Quadro – Bosch BME280 lab

Attila Mák
01.08.2017

Five Years Out

Five Years Out

Revision History

Revision, Date	Editor	Subject(major changes)
Revision 1.0, 01.08.2017	Attila Mák	Initial release

Contents

1 Scope/ description

This document is guiding to set up Arrow Quadro board with Bosch BME280 shuttle board. We will create a project which will send the all the sensor data to the Quadro board. And the Quadro will send the received data to the cloud via Wi-Fi.

The connection between Quadro and BME280 shuttle board is based on SPI.

2 Hardware setup

The required hardware to perform the steps described in this application note consists of:

Developer PC:

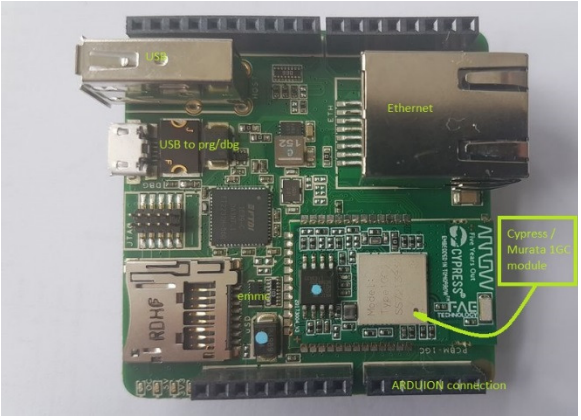
This platform will be used for setup and writing and downloading the two firmware into the microcontrollers. The power supply task will be solved via USB.
Desktop computer or laptop with x86 architecture and USB 2.0

The ARROW's Quadro board as it is

The BME280 Shuttle board.

2.1 Arrow Quadro board

https://www.arrow.com/en/campaigns/cypress-wiced-iot

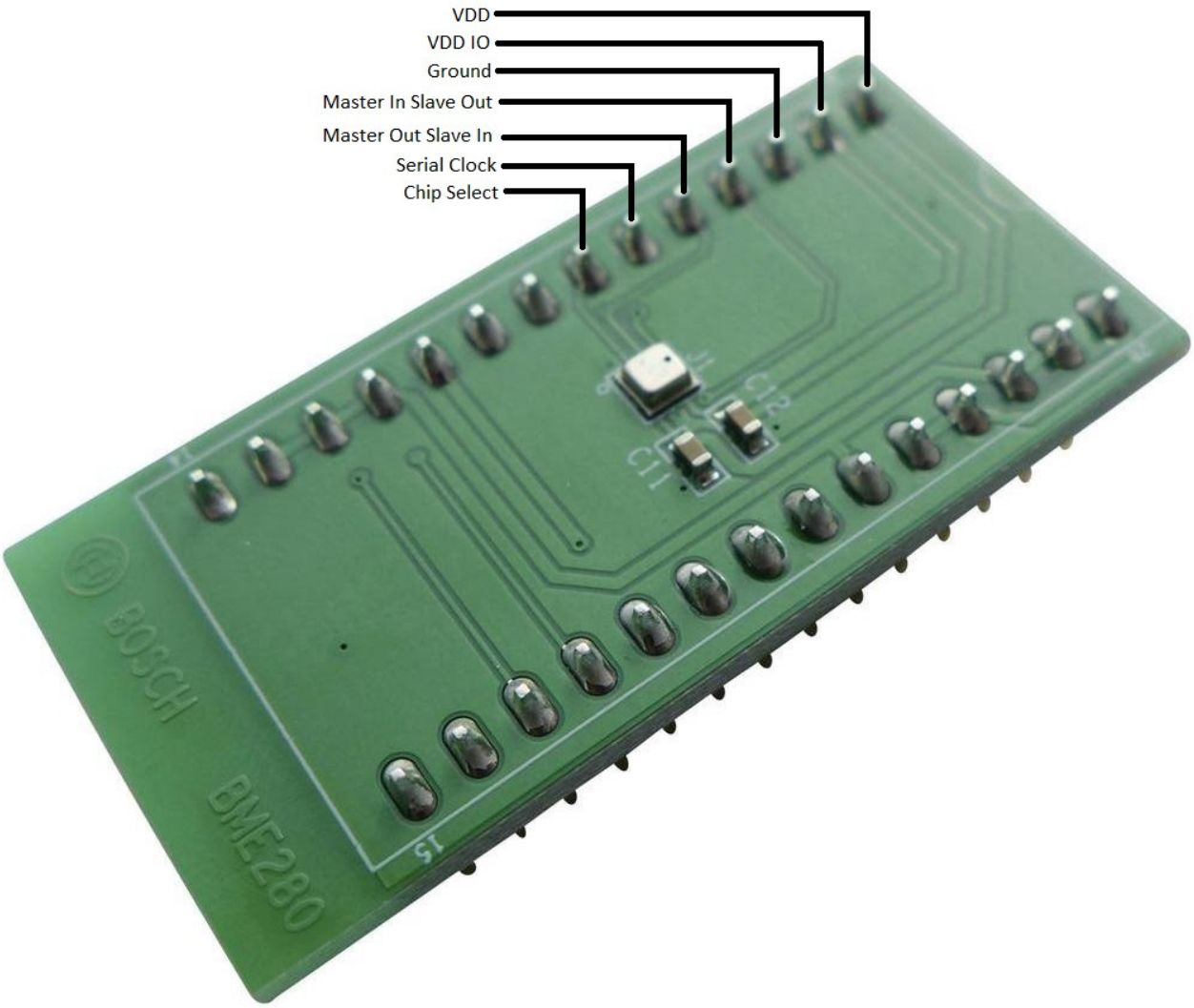


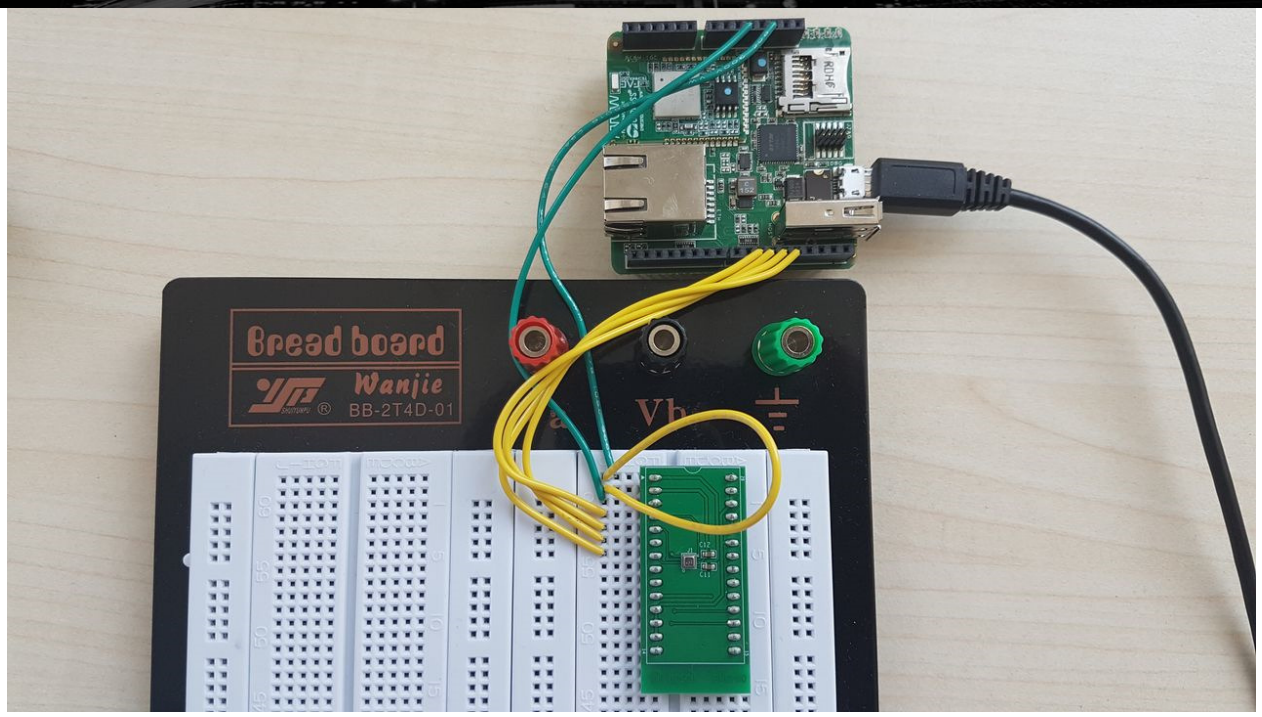
Arduino connection:

X	SCL/D15
X	SDA/D14
X	SCK/D13
NOT CONNECTED	MISO/D12
IOREF	PWM/MOSI/D11
NRST	PWM/CS/D10
NRST	PWM/D9
+3.3V	D8
+5V	NOT CONNECTED
GND	NOT CONNECTED
GND	D7
+VIN	PWM/D6
AN0	PWM/D5
AN1	D4
AN2	PWM/D3
AN3	D2
AN4	UART_TXD_RXuc
AN5	UART_RX_TXuc

2.2 BME280 Shuttle board

https://www.bosch-sensortec.com/bst/support_tools/downloads/overview_downloads





Because of the pinout of Shuttle board system doesn't compatible with Arduino system wiring is preferred.

3 Software setup

3.1 IDE installation to the PC

3.1.1 WICED installation

Pay ATTENTION! Some antivirus like BitDefender may cause problems during the installation phase and during normal use of the Wiced Studio

1) Download and install 32bit and 64bit JRE (Java Runtime Environment)

   http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

2) Download and install Cypress WICED Studio 6.0

   https://community.cypress.com/community/wiced-wifi/wiced-wifi-documentation

3) Plug the Board with USB Cable

   Verify Driver installation

4) Download and install Quadro BSP file

   BCM943907_QUADRO_w6.zip for WICED 6.0 IDE

5) Download the Quadro IoT Starter Kit Getting Started Guide

   Bluemix IoT LAB

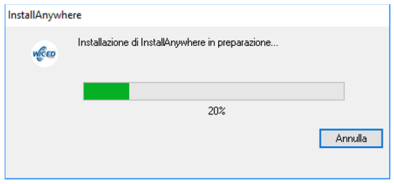
Setup JRE 32 and JRE64 java runtime

1) Check/Fix your JRE (Java Runtime Environment) installation:
   a. 32-bit JRE is needed for Cypress WICED Studio
   b. 64-bit JRE is needed for SDK's installer
      (JRE is designed to allow both 32 and 64 bit variants to be installed on same system)
2) Not normally required, but if you have a JRE related issue, check your Windows PATH. This should include a path to your Java installation:

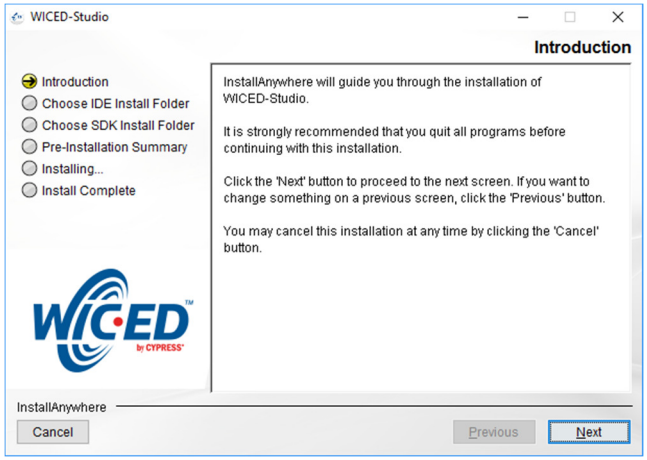
   C:\ProgramData\Oracle\Java\javapath

Setup Wiced Studio (WINDOWS)

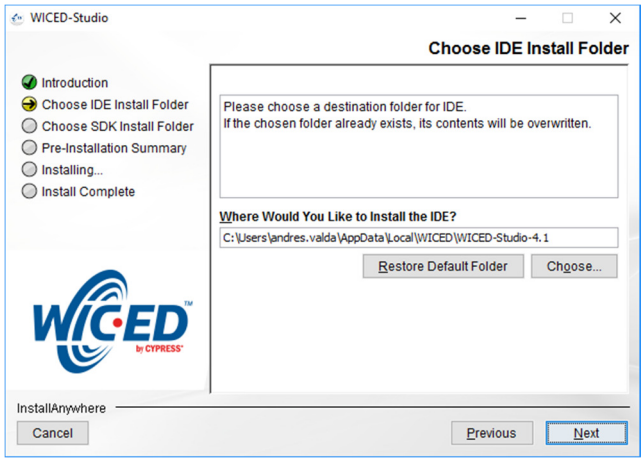
1) install Cypress WICED Studio 4.1.1 development tools by unzip file WICED-Studio-4.1.1.8-IDE-Installer.exe.zip and execute WICED-Studio-4.1.1.8-IDE-Installer.exe
2) When Setup start Appear window below



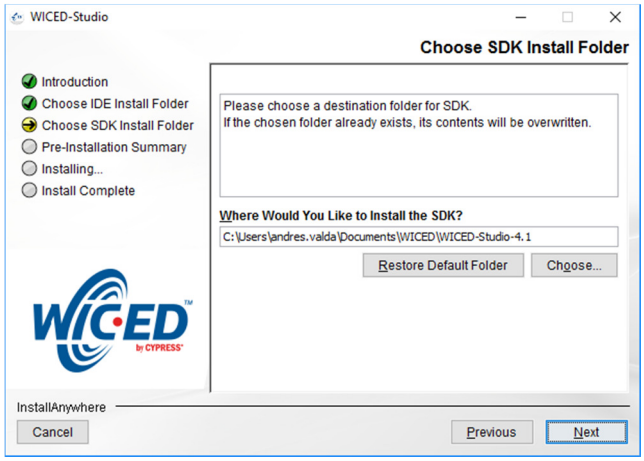
And then



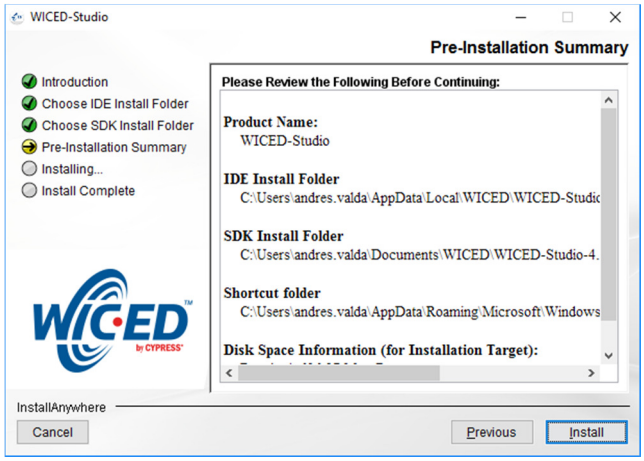
Press Next Button



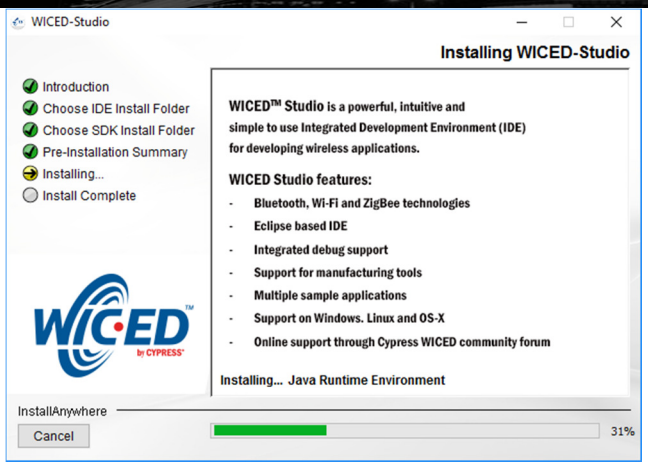
Press Next Button



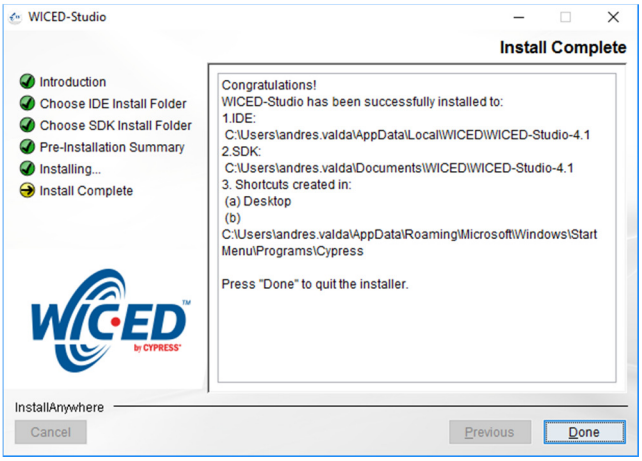
Press Next Button



Press Install

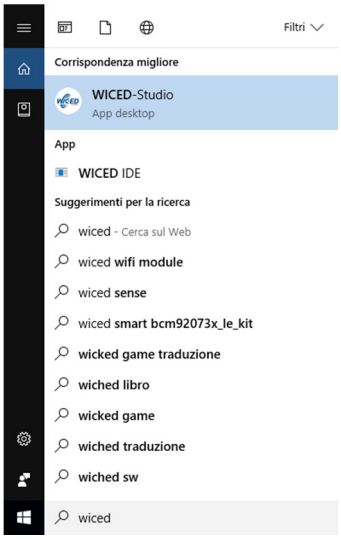


Wait while setup finish

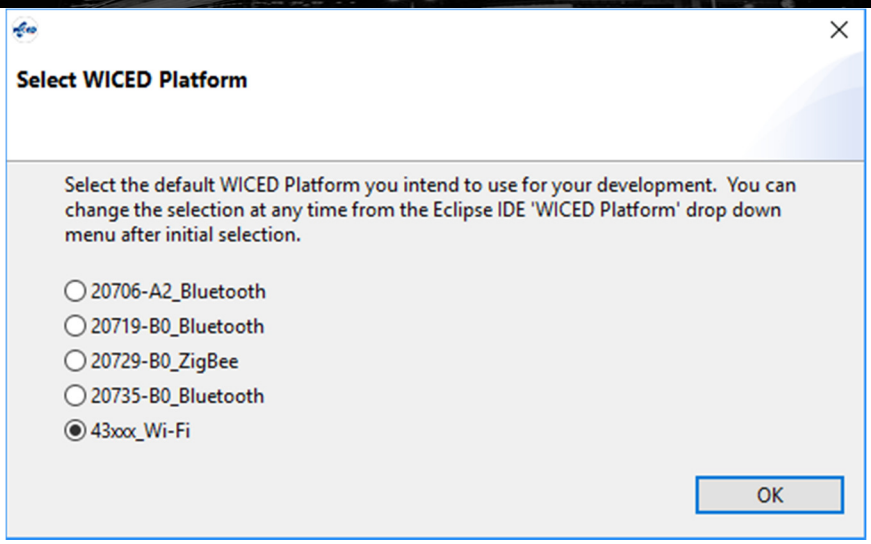


And press Done Button

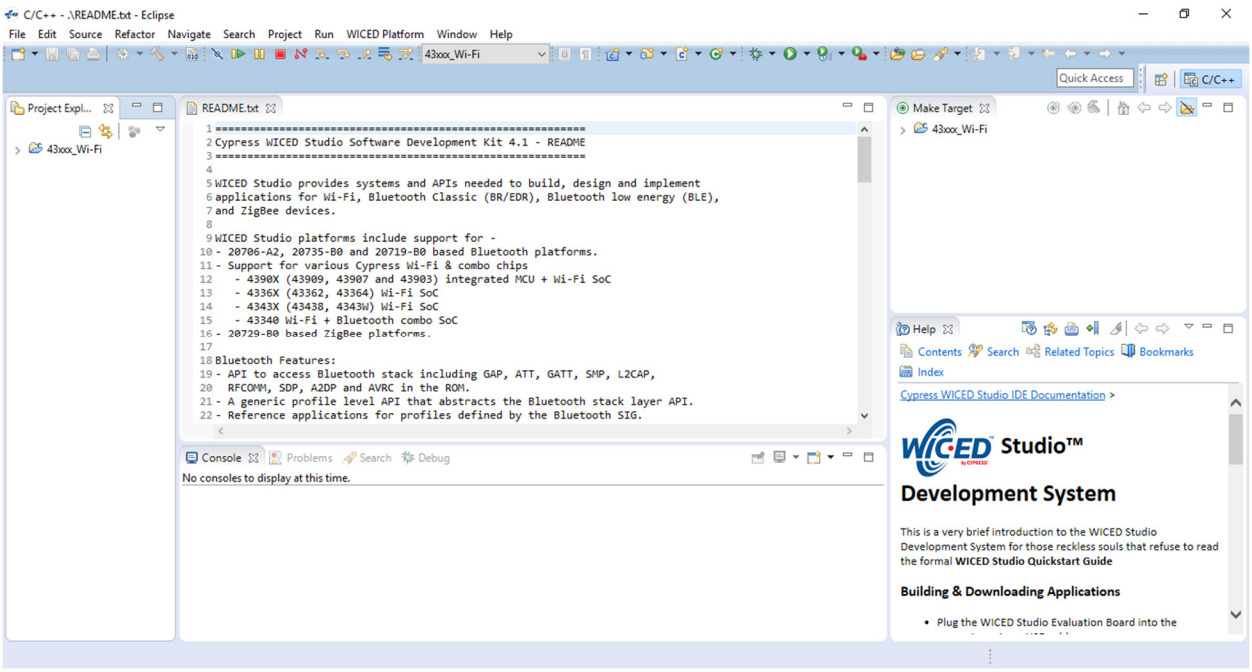
3) Launch the WICED SDK



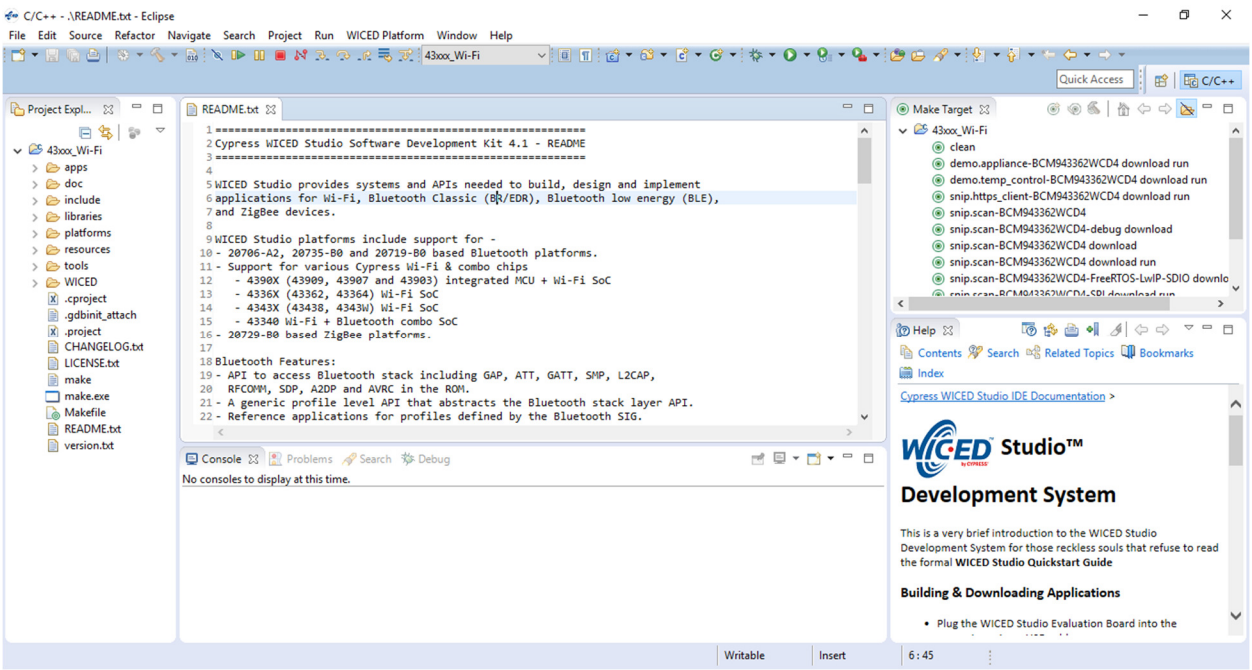
Select 43xxx_Wi-Fi Platform


Select WICED Platform
Select the default WICED Platform you intend to use for your development. You can change the selection at any time from the Eclipse IDE 'WICED Platform' drop down menu after initial selection.
20706-A2_Bluetooth
20719-B0_Bluetooth
20729-B0_ZigBee
20735-B0_Bluetooth
43xxx_Wi-Fi
OK

The WICED IDE must appear as below



Expand 43xxx_Wi-Fi Project Explorer and 43xxx_Wi-Fi Make Target



SETUP WICED STUDIO (OTHER OS)

OSX

https://community.cypress.com/docs/DOC-3988

Linux 64-bit

https://community.cypress.com/docs/DOC-3989

Linux 32 bit

https://community.cypress.com/docs/DOC-3990

3.1.2 BME280

The latest driver is available in the GitHub website:

https://github.com/BoschSensortec/BME280_driver

4 Using BME280 driver

The needed structs and routines:

```
52 wiced_spi_device_t spi_bosch;
53 wiced_spi_message_segment_t spi_bosch_msg;
```

```
67 //bme280 routines
68 void print_sensor_data(struct bme280_data *comp_data)
69 {
70 #ifdef FLOATING_POINT_REPRESENTATION
71         WPRINT_APP_INFO(("%0.2f\t\t%0.2f\t\t%0.2f\t\n",comp_data->temperature, comp_data->pressure, comp_data->humidity));
72 #else
73         WPRINT_APP_INFO(("temperature:%ld\t\tpressure:%ld\t\thumidity:%ld\t\n",comp_data->temperature, comp_data->pressure,
74                 comp_data->humidity));
75 #endif
76 }
```

In the application_start function:

```
209     spi_bosch.port      = WICED_SPI_1;
210     spi_bosch.chip_select = WICED_GPIO_NONE;//PIN_SPI_1_CS;//PIN_SPI_0_CS;
211     spi_bosch.speed       = 1000000;
212     spi_bosch.mode        = (SPI_CLOCK_RISING_EDGE | SPI_CLOCK_IDLE_HIGH | SPI_NO_DMA | SPI_MSB_FIRST);
213     spi_bosch.bits        = 8;
214
215
216     wiced_spi_init( &spi_bosch );
217     wiced_spi_init(&wiced_spi_flash);
```

```
245     mybme280_dev.read = BME280_SPI_bus_read;
246     mybme280_dev.write = BME280_SPI_bus_write;
247     mybme280_dev.delay_ms = BME280_delay_msek;
248
249
250     rslt = bme280_init(& mybme280_dev);
251
252     mybme280_dev.settings.osr_h = BME280_OVERSAMPLING_4X;
253     mybme280_dev.settings.osr_p = BME280_OVERSAMPLING_4X;
254     mybme280_dev.settings.osr_t = BME280_OVERSAMPLING_4X;
255
256     settings_sel = BME280_OSR_PRESS_SEL|BME280_OSR_TEMP_SEL|BME280_OSR_HUM_SEL;
257     rslt = bme280_set_sensor_settings(settings_sel, &mybme280_dev);
258
259
260     rslt = bme280_set_sensor_mode(BME280_NORMAL_MODE, &mybme280_dev);
261                 /* Give some delay for the sensor to go into normal mode */
262
263     WPRINT_APP_INFO( ( "INIT OK\r\n" ) );
264
```

```
336         while ( 1 )
337         {
338             {
339                 rslt = bme280_get_sensor_data(BME280_PRESS | BME280_HUM | BME280_TEMP, &comp_data, &mybme280_dev);
340                 WPRINT_APP_INFO(("[MQTT] Publishing..."));
341
342                 #ifdef FLOATING_POINT_REPRESENTATION
343                 sprintf(newmsg, "{\"_|deviceHid\": \"%s\",\"tmp\": \"%.1f\",\"prs\": \"%.2f\",\"hum\": \"%.2f\"}",
344                                                         HID, \
345                                                         comp_data.temperature, \
346                                                         comp_data.humidity,  comp_data.pressure);
347                 #else
348                 //WPRINT_APP_INFO(("temperature:%ld\t\tpressure:%ld\t\thumidity:%ld\t\n",comp_data->temperature, comp_data->pressure, com
349                 sprintf(newmsg, "{\"_|deviceHid\": \"%s\",\"tmp\": \"%ld\",\"prs\": \"%ld\",\"hum\": \"%ld\"}",
350                                                         HID, \
351                                                         comp_data.temperature, \
352                                                         comp_data.humidity,  comp_data.pressure);
353                 #endif
354                 do
355                 {
356                     ret = mqtt_app_publish( mqtt_object, WICED_MQTT_QOS_DELIVER_AT_LEAST_ONCE, (uint8_t*) WICED_TOPIC, (uint8_t*) newmsg,
357                     retries++ ;
358                 } while ( ( ret != WICED_SUCCESS ) && ( retries < MQTT_PUBLISH_RETRY_COUNT ) );
359                 if ( ret != WICED_SUCCESS )
360                 {
361                     WPRINT_APP_INFO((" Failed\n"));
362                     break;
363                 }
364                 else
365                 {
366                     WPRINT_APP_INFO((" Success\n"));
367                     print_sensor_data(&comp_data);
368                 }
369
370                 pub_in_progress = 0;
371                 count++ ;
372             }
373             wiced_rtos_delay_milliseconds( 5000 );
374         }
```

And what we need is the “support file:”

We need read-write and delay function.

```
int8_t BME280_SPI_bus_read(uint8_t dev_addr, uint8_t reg_addr, uint8_t *reg_data, uint8_t cnt)
{

    int32_t iError=0;
    uint8_t array[SPI_BUFFER_LEN]={0,};
    uint8_t arrayRX[SPI_BUFFER_LEN];
    uint8_t stringpos;
    wiced_spi_message_segment_t       message;
    /*  For the SPI mode only 7 bits of register addresses are used.
    The MSB of register address is declared the bit what functionality it is
    read/write (read as 1/write as BME280_INIT_VALUE)*/
    array[0] = reg_addr|SPI_READ;/*read routine is initiated register address is mask with 0x80*/


    message.length      = (cnt + 1);
    message.tx_buffer   = array;
    message.rx_buffer   = arrayRX;

    //Send the command
    wiced_spi_transfer( &spi_bosch, &message, 1 );

    WPRINT_APP_INFO( ( "SPI READ\r\n" ) );

    memcpy(reg_data, &arrayRX[1], cnt);

    return (int8_t)iError;
}

int8_t BME280_SPI_bus_write(uint8_t dev_addr, uint8_t reg_addr, uint8_t *reg_data, uint8_t cnt)
{
    int32_t iError = 0;
    uint8_t array[SPI_BUFFER_LEN * BME280_ADDRESS_INDEX];
    uint8_t arrayrx[SPI_BUFFER_LEN * BME280_ADDRESS_INDEX];
    uint8_t stringpos = 0;
    uint8_t index = 0;
    wiced_spi_message_segment_t       message;

    for (stringpos = 0; stringpos < cnt; stringpos++) {
        /* the operation of (reg_addr++)&0x7F done as per the
        SPI communication protocol specified in the data sheet*/
        index = stringpos * BME280_ADDRESS_INDEX;
        array[index] = (reg_addr++) & SPI_WRITE;
        array[index + BME280_DATA_INDEX] = *(reg_data + stringpos);
    }

    message.length      = (cnt + 1);
    message.tx_buffer   = array;
    message.rx_buffer   = arrayrx;

    //Send the command
    wiced_spi_transfer( &spi_bosch, &message, 1 );

    WPRINT_APP_INFO( ( "SPI WRITE\r\n" ) );
    /* Please take the below function as your reference
     * for write the data using SPI communication
     * add your SPI write function here.
     * "IERROR = SPI_WRITE_STRING(ARRAY, CNT*2)"
     * iError is an return value of SPI write function
     * Please select your valid return value
     * In the driver SUCCESS defined as 0
     * and FAILURE defined as -1
     */
    return (int8_t)iError;
}

void BME280_delay_msek(uint32_t msek)
{
    /*Here you can write your own delay routine*/
    wiced_rtos_delay_milliseconds(msek );

    WPRINT_APP_INFO( ( "Delay %d ms\r\n", msek ) );
}
```

This is how its looks when working: