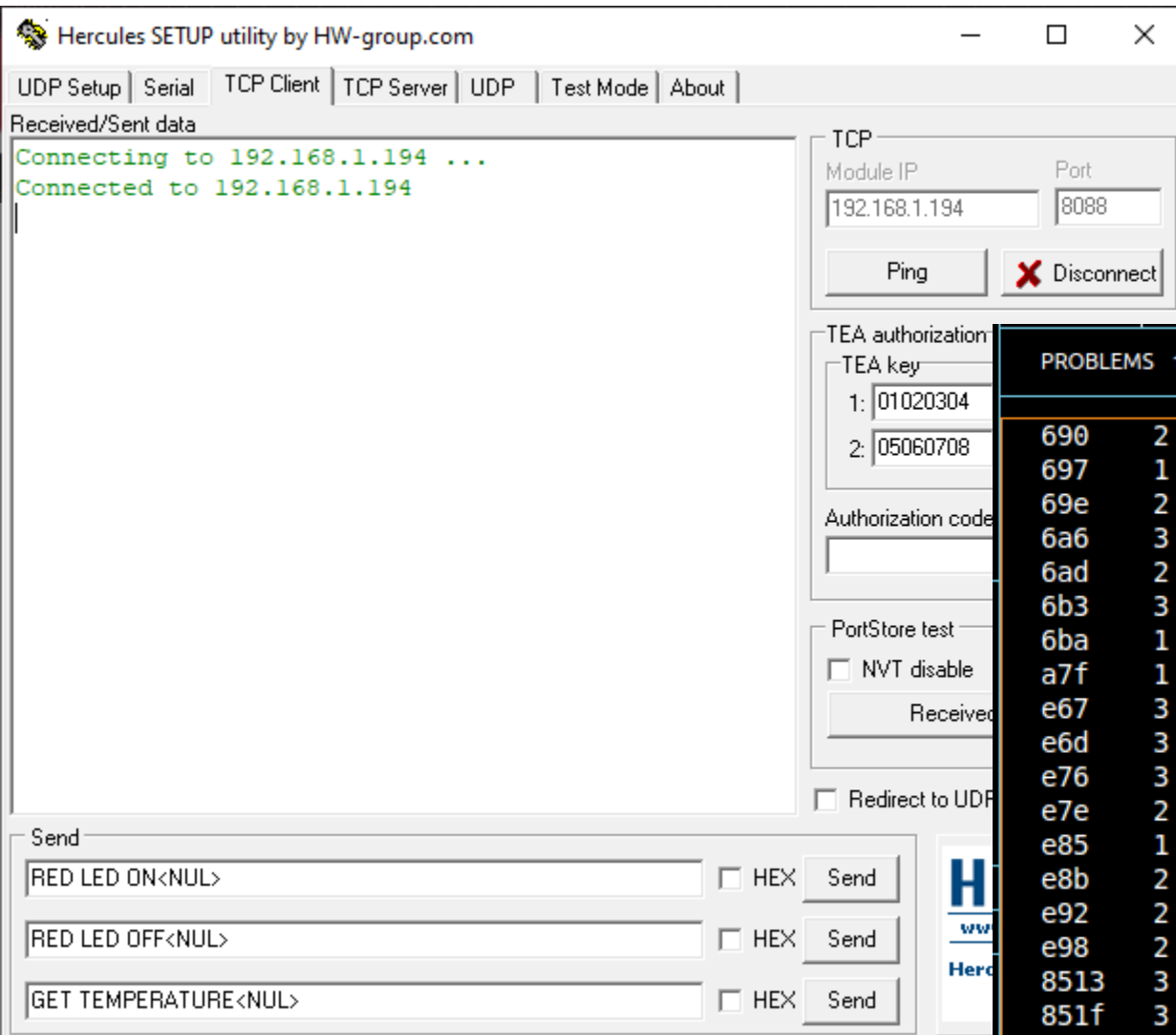
C main.c 9+ X

Core &gt; Src &gt; C main.c &gt; server\_fn

```
158 //*****
159 /* SERVER EVENT HANDLER - GET TEMPERATURE IN CELCIUS
160 //*****
161 else if(strncmp(c->recv.buf,"GET TEMPERATURE",16) == 0)
162 {
163     HAL_GPIO_TogglePin(GPIOx: LED2B_GPIO_Port, GPIO_Pin: LED2B_Pin);
164     temptosend = tmp102_read_temperature();
165     sprintf(c->recv.buf, "%.2fC", temptosend);
166     c->recv.len = strlen(c->recv.buf);
167     MG_INFO((fmt: "INFO: Current Temperature -> %.2f deg C\n\n",temptosend));
168 }
169 mg_send(c, c->recv.buf, c->recv.len);
170 c->recv.len = 0; // Tell Mongoose we've consumed data
171 }
172 else if (ev == MG_EV_CLOSE)
173 {
174     MG_INFO((fmt: "SERVER DISCONNECTED!"));
175 }
176 else if (ev == MG_EV_ERROR)
177 {
178     MG_INFO((fmt: "SERVER ERROR: %s", (char *) ev_data));
179 }
180 }
```



## Let's Put MAX to Work – Connect to the Server



```

PROBLEMS 17  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  STM32CUBE RTOS
690  2 mongoose.c:22174:mg_phy_init  PHY ID: 0x07 0xc131 (LAN87x)
697  1 mongoose.c:19853:mg_random  Weak RNG: using rand()
69e  2 mongoose.c:4139:mg_mgr_init  Driver: stm32f, MAC: 2a:d3:94:0a:5e:1e
6a6  3 mongoose.c:4146:mg_mgr_init  MG_IO_SIZE: 512, TLS: builtin
6ad  2 main.c:87:server_fn  INFO: SERVER IS LISTENING
6b3  3 mongoose.c:4063:mg_listen  1 0 tcp://0.0.0.0:8088
6ba  1 mongoose.c:5154:mg_tcpip_poll  Network is down
a7f  1 mongoose.c:5154:mg_tcpip_poll  Network is down
e67  3 mongoose.c:23871:mg_tcpip_driv  Link is 100M full-duplex
e6d  3 mongoose.c:4484:tx_dhcp_discov  DHCP discover sent. Our MAC: 2a:d3:94:0a:5e:1e
e76  3 mongoose.c:4463:tx_dhcp_reques  DHCP req sent
e7e  2 mongoose.c:4608:rx_dhcp_client  Lease: 86400 sec (86403)
e85  1 mongoose.c:19853:mg_random  Weak RNG: using rand()
e8b  2 mongoose.c:4366:onstatechange  READY, IP: 192.168.1.194
e92  2 mongoose.c:4367:onstatechange  GW: 192.168.1.1
e98  2 mongoose.c:4368:onstatechange  MAC: 2a:d3:94:0a:5e:1e
8513 3 mongoose.c:4521:rx_arp  ARP: tell 192.168.1.235 we're 2a:d3:94:0a:5e:1e
851f 3 mongoose.c:4756:accept_conn  2 accepted 192.168.1.235:63625
8526 2 main.c:91:server_fn  INFO: SERVER ACCEPTED A CONNECTION
  
```

# Let's Put MAX to Work – Control the RED LED

Hercules SETUP utility by HW-group.com

UDP Setup | Serial | TCP Client | TCP Server | UDP | Test Mode | About

Received/Sent data

```
Connecting to 192.168.1.194 ...
Connected to 192.168.1.194
RED LED ONRED LED ON|
```

TCP

Module IP: 192.168.1.194 Port: 8088

Ping Disconnect

TEA authorization

TEA key

1: 01020304 3: 090A0B0C  
 2: 05060708 4: 0D0E0F10

Authorization code

PortStore test

NVT disable

Received test data

Redirect to UDP

Send

RED LED ON<NUL>  HEX Send

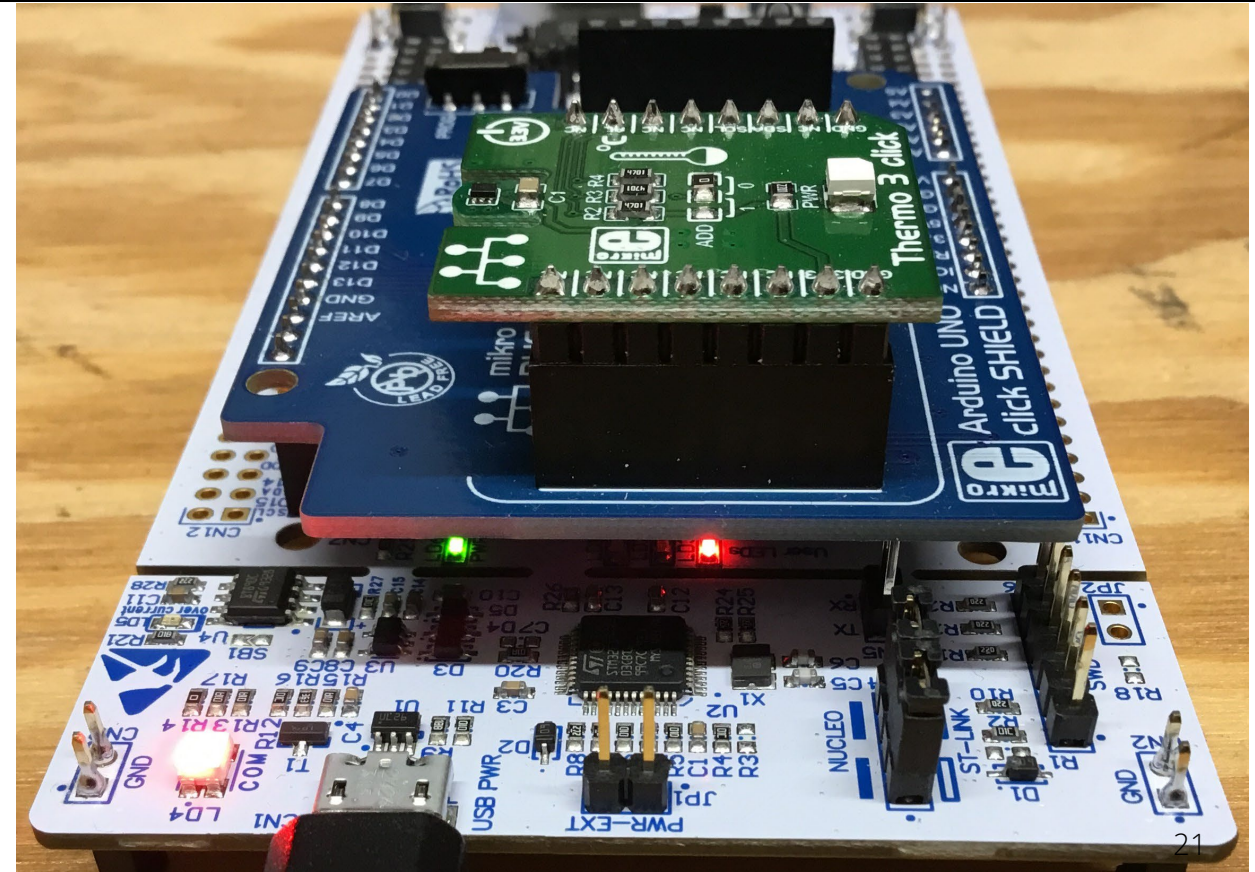
RED LED OFF<NUL>  HEX Send

GET TEMPERATURE<NUL>  HEX Send

HWgroup  
 www.HW-group.com  
 Hercules SETUP utility  
 Version 3.2.8

PROBLEMS 17 OUTPUT DEBUG CONSOLE **TERMINAL** PORTS STM32CUBE RTOS

```
77b8e 2 main.c:95:server_fn INFO: SERVER RECEIVED DATA: RED LED ON
77b96 2 main.c:125:server_fn INFO: RED LED IS ON!
```



# Let's Put MAX to Work – Get the Temperature

Hercules SETUP utility by HW-group.com

UDP Setup | Serial | TCP Client | TCP Server | UDP | Test Mode | About

Received/Sent data

```
Connecting to 192.168.1.194 ...
Connected to 192.168.1.194
RED LED ONRED LED ONGET TEMPERATURE26.31C
```

TCP Module IP: 192.168.1

Ping Disconnect

TEA authorization

TEA key

1: 01020304 3: 090A0B0C  
 2: 05060708 4: 0D0E0F10

Authorization code

PortStore test

NVT disable

Received test data

Redirect to UDP

Send

RED LED ON<NUL>  HEX Send

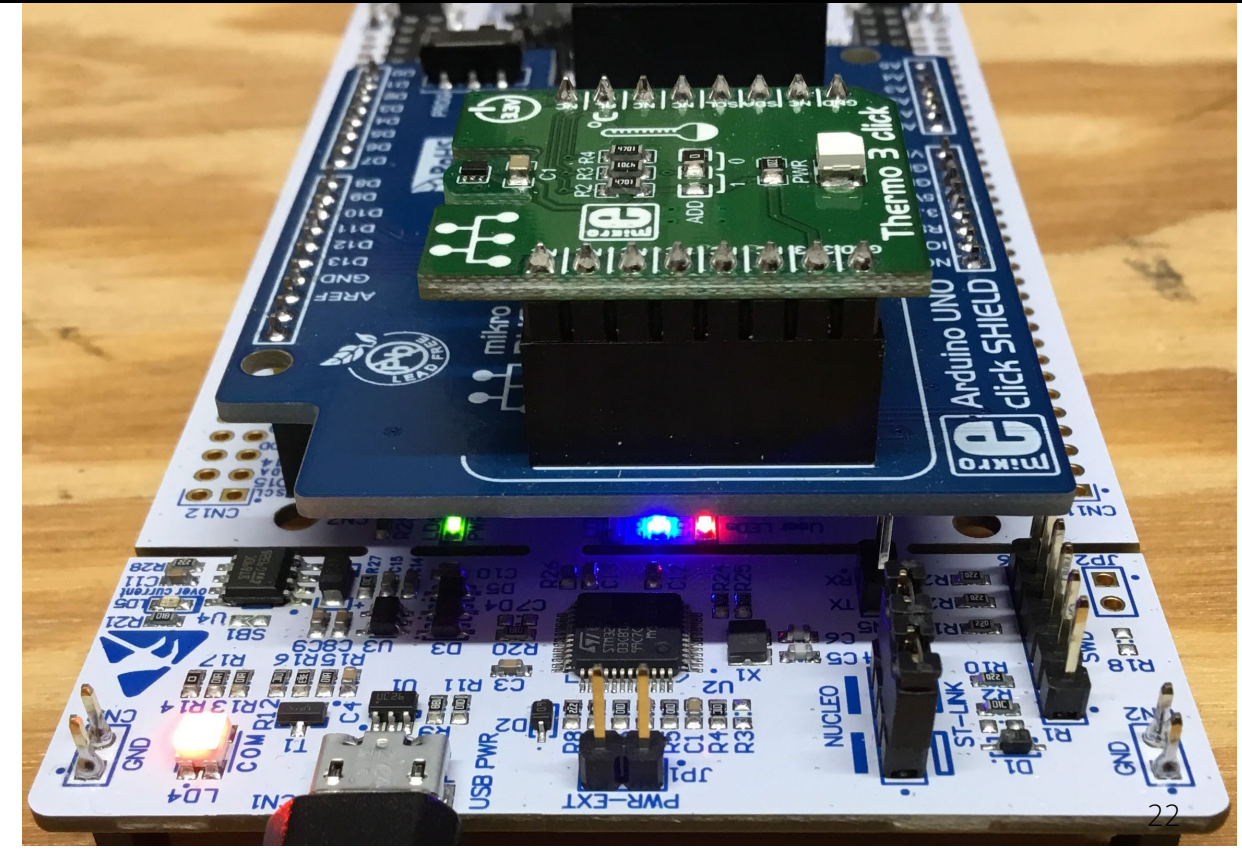
RED LED OFF<NUL>  HEX Send

GET TEMPERATURE<NUL>  HEX Send

**HWgroup**  
 www.HW-group.com  
 Hercules SETUP utility  
 Version 3.2.8

PROBLEMS 17 OUTPUT DEBUG CONSOLE **TERMINAL** PORTS STM32CUBE RTOS

```
bc466 2 main.c:95:server_fn INFO: SERVER RECEIVED DATA: GET TEMPERATURE
bc46e 2 tmp102.c:82:tmp102_read_temper INFO: 2 bytes were read-> rxBuf[0] 0x1a -- rxBuf[1] 0x50
bc478 2 tmp102.c:87:tmp102_read_temper INFO: 16-bit value = 0x01a5
bc47f 2 main.c:167:server_fn INFO: Current Temperature -> 26.31 deg C
```



## Next Time...

## MORE TO COME..

# Thank you for attending!!!

Please consider the resources below:

- [Today's Download Package](#)
- [www.st.com](http://www.st.com)
- [digikey.com](http://digikey.com)

```

0xDFFF FFFF
0xD000 0000
0xCFFF FFFF
0xC000 0000

0x9FFF FFFF
0x9000 0000
0x8FFF FFFF
0x8000 0000
0x7FFF FFFF
0x7000 0000
0x6FFF FFFF
0x6000 0000

```

SDRAM2
SDRAM1 REMAP NOR
XSPI1
NAND
XSPI2
NOR / SRAM



Thank You

Sponsored by

