ARM Your Sensors

Developing Sensor IoT Nodes in an ARMED Playground

August 28, 2018

Fred Eady

Presented by:

DesignNews

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

ARM Your Sensors

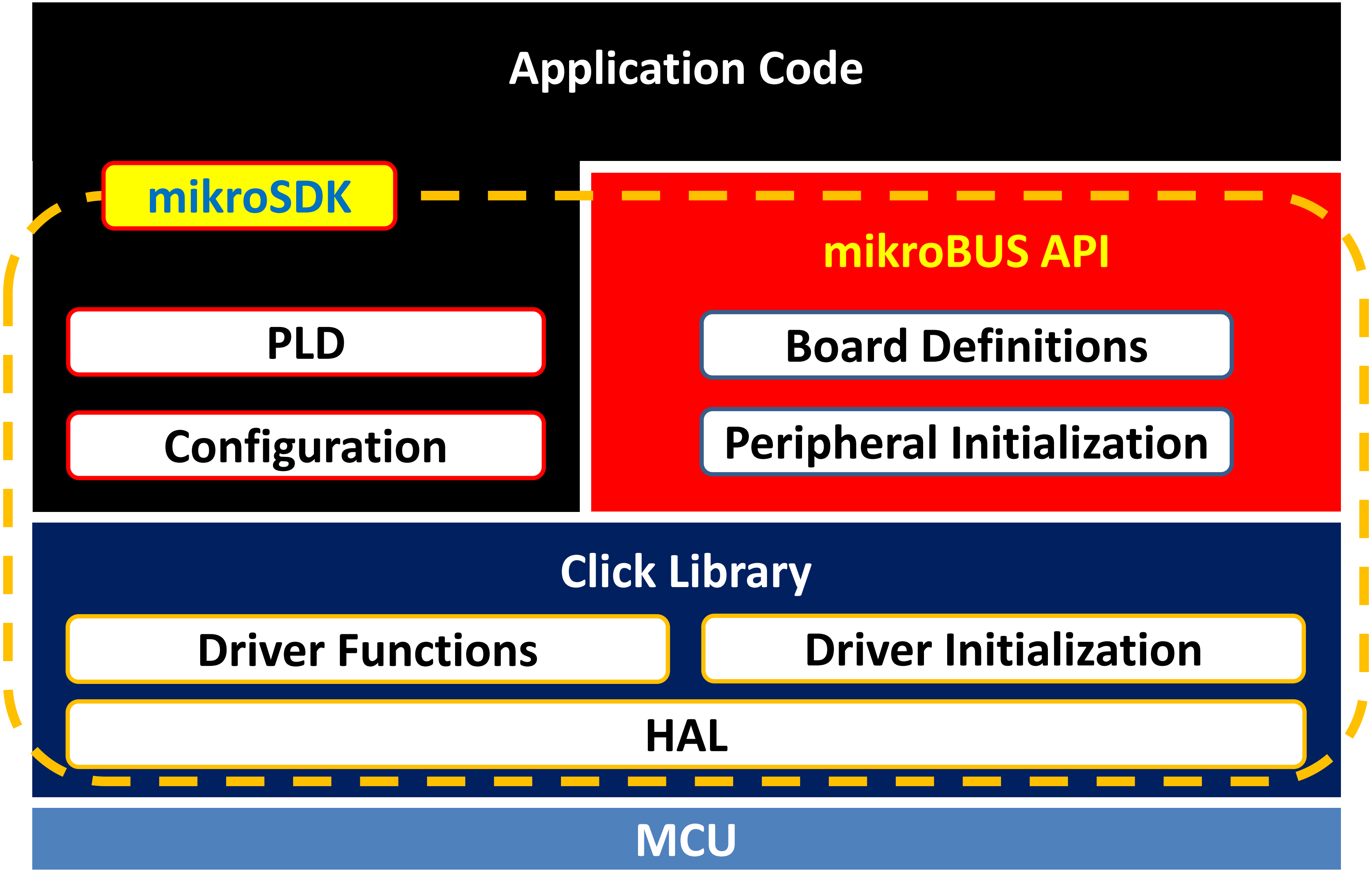
AGENDA

- ARMed Abstraction II
- Lock and Load
- Temperature and Humidity
- ARMed Physics Sans Abstraction
- Day 2 Summary

Presented by:

ARM Your Sensors

ARMed Abstraction II - mikroSDK



Application Code

mikroSDK

PLD

Configuration

mikroBUS API

Board Definitions

Peripheral Initialization

Click Library

Driver Functions

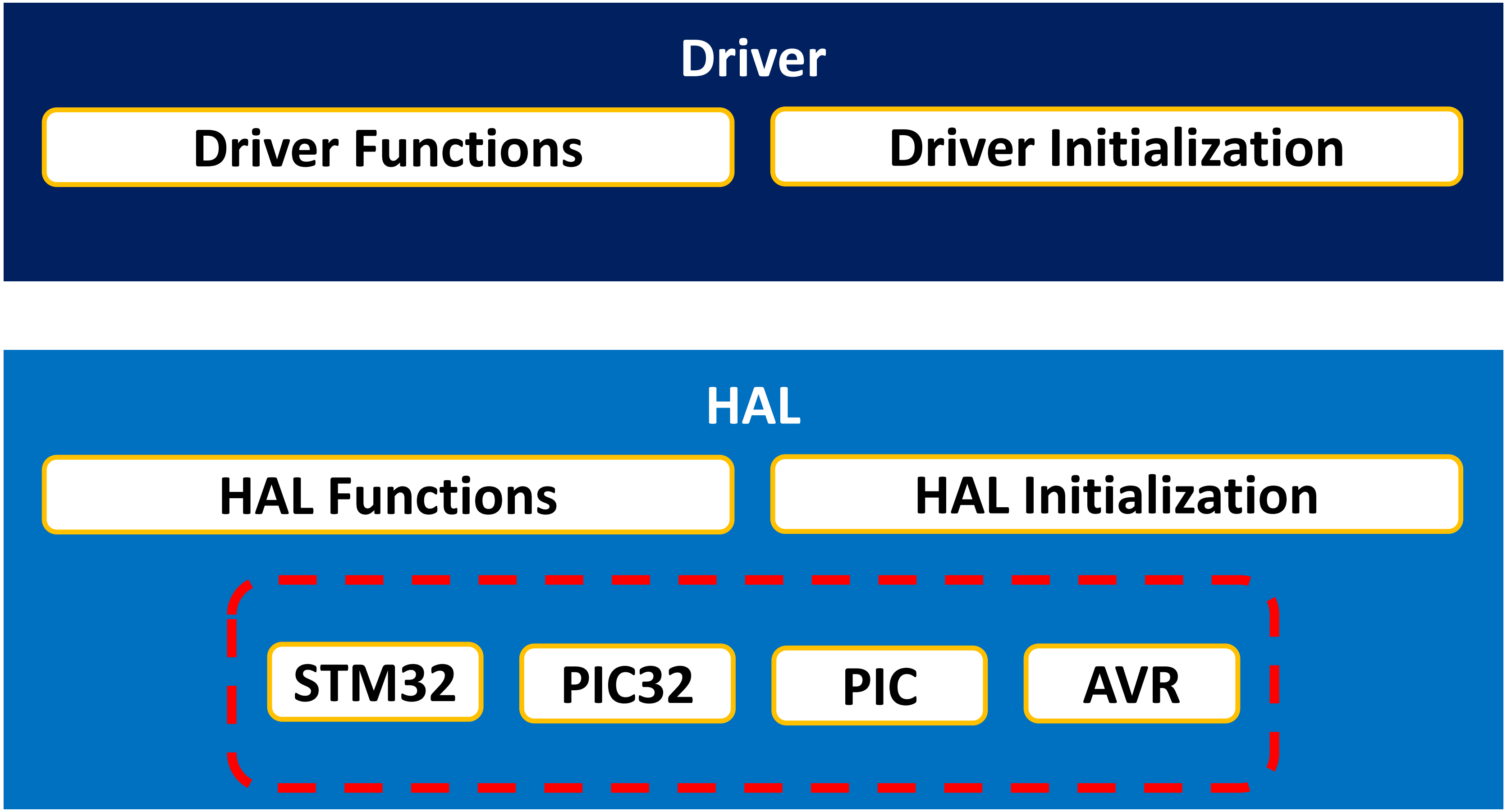
Driver Initialization

HAL

MCU

ARM Your Sensors

ARMed Abstraction II – Click Library



Driver

Driver Functions

Driver Initialization

HAL

HAL Functions

HAL Initialization

STM32

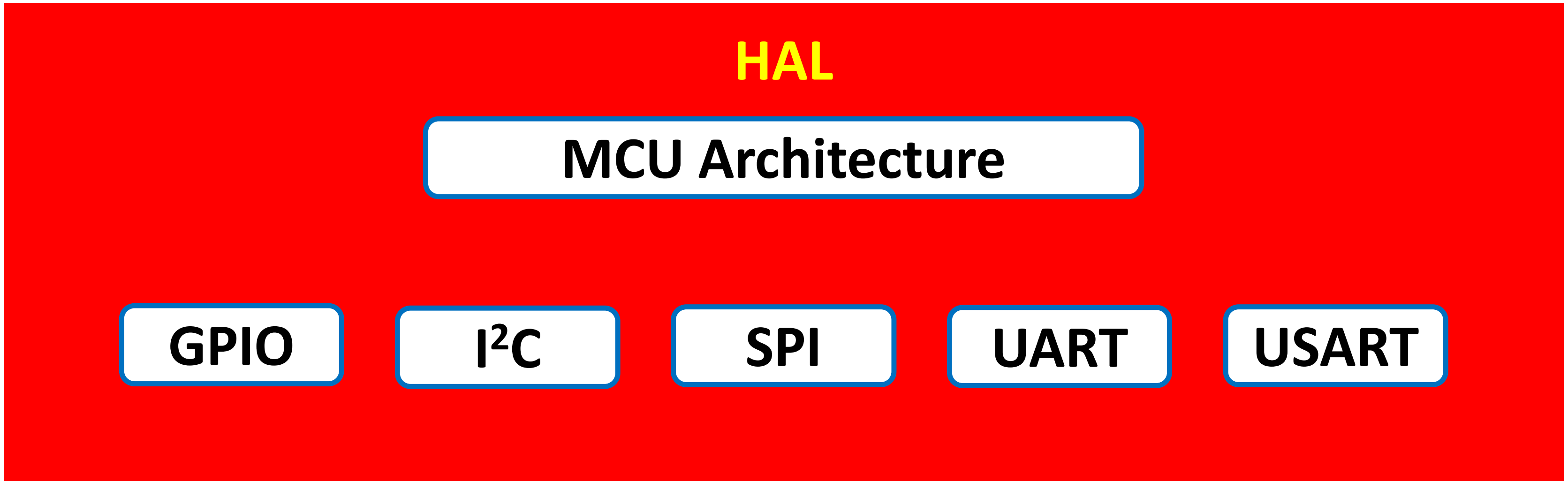
PIC32

PIC

AVR

ARM Your Sensors

ARMed Abstraction II – Click Library



HAL

MCU Architecture

GPIO

I^{2}C

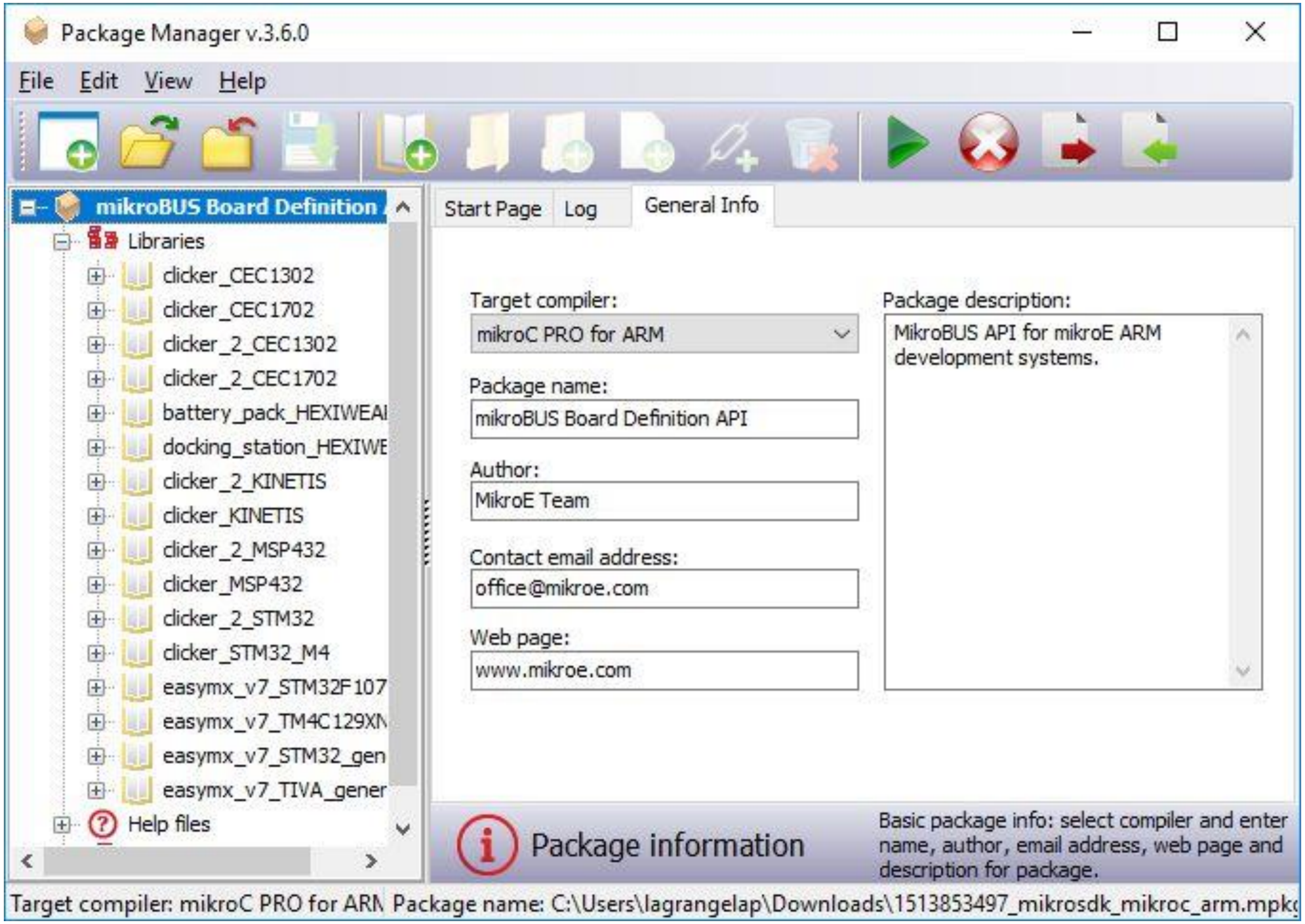
SPI

UART

USART

ARM Your Sensors

Lock and Load



Target compiler: mikroC PRO for ARM

Package name: mikroBUS Board Definition API

Author: MikroE Team

Contact email address: office@mikroe.com

Web page: www.mikroe.com

Package description: MikroBUS API for mikroE ARM development systems.

Package information

Basic package info: select compiler and enter name, author, email address, web page and description for package.

Target compiler: mikroC PRO for ARN Package name: C:\Users\lagrangelap\Downloads\1513853497_mikrosdk_mikroc_arm.mpk

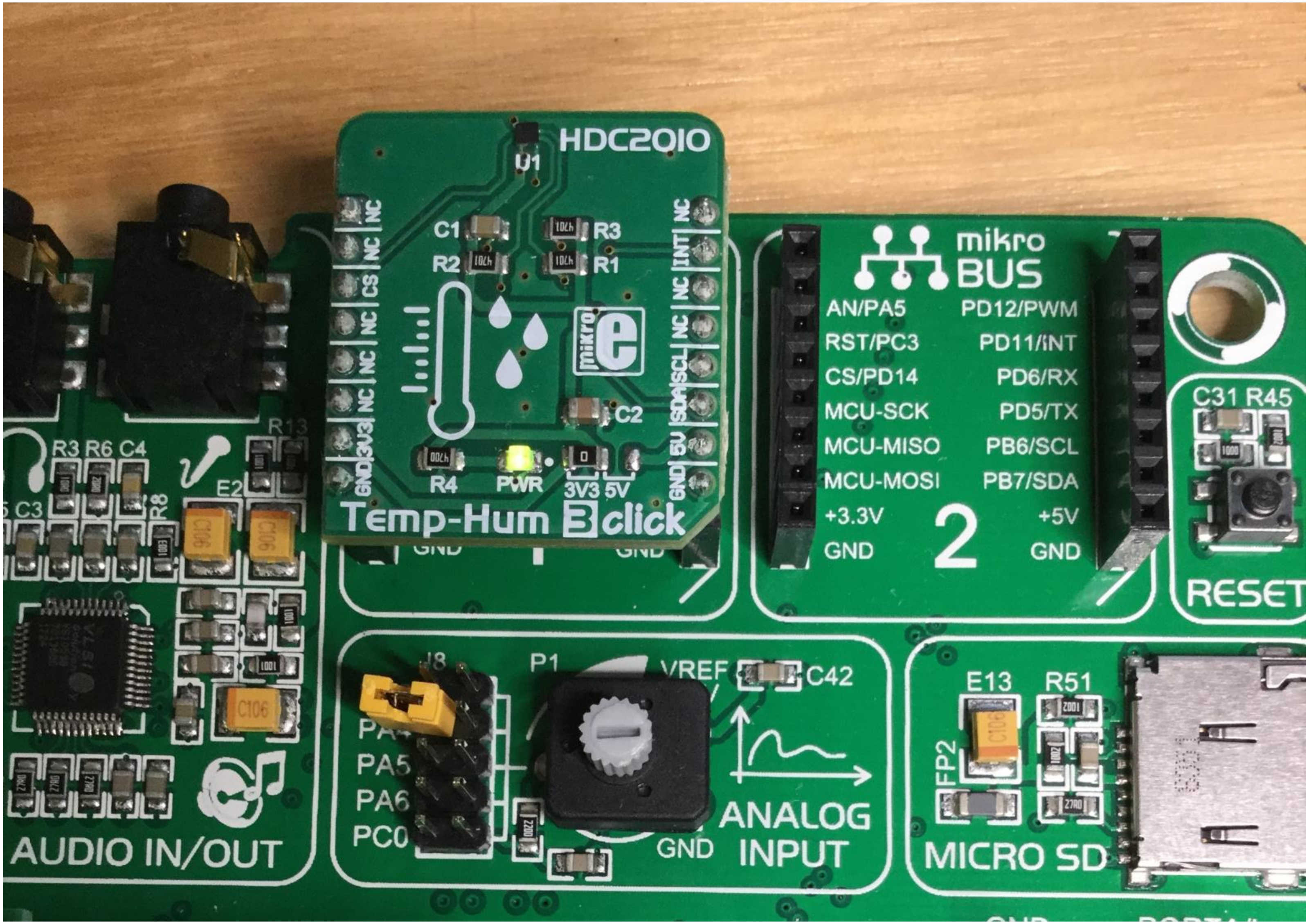
DesignNews

CEC CONTINUING EDUCATION CENTER

Presented by: Digi-Key ELECTRONICS

ARM Your Sensors

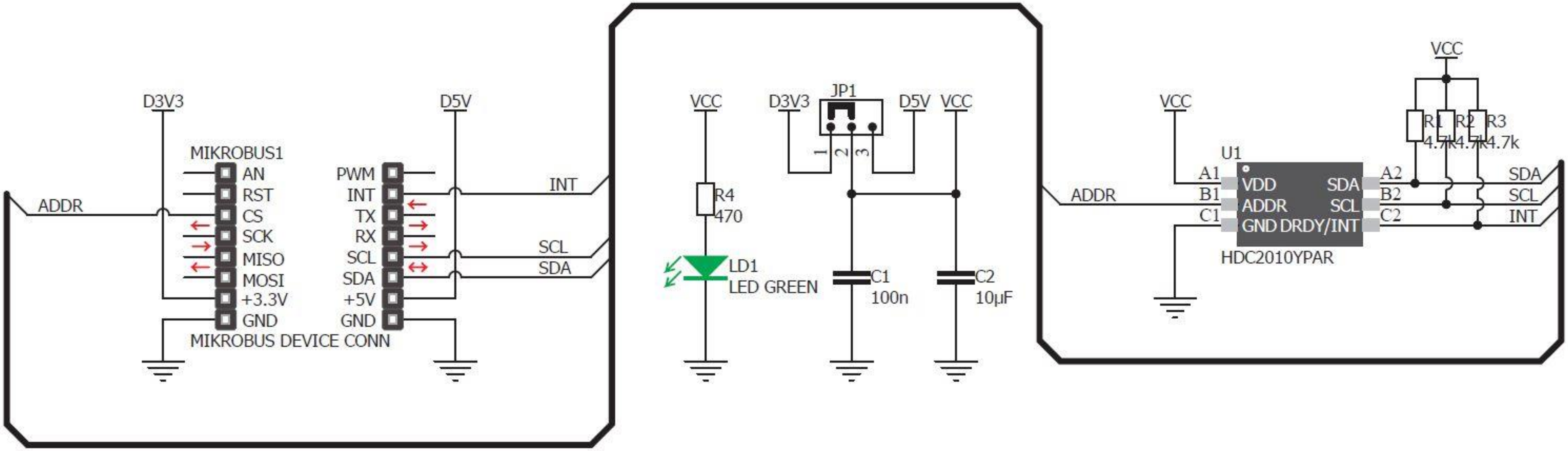
Temperature and Humidity



Presented by:

ARM Your Sensors

Temperature and Humidity



Presented by:

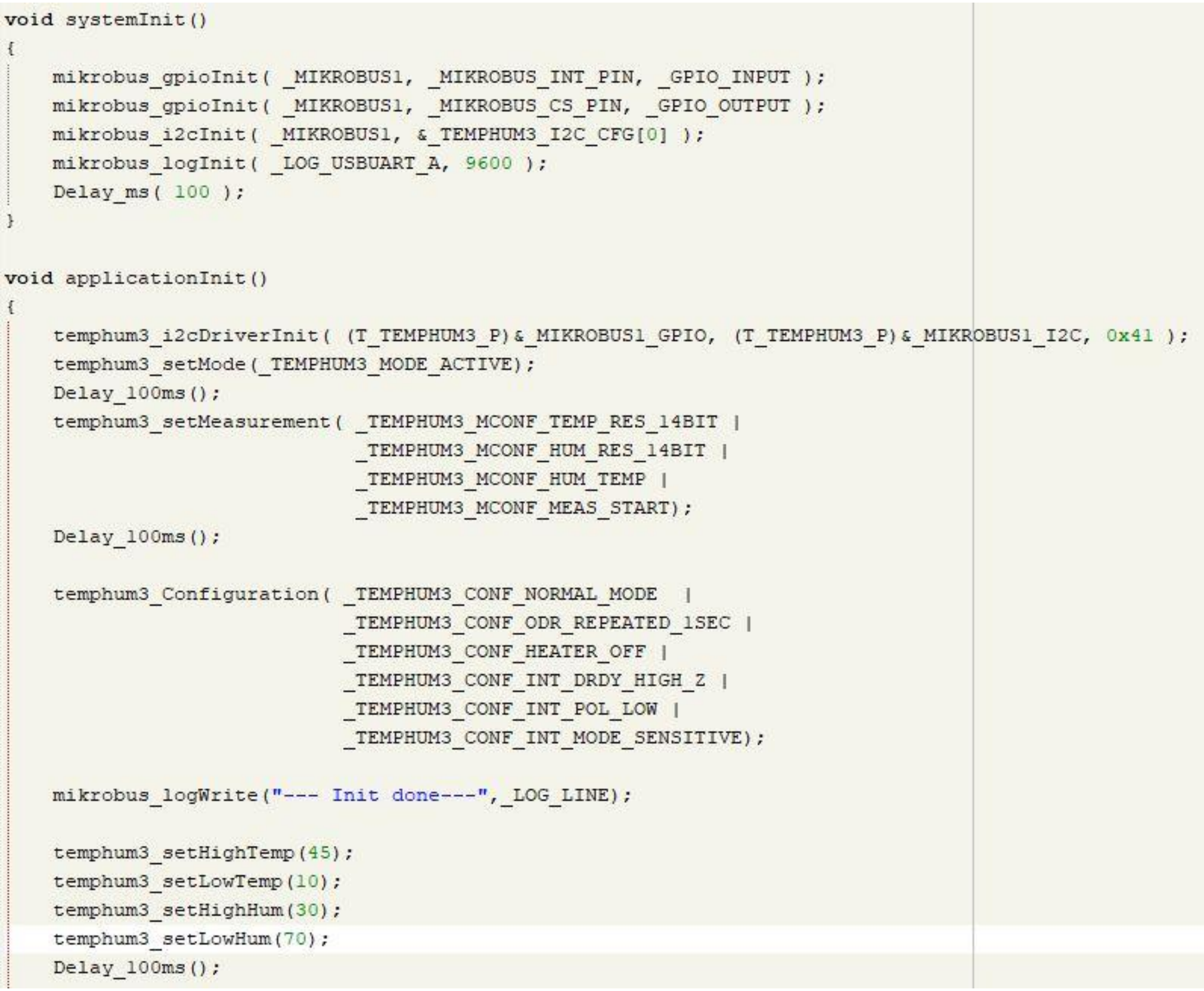
DesignNews

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

ARM Your Sensors

Temperature and Humidity

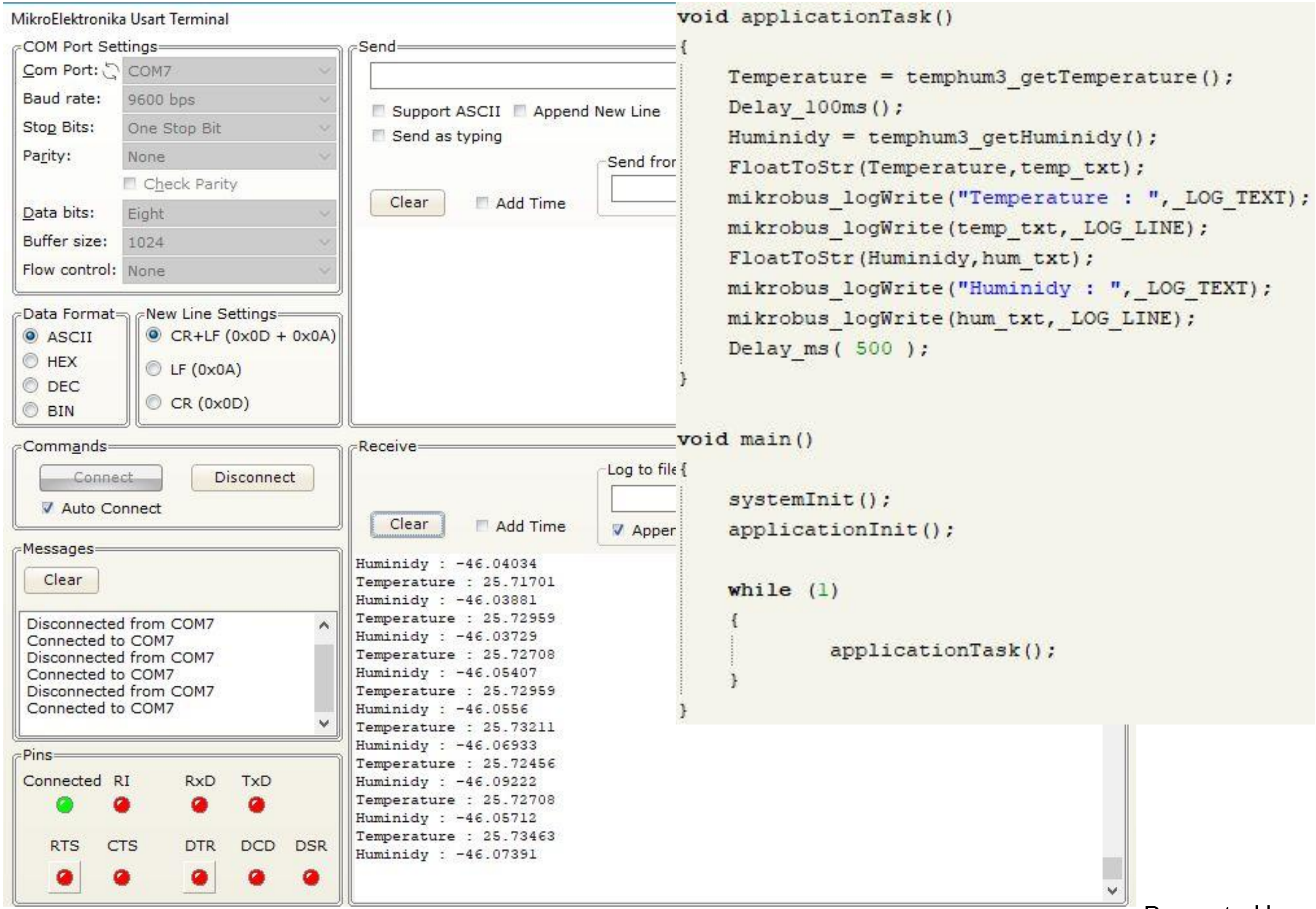


```
void systemInit()
{
    mikrobus_gpioInit( _MIKROBUS1, _MIKROBUS_INT_PIN, _GPIO_INPUT );
    mikrobus_gpioInit( _MIKROBUS1, _MIKROBUS_CS_PIN, _GPIO_OUTPUT );
    mikrobus_i2cInit( _MIKROBUS1, & _TEMPHUM3_I2C_CFG[0] );
    mikrobus_logInit( _LOG_USBUART_A, 9600 );
    Delay_ms( 100 );
}

void applicationInit()
{
    temphum3_i2cDriverInit( (T_TEMPHUM3_P)&_MIKROBUS1_GPIO, (T_TEMPHUM3_P)&_MIKROBUS1_I2C, 0x41 );
    temphum3_setMode(_TEMPHUM3_MODE_ACTIVE);
    Delay_100ms();
    temphum3_setMeasurement( _TEMPHUM3_MCONF_TEMP_RES_14BIT |
                             _TEMPHUM3_MCONF_HUM_RES_14BIT |
                             _TEMPHUM3_MCONF_HUM_TEMP |
                             _TEMPHUM3_MCONF_MEAS_START);
    Delay_100ms();

    temphum3_Configuration( _TEMPHUM3_CONF_NORMAL_MODE  |
                            _TEMPHUM3_CONF_ODR_REPEATED_1SEC |
                            _TEMPHUM3_CONF_HEATER_OFF |
                            _TEMPHUM3_CONF_INT_DRDY_HIGH_Z |
                            _TEMPHUM3_CONF_INT_POL_LOW |
                            _TEMPHUM3_CONF_INT_MODE_SENSITIVE);

    mikrobus_logWrite("--- Init done---",_LOG_LINE);

    temphum3_setHighTemp(45);
    temphum3_setLowTemp(10);
    temphum3_setHighHum(30);
    temphum3_setLowHum(70);
    Delay_100ms();
```

Presented by:

ARM Your Sensors

Temperature and Humidity



MikroElektronika Usart Terminal

COM Port Settings
Com Port: COM7
Baud rate: 9600 bps
Stop Bits: One Stop Bit
Parity: None
□ Check Parity
Data bits: Eight
Buffer size: 1024
Flow control: None

Data Format: ● ASCII ○ HEX ○ DEC ○ BIN

New Line Settings: ● CR+LF (0x0D + 0x0A) ○ LF (0x0A) ○ CR (0x0D)

Commands: ☑ Auto Connect

```
Disconnected from COM7
Connected to COM7
Disconnected from COM7
Connected to COM7
Disconnected from COM7
Connected to COM7
```

```
Huminidy : -46.04034
Temperature : 25.71701
Huminidy : -46.03881
Temperature : 25.72959
Huminidy : -46.03729
Temperature : 25.72708
Huminidy : -46.05407
Temperature : 25.72959
Huminidy : -46.0556
Temperature : 25.73211
Huminidy : -46.06933
Temperature : 25.72456
Huminidy : -46.09222
Temperature : 25.72708
Huminidy : -46.05712
Temperature : 25.73463
Huminidy : -46.07391
```

```
void applicationTask()
{
    Temperature = temphum3_getTemperature();
    Delay_100ms();
    Huminidy = temphum3_getHuminidy();
    FloatToStr(Temperature,temp_txt);
    mikrobus_logWrite("Temperature : ",_LOG_TEXT);
    mikrobus_logWrite(temp_txt,_LOG_LINE);
    FloatToStr(Huminidy,hum_txt);
    mikrobus_logWrite("Huminidy : ",_LOG_TEXT);
    mikrobus_logWrite(hum_txt,_LOG_LINE);
    Delay_ms( 500 );
}

void main()
{
    systemInit();
    applicationInit();

    while (1)
    {
            applicationTask();
    }
}
```

DesignNews

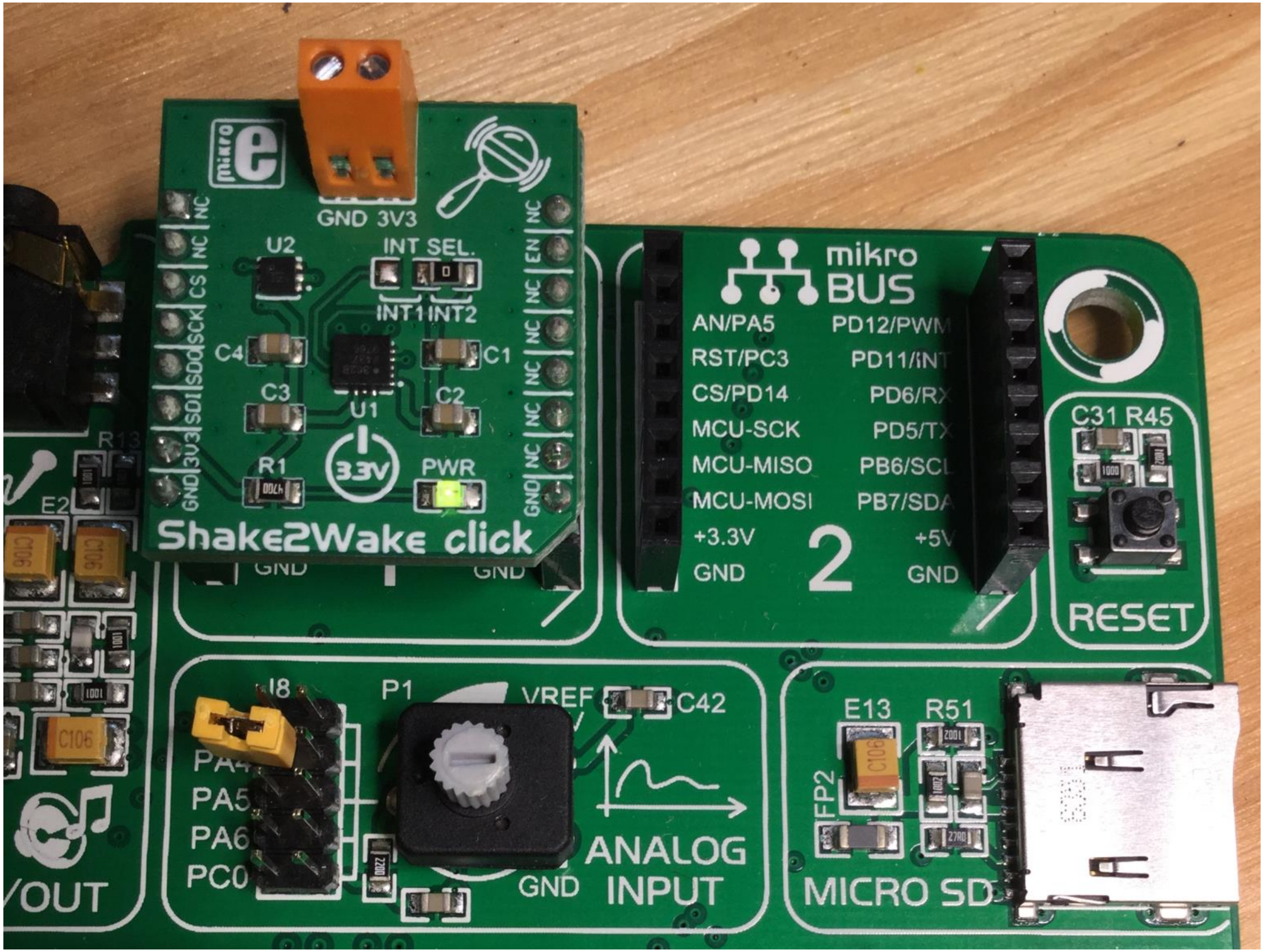
CEC CONTINUING EDUCATION CENTER

Presented by:

Digi-Key ELECTRONICS

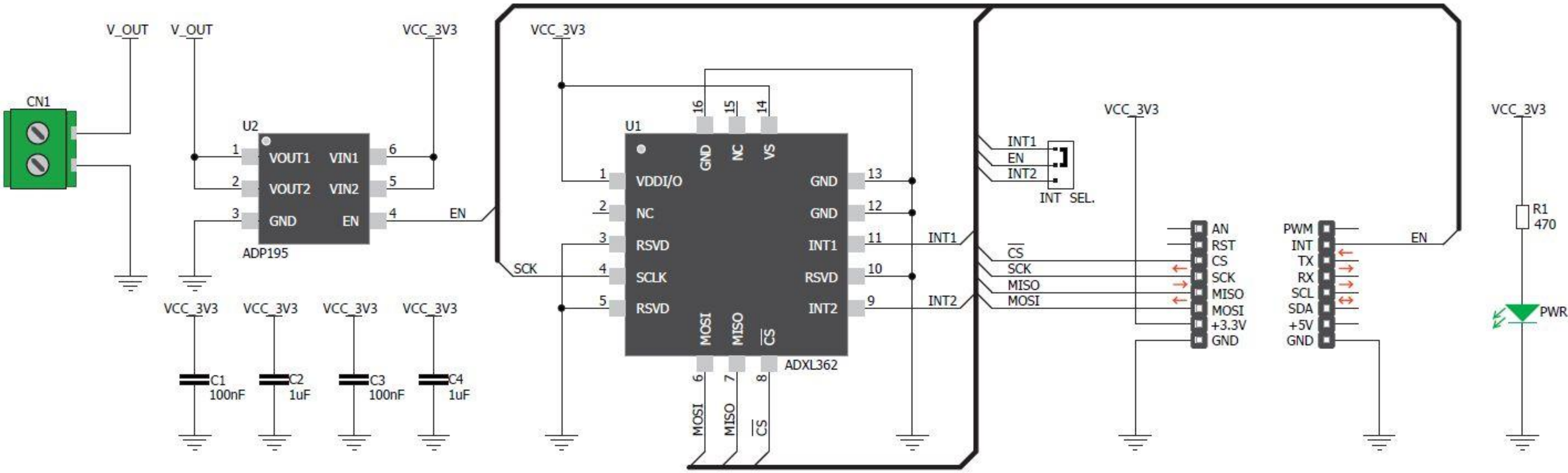
ARM Your Sensors

ARMed Physics Sans Abstraction



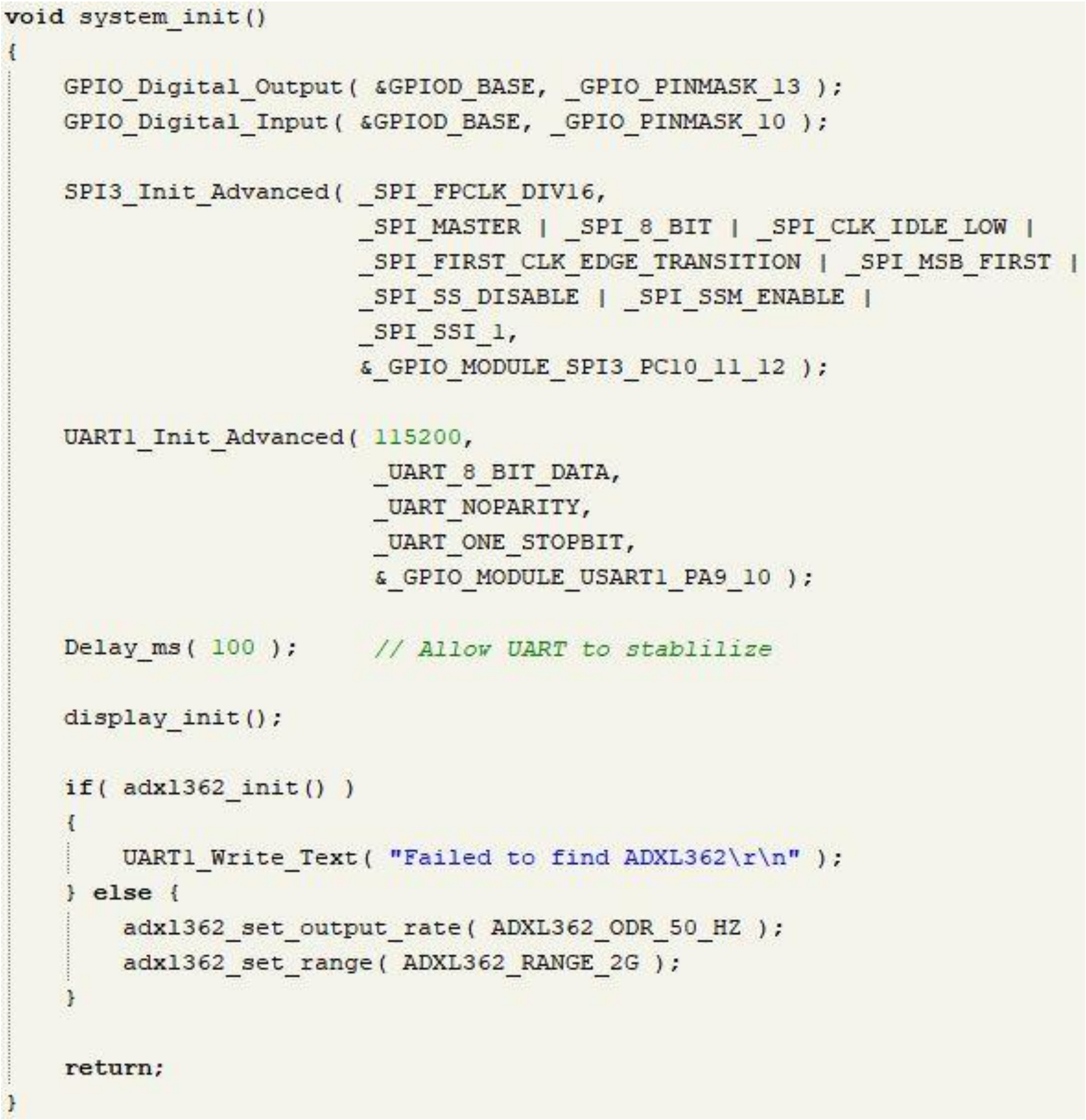
ARM Your Sensors

ARMed Physics Sans Abstraction



ARM Your Sensors

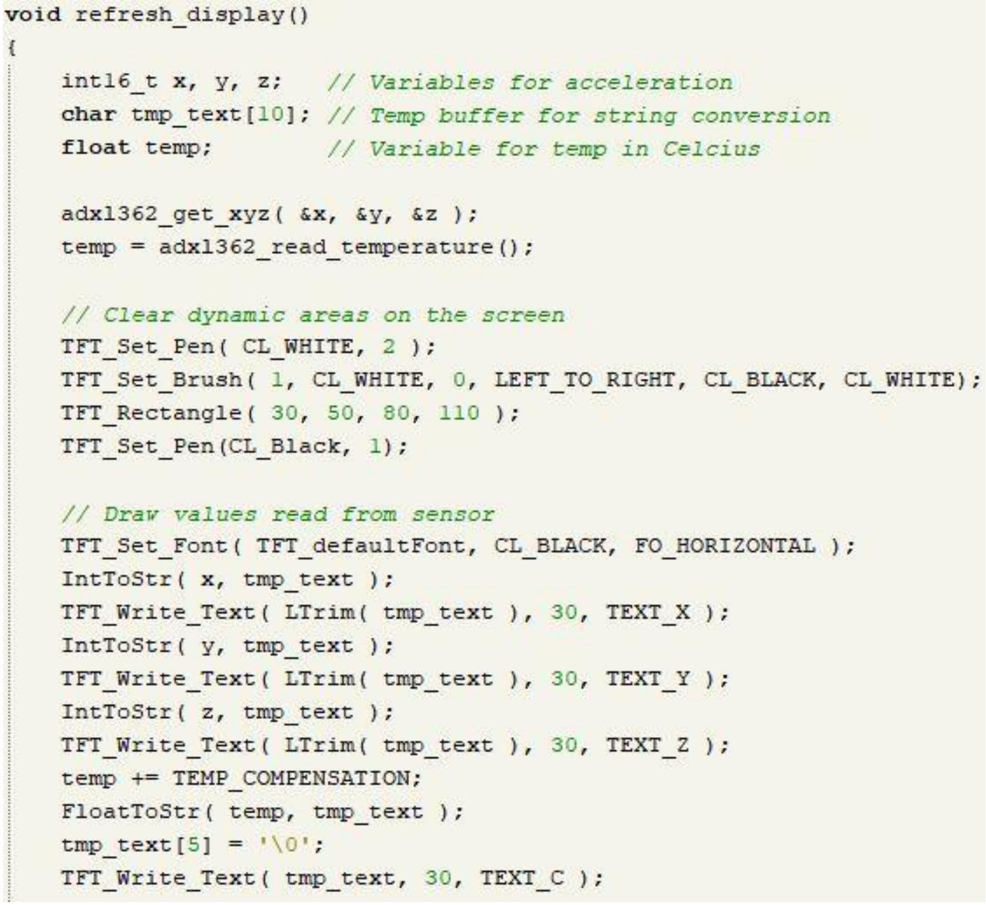
ARMed Physics Sans Abstraction



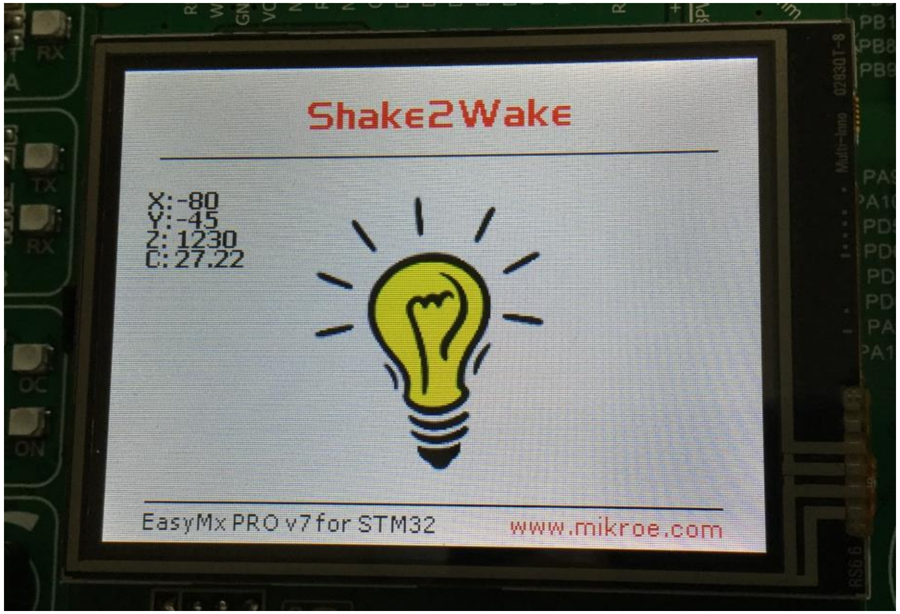
```
void system_init()
{
    GPIO_Digital_Output( &GPIOD_BASE, _GPIO_PINMASK_13 );
    GPIO_Digital_Input( &GPIOD_BASE, _GPIO_PINMASK_10 );

    SPI3_Init_Advanced( _SPI_FPCLK_DIV16,
                        _SPI_MASTER | _SPI_8_BIT | _SPI_CLK_IDLE_LOW |
                        _SPI_FIRST_CLK_EDGE_TRANSITION | _SPI_MSB_FIRST |
                        _SPI_SS_DISABLE | _SPI_SSM_ENABLE |
                        _SPI_SSI_1,
                        &_GPIO_MODULE_SPI3_PC10_11_12 );

    UART1_Init_Advanced( 115200,
                         _UART_8_BIT_DATA,
                         _UART_NOPARITY,
                         _UART_ONE_STOPBIT,
                         &_GPIO_MODULE_USART1_PA9_10 );

    Delay_ms( 100 );     // Allow UART to stablilize

    display_init();

    if( adxl362_init() )
    {
        UART1_Write_Text( "Failed to find ADXL362\r\n" );
    } else {
        adxl362_set_output_rate( ADXL362_ODR_50_HZ );
        adxl362_set_range( ADXL362_RANGE_2G );
    }

    return;
}
```

ARM Your Sensors

ARMed Physics Sans Abstraction



```
void refresh_display()
{
    int16_t x, y, z;    // Variables for acceleration
    char tmp_text[10]; // Temp buffer for string conversion
    float temp;         // Variable for temp in Celcius

    adxl362_get_xyz( &x, &y, &z );
    temp = adxl362_read_temperature();

    // Clear dynamic areas on the screen
    TFT_Set_Pen( CL_WHITE, 2 );
    TFT_Set_Brush( 1, CL_WHITE, 0, LEFT_TO_RIGHT, CL_BLACK, CL_WHITE);
    TFT_Rectangle( 30, 50, 80, 110 );
    TFT_Set_Pen(CL_Black, 1);

    // Draw values read from sensor
    TFT_Set_Font( TFT_defaultFont, CL_BLACK, FO_HORIZONTAL );
    IntToStr( x, tmp_text );
    TFT_Write_Text( LTrim( tmp_text ), 30, TEXT_X );
    IntToStr( y, tmp_text );
    TFT_Write_Text( LTrim( tmp_text ), 30, TEXT_Y );
    IntToStr( z, tmp_text );
    TFT_Write_Text( LTrim( tmp_text ), 30, TEXT_Z );
    temp += TEMP_COMPENSATION;
    FloatToStr( temp, tmp_text );
    tmp_text[5] = '\0';
    TFT_Write_Text( tmp_text, 30, TEXT_C );
```



Shake2Wake

```
X:-80
Y:-45
Z:1230
C:27.22
```

EasyMx PRO v7 for STM32 www.mikroe.com

ARM Your Sensors

Day 2 Summary