AN3406

Fast Prototyping of BLE Sensors for AWS Cloud Using the ATmega4809 Curiosity Nano Development Platform

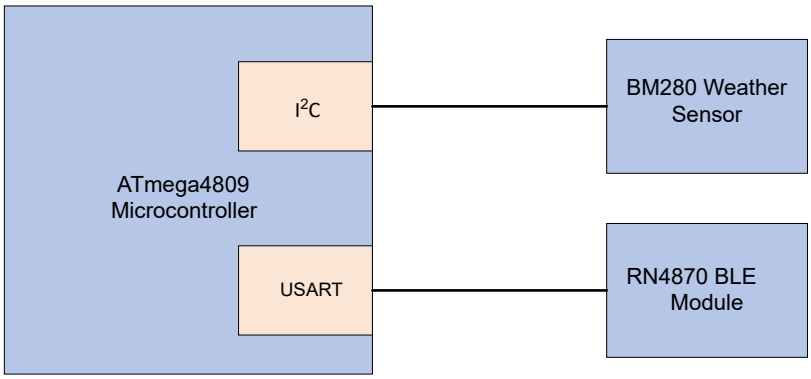
Introduction

Author: Ioan Pop, Alin Stoicescu, Microchip Technology Inc.

This application note describes how to use the ATmega4809 Curiosity Nano Development Platform to create a low-power BLE (Bluetooth® Low Energy) sensor node that connects to the Amazon Web Services (AWS) Cloud through a gateway. This document includes a detailed tutorial and links to a GitHub repository containing the documented code.

The sensor edge node is controlled by the ATmega4809 microcontroller, a robust, low-power microcontroller engineered for real-time control applications. To prototype the sensor node, the ATmega4809 Curiosity Nano was used in conjunction with the Curiosity Nano Base for click boards™, which has three mikroBUS™ slots. These mikroBUS sockets are used to connect the RN4870 click board, which incorporates a Microchip RN4870 Bluetooth Low Energy module, as well as the Weather click board, which contains a BME280 sensor to provide humidity, temperature, and pressure readings. The application is intended to be an off-the-shelf system – each of the components is designed for seamless prototyping on the Curiosity Nano Platform and can quickly be configured in software through MPLAB® Code Configurator (MCC).

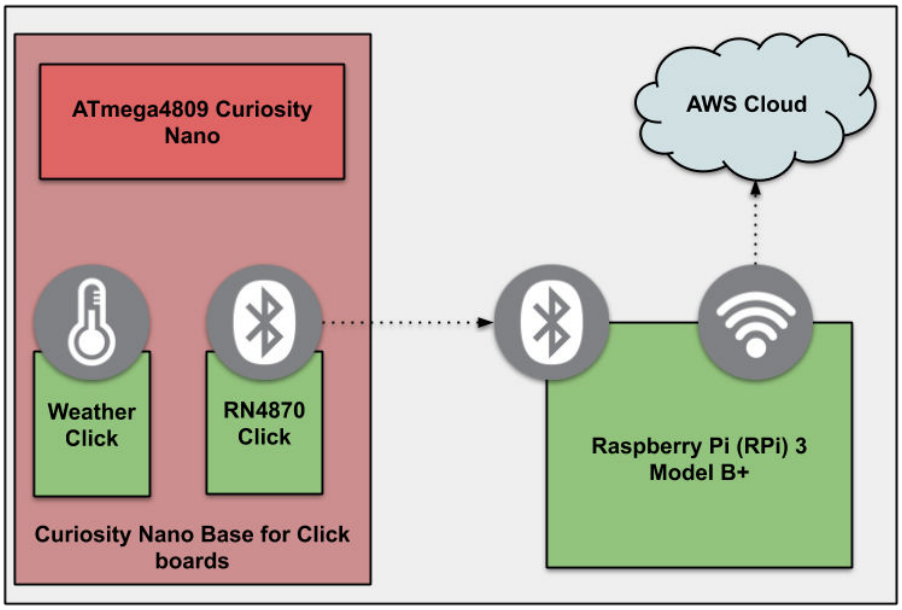
Figure 1. Sensor Node Block Diagram



When the application starts, the microcontroller creates three public characteristics for pressure, temperature, and humidity data in the BLE module for the sensor data. The microcontroller reads the data from the sensor every five seconds and updates the characteristics with the respective values. A gateway is required to transmit the information to the cloud. The gateway establishes a connection with the BLE module of the end node and connects to the cloud services from a local Internet network through Ethernet or Wi-Fi®. Every five seconds, the gateway reads the characteristics and publishes the data to the cloud.

© 2020 Microchip Technology Inc.

Figure 2. Application Functionality Diagram



A Raspberry Pi® (RPi) 3 Model B+ board is used as a gateway since it incorporates BLE, Wi-Fi, and Ethernet modules, and it is easy to use and provides good documentation.

AWS is used for cloud services. It provides a gateway core software called AWS IoT Greengrass that allows the users to interact with the cloud. Publishing and subscription to topics can be done just by utilizing Application Programming Interfaces (API) in Lambda.

The source code for peripherals used in the microcontroller is generated using MCC. The code generator creates the structure of the project, the files, and the source code. MCC provides support for every peripheral and development extension boards such as the RN4870 click board.

The following GitHub repository contains the project source code for the ATmega4809 Curiosity Nano board together with the Lambda used by the gateway to read characteristics and publish the data to the cloud.

View Code Example on GitHub
Click to browse repository

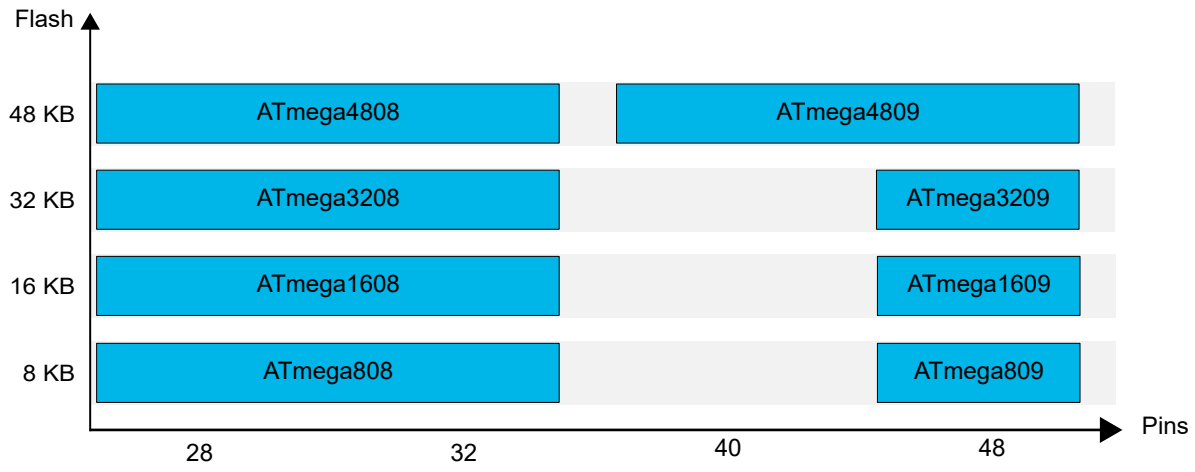
Table of Contents

1. Relevant Devices

This section lists the relevant devices for this document. The following figures show the different family devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features.
- Horizontal migration to the left reduces the pin count and, therefore, the available features
- Devices with different Flash memory sizes typically also have different SRAM and EEPROM

Figure 1-1. megaAVR® 0-series Overview



2. Overview

This application note explains how to obtain the hardware and software configurations required by the BLE Internet of Things (IoT) project. The functionality of the example project and the AWS configurations are also detailed here.

In the example project, the featured device reads pressure, temperature, and humidity data from the weather sensor. The sensor sends data through an I^2C interface, and a five-second reading interval is used for transmitting the data to the cloud. The device exchanges information with the RN4870 BLE device through the USART interface and starts creating services and characteristics for the sensor and updates them after every reading.

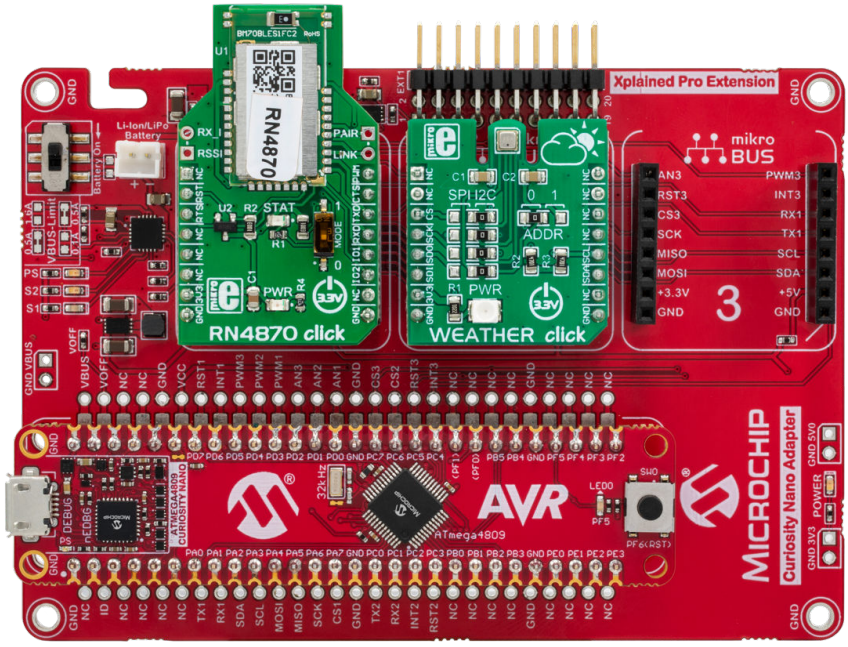
The Raspberry Pi board connects to the BLE device, reads the characteristics, and, using the Greengrass core, publishes the data to the cloud. Greengrass is a gateway software provided by AWS capable of using the hardware components of the Raspberry Pi board. Thus, the gateway core can interact with nearby BLE node devices and can provide secure communication to the cloud.

Prerequisites:

- Integrated Development Environment (IDE):
    1. Atmel Studio v7 with the latest device packages installed.
    2. MPLAB® X v5.30 with XC8 v2.10 compiler (used in this project).
        - 2.1. MCC version 3.85.
        - 2.2. AVR8 Peripheral Library AVR® microcontroller version 2.0.1.
- Microchip ATmega4809 Curiosity Nano development board
- Microchip Curiosity Nano Base for click boards™
- BLE RN4870 click board from MikroE
- Raspberry Pi 3 Model B+
- Weather click board from MikroE
- Amazon Web Services (AWS) Account

3. Hardware Description

Figure 3-1. Sensor Edge Node Prototyping Platform

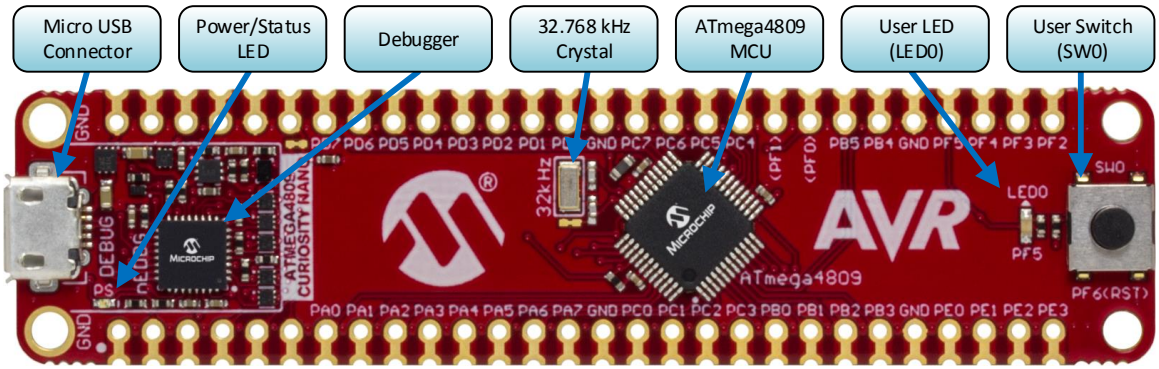


3.1 ATmega4809 Curiosity Nano Board and the Curiosity Nano Adapter

ATmega4809 is a microcontroller featuring the 8-bit AVR® processor with hardware multiplier, running at up to 20 MHz and with up to 48 KB Flash, 6 KB SRAM, and 256 bytes of EEPROM in 48-pin packages. It also provides Core Independent Peripherals (CIPs) and flexible low-power architecture, including Event System and SleepWalking.

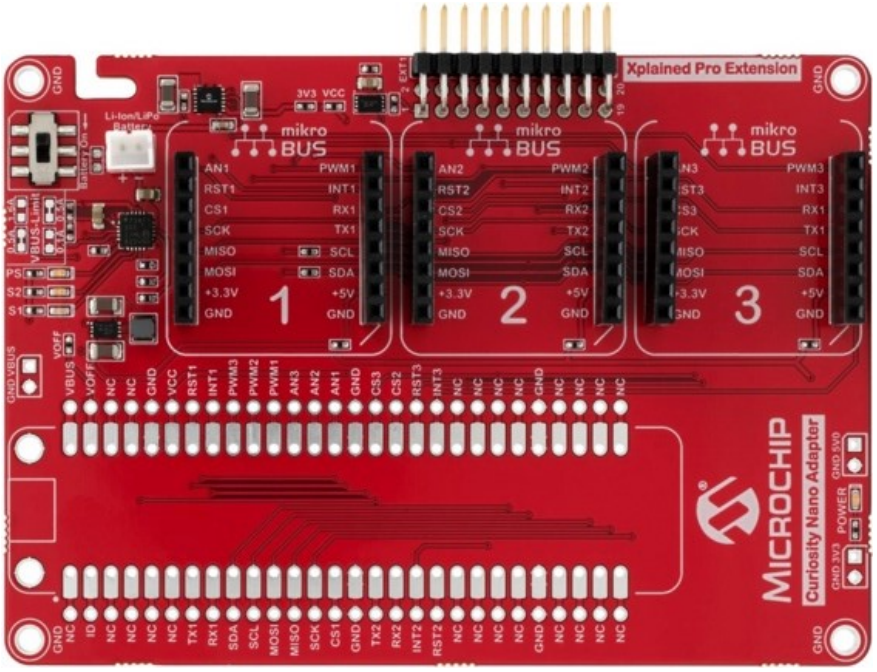
The microcontroller is intended for developing IoT and communications applications, as it offers a sizeable memory, a powerful core, and fast clock speeds.

Figure 3-2. ATmega4809 Curiosity Nano



ATmega4809 Curiosity Nano (DM320115) contains an embedded debugger (EDBG) that allows fast programming of the microcontroller and provides a USART – USB bridge for virtual COM port communication. The debugger supports both IDE programming and drag-and-drop programming.

Figure 3-3. Curiosity Nano Base for Click boards



The Curiosity Nano Base for click boards (AC164162) is a board that facilitates easy connections for Curiosity Nano boards. It offers three mikroBUS slots for easy interfacing with click boards, an Xplained Pro extension header and a Li-Ion/LiPo charger and management circuit for connecting a battery to power up the connected devices.

3.2 MIKROE RN4870 Click Board

The RN4870 click board is a BLE board which can be inserted in the mikroBUS slot. It is powered by 3.3V and is connected to the host microcontroller by the USART peripheral, allowing control via ASCII commands. The board also provides support and channels for Ready-to-Send (RTS) and Clear-to-Send (CTS) USART signals, which are optional and will not be used in this application.

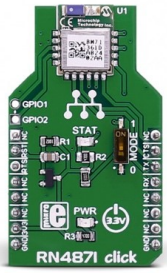
The RN4870 is a Bluetooth low-energy module. It has a completely integrated software based on the Bluetooth stack version 5.

The module supports transparent USART, which is used for data transfer from USART to a peer BLE device. The default baud rate is 115200 symbols/s, but it can be changed from the command interface. The USART data exchange is done through two channels, one for transmitting and one for receiving.

Note: Either the RN4870 or RN4871 Bluetooth modules can be used in this application.

Figure 3-4. RN4870 Click Board

Figure 3-5. RN4871 Click Board



The module supports Generic Attribute Profile characteristics (GATT), and it can implement up to five public and four private user-defined GATT services, each service providing up to eight characteristics.

When interfaced with a BLE-enabled smartphone or Bluetooth Internet Gateway, the applications can be monitored, controlled, and updated from anywhere in the world. Thus, the RN4870 module is perfect for IoT applications.

3.3 Raspberry Pi® 3 Model B+ Board

The Raspberry Pi 3 Model B+ development board runs various distributions of the Linux® operating system, granting open-source software libraries, drivers, and applications to the users. These allow easy interfacing with other devices. The board provides a built-in BLE module for communication with the end nodes and Wi-Fi and Ethernet socket for the cloud connection and communication, favorable for IoT. The RPi 3 B+ has an armv71 processor architecture and will be used in this demo application, as a gateway. A microSD card, which is at least 8 GB in size, is needed for the operating system of RPi and the gateway software.

3.4 MikroE Weather Click Board

The Weather click board contains a BME280 Bosch sensor that reads pressure, temperature, and humidity data from the environment. The sensor can communicate through SPI or I^2C. The default communication is based on I^2C, and using SPI requires resoldering 0-Ohm resistors on the click board.

The sensor provides great accuracy with ±0.1 kPa for pressure, ±1°C for temperature, and ±3% relative humidity. The sensor is also very small and is intended for use in smartwatches, smartphones, or similar applications, making it perfect for IoT development.

Figure 3-6. Mikroe Weather click

4. Application Overview

4.1 Software Requirements

Microchip's MPLAB X and Atmel Studio are Integrated Development Environments used to develop applications for microcontrollers. The application presented in this document has been developed in MPLAB X. The MPLAB X IDE v5.30 and the XC8 compiler v2.10 or later are required for the correct functionality of this project.

MPLAB X IDE gives a seamless and easy-to-use environment to write, build, and debug user applications written in C/C++ or assembly code. MPLAB X supports all Microchip microcontrollers and development tools.

There is a plug-in for MPLAB X called MPLAB Code Configurator, which provides code generation features to facilitate the use of peripherals. These tools can also be used online on MPLAB Xpress.

4.2 Amazon Web Service Software Overview

An account is needed to use Amazon Web Services. It can be created at no charge and allows the use of the gateway software and AWS for up to one year for multiple devices. The cloud account and gateway (which is called Greengrass) configurations are not the focus of this document and will not be explained in detail. AWS already provides a well-documented and written developer guide which can be found at Getting Started with AWS IoT Greengrass.

The Greengrass developer guide is divided into chapters, based on the focus of the information presented. These chapters are called modules.

Module 1 explains how to download and load the operating system for Raspberry Pi and how to establish a connection between the board and the computer. This module also gives a step-by-step explanation of what configurations are needed for the board to safely run the Greengrass.

Module 2 describes how to download and install the gateway software on Raspberry Pi. It comes with public and private keys and a certificate, which are mandatory for cloud connection. This module provides details on how to start the gateway and how to check the process which handles the gateway.

Therefore, these two modules are mandatory for configuring the Raspberry Pi board and getting started with the Greengrass core.

Module 3 is focused on Lambda and data exchange between the cloud and the RPi. The module explains how to add and deploy Lambda on Raspberry Pi. AWS Lambda is a function that runs code without provisioning or managing servers. The Lambda can access the cloud by sending Message Queuing Telemetry Transport (MQTT) messages on specific topics. The sending procedure is called publishing. The client can subscribe to a specific topic and receive the messages.

Module 3 is also mandatory since the purpose of the application presented in this document is to send BLE data to the cloud, and this will be implemented by using Lambda. The other modules are not required nor used throughout this application note.

Note: Greengrass core (GGC) v1.10 is used in this project.

4.3 ATmega4809 Software Overview

The project is developed using MPLAB X, and the peripherals required were configured with MCC. A new project is created by pressing *File>New Project* within MPLAB X IDE. The ATmega4809 device must be selected.

The whole project is provided in the GitHub repository. The purpose of this section is to familiarize users with MPLAB Code Configurator, the structure of the generated code, and to explain the usage of every peripheral and its configurations.

1. **System Module Configuration**
   This is where the system's clock is configured. This application uses a 20 MHz oscillator prescaled ten times to 2 MHz.

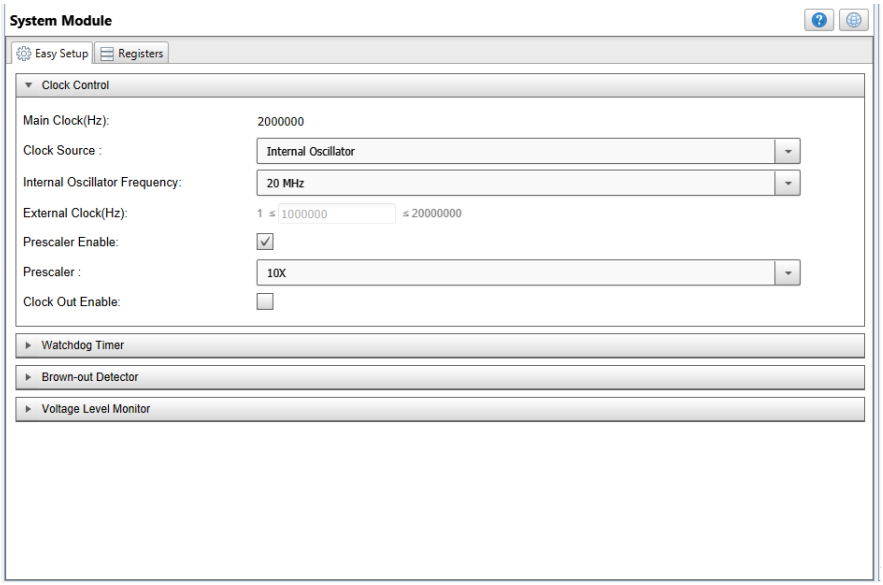
Clock Source: Internal Oscillator

Internal Oscillator Frequency: 20 MHz

Prescaler Enable: Yes

Prescaler: 10X

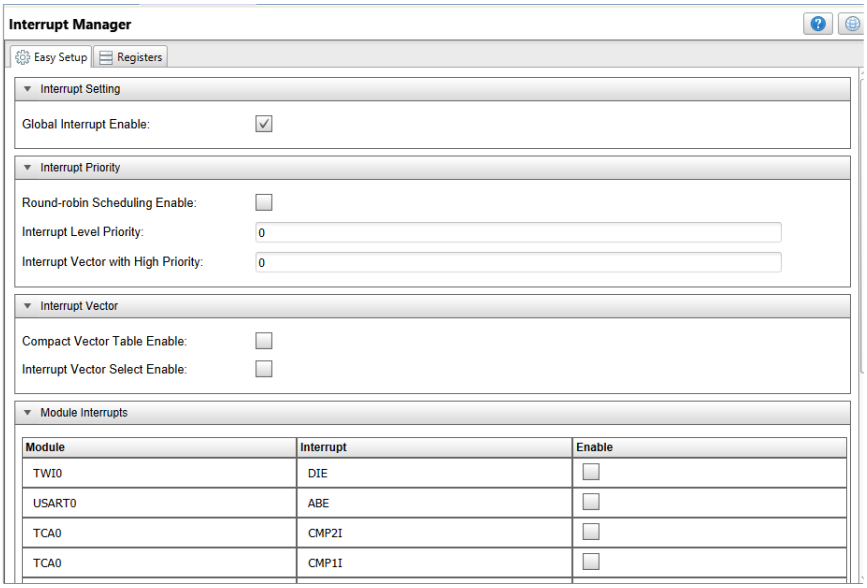
Figure 4-1. System Module Configuration



2. **Interrupt Manager Configuration**
The peripherals that use the interrupts will be configured further. For now, Global Interrupts need to be enabled.

Global Interrupt Enable: Yes

Figure 4-2. Interrupt Manager Configuration



The peripherals configured next can be added from the **Devices Resources** tab found under the **Project Resources** tab.

3. **USART0 Configuration**
This is the USART instance that communicates with the RN4870 module and sends commands to it. The API prefix has been changed and needs to be called exactly that, to work with the code.

API Prefix: USART_RN4870

Interrupt Driven: Yes

RX Buffer Size: 8 bytes

TX Buffer Size: 8 bytes

Printf support: Yes

Mode: Async Mode

Baud Rate: 115200

Enable USART Receiver: Yes

Enable USART Transmitter: Yes

Parity Mode: No parity

Stop Bit Mode: 1 stop bit

Character Size: 8 bit

The receive interrupt also needs to be enabled in Interrupt Settings.

Figure 4-3. USART0 Configuration

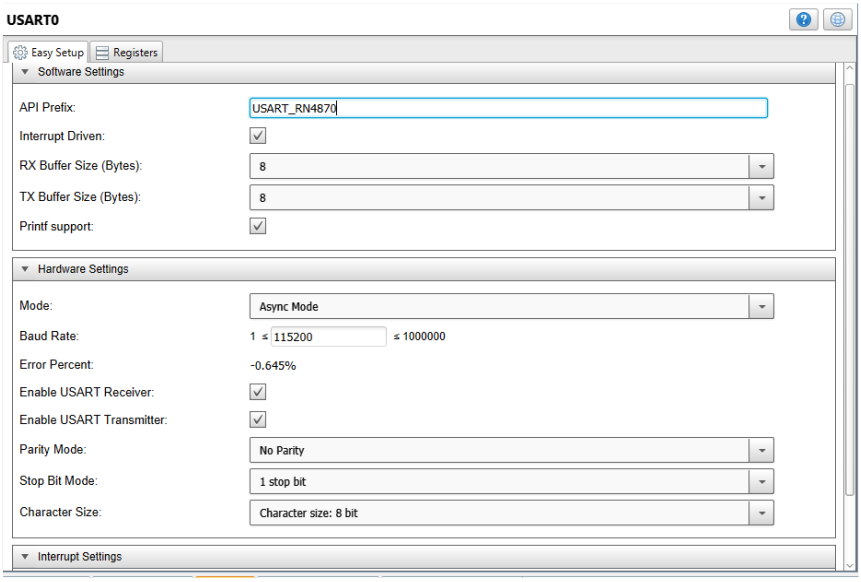
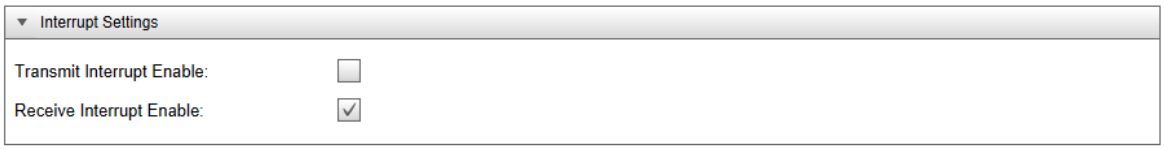


Figure 4-4. USART0 Interrupt Settings Configuration



4. **USART3 Configuration**
This USART instance communicates with the embedded debugger, which acts as a serial bridge and sends the information to the USB connected device. The statement about the API prefix also applies here.

API Prefix: USART_TERMINAL

Interrupt Driven: Yes

RX Buffer Size: 8 bytes

TX Buffer Size: 8 bytes

Printf support: No

Mode: Async Mode

Baud Rate: 115200

Enable USART Receiver: Yes

Enable USART Transmitter: Yes

Parity Mode: No parity

Stop Bit Mode: 1 stop bit

Character Size: 8 bit

Figure 4-5. USART3 Configuration

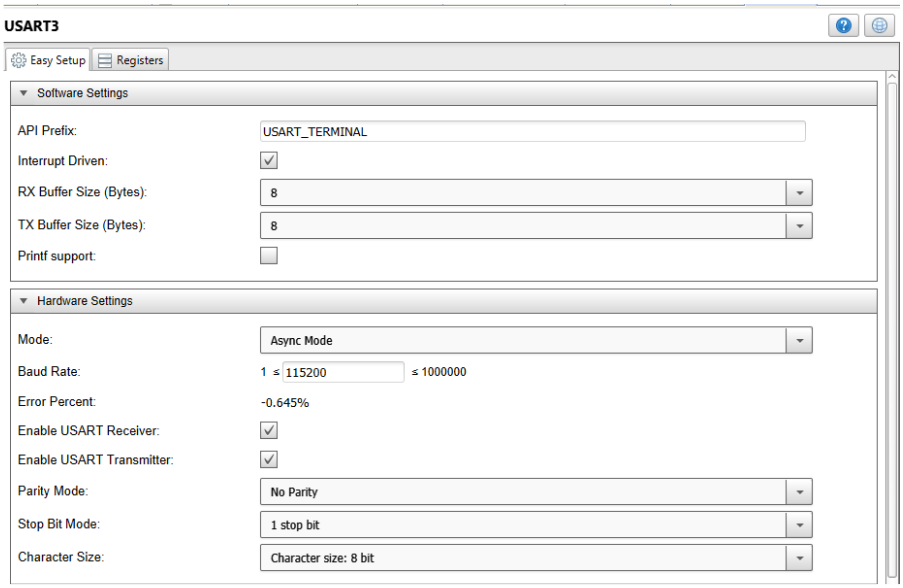
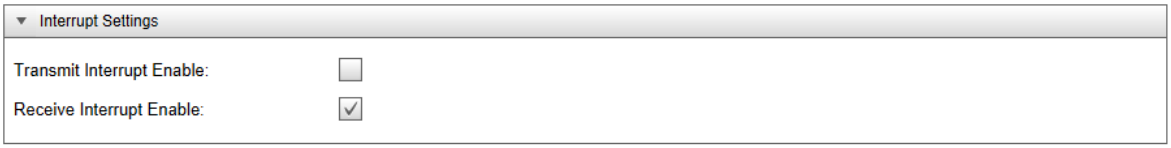


Figure 4-6. USART3 Interrupt Settings Configuration



5. **Timer A (TCA0) Configuration**
   This timer is used for counting to five seconds, after which the microcontroller reads the sensor values and stores them into the RN4870 module's characteristics. The timer will be enabled at a later time in the code.

   API Prefix: TIMER_0

   Enable Timer: No

   Clock Selection: System Clock/256

   Timer mode: 16 Bit (Normal)

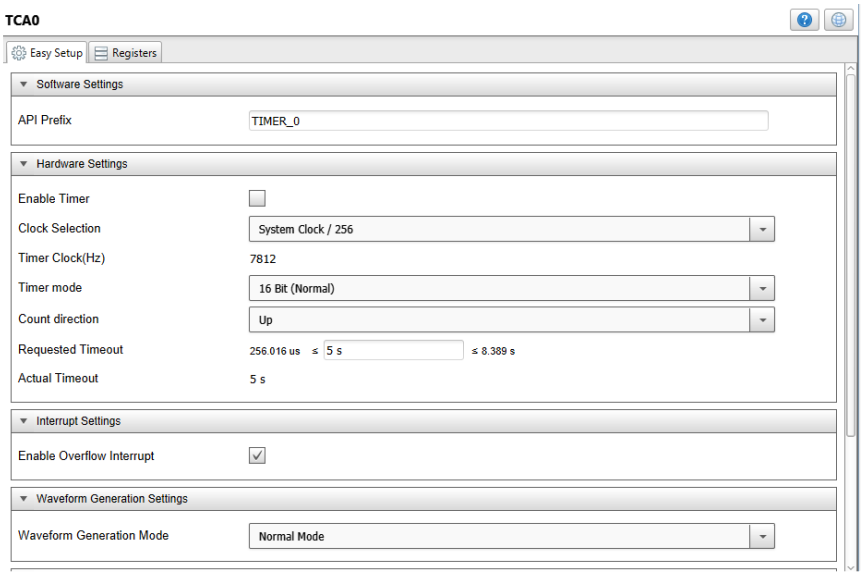
   Count Direction: Up

   Requested Timeout: 5s

   Enable Overflow Interrupt: Yes

   Waveform Generation Mode: Normal Mode

Figure 4-7. TCA0 Configuration



6. **Two Wire Interface (TWI0) Configuration**
The TWI or I^2C interface is used for communicating with the sensor.

API Prefix: I2C0

Interrupt Driven: No

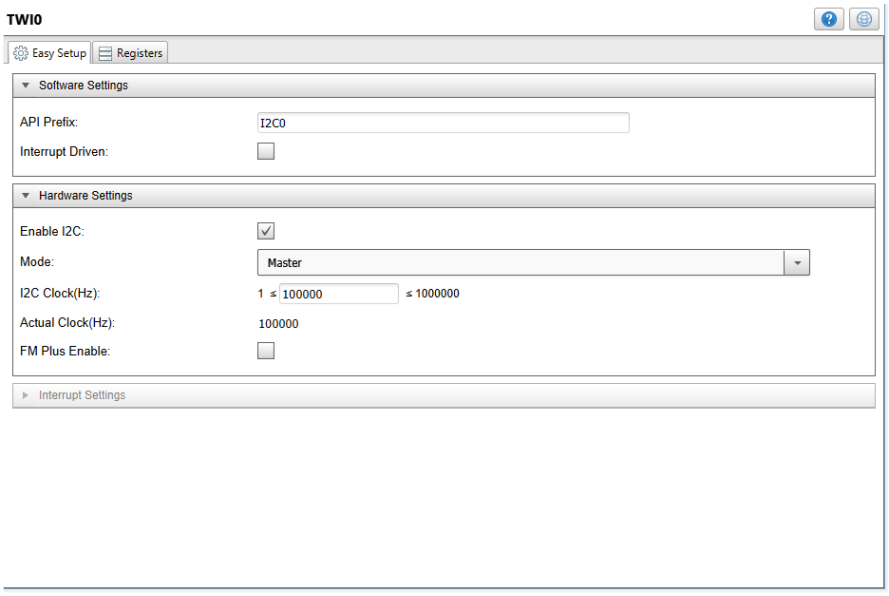
Enable I2C: Yes

Mode: Master

I^2C Clock (Hz): 100000

FM Plus Enable: No

Figure 4-8. TWI0 Configuration



7. **Weather Click Configuration**
The click libraries can be found at the bottom of Device Resources in the MikroE Clicks drop down and then in the Sensors drop-down menu. The configuration can be changed depending on the application, but this

document will present the simplest way to configure it. Setting the Sample Application field to anything else except Custom will ignore the chosen settings in favor of that application's preset.

Sensor Mode: Forced

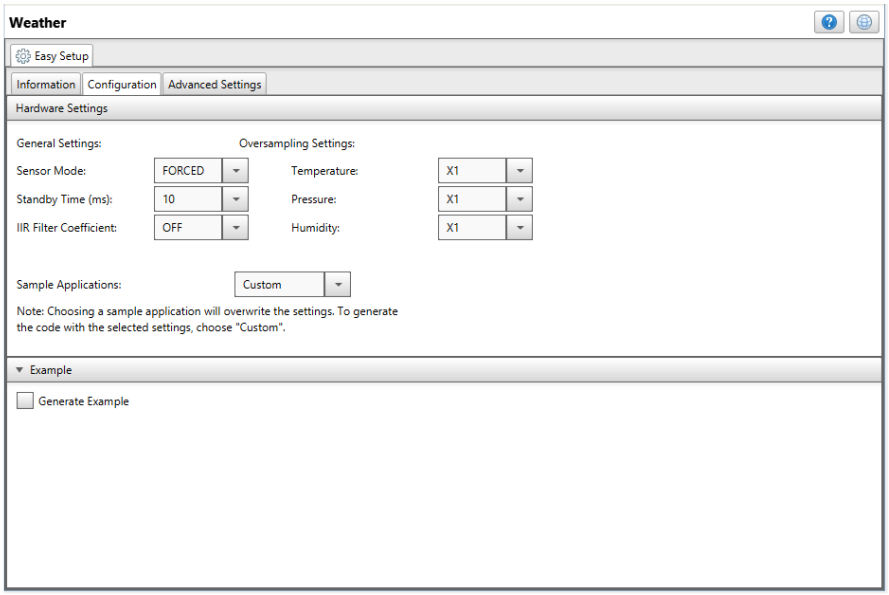
Standby Time (ms): 10

IIR Filter Coefficient: OFF

Oversampling for temperature, pressure and humidity: X1

Sample Applications: Custom

Figure 4-9. Weather Click Configuration

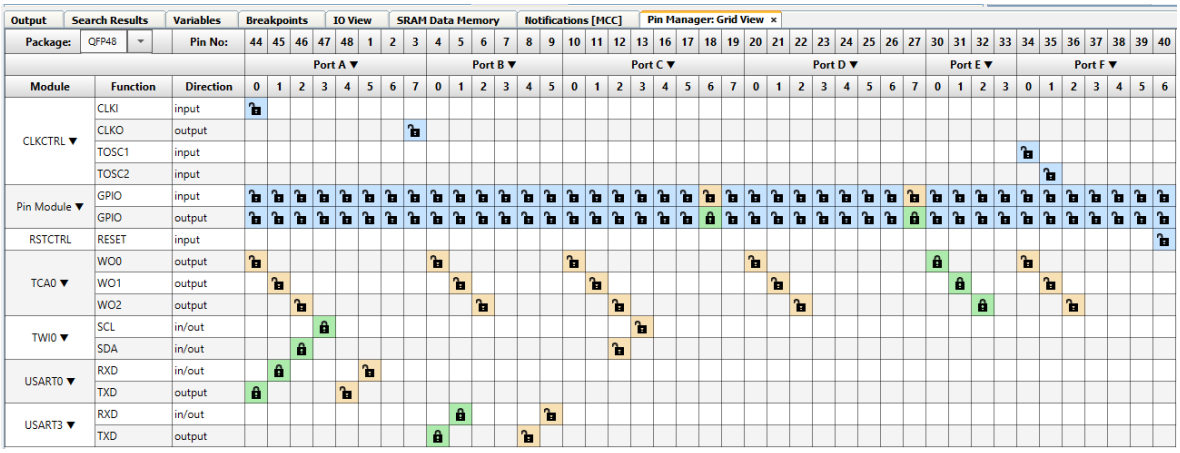


When the Weather Click resource is added, two other modules will be loaded in the Libraries and then in the Foundation Services drop-down menu. They are DELAY and I2CSIMPLE. No changes can be done in DELAY, while the I2CSIMPLE module requires a peripheral instance of TWI as the master. If only TWI0 is enabled, as in the current application, the MCC will choose it by default, but care must be taken when there are several instances of the TWI peripheral.

8. **Pin Module Configuration**
   The pins have to be configured similarly to how the hardware is mounted on the adapter. This document will provide the hardware configuration used here and its pin configuration, but it can be changed to suit various needs.

   The RN4870 module is set into the first mikroBUS slot and the Weather click in the second one. This allows the instances of the USART and the TWI to communicate with them with no issues.

Figure 4-10. Pin Assignments



Firstly, pin PC6 and PD7 are configured as outputs in the pin module interface. The waveform generation of TCA0 is not used in this project, but the module requires a pin assignment. If they do not interfere with the other functions, it is irrelevant which ones are used. They are set to PE0, PE1, and PE2. The TWI0 is set to PA2 and PA3 to communicate with the Weather click. USART0 communicates with the RN4870 module and is set to PA0 and PA1. USART3 communicates with the debugger and is set to PB0 and PB1.

Pin module: GPIO Output: PC6, PD7

TCA0: PE0, PE1, PE2

TWI0: PA2, PA3

USART0: PA0, PA1

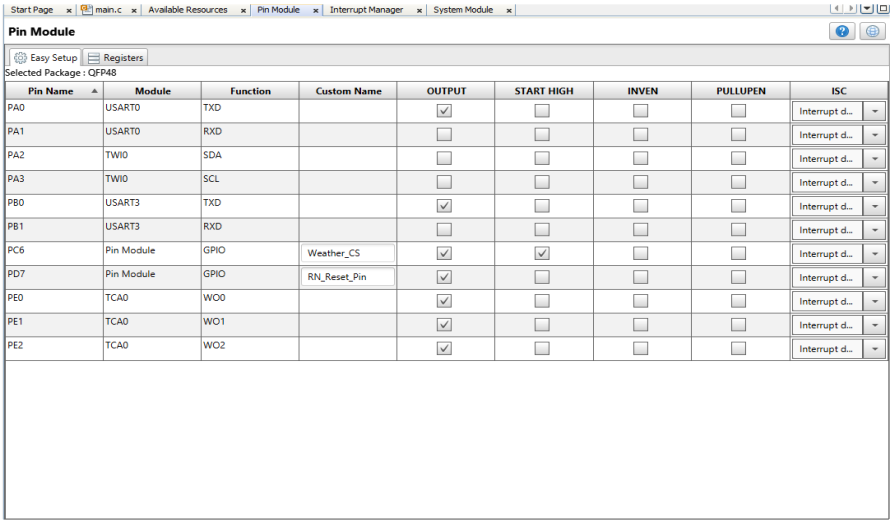
USART3: PB0, PB1

Finally, in the Pin Module in Project Resources, pin PC6 needs to be ‘Weather_CS’ and set as Start High, and PD7 needs to be named ‘RN_Reset_Pin’.

PC6: Name: ‘Weather_CS’, Start High: Yes

PD7: Name: ‘RN_Reset_Pin’

Figure 4-11. Pin Module Configuration



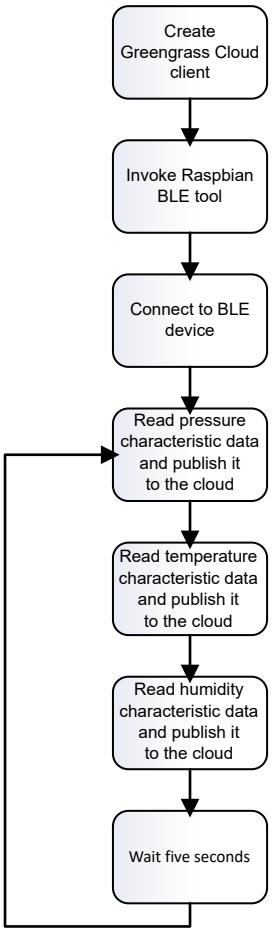
4.4 Lambda Overview

Lambda is one way of how the Greengrass gateway can interact with end devices and the cloud. A Lambda is configured or edited in the cloud and deployed on the Raspberry Pi board. The function comes with a Software Development Kit (SDK), which provides the Application Programming Interface used for cloud connection and topic-specific subscriptions and publications. The messages between the Raspberry Pi and the cloud are sent through the MQTT protocol, but the Greengrass core takes care of the procedures and, therefore, these details are more transparent.

The Lambda can be written in several programming languages and versions of programming languages. The one used in this application is written in Python™ 2.7. Since **Module 3** of the tutorial provided by AWS for Greengrass, is based on Python 2.7, it will be easier to understand and get familiarized with the structure and the procedures of the Lambda.

The flow diagram of the Lambda is described below:

Figure 4-12. Lambda Workflow



1. The Lambda will create a Greengrass client responsible for cloud communication.
2. It will invoke a tool responsible for BLE connection and communication, and it will enter its virtual user interface. Other BLE-oriented tools that are available on Raspbian can be used.
3. It will connect to the BLE end device using its MAC address. The tool does not provide device scanning and the MAC address is different for every device, so the user is responsible for providing it. More information on how to obtain the MAC address can be found in Command State.
4. Once the connection has been established, it will read the bytes available in the temperature, pressure, and humidity characteristics using the `char-read-hnd` command, followed by the handler number. It will then pack the data and publish it to the cloud with the topic ‘BLE/data’. This step is in a loop. The actions are executed once every five seconds. The topic name and the time between executions are the user’s choice and

can be changed. The read of the characteristics is done through a handler, which is assigned when the characteristics are created. More information on how to obtain the handler can be found in Command State.

5. Application Demo

5.1 Main Application Configurations

The MCC will create the peripherals' files and APIs together with the structure of the project and the initialization of the system. The Reset sequence of the BLE module must be added to ensure everything starts correctly. The sensor needs to be read, and the characteristics updated occasionally. All these steps have already been written and provided in the GitHub repository project, but are provided for developers who wish to further customize this application.

5.2 Getting the Board Ready

The pins must be soldered to the Curiosity Nano board and then connected to the Curiosity Nano Adapter board. The RN4870 click must be connected to the mikroBUS slot 1 and the Weather click in the slot 2.

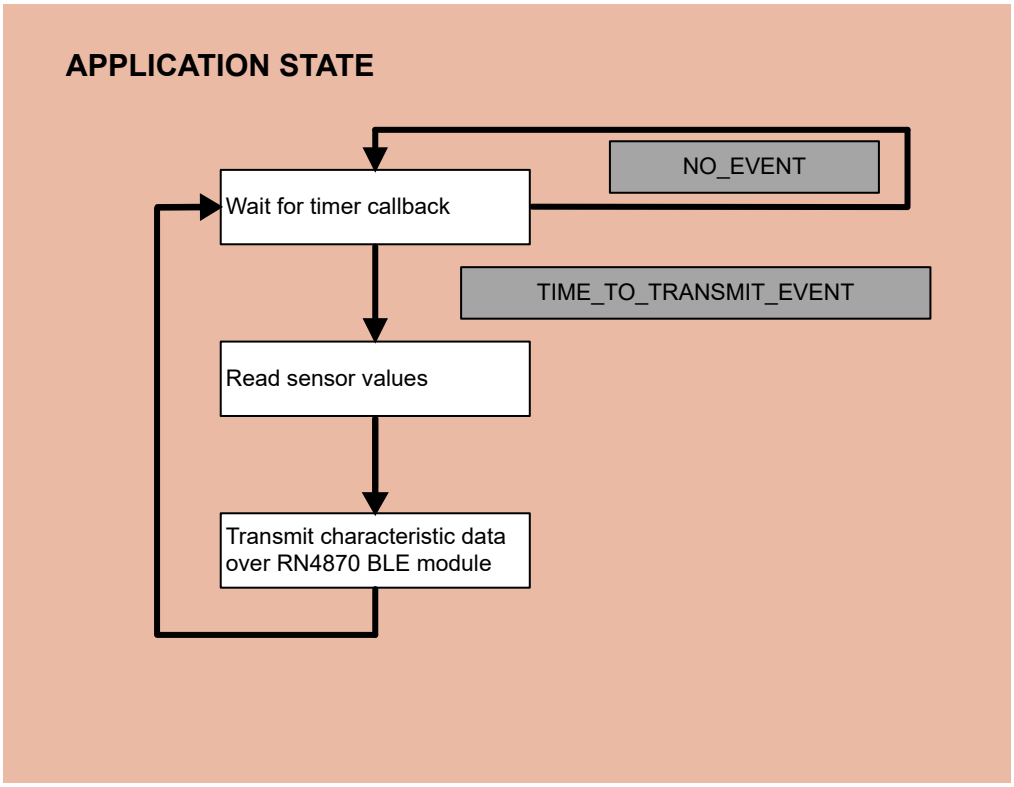
Finally, the Curiosity Nano board has to be connected to a PC through a USB port. The main project, downloaded from the GitHub repository, can be found in the atmega4809-cnano-ble-aws.X folder. The project will be opened with MPLAB X to program the device.

The project is divided into two states: The Application state, in which the application reads the sensors and updates the characteristic values every five seconds, and the Command state, which can be used to interact with the RN4870 board. To switch between the two states, the '/' character must be sent from a serial software terminal.

5.2.1 Application State

The BLE-connected weather sensor operation takes place during the Application state. During this state, the microcontroller reads data from the weather sensor and updates the characteristic values. The characteristic values are then read by the cloud gateway from the RN4870 BLE module.

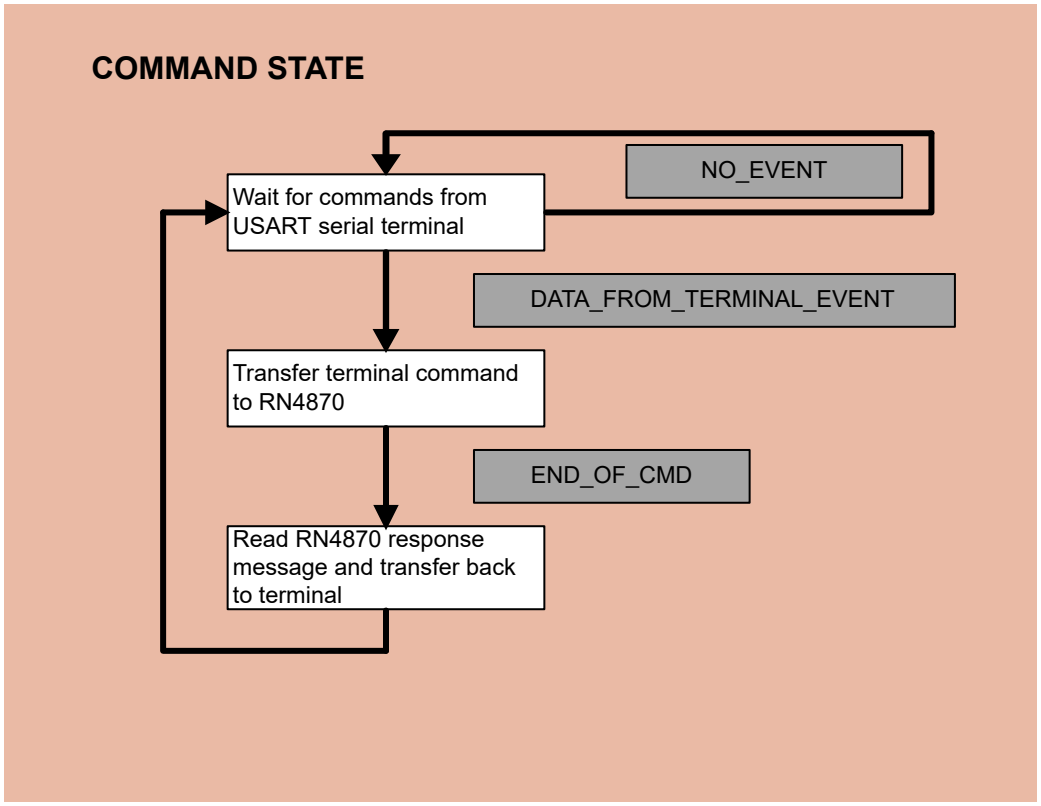
Figure 5-1. Application State



5.2.2 Command State

During the Command state, the microcontroller acts as a USART bridge between the serial terminal and the RN4870 module.

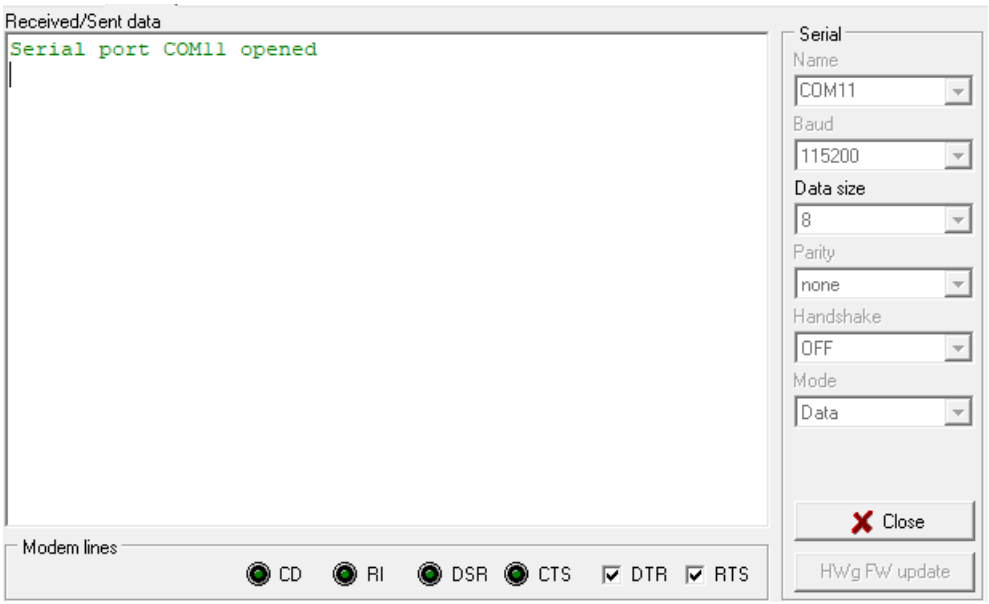
Figure 5-2. Command State



This functionality opens a line of communication to the RN4870 module so that the PC can be used to retrieve the module MAC address. The Command state can also be entered to make any desired configurations, such as changing the name of the module.

The Command state is only for prototyping purposes, and will not be used in the final application.

Figure 5-3. Hercules Terminal



In Command state, use the following commands and:

1. Check for the MAC address of the device. This info will be required later by the Lambda. The MAC address will be on the first line of the reply:
   - Send: '`d`'
   - Reply: '`BTA=D88039F37559`'
   - Reply: '`Name=RN4870-7559`'
   - Reply: '`Connected=no`'
   - Reply: '`Authen=2`'
   - Reply: '`Features=0000`'
   - Reply: '`Services=C0`'
2. Ensure the board uses the latest firmware available, which at this moment is v1.30:
   - Send: '`v`'
   - Reply: '`RN4870 V1.30 3/18/2018 (c)Microchip Technology Inc`'

In Command state, utilize any command described in the BLE RN4870 Module User's Guide.

Note: In this state, the application is no longer reading sensors and updating the characteristics.

5.3 Creating and Loading Lambda

The Getting Started with AWS IoT Greengrass developer guide needs to be followed to install and prepare the Greengrass core on the Raspberry Pi. The developer guide must be followed up to at least **Module 3**.

After the Greengrass procedure is completed, continue configuring the Lambda following these steps:

1. Ensure the Greengrass core is not active yet. To stop the core, open a new terminal window in RPi, go to the location of the core, and perform a Stop command:
   - press **CTRL** + **ALT** + **T** to open a new terminal window
   - enter '`cd /greengrass/ggc/core`' to open the location of the core
   - enter '`sudo ./greengrassd stop`' to stop the core
2. The Lambda and its dependencies are provided in the GitHub repository. Open the 'AWS_Lambda' folder and open the '`lambda_function.py`' file with a text editor.
3. The 'DEVICE' variable represents the MAC address of the RN4870 module and must be changed with the one obtained in Command State. A MAC address has six bytes, and a colon must be added between every byte:

- `DEVICE = 'D8:80:39:F3:75:59'`

4. Archive the entire content of the 'AWS_Lambda' folder as `.zip`, not the folder itself. Then, upload the archive to the cloud and deploy it on Raspberry Pi, as described in the AWS Greengrass tutorial **Module 3**. The topic used when creating the subscription is 'BLE/data'.
5. By default, the RN4870 requires a connection with authentication and, encryption and the BLE module from RPi does not implement it by default. The name of the device responsible for BLE in RPi will be '`hci0`' and using the `hciconfig` command will show the default state of the module:

```
pi@raspberrypi:~ $ sudo hciconfig hci0
hci0:    Type: Primary  Bus: UART
    BD Address: B8:27:EB:09:B1:6A  ACL MTU: 1021:8  SCO MTU: 64:1
    UP RUNNING
    RX bytes:813 acl:0 sco:0 events:53 errors:0
    TX bytes:2524 acl:0 sco:0 commands:53 errors:0
```

The authentication with encryption must be activated using the `encrypt` argument. Using the same command as above for module status checking, new '`AUTH`' and '`ENCRYPT`' functionalities can be observed:

```
pi@raspberrypi:~ $ sudo hciconfig hci0 encrypt
pi@raspberrypi:~ $ sudo hciconfig hci0
hci0:    Type: Primary  Bus: UART
    BD Address: B8:27:EB:09:B1:6A  ACL MTU: 1021:8  SCO MTU: 64:1
    UP RUNNING AUTH ENCRYPT
    RX bytes:827 acl:0 sco:0 events:55 errors:0
    TX bytes:2534 acl:0 sco:0 commands:55 errors:0
```

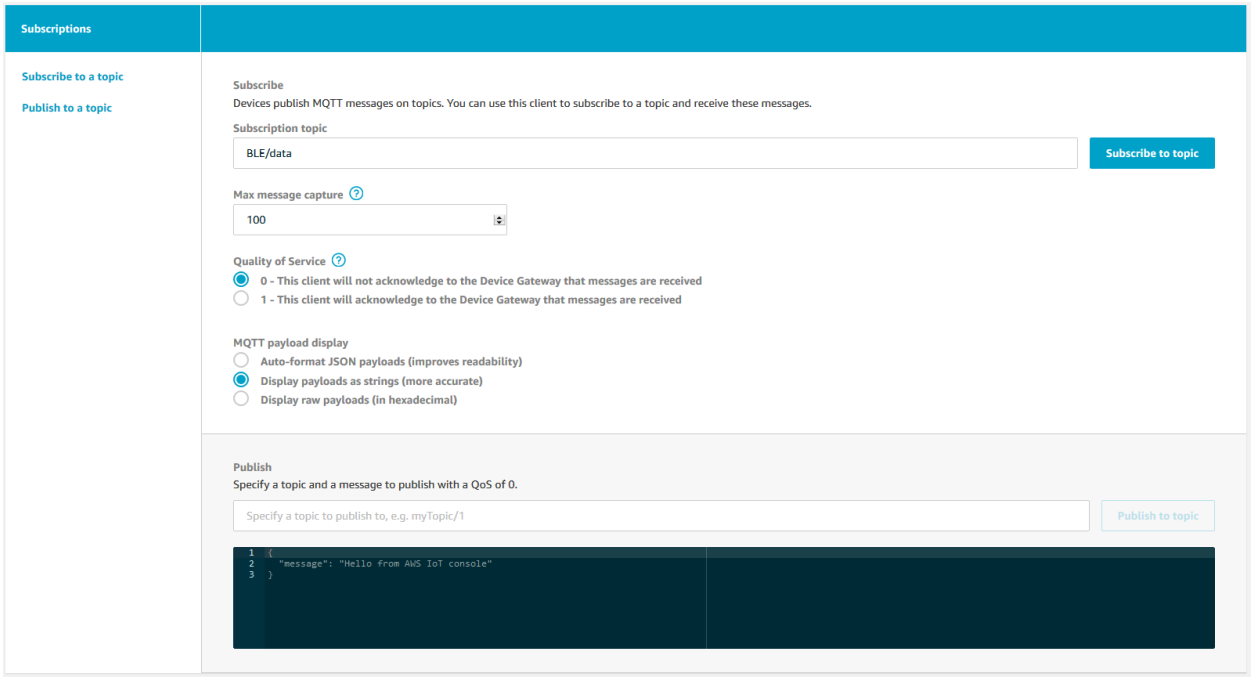
6. All the configurations have been implemented, and the Greengrass core can start.
   - press **CTRL** + **ALT** + **T** to open a new terminal window
   - enter '`cd /greengrass/ggc/core`' to open the location of the core
   - enter '`sudo ./greengrassd start`' to start the core

Note: All of the above configurations must be followed only the first time when loading a new Lambda. After performing a Reset on the RPi board, only steps 5 and 6 need to be followed.

5.4 Visualizing Sensor Data in Cloud

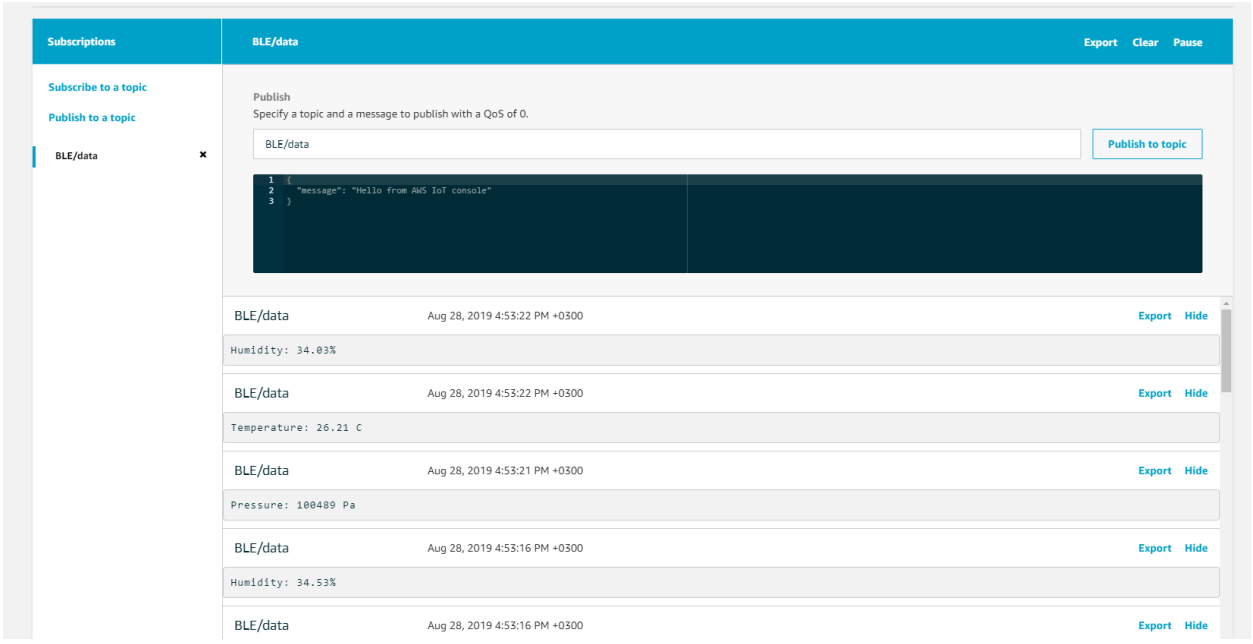
Open the personal AWS account and test the subscriptions as presented in **Module 3** of the AWS Greengrass Core Developer Guide. The topic in which the Lambda will publish data is 'BLE/data', and a subscription to it is needed.

Figure 5-4. Cloud Test



After the subscriptions, the window will print the sensor's value every five seconds, as in the image below.

Figure 5-5. Data Received from Sensor



6. Conclusion

The steps presented in this application note provide a simple solution of sending BLE data through a gateway to the cloud. Microchip's ATmega4809 Curiosity Nano board and Curiosity Nano Adapter provide three mikroBUS sockets which allow users to add new features to their project, such as the RN4870 click board. MCC offers support for this board, configuring the interface and the default settings of the board. The gateway software is provided by AWS together with a developer guide, which provides details regarding the configurations and features.

7. References

1. BLE RN4870 Click board.
2. BLE RN4870 Module User's Guide.
3. Raspberry Pi 3 Model B+.
4. AWS Developer Guide.
5. GitHub Repository.
6. Weather click board.
7. Curiosity Nano ATmega4809.
8. Curiosity Nano Adapter.

8. Revision History

Doc. Rev.	Date	Comments
A	03/2020	Initial document release

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with

your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2020, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-5669-8

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820