

# Quadro – Bosch BME280 lab

Attila Mák  
01.08.2017



©2017 ARROW

All rights reserved. No part of this manual shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical photocopying, desktop publishing, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of information contained herein. While every precaution has been taken in the preparation of this document, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein. All terms mentioned in this manual that are known to be trademarks or service marks are listed below. In addition, terms suspected of being trademarks or service marks have been appropriately capitalized. ARROW cannot attest to the accuracy of this information. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Revision History

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

# Contents

1	Scope/ description .....	5
2	Hardware setup.....	5
2.1	Arrow Quadro board.....	6
2.2	ONSEMI IDK board .....	7
3	Software setup.....	8
3.1	IDE installation to the PC .....	8
3.1.1	WICED installation.....	8
	Setup JRE 32 and JRE64 java runtime .....	9
	Setup Wiced Studio (WINDOWS).....	9
	SETUP WICED STUDIO (OTHER OS).....	14
	OSX.....	14
	Linux 64-bit .....	14
	Linux 32 bit.....	14
3.1.2	ONSEMI IDK installation.....	<b>Error! Bookmark not defined.</b>
4	Using ONSEMI IDK.....	14
5	Quadro board.....	<b>Error! Bookmark not defined.</b>

# 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.

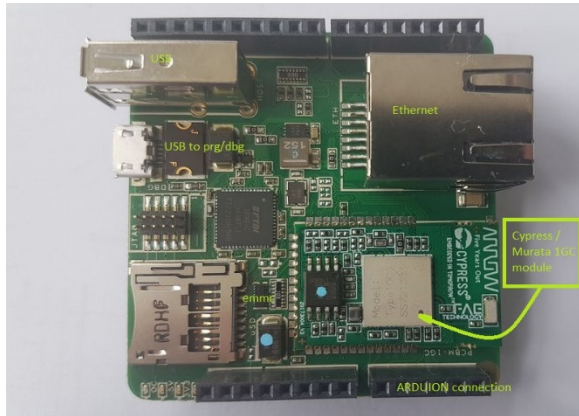
Desktop computer or laptop with x86 architecture and USB 2.0

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>

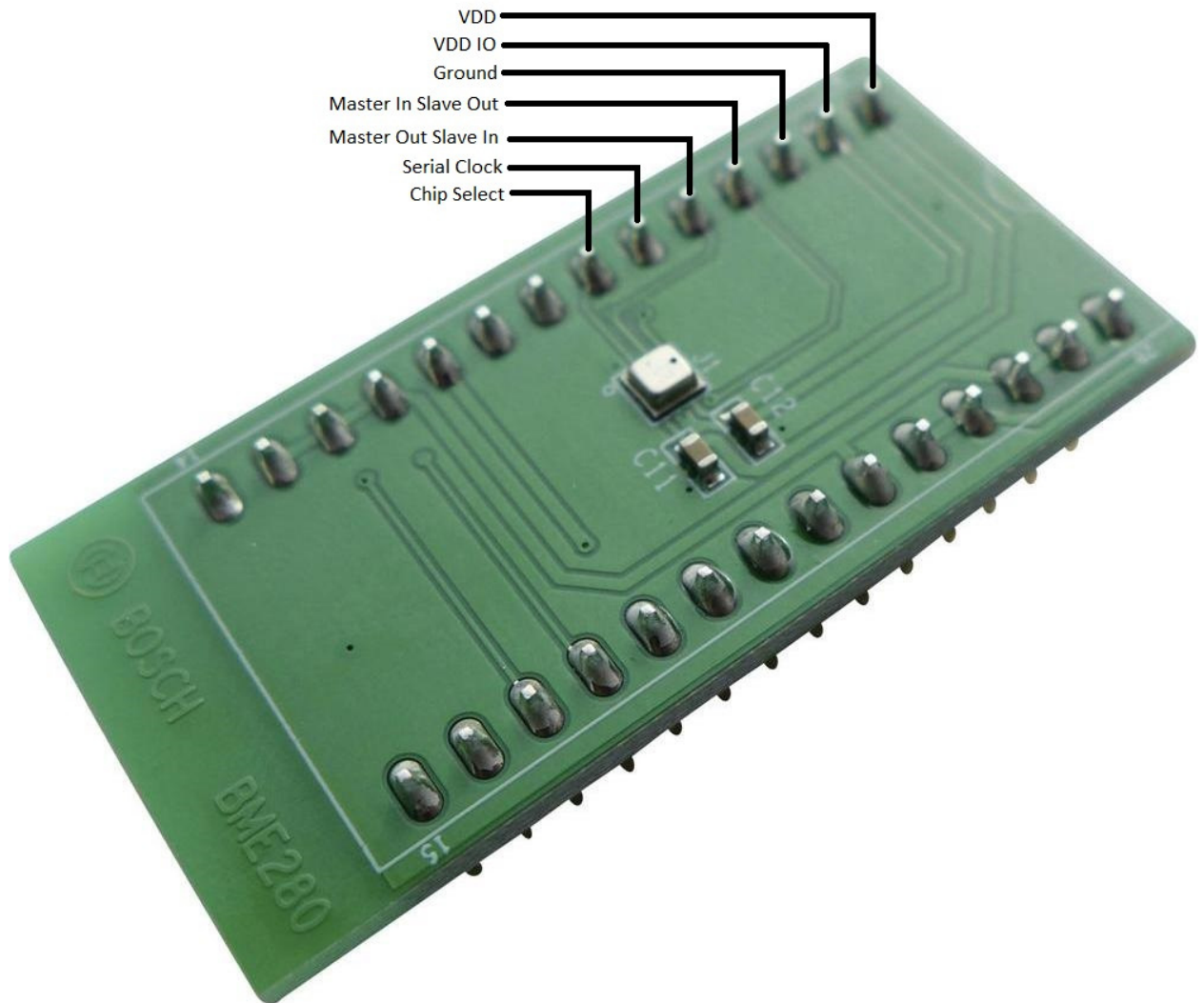


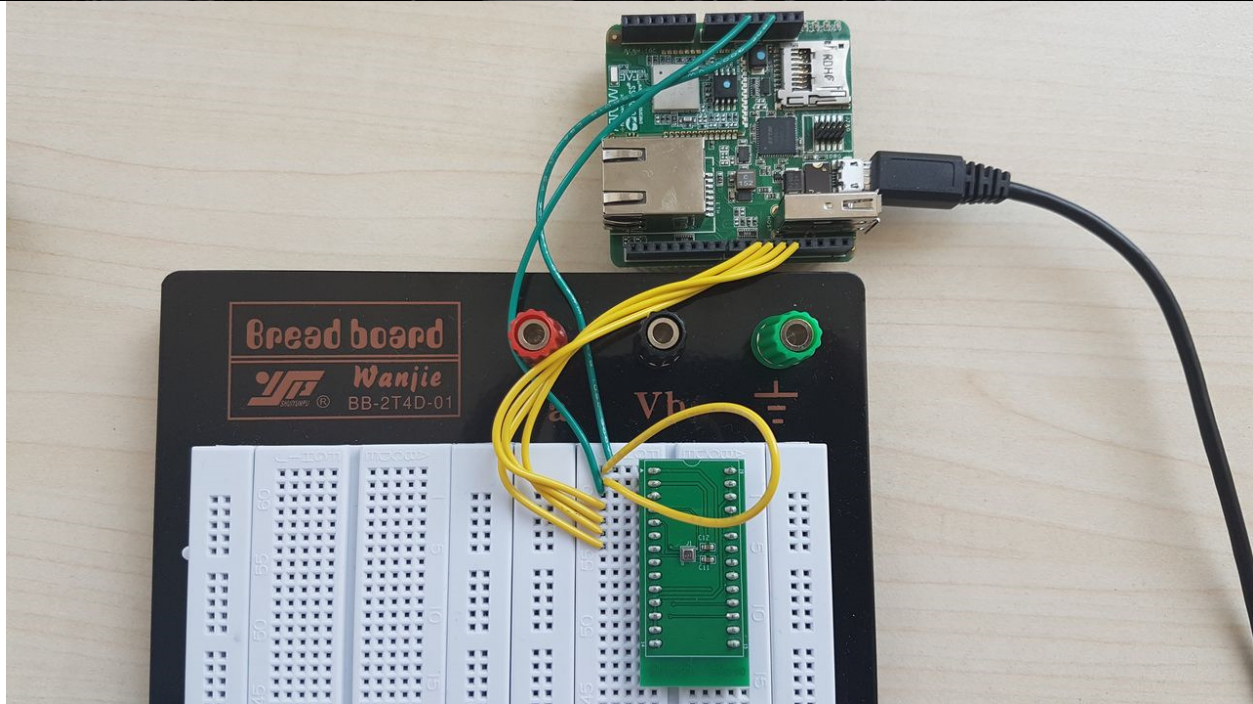
Arduino connection:

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](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)

<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

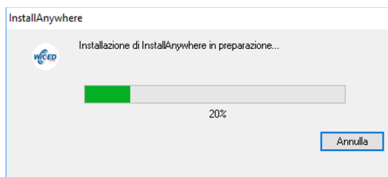
### Setup JRE 32 and JRE64 java runtime

- 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:

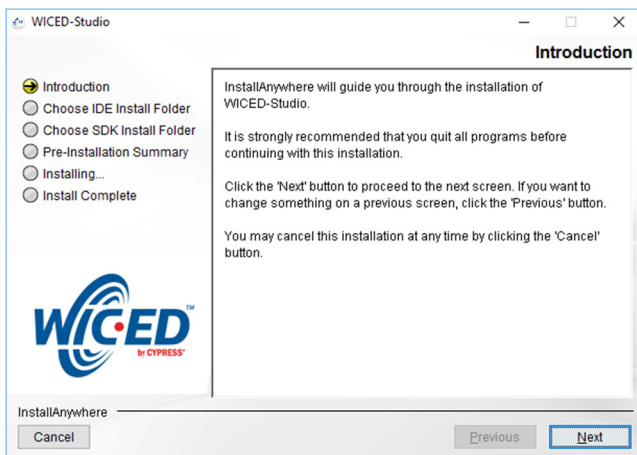
C:\ProgramData\Oracle\Java\javapath

### Setup Wiced Studio (WINDOWS)

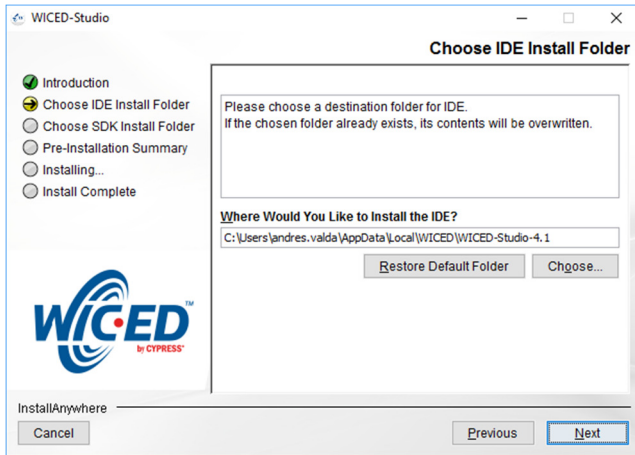
- 1) install Cypress WICED Studio 4.1.1 development tools by unzip file WICED-Studio-4.1.1.8-IDE-Installer.exe.zip and execute WICED-Studio-4.1.1.8-IDE-Installer.exe
- 2) When Setup start Appear window below



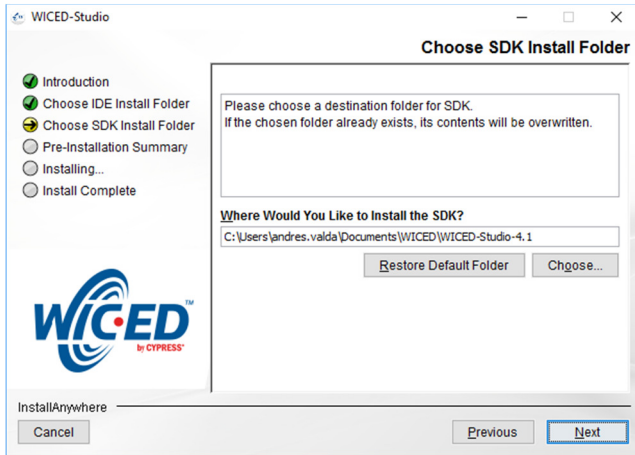
And then



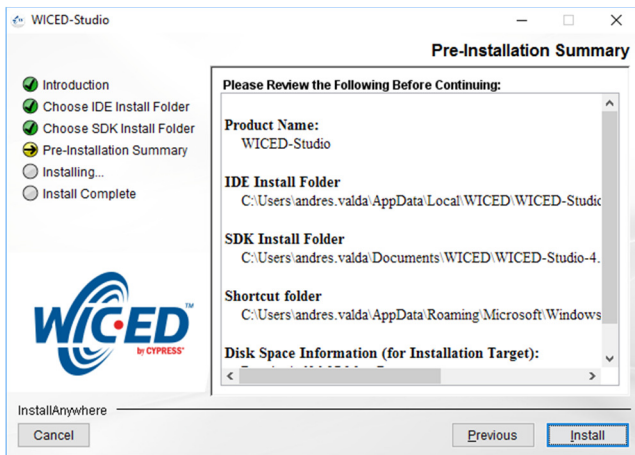
Press Next Button



Press Next Button

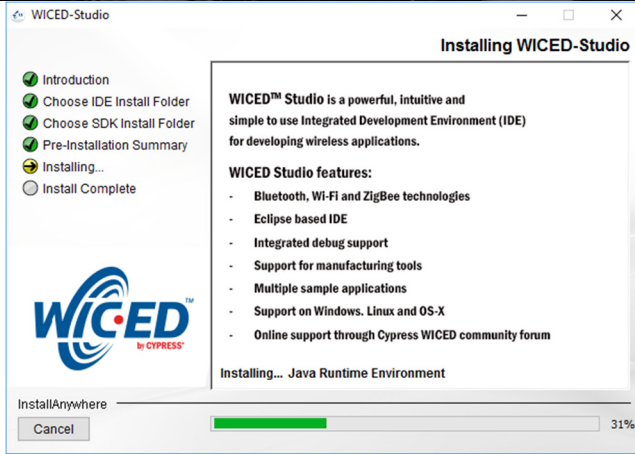


Press Next Button

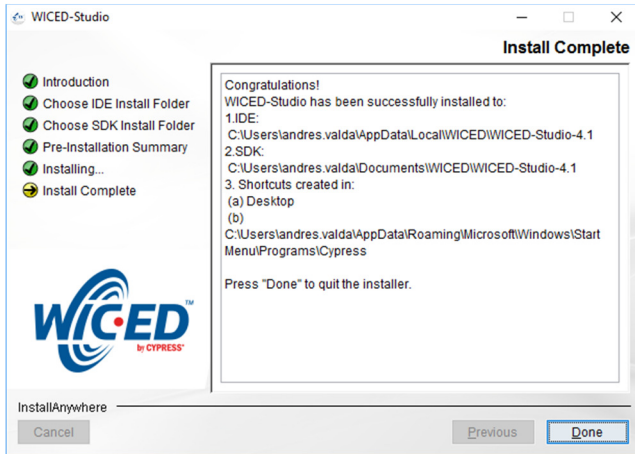


Press Install

# Quadro – Bosch BME280 driver integration

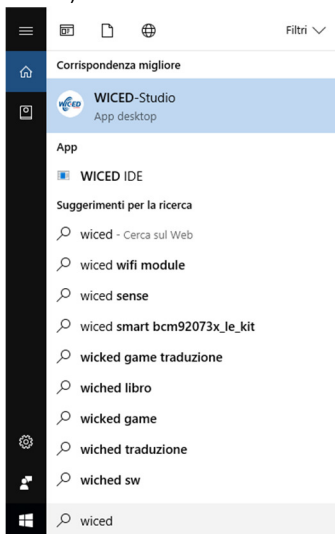


Wait while setup finish

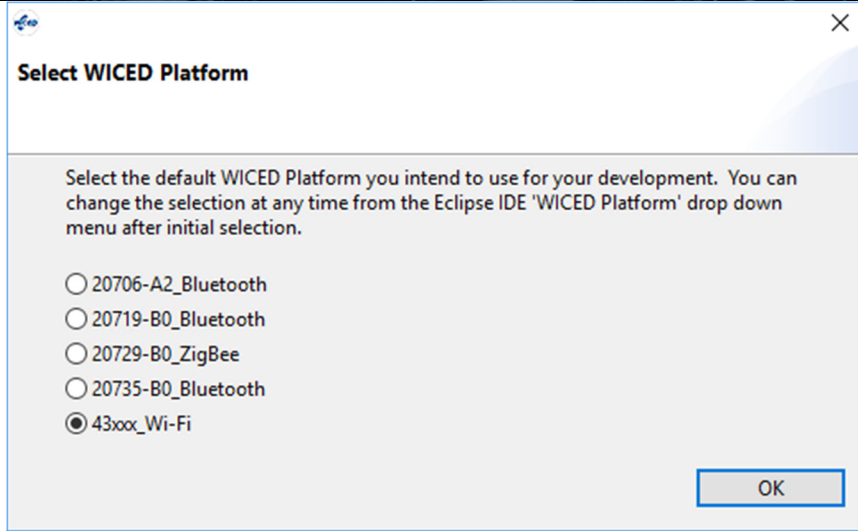


And press Done Button

3) Launch the WICED SDK

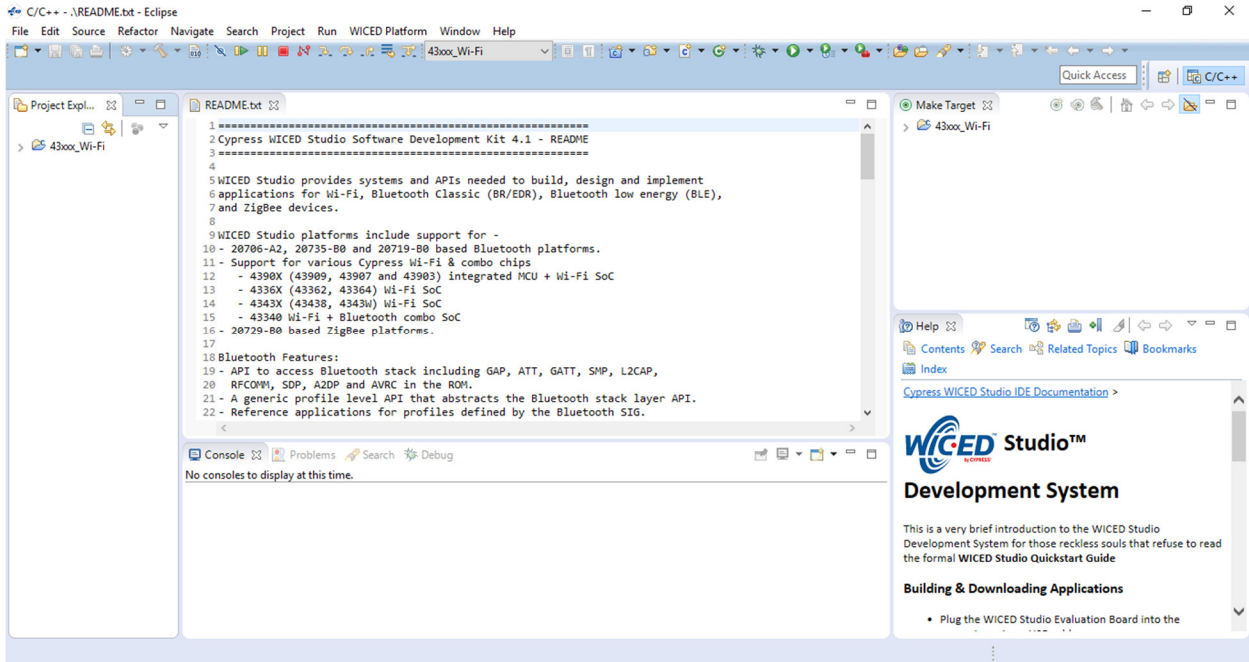


Select 43xxx\_Wi-Fi Platform

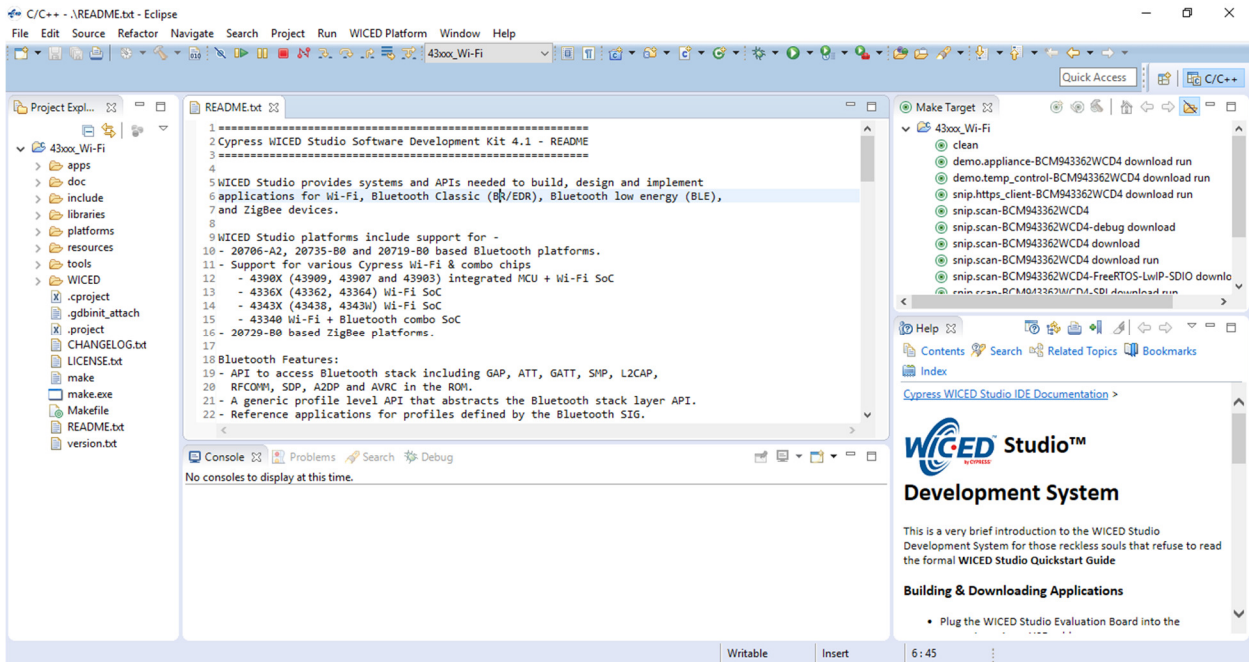


# Quadro – Bosch BME280 driver integration

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](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     #ifndef FLOATING_POINT_REPRESENTATION
71         WPRINT_APP_INFO(("%.2f\t%.2f\t%.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);
218
    
```

```

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
251
252
253
254
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
264 WPRINT_APP_INFO( ( "INIT OK\r\n" ) );
265
336
337
338
339
340
341
342 #ifndef 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
348 #else
349 //WPRINT_APP_INFO(("temperature:%ld\t\tpressure:%ld\t\thumidity:%ld\t\n", comp_data->temperature, comp_data->pressure, com
350 sprintf(newmsg, "{ \"_deviceHid\": \"%s\", \"tmp\": \"%.1d\", \"prs\": \"%.1d\", \"hum\": \"%.1d\" }",
351         HID, \
352         comp_data.temperature, \
353         comp_data.humidity, comp_data.pressure);
354
355 #endif
356 do
357 {
358     ret = mqtt_app_publish( mqtt_object, WICED_MQTT_QOS_DELIVER_AT_LEAST_ONCE, (uint8_t*) WICED_TOPIC, (uint8_t*) newmsg,
359     retries++ ;
360 } while ( ( ret != WICED_SUCCESS ) && ( retries < MQTT_PUBLISH_RETRY_COUNT ) );
361 if ( ret != WICED_SUCCESS )
362 {
363     WPRINT_APP_INFO((" Failed\n"));
364     break;
365 }
366 #else
367 {
368     WPRINT_APP_INFO((" Success\n"));
369     print_sensor_data(&comp_data);
370 }
371
372 pub_in_progress = 0;
373 count++ ;
374 wiced_rtos_delay_milliseconds( 5000 );
375
376 }
377
378 }

```

And what we need is the “support file:”

We need read-write and delay function.

```

100 int8_t BME280_SPI_bus_read(uint8_t dev_addr, uint8_t reg_addr, uint8_t *reg_data, uint8_t cnt)
11 {
12     {
13         int32_t iError=0;
14         uint8_t array[SPI_BUFFER_LEN]={0,};
15         uint8_t arrayRX[SPI_BUFFER_LEN];
16         uint8_t stringpos;
17         wiced_spi_message_segment_t message;
18         /* For the SPI mode only 7 bits of register addresses are used.
19          The MSB of register address is declared the bit what functionality it is
20          read/write (read as 1/write as BME280_INIT_VALUE)*/
21         array[0] = reg_addr|SPI_READ;/*read routine is initiated register address is mask with 0x80*/
22     }
23     {
24         message.length = (cnt + 1);
25         message.tx_buffer = array;
26         message.rx_buffer = arrayRX;
27     }
28     //Send the command
29     wiced_spi_transfer( &spi_bosch, &message, 1 );
30 }
31 WPRINT_APP_INFO( ( "SPI READ\r\n" ) );
32 {
33     memcpy(reg_data, &arrayRX[1], cnt);
34 }
35 return (int8_t)iError;
36 }
37 }
38
39 int8_t BME280_SPI_bus_write(uint8_t dev_addr, uint8_t reg_addr, uint8_t *reg_data, uint8_t cnt)
40 {
41     int32_t iError = 0;
42     uint8_t array[SPI_BUFFER_LEN * BME280_ADDRESS_INDEX];
43     uint8_t arrayrx[SPI_BUFFER_LEN * BME280_ADDRESS_INDEX];
44     uint8_t stringpos = 0;
45     uint8_t index = 0;
46     wiced_spi_message_segment_t message;
47 }
48 for (stringpos = 0; stringpos < cnt; stringpos++) {
49     /* the operation of (reg_addr++)&0x7F done as per the
50     SPI communication protocol specified in the data sheet*/
51     index = stringpos * BME280_ADDRESS_INDEX;
52     array[index] = (reg_addr++) & SPI_WRITE;
53     array[index + BME280_DATA_INDEX] = *(reg_data + stringpos);
54 }
55
56 message.length = (cnt + 1);
57 message.tx_buffer = array;
58 message.rx_buffer = arrayrx;
59
60 //Send the command
61 wiced_spi_transfer( &spi_bosch, &message, 1 );
62
63 WPRINT_APP_INFO( ( "SPI WRITE\r\n" ) );
64 /* Please take the below function as your reference
65 * for write the data using SPI communication
66 * add your SPI write function here.
67 * "iERROR = SPI_WRITE_STRING(ARRAY, CNT*2)"
68 * iError is an return value of SPI write function
69 * Please select your valid return value
70 * In the driver SUCCESS defined as 0
71 * and FAILURE defined as -1
72 */
73 return (int8_t)iError;
74 }
75 }
76
77 void BME280_delay_msek(uint32_t msek)
78 {
79     /*Here you can write your own delay routine*/
80     wiced_rtos_delay_milliseconds(msek );
81 }
82 WPRINT_APP_INFO( ( "Delay %d ms\r\n", msek ) );
83 }
84 }
85 }

```



This is how its looks when working:

```
Starting WICED v4.1.1
Platform BCM943907_QUADRO initialised
Started ThreadX v5.6
Initialising NetX_Duo v5.7_sp2
Creating Packet pools
WMD SoC-43999 interface initialised
WLAN MAC address : 00:CC:2B:70:CF:18
WLAN Firmware : w10: Dec 19 2016 19:29:37 version 7.15.168.78 (<r663126> FWID 01-8ba7c839
SPI MAC ADDRESS = FFFFFFF8:FFFFFF80:39:4E:FFFFFFF88:4A
Function OK
Function OK
SPI READ
Function OK
Function OK
SPI WRITE
Delay 2 ms
Function OK
SPI READ
Function OK
SPI READ
Function OK
Function OK
Function OK
Function OK
SPI READ
Function OK
SPI WRITE
Function OK
SPI READ
Function OK
SPI WRITE
Function OK
SPI READ
Function OK
SPI READ
Function OK
SPI WRITE
Function OK
SPI READ
Function OK
SPI READ
Function OK
SPI WRITE
INIT OK
Joining : ANE-WIFI
Successfully joined : ANE-WIFI
Obtaining IPv4 address via DHCP
DHCP CLIENT hostname WICED IP
IPv4 network ready IP: 192.168.0.107
Setting IPv6 link-local address
IPv6 network ready IP: FE80:0000:0000:0000:A2CC:2BFF:FE70:CF18
Resolving IP address of MQTT broker...
Resolved Broker IP: 159.8.169.212
[MQTT] Opening connection...Success
Function OK
Function OK
SPI READ
[MQTT] Publishing... Success
temperature:2730 pressure:10068853 humidity:59754
Function OK
SPI READ
[MQTT] Publishing... Success
temperature:2665 pressure:10059042 humidity:59599
Function OK
SPI READ
[MQTT] Publishing... Success
temperature:2633 pressure:10053525 humidity:59546
```