



DesignNews

Embedded Studio Primer

DAY 2: Embedded Studio and the STM32

Sponsored by



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 ‘Group Chat’ by maximizing the chat widget in your dock.
- Submit questions for the lecturer using the Q&A widget. They will follow-up after the lecture portion concludes.

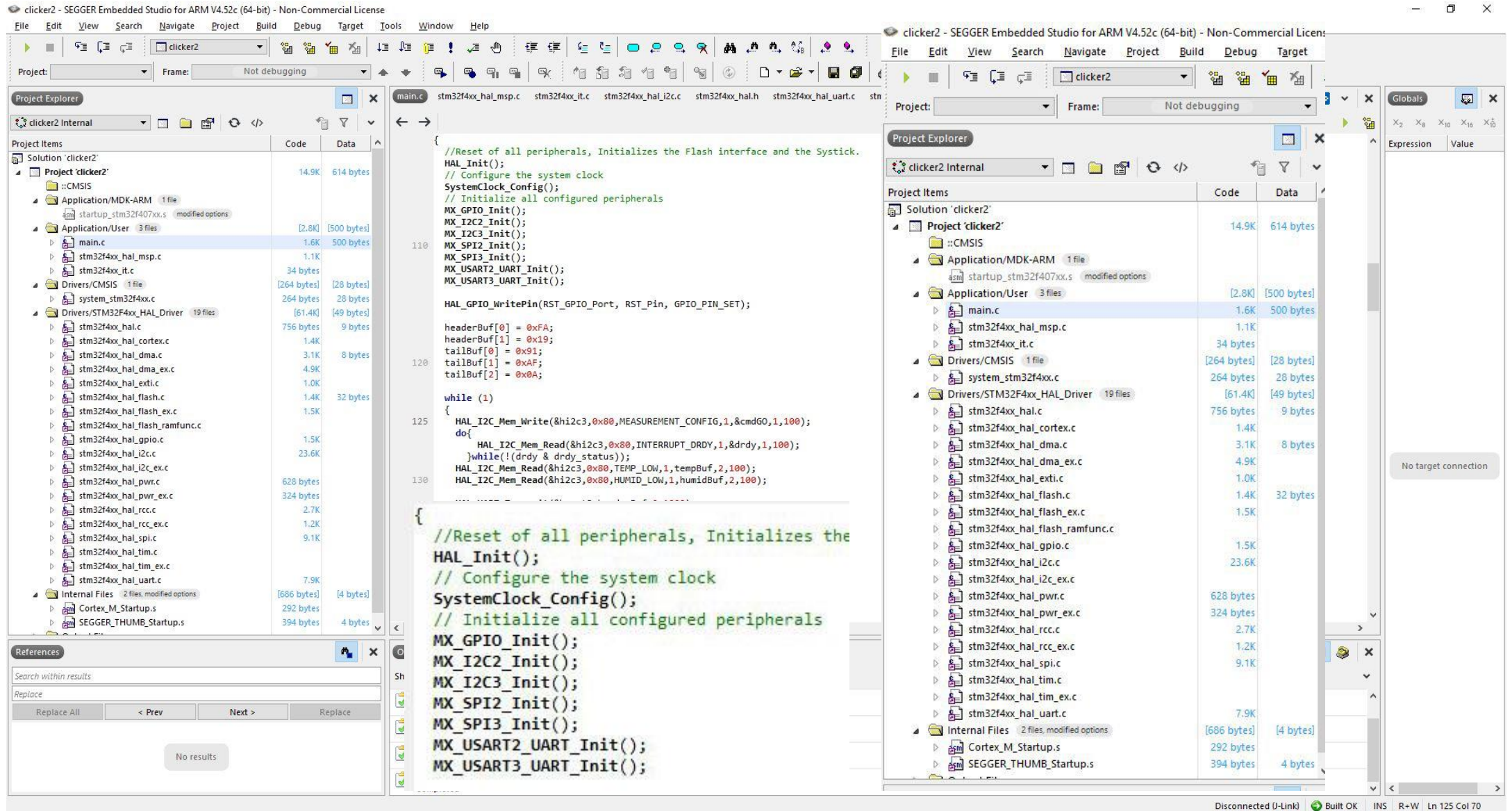


Fred Eady

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

Embedded Studio Primer

Embedded Studio and the STM32



The screenshot displays the SEGGER Embedded Studio interface for ARM V4.52c. The main window shows the source code for a project named 'clicker2'. The code includes initialization functions for HAL, system clock, and various peripherals (GPIO, I2C, SPI, USART). A while loop is present, likely for a measurement or data reading process.

```

//Reset of all peripherals, Initializes the Flash interface and the SysTick.
HAL_Init();
// Configure the system clock
SystemClock_Config();
// Initialize all configured peripherals
MX_GPIO_Init();
MX_I2C2_Init();
MX_I2C3_Init();
MX_SPI2_Init();
MX_SPI3_Init();
MX_USART2_UART_Init();
MX_USART3_UART_Init();

HAL_GPIO_WritePin(RST_GPIO_Port, RST_Pin, GPIO_PIN_SET);

headerBuf[0] = 0xFA;
headerBuf[1] = 0x19;
tailBuf[0] = 0x91;
tailBuf[1] = 0xAF;
tailBuf[2] = 0x0A;

while (1)
{
    HAL_I2C_Mem_Write(&hi2c3,0x80,MEASUREMENT_CONFIG,1,&cmdGO,1,100);
    do{
        HAL_I2C_Mem_Read(&hi2c3,0x80,INTERRUPT_DRDY,1,&drdy,1,100);
    }while(!(&drdy & drdy_status));
    HAL_I2C_Mem_Read(&hi2c3,0x80,TEMP_LOW,1,tempBuf,2,100);
    HAL_I2C_Mem_Read(&hi2c3,0x80,HUMID_LOW,1,humidBuf,2,100);
}

//Reset of all peripherals, Initializes the
HAL_Init();
// Configure the system clock
SystemClock_Config();
// Initialize all configured peripherals
MX_GPIO_Init();
MX_I2C2_Init();
MX_I2C3_Init();
MX_SPI2_Init();
MX_SPI3_Init();
MX_USART2_UART_Init();
MX_USART3_UART_Init();
    
```

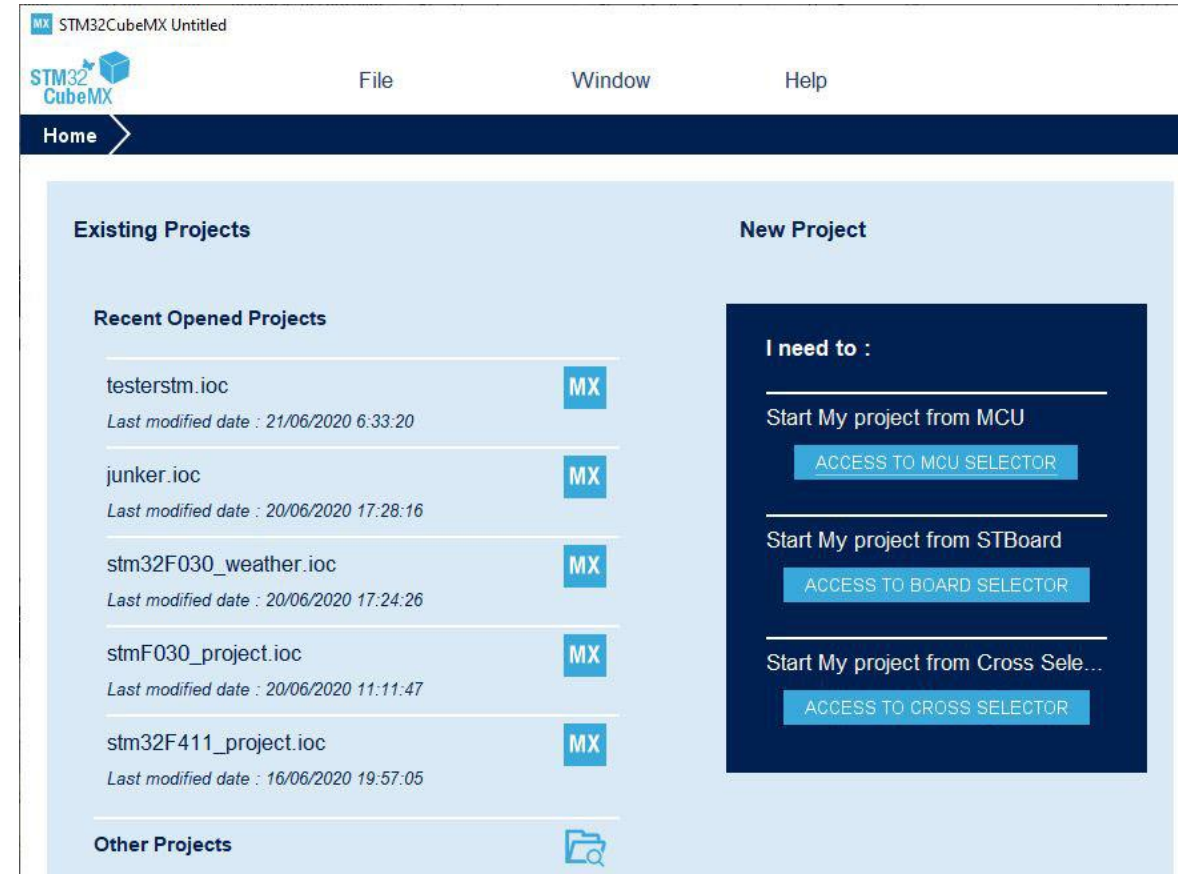
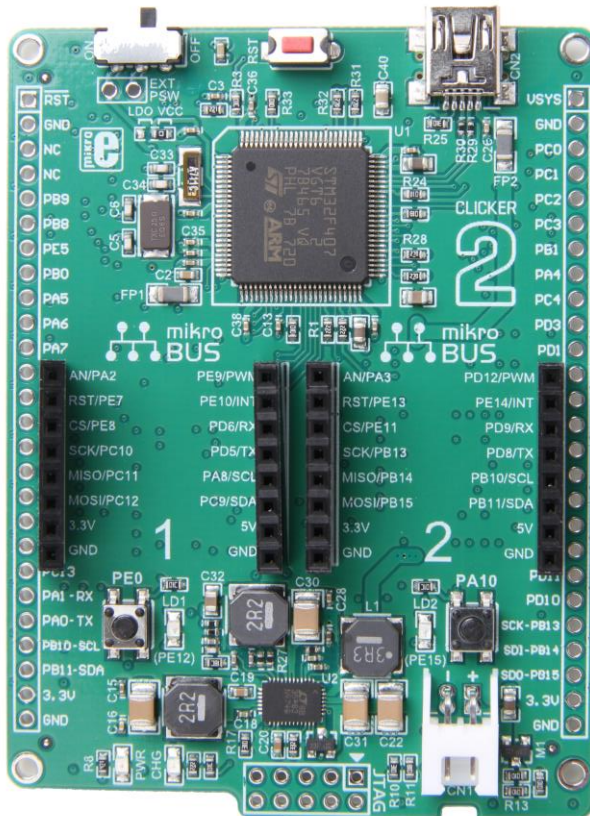
The Project Explorer on the left shows the project structure for 'clicker2', including folders for CMSIS, Application/MDK-ARM, Application/User, Drivers/CMSIS, and Drivers/STM32F4xx_HAL_Driver. The References window at the bottom left shows a search for 'Replace' with no results.

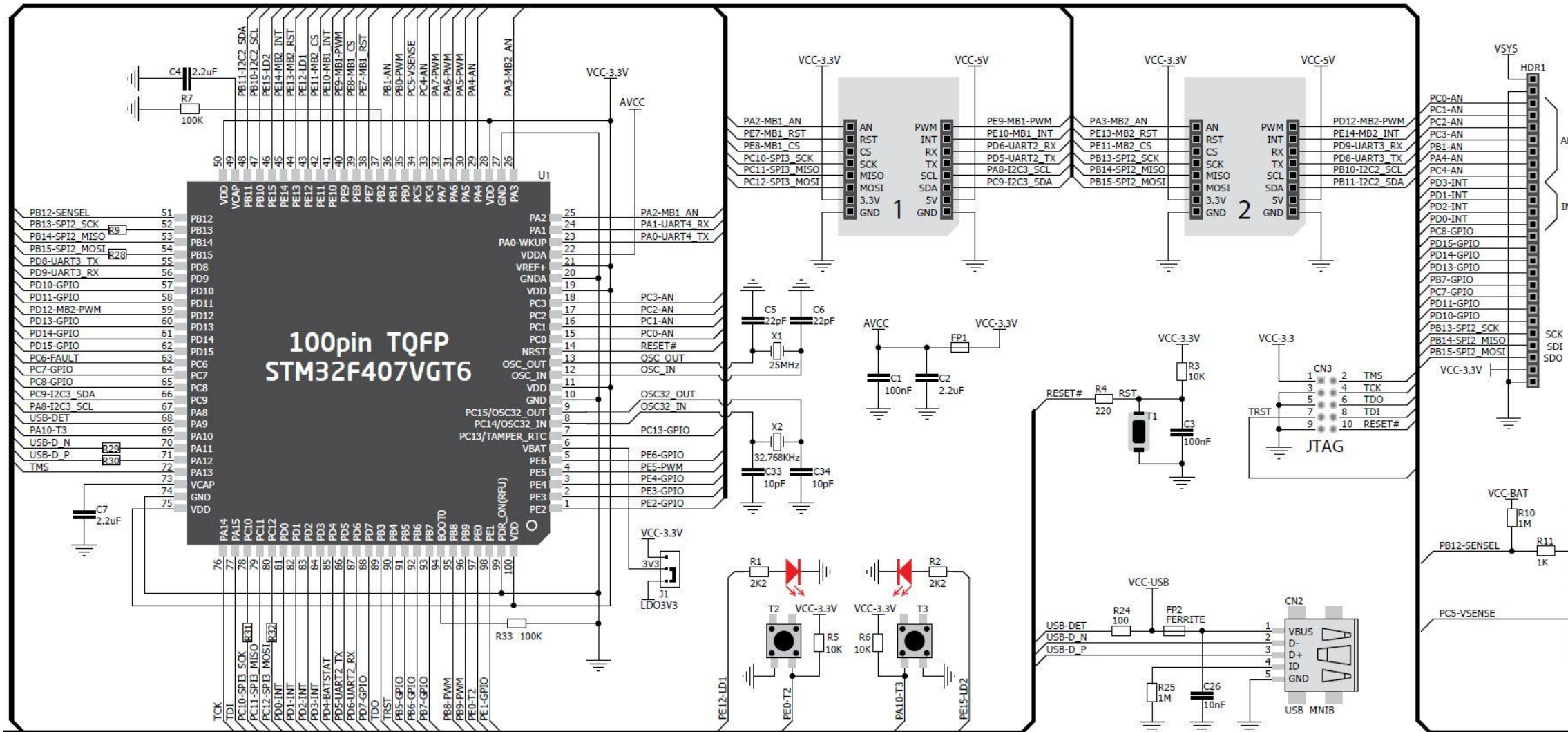
The Project Explorer on the right shows a table of project items with their code and data sizes:

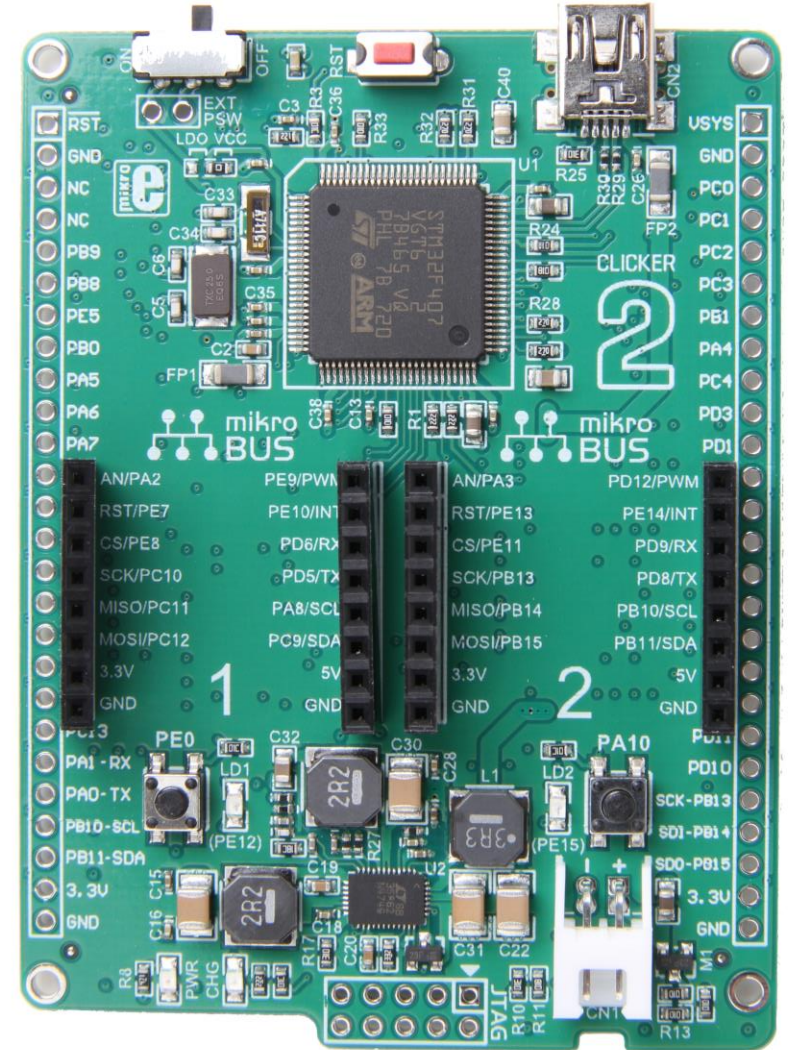
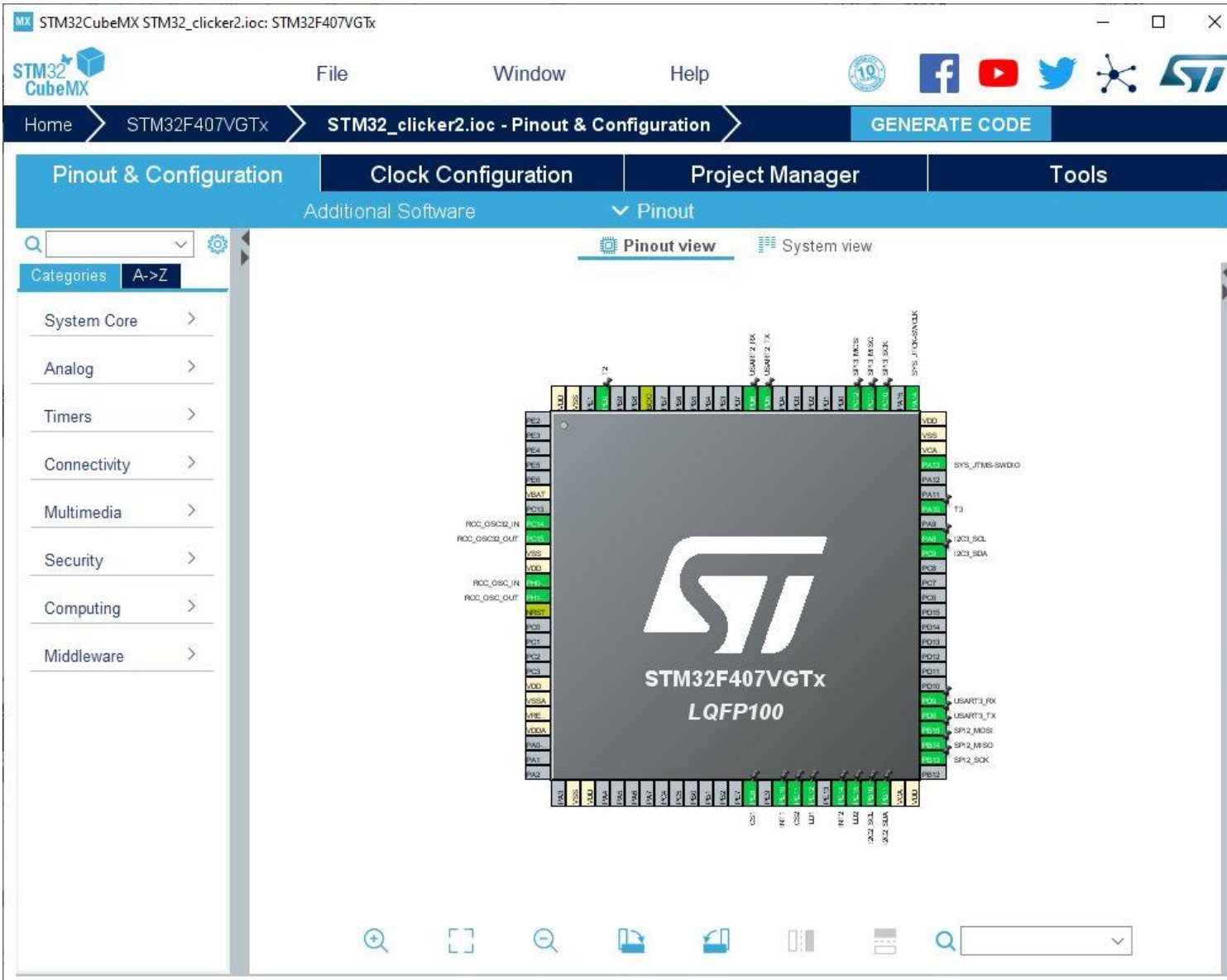
Project Item	Code	Data
Solution 'clicker2'	14.9K	614 bytes
Project 'clicker2'	14.9K	614 bytes
Application/MDK-ARM	1 file	
Application/User	3 files	[500 bytes]
main.c	1.6K	500 bytes
stm32f4xx_hal_msp.c	1.1K	
stm32f4xx_it.c	34 bytes	
Drivers/CMSIS	1 file	[264 bytes]
system_stm32f4xx.c	264 bytes	28 bytes
Drivers/STM32F4xx_HAL_Driver	19 files	[61.4K]
stm32f4xx_hal.c	756 bytes	9 bytes
stm32f4xx_hal_cortex.c	1.4K	
stm32f4xx_hal_dma.c	3.1K	8 bytes
stm32f4xx_hal_dma_ex.c	4.9K	
stm32f4xx_hal_exti.c	1.0K	
stm32f4xx_hal_flash.c	1.4K	32 bytes
stm32f4xx_hal_flash_ex.c	1.5K	
stm32f4xx_hal_flash_ramfunc.c	1.5K	
stm32f4xx_hal_gpio.c	1.5K	
stm32f4xx_hal_i2c.c	23.6K	
stm32f4xx_hal_i2c_ex.c	4.9K	
stm32f4xx_hal_pwr.c	628 bytes	
stm32f4xx_hal_pwr_ex.c	324 bytes	
stm32f4xx_hal_rcc.c	2.7K	
stm32f4xx_hal_rcc_ex.c	1.2K	
stm32f4xx_hal_spi.c	9.1K	
stm32f4xx_hal_tim.c	7.9K	
stm32f4xx_hal_tim_ex.c	7.9K	
stm32f4xx_hal_uart.c	7.9K	
Internal Files	2 files, modified options	[686 bytes]
Cortex_M_Startup.s	292 bytes	
SEGGER_THUMB_Startup.s	394 bytes	4 bytes

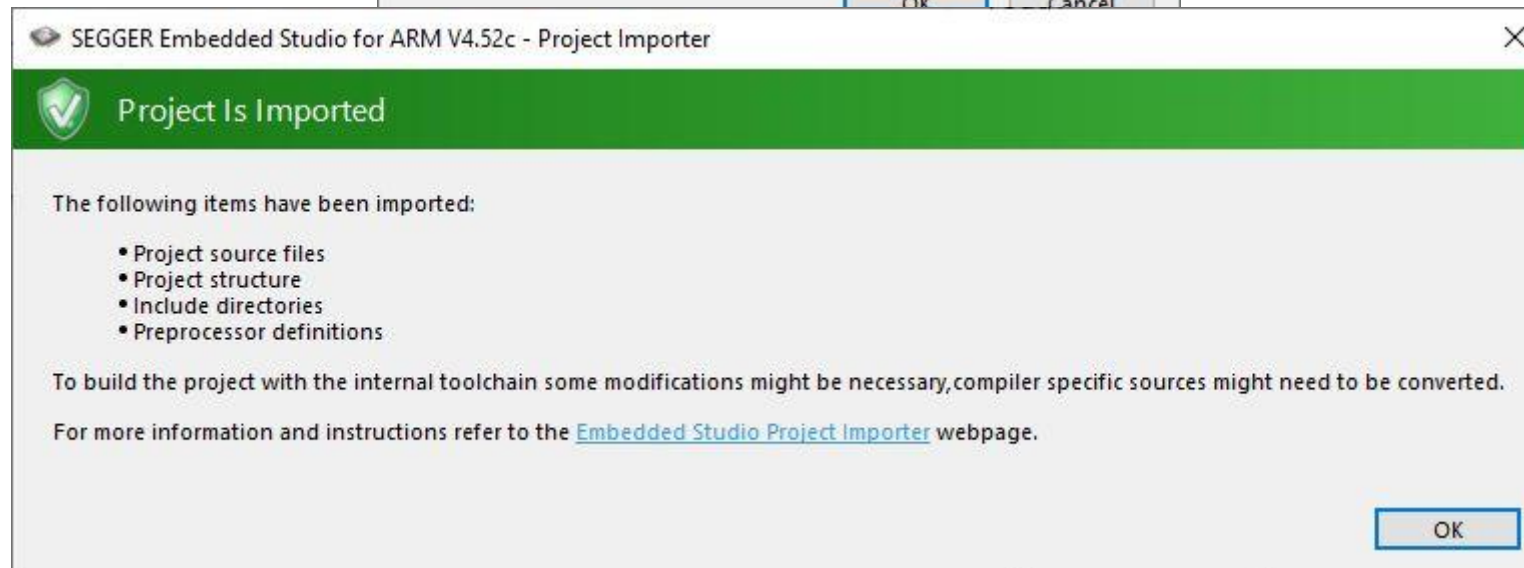
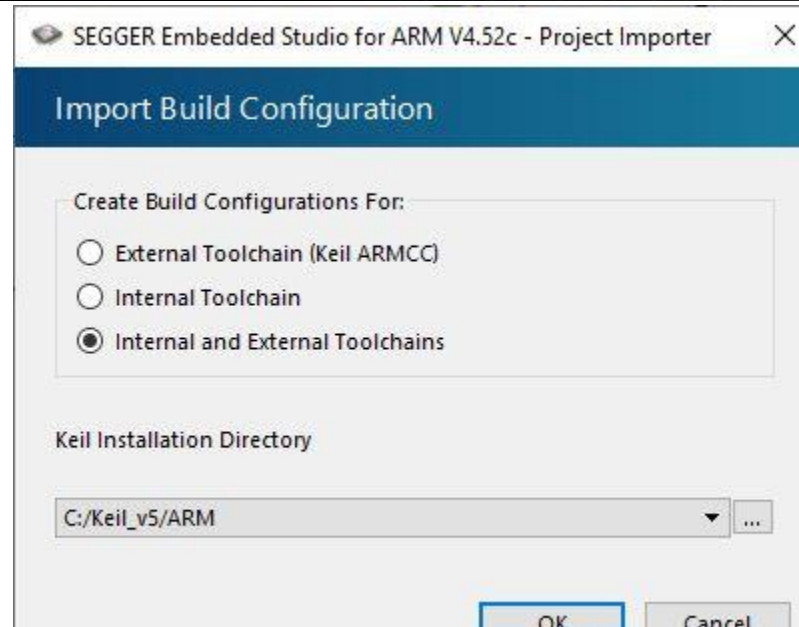
AGENDA

- **Importing a Keil/STM32CubeMX Project**
- **Coding the Imported Keil Project**
- **Weather Over Wi-Fi**









The screenshot displays the SEGGER Embedded Studio interface. At the top, the title bar reads "clicker2 - SEGGER Embedded Studio for ARM V4.52c (64-bit) - Non-Commercial License". The menu bar includes File, Edit, View, Search, Navigate, Project, Build, Debug, Target, Tools, Window, and Help. The toolbar contains various icons for project management and debugging.

The main workspace shows the "clicker2 - SEGGER Embedded Studio for ARM" splash screen with a central chip icon and the text "SEGGER Embedded Studio". Below this, there are two sections: "clicker2 External" and "clicker2 Internal", each with a "Check for Updates" button and a toggle switch. The "clicker2 Internal" section also has a "Check for Packages" button and a toggle switch.

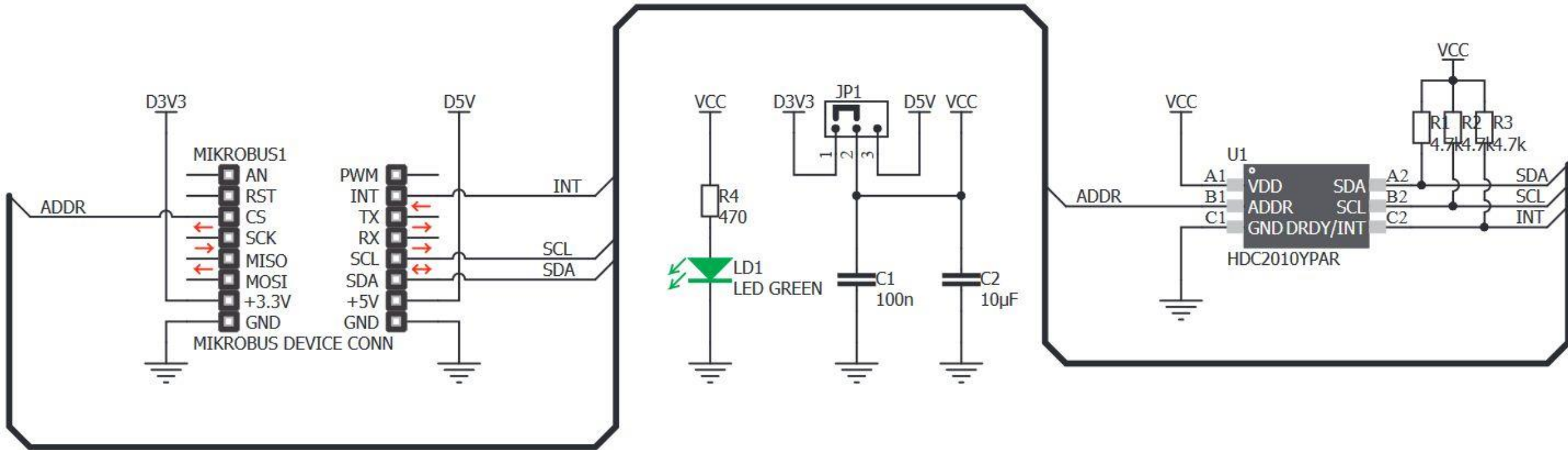
The left sidebar shows a file explorer with the following structure:

- clicker2 External
- clicker2 Internal
- <Edit Build Configurations...>
- ::CMSIS
- Application/MDK-ARM (1 file)
 - startup_stm32f407xx.s (modified options)
- Application/User (3 files)
 - main.c
 - stm32f4xx_hal_msp.c
 - stm32f4xx_it.c
- Drivers/CMSIS (1 file)
 - system_stm32f4xx.c
- Drivers/STM32F4xx_HAL_Driver (19 files)
 - stm32f4xx_hal.c
 - stm32f4xx_hal_cortex.c
 - stm32f4xx_hal_dma.c
 - stm32f4xx_hal_dma_ex.c
 - stm32f4xx_hal_exti.c
 - stm32f4xx_hal_flash.c
 - stm32f4xx_hal_flash_ex.c
 - stm32f4xx_hal_flash_ramfunc.c
 - stm32f4xx_hal_gpio.c
 - stm32f4xx_hal_i2c.c
 - stm32f4xx_hal_i2c_ex.c
 - stm32f4xx_hal_pwr.c
 - stm32f4xx_hal_pwr_ex.c
 - stm32f4xx_hal_rcc.c
 - stm32f4xx_hal_rcc_ex.c
 - stm32f4xx_hal_spi.c
 - stm32f4xx_hal_tim.c
 - stm32f4xx_hal_tim_ex.c
 - stm32f4xx_hal_uart.c
- Internal Files (2 files, modified options)
 - Cortex_M_Startup.s
 - SEGGER_THUMB_Startup.s

The bottom status bar shows the following information:

- Show: Transcript
- Tasks
- Loading solution clicker2.emProject Completed (6 files in 0.0s, 6000 files/s)
- Mapping project information Completed
- Preparing solution 'clicker2' Completed
- Restoring state from previous session Completed
- SEGGER Embedded Studio is ready to use Completed
- Disconnected (J-Link) Built OK INS (No editor)

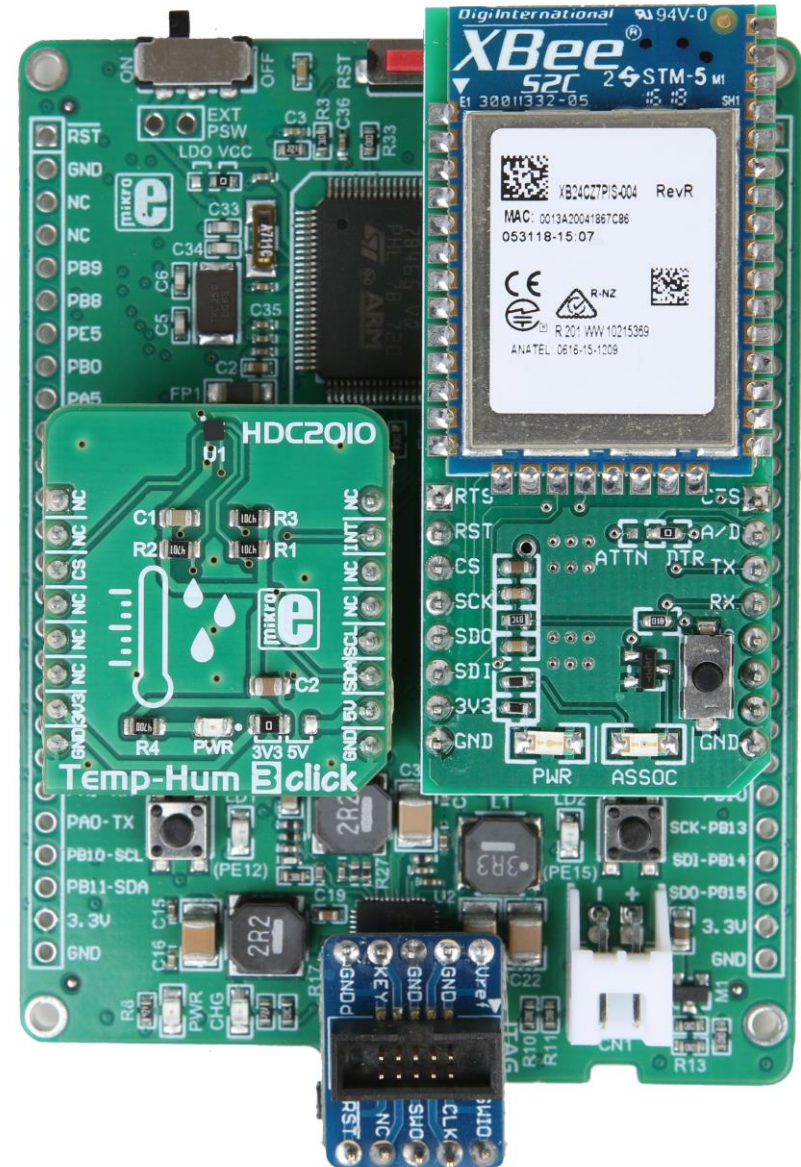
The right sidebar contains a "Projects" list with buttons for "Open existing" and "Create new". The list includes "clicker2", "samL11_Project", "TeensyProject", "Start_SAML11E16", "samD21_Project", "same54_Project", and "SAMD21 Samples". Below the projects list is a "Search Symbols" section with a list of symbols including "uint_fast8_t", "uint_least16_t", "uint_least32_t", "uint_least64_t", "uint16_t", "uint32_t", "uint64_t", "uint8_t", "uintmax_t", "uintptr_t", "USART_TypeDef", "USB_OTG_DeviceTypeDef", "USB_OTG_GlobalTypeDef", "USB_OTG_HostChannelT", "USB_OTG_HostTypeDef", "USB_OTG_INEndpointTyp", "USB_OTG_OUTEndpoint", "wchar_t", "WWDG_TypeDef", and "xPSR_Type".



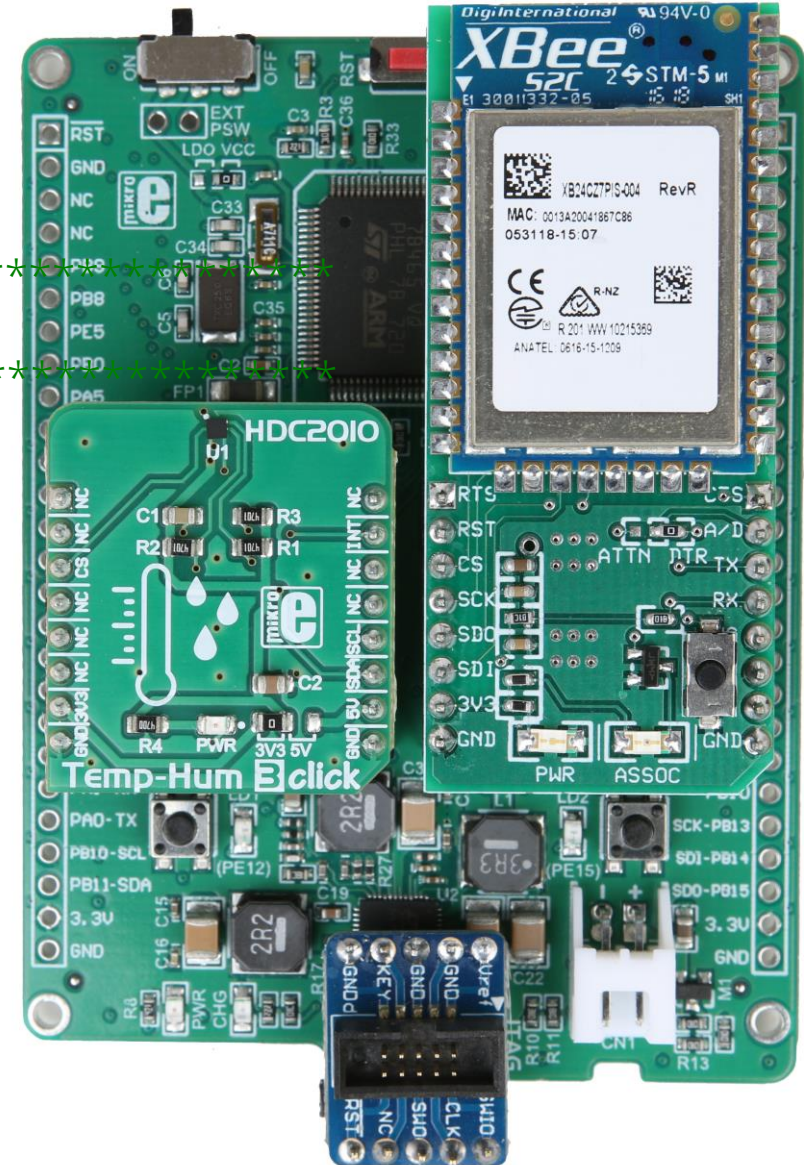
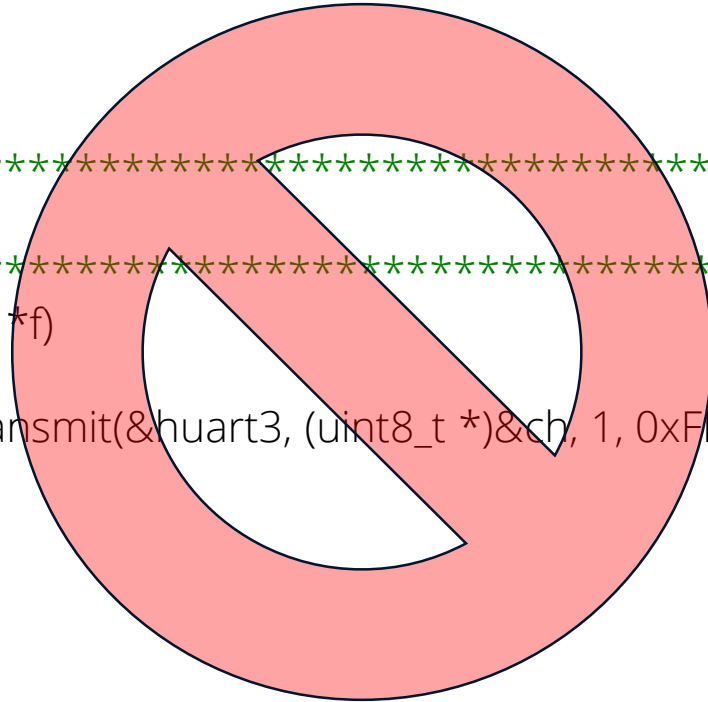

```

static void MX_USART3_UART_Init(void)
{
    huart3.Instance = USART3;
    huart3.Init.BaudRate = 9600;
    huart3.Init.WordLength = UART_WORDLENGTH_8B;
    huart3.Init.StopBits = UART_STOPBITS_1;
    huart3.Init.Parity = UART_PARITY_NONE;
    huart3.Init.Mode = UART_MODE_TX_RX;
    huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart3.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart3) != HAL_OK)
    {
        Error_Handler();
    }
}

```




```
/**  
** PRINTF KEIL  
**  
int fputc(int ch, FILE *f)  
{  
    HAL_UART_Transmit(&huart3, (uint8_t *)&ch, 1, 0xFFFF);  
    return ch;  
}
```



```

//*****
//* PRINTF STUFF
//*****
typedef struct __printf_tag
{
    size_t charcount;
    size_t maxchars;
    char *string;
    int (*output_fn)(int, struct __printf_tag *ctx);
} __printf_t;

void uart3_putc(char ch)
{
    HAL_UART_Transmit(&huart3, (uint8_t *)&ch, 1, 0xFFFF);
}

int uart3_printf(const char *fmt, ...)
{
    int n;
    va_list ap;
    __printf_t iod;
    va_start(ap, fmt);
    iod.string = 0;
    iod.maxchars = INT_MAX;
    iod.output_fn = uart3_putc;
    n = __vfprintf(&iod, fmt, ap);
    va_end(ap);
    return n;
}

```

SEGGER Embedded Studio for ARM printf-style output

SEGGER Embedded Studio for ARM provides a solution for just this case by using some internal functions and data types in the SEGGER Embedded Studio for ARM library. These functions and types are define in the header file <__vfprintf.h>.

The first thing to introduce is the **__printf_t** type which captures the current state and parameters of the format conversion:

```

typedef struct __printf_tag
{
    size_t charcount;
    size_t maxchars;
    char *string;
    int (*output_fn)(int, struct __printf_tag *ctx);
} __printf_t;

```

This type is used by the library functions to direct what the formatting routines do with each character they need to output. If **string** is non-zero, the character is appended to the string pointed to by **string**; if **output_fn** is non-zero, the character is output through the function **output_fn** with the context passed as the second parameter.

The member **charcount** counts the number of characters currently output, and **maxchars** defines the maximum number of characters output by the formatting routine **__vfprintf**.

We can use this type and function to rewrite **uart0_printf**:

```

int uart0_printf(const char *fmt, ...)
{
    int n;
    va_list ap;
    __printf_t iod;
    va_start(ap, fmt);
    iod.string = 0;
    iod.maxchars = INT_MAX;
    iod.output_fn = uart0_putc;
    n = __vfprintf(&iod, fmt, ap);
    va_end(ap);
    return n;
}

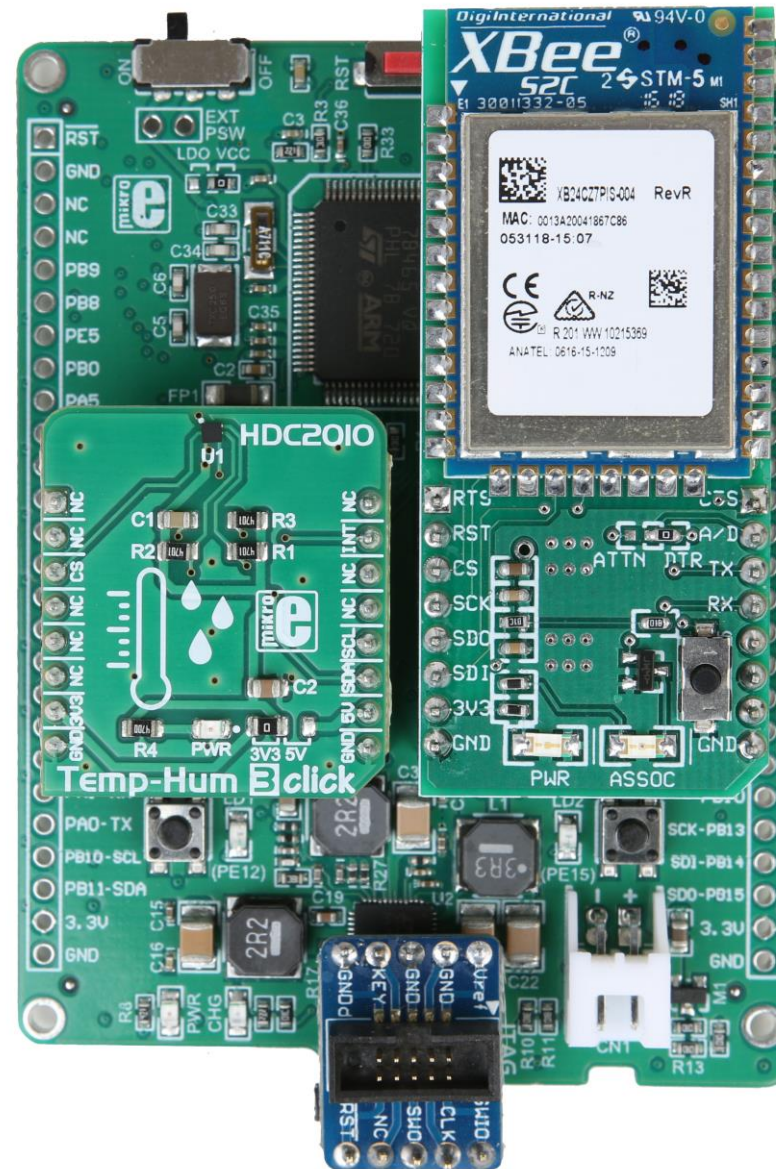
```



```

static void MX_I2C2_Init(void)
{
    hi2c2.Instance = I2C2;
    hi2c2.Init.ClockSpeed= 100000;
    hi2c2.Init.DutyCycle = I2C_DUTYCYCLE_2;
    hi2c2.Init.OwnAddress1 = 0;
    hi2c2.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
    hi2c2.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
    hi2c2.Init.OwnAddress2 = 0;
    hi2c2.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
    hi2c2.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
    if (HAL_I2C_Init(&hi2c2) != HAL_OK)
    {
        Error_Handler();
    }
}

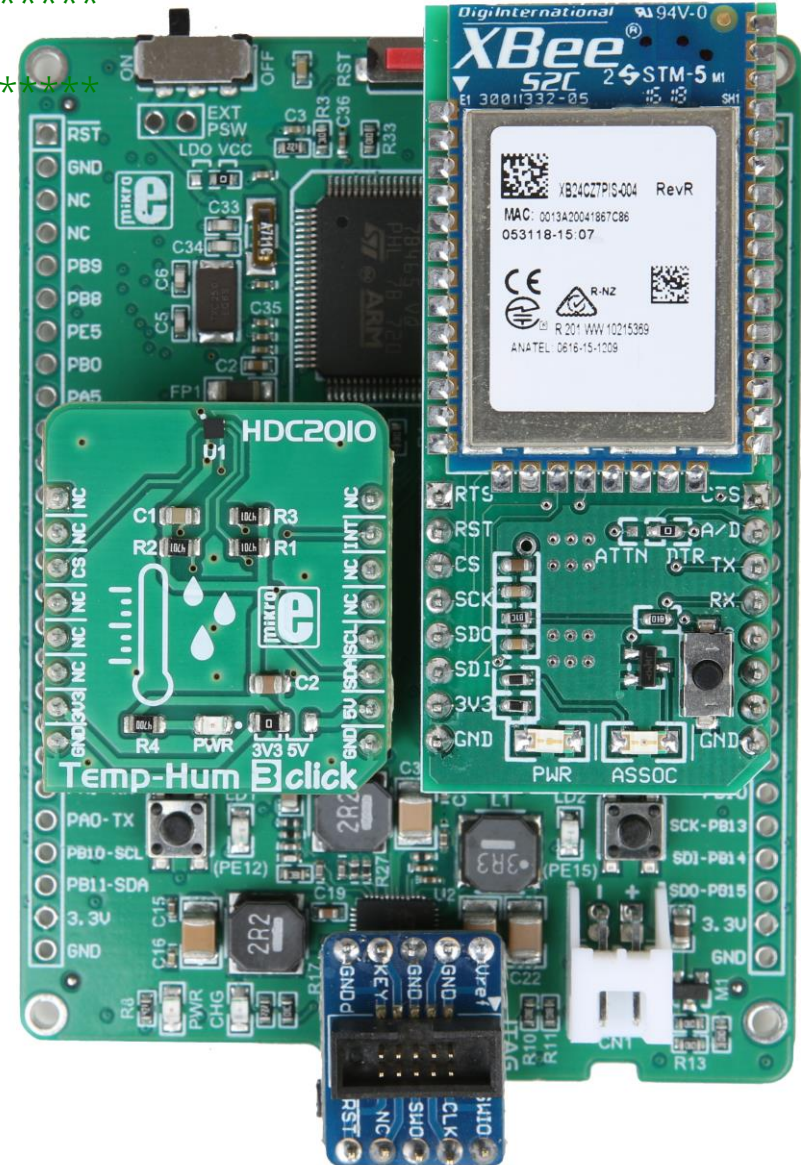
```



```

//*****
/** MAIN
//*****
int main(void)
{
//Reset of all peripherals, Initializes the Flash interface and the Systick.
HAL_Init();
// Configure the system clock
SystemClock_Config();
// Initialize all configured peripherals
MX_GPIO_Init();
MX_I2C2_Init();
MX_I2C3_Init();
MX_SPI2_Init();
MX_SPI3_Init();
MX_USART2_UART_Init();
MX_USART3_UART_Init();
//Drive XBee RESET pin HI
HAL_GPIO_WritePin(RST_GPIO_Port, RST_Pin, GPIO_PIN_SET);
//packet header and footer
headerBuf[0] = 0xFA;
headerBuf[1] = 0x19;
tailBuf[0] = 0x91;
tailBuf[1] = 0xAF;
tailBuf[2] = 0x0A;

```




```

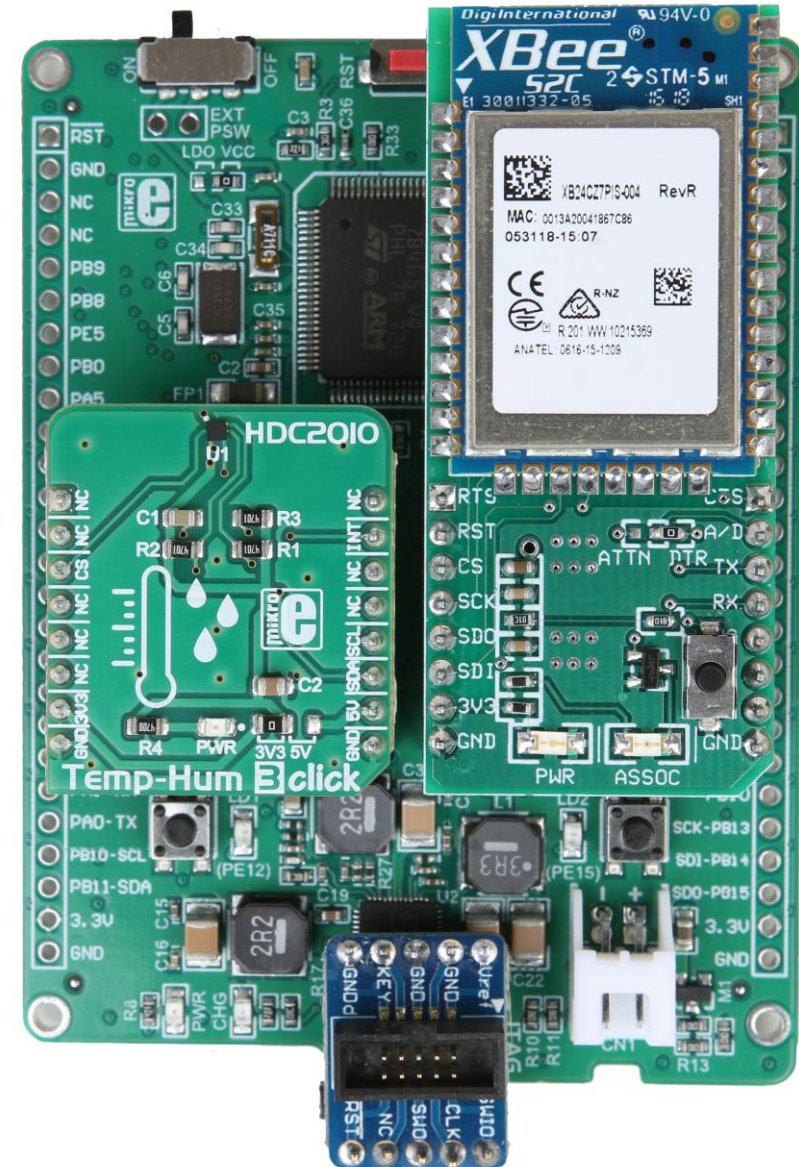
while (1)
{
  HAL_I2C_Mem_Write(&hi2c3,0x80,MEASUREMENT_CONFIG,1,&cmdGO,1,100);
  do{
    HAL_I2C_Mem_Read(&hi2c3,0x80,INTERRUPT_DRDY,1,&drdy,1,100);
  }while(!((drdy & drdy_status));
  HAL_I2C_Mem_Read(&hi2c3,0x80,TEMP_LOW,1,tempBuf,2,100);
  HAL_I2C_Mem_Read(&hi2c3,0x80,HUMID_LOW,1,humidBuf,2,100);

  HAL_UART_Transmit(&huart3,headerBuf,2,1000);
  HAL_UART_Transmit(&huart3,tempBuf,2,1000);
  HAL_UART_Transmit(&huart3,humidBuf,2,1000);
  HAL_UART_Transmit(&huart3,tailBuf,3,1000);

  tempVal = make16(tempBuf[1],tempBuf[0]);
  humidVal = make16(humidBuf[1],humidBuf[0]);
  tempC = (((float)tempVal/65536)*165)-40;
  tempF = ((float)tempC * 1.8)+32;
  humidity = ((float)humidVal/65536)*100;

  uart3_printf("\r\nTemperature C = %3.2f\r\n",tempC);
  HAL_Delay(10);
  uart3_printf("Temperature F = %3.2f\r\n",tempF);
  HAL_Delay(10);
  uart3_printf("Humidity = %3.2f%%\r\n\r\n",humidity);
  HAL_Delay(1000);
}
}

```






```

while (1)
{
  HAL_I2C_Mem_Write(&hi2c3,0x80,MEASUREMENT_CONFIG,1,&cmdGO,1,100);
  do{
    HAL_I2C_Mem_Read(&hi2c3,0x80,INTERRUPT_DRDY,1,&drdy,1,100);
  }while(!((drdy & drdy_status));
  HAL_I2C_Mem_Read(&hi2c3,0x80,TEMP_LOW,1,tempBuf,2,100);
  HAL_I2C_Mem_Read(&hi2c3,0x80,HUMID_LOW,1,humidBuf,2,100);

```

```

  HAL_UART_Transmit(&huart3,headerBuf,2,1000);
  HAL_UART_Transmit(&huart3,tempBuf,2,1000);
  HAL_UART_Transmit(&huart3,humidBuf,2,1000);
  HAL_UART_Transmit(&huart3,tailBuf,3,1000);

```

```

  tempVal = make16(tempBuf[1],tempBuf[0]);
  humidVal = make16(humidBuf[1],humidBuf[0]);
  tempC = (((float)tempVal/65536)*165)-40;
  tempF = ((float)tempC * 1.8)+32;
  humidity = ((float)humidVal/65536)*100;

```

```

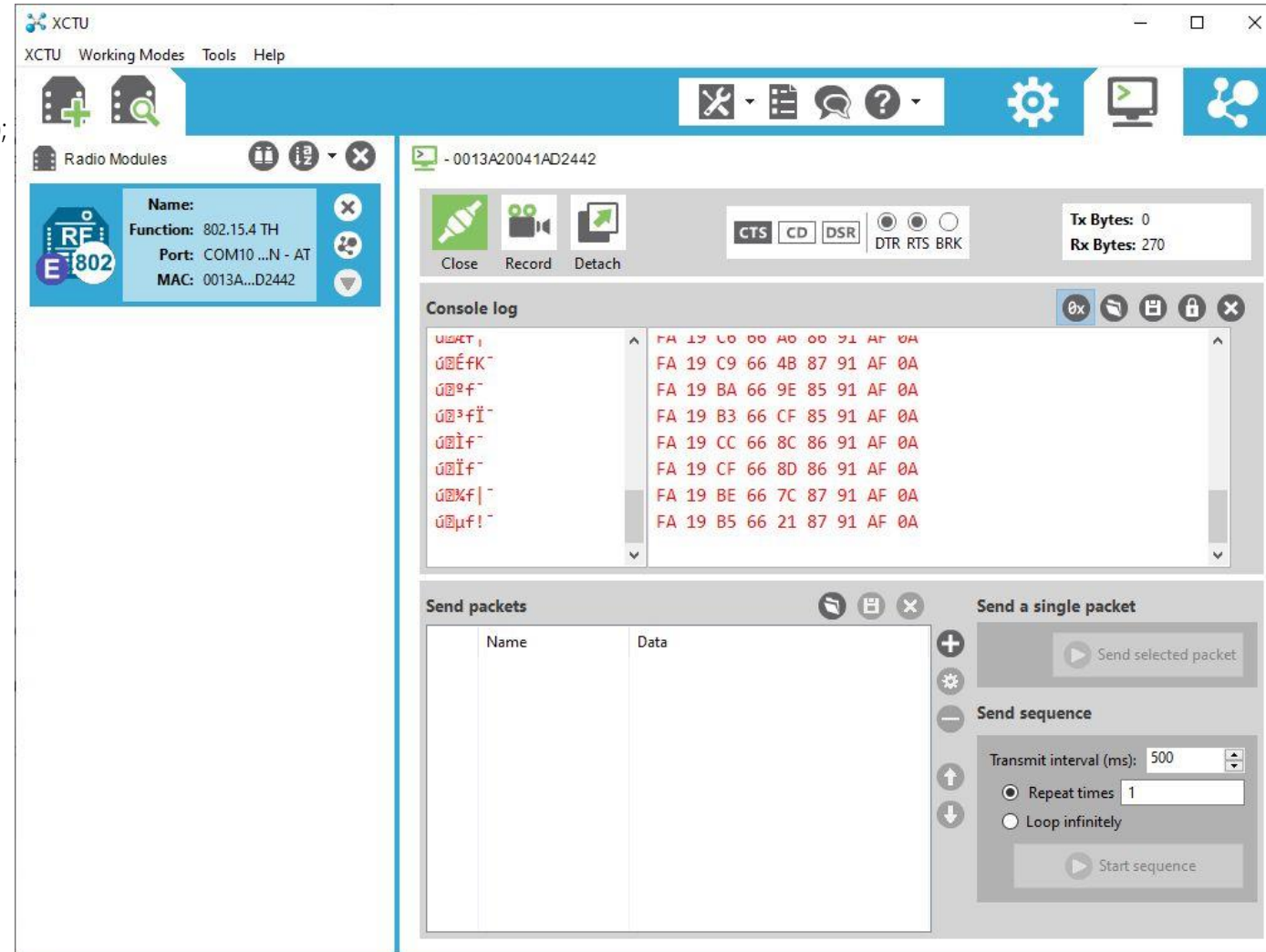
  //uart3_printf("\r\nTemperature C = %3.2f\r\n",tempC);
  //HAL_Delay(10);
  //uart3_printf("Temperature F = %3.2f\r\n",tempF);
  //HAL_Delay(10);
  //uart3_printf("Humidity = %3.2f%%\r\n\r\n",humidity);
  HAL_Delay(1000);

```

```

}

```



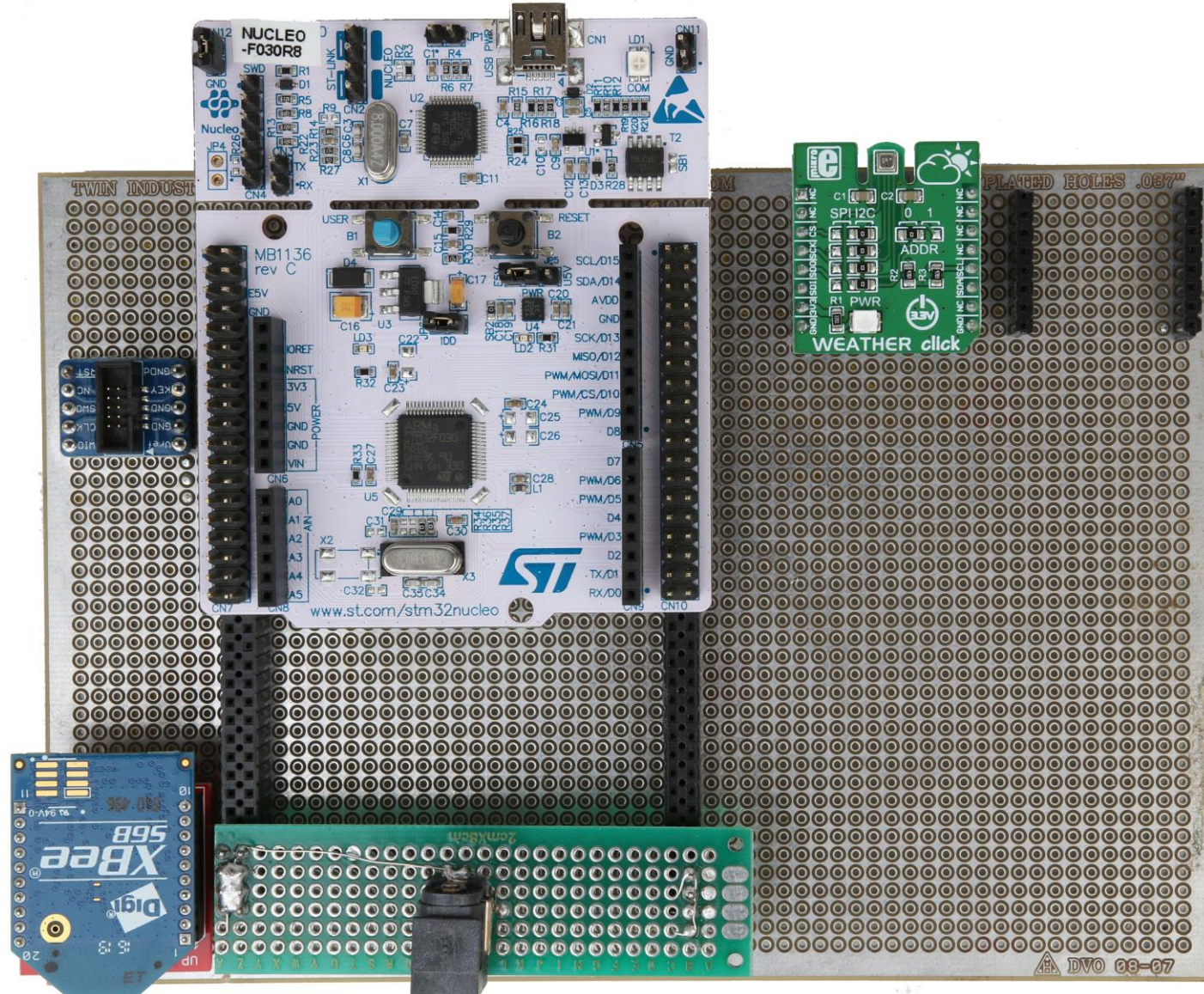
The screenshot shows the XCTU software interface. The top menu bar includes XCTU, Working Modes, Tools, and Help. The main window is divided into several sections:

- Radio Modules:** A list of modules is shown, with one selected:

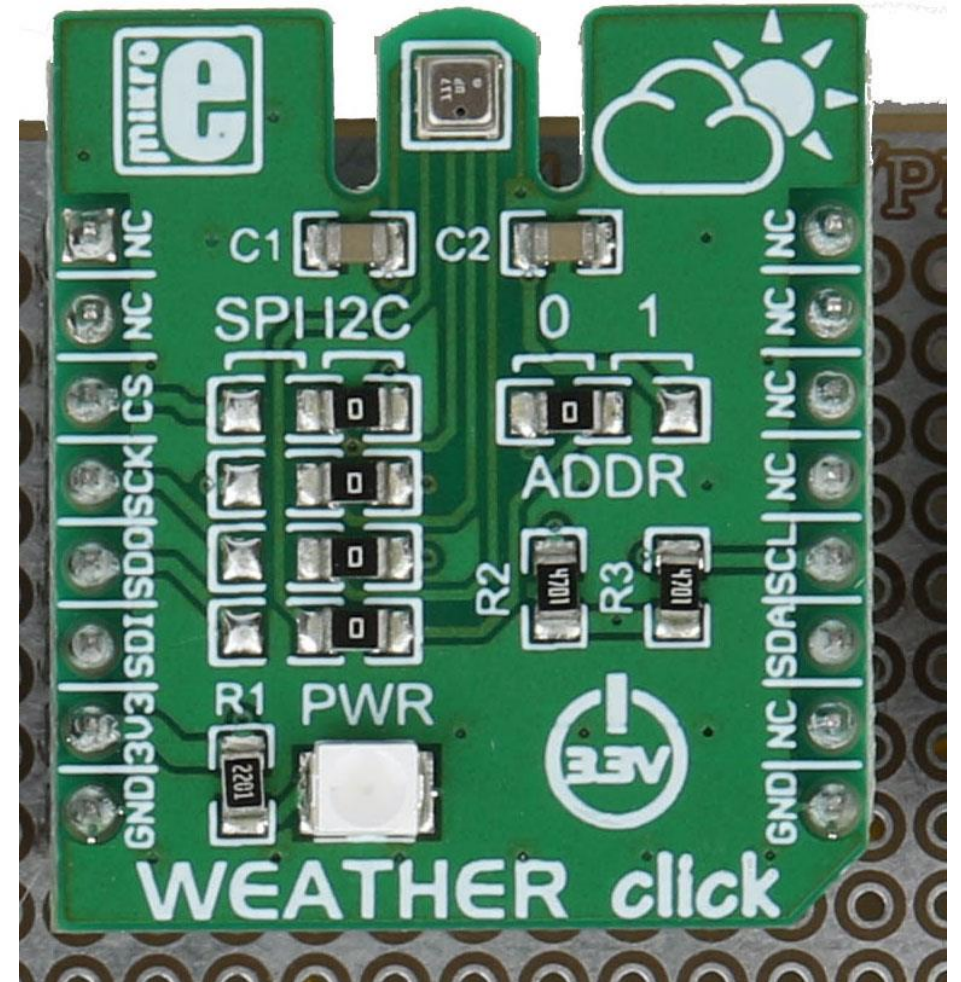
Name:	
Function:	802.15.4 TH
Port:	COM10 ...N - AT
MAC:	0013A...D2442
- Console log:** A window showing received data packets in hexadecimal and ASCII. The data is:


```

FA 19 C0 00 A0 00 91 AF 0A
FA 19 C9 66 4B 87 91 AF 0A
FA 19 BA 66 9E 85 91 AF 0A
FA 19 B3 66 CF 85 91 AF 0A
FA 19 CC 66 8C 86 91 AF 0A
FA 19 CF 66 8D 86 91 AF 0A
FA 19 BE 66 7C 87 91 AF 0A
FA 19 B5 66 21 87 91 AF 0A
      
```
- Send packets:** A section for sending data, including a table for Name and Data, and controls for sending a single packet or a sequence. The sequence settings are:
 - Transmit interval (ms): 500
 - Repeat times: 1
 - Loop infinitely:




```
SysTick_Config(SystemCoreClock/1000);
Weather_initializeClick();
while (1)
{
    Weather_readSensors();
    tc = BME280_getTemperature();
    tf = (tc * 1.8) + 32;
    th = BME280_getHumidity();
    tk = BME280_getPressure();
    tm = tk * 0.30;
    uart2_printf("\r\nTemperature = %3.2fC\r\n",tc);
    uart2_printf("Temperature = %3.2fF\r\n",tf);
    uart2_printf("Humidity = %3.2f%%\r\n",th);
    uart2_printf("Barometric Pressure = %3.2fkPa\r\n",tk);
    uart2_printf("Barometric Pressure = %3.2finHg\r\n",tm);
    HAL_Delay(100);
    HAL_GPIO_WritePin(LEDGRN_GPIO_Port, LEDGRN_Pin, GPIO_PIN_RESET);
    HAL_Delay(500);
    HAL_GPIO_WritePin(LEDGRN_GPIO_Port, LEDGRN_Pin, GPIO_PIN_SET);
    HAL_Delay(500);
}
```




```
/** *****  
/** PRINTF STUFF  
/** *****  
typedef struct __printf_tag  
{  
    size_t charcount;  
    size_t maxchars;  
    char *string;  
    int (*output_fn)(int, struct __printf_tag *ctx);  
} __printf_t;  
  
void uart2_putc(char ch)  
{  
    HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF);  
}  
  
int uart2_printf(const char *fmt, ...)  
{  
    int n;  
    va_list ap;  
    __printf_t iod;  
    va_start(ap, fmt);  
    iod.string = 0;  
    iod.maxchars = INT_MAX;  
    iod.output_fn = uart2_putc;  
    n = __vfprintf(&iod, fmt, ap);  
    va_end(ap);  
    return n;  
}
```



Embedded Studio Primer

Weather Over Wi-Fi



*Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 192.168.1.30

No.	Time	Source	Destination	Protocol	Length	Info
14723	152.157525	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14728	153.398750	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14731	154.642908	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14739	155.886073	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14743	157.121860	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14749	158.363405	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14752	159.603217	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14758	160.846706	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14761	162.086173	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14764	163.333327	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14767	164.576365	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14774	165.812250	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14782	167.049324	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14786	168.294027	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14790	169.531173	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14799	170.784828	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131

> Frame 14463: 173 bytes on wire (1384 bits), 173 bytes captured (1384 bits) on interface \Device\NPF...
 > Ethernet II, Src: FSFORTH-1d:82:09 (00:04:f3:1d:82:09), Dst: IntelCor_49:e4:a4 (dc:fb:48:49:e4:)
 > Internet Protocol Version 4, Src: 192.168.1.30, Dst: 192.168.1.227
 > User Datagram Protocol, Src Port: 9750, Dst Port: 9750
 > Data (131 bytes)

```

0000 dc fb 48 49 e4 a4 00 04 f3 1d 82 09 08 00 45 00  ..HI.....E..
0010 00 9f 00 19 00 00 40 11 f5 e3 c0 a8 01 1e c0 a8  .....@.....
0020 01 e3 26 16 26 16 00 8b c9 bb 0d 0a 54 65 6d 70  ..&&...Temp
0030 65 72 61 74 75 72 65 20 3d 20 32 34 2e 30 32 43  erature = 24.02C
0040 0d 0a 54 65 6d 70 65 72 61 74 75 72 65 20 3d 20  ..Temper ature =
0050 37 35 2e 32 34 46 0d 0a 48 75 6d 69 64 69 74 79  75.24F.. Humidity
0060 20 3d 20 35 32 2e 30 38 25 0d 0a 42 61 72 6f 6d   = 52.08 %..Barom
0070 65 74 72 69 63 20 50 72 65 73 73 75 72 65 20 3d  etric Pr essure =
0080 20 31 31 35 2e 37 38 6b 50 61 0d 0a 42 61 72 6f   115.78k Pa..Baro
0090 6d 65 74 72 69 63 20 50 72 65 73 73 75 72 65 20  metric P ressure
00a0 3d 20 33 34 2e 37 33 69 6e 48 67 0d 0a             = 34.73i nHg..
  
```

*Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 192.168.1.30

No.	Time	Source	Destination	Protocol	Length	Info
14723	152.157525	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14728	153.398750	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14731	154.642908	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14739	155.886073	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14743	157.121860	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14749	158.363405	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14752	159.603217	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14758	160.846706	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14761	162.086173	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14764	163.333327	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14767	164.576365	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14774	165.812250	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14782	167.049324	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14786	168.294027	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14790	169.531173	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131
14799	170.784828	192.168.1.30	192.168.1.227	UDP	173	9750 → 9750 Len=131

```

0000 dc fb 48 49 e4 a4 00 04 f3 1d 82 09 08 00 45 00  ..HI.....E..
0010 00 9f 00 19 00 00 40 11 f5 e3 c0 a8 01 1e c0 a8  .....@.....
0020 01 e3 26 16 26 16 00 8b c9 bb 0d 0a 54 65 6d 70  ..&&...Temp
0030 65 72 61 74 75 72 65 20 3d 20 32 34 2e 30 32 43  erature = 24.02C
0040 0d 0a 54 65 6d 70 65 72 61 74 75 72 65 20 3d 20  ..Temper ature =
0050 37 35 2e 32 34 46 0d 0a 48 75 6d 69 64 69 74 79  75.24F.. Humidity
0060 20 3d 20 35 32 2e 30 38 25 0d 0a 42 61 72 6f 6d   = 52.08 %..Barom
0070 65 74 72 69 63 20 50 72 65 73 73 75 72 65 20 3d  etric Pr essure =
0080 20 31 31 35 2e 37 38 6b 50 61 0d 0a 42 61 72 6f   115.78k Pa..Baro
0090 6d 65 74 72 69 63 20 50 72 65 73 73 75 72 65 20  metric P ressure
00a0 3d 20 33 34 2e 37 33 69 6e 48 67 0d 0a             = 34.73i nHg..
  
```


Thank you for attending

Please consider the resources below:

- <https://www.mikroe.com>
- <https://www.st.com/en/development-tools/stm32cubemx.html>
- <https://www.digi.com>



Thank You

Sponsored by

