

# Smart Cellular Module

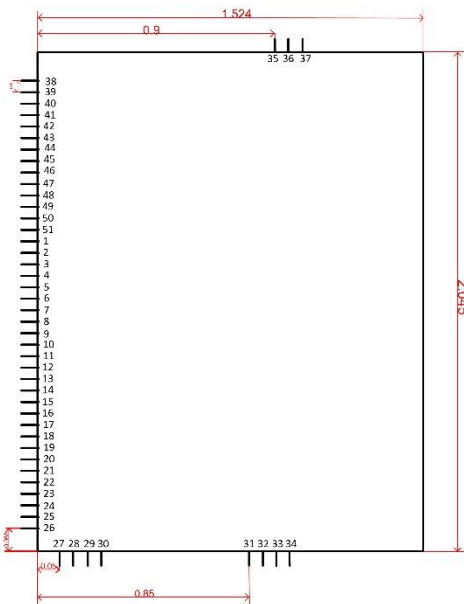
## 1.0 Device Overview:

The SCM-1 is a small form factor, low power cellular cloud-based module, it provides verity of analog and digital functionality. It has been designed with LTE Category-M1 and NB-IoT cellular technologies. LTE Category-M1 and NB-IoT are low-power wide area (LPWA) cellular technologies, both are specifically designed for the Internet of Things (IoT) and machine-to-machine (M2M) communications.



## Features

- Supports LTE Cat. -M1 and NB-IoT technologies.
- Low Power
- Small form factor (1.524"X2.045")
- Better coverage than regular cellular networks.
- Has the priority over other cellular networks



## Applications:

- Measuring and controlling systems
- Industrial applications
- Smart lighting control
- Smart irrigation system
- Gas leak detection

## 2.0 Pin Assignments:

This part shows pin assignments of the SCM-1. It has 37 pins with different functions which are shown in the figure and the table below.

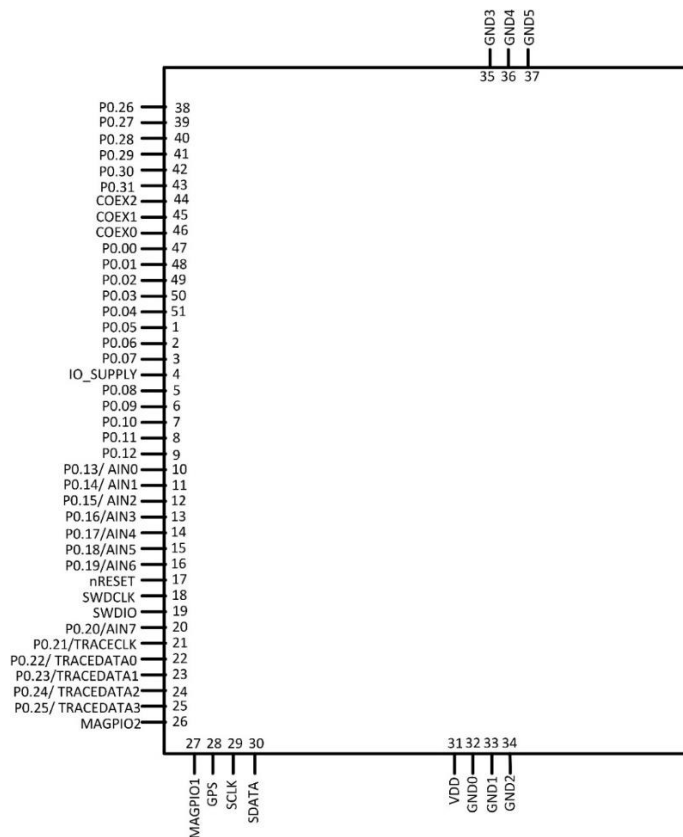


Fig2.1: SCM-1 Diagram

Table1: Pin Assignments

#	Pin name	Function	Description
1	P0.05	Digital I/O (SoC)	General purpose I/O
2	P0.06	Digital I/O (SoC)	General purpose I/O
3	P0.07	Digital I/O (SoC)	General purpose I/O
4	IO_SUPPLY	Power	Reserved for Nordic use
5	P0.08	Digital I/O (SoC)	General purpose I/O
6	P0.09	Digital I/O (SoC)	General purpose I/O
7	P0.10	Digital I/O (SoC)	General purpose I/O
8	P0.11	Digital I/O (SoC)	General purpose I/O
9	P0.12	Digital I/O (SoC)	General purpose I/O
10	P0.13/ AIN0	Digital I/O (SoC)/ Analog input	General purpose I/O/ Analog input
11	P0.14/ AIN1	Digital I/O (SoC)/ Analog input	General purpose I/O/ Analog input

12	P0.15/ AIN2	Digital I/O (SoC)/ Analog input	General purpose I/O/ Analog input
13	P0.16/AIN3	Digital I/O (SoC)/ Analog input	General purpose I/O/ Analog input
14	P0.17/AIN4	Digital I/O (SoC)/ Analog input	General purpose I/O/ Analog input
15	P0.18/AIN5	Digital I/O (SoC)/ Analog input	General purpose I/O/ Analog input
16	P0.19/AIN6	Digital I/O (SoC)/ Analog input	General purpose I/O/ Analog input
17	nRESET	Digital I/O (SoC)	System reset
18	SWDCLK	Digital input	Serial wire debug clock input for debug and programming
19	SWDIO	Digital I/O	Serial wire debug I/O for debug and programming
20	P0.20/AIN7	Digital I/O (SoC)/ Analog input	General purpose I/O/ Analog input
21	P0.21/TRACECLK	Digital I/O (SoC)/ Trace clock	General purpose I/O/ Trace buffer clock (optional).
22	P0.22/ TRACEDATA0	Digital I/O (SoC)/ Trace data	General purpose I/O/Trace buffer TRACEDATA [0] (optional).
23	P0.23/TRACEDATA1	Digital I/O (SoC)/ Trace data	General purpose I/O /Trace buffer TRACEDATA [1] (optional)
24	P0.24/ TRACEDATA2	Digital I/O (SoC)/ Trace data	General purpose I/O/ Trace buffer TRACEDATA [2] (optional).
25	P0.25/ TRACEDATA3	Digital I/O (SoC)/ Trace data	General purpose I/O/ Trace buffer TRACEDATA [3] (optional).
26	MAGPIO2	Digital I/O (SoC)	Reserved for Nordic use
27	MAGPIO1	Digital I/O (SoC)	Reserved for Nordic use
28	GPS	RF	GPS receiver input
29	SCLK	Digital I/O (SoC)	Reserved for Nordic use
30	SDATA	Digital I/O (SoC)	Reserved for Nordic use
31	VDD	Power	Supply voltage
32	GND0	Power	Ground
33	GND1	Power	Ground
34	GND2	Power	Ground
35	GND3	Power	Ground
36	GND4	Power	Ground
37	GND5	Power	Ground
38	P0.26	Digital I/O (SoC)	General purpose I/O
39	P0.27	Digital I/O (SoC)	General purpose I/O
40	P0.28	Digital I/O (SoC)	General purpose I/O
41	P0.29	Digital I/O (SoC)	General purpose I/O
42	P0.30	Digital I/O (SoC)	General purpose I/O

43	P0.31	Digital I/O (SoC)	General purpose I/O
44	COEX2	Digital I/O (SoC)	Coexistence interface
45	COEX1	Digital I/O (SoC)	Coexistence interface
46	COEX0	Digital I/O (SoC)	Coexistence interface
47	P0.00	Digital I/O (SoC)	General purpose I/O
48	P0.01	Digital I/O (SoC)	General purpose I/O
49	P0.02	Digital I/O (SoC)	General purpose I/O
50	P0.03	Digital I/O (SoC)	General purpose I/O
51	P0.04	Digital I/O (SoC)	General purpose I/O

### 3.0 Dimensions and Footprint:

Figure 2 shows the dimensions and footprint of the SCM-1.

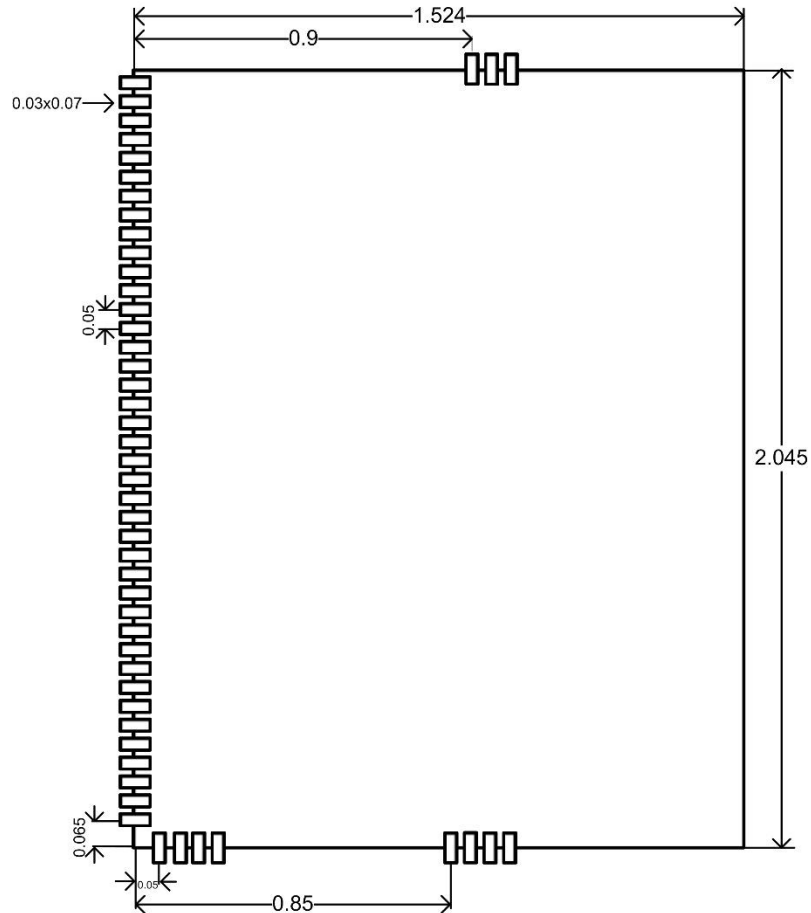


Fig3.1: SCM-1 Dimensions and footprint

---

---

## 4.0 Technical and Environmental Specifications:

Tables 4.1 and 4.2 show the environmental and technical specifications for the SCM-1.

### 4.1 Recommended Operation Conditions

Table2: Recommended Operation Conditions

Symbol	Parameter	Notes	Min.	Nom.	Max.	Units
VDD	Battery input voltage	- Including voltage drop, ripple and spikes. - RF 3GPP compliancy requires 3.3 V.	3.0	3.8	5.5	V
GPIOH	GPIO high level voltage				VDD_GPIO	V
MAGPIOH	MAGPIO high level voltage			1.8	1.8	V
TA	Operating temperature		-40	25	85	°C

### 4.2 Absolute Maximum Ratings

Table3: Absolute Ratings

	Min.	Max.	Unit
VDD	-0.3	5.5	V
Storage temperature	-40	125	°C
Moisture Sensitivity Level (MSL)		2	

---

## 4.3 Block Diagram

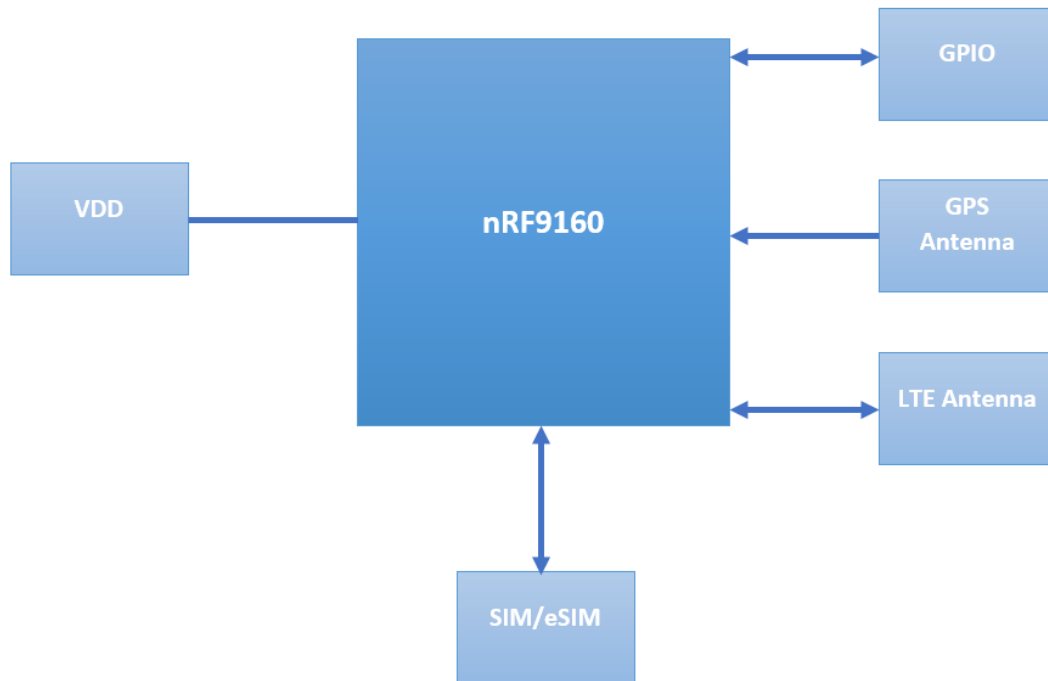


Fig4.1: SCM-1 Block Diagram

## 5.0 Software configuration:

This section is a detailed introduction to the software configuration including development tools and coding.

### 5.1 Getting Started

#### *Requirements:*

- SCM-1
- nRF9160 DK
- Windows, macOS or Linux PC
- nanoSIM card supporting LTE-Cat.M1
- microUSB cable

Header Pin (SCM-1)	Function	DK Connection
1	3.3V	VTG
2	SWDIO	SWDIO
3	SWCLK	SWDCLK
4	P1.00	SWO
5	RESET	RESET
6	GND	GNDDET

## 5.2 Development Tools

To get started with nRF DK and the SCM-1 install the **nRF connect for desktop**.

1- Go to Nordic Semiconductor Website. Click [here](#)

2- Click on **Software and tools**.

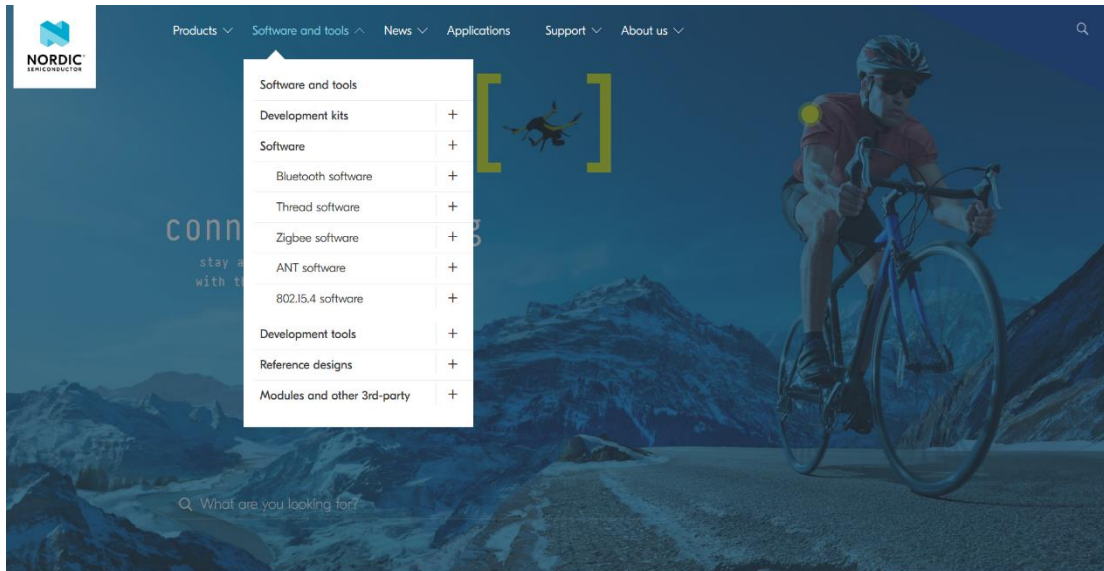


Fig5.1: Software and Tools

3- Click on Development Tools and choose **nRF Connect for Desktop**.

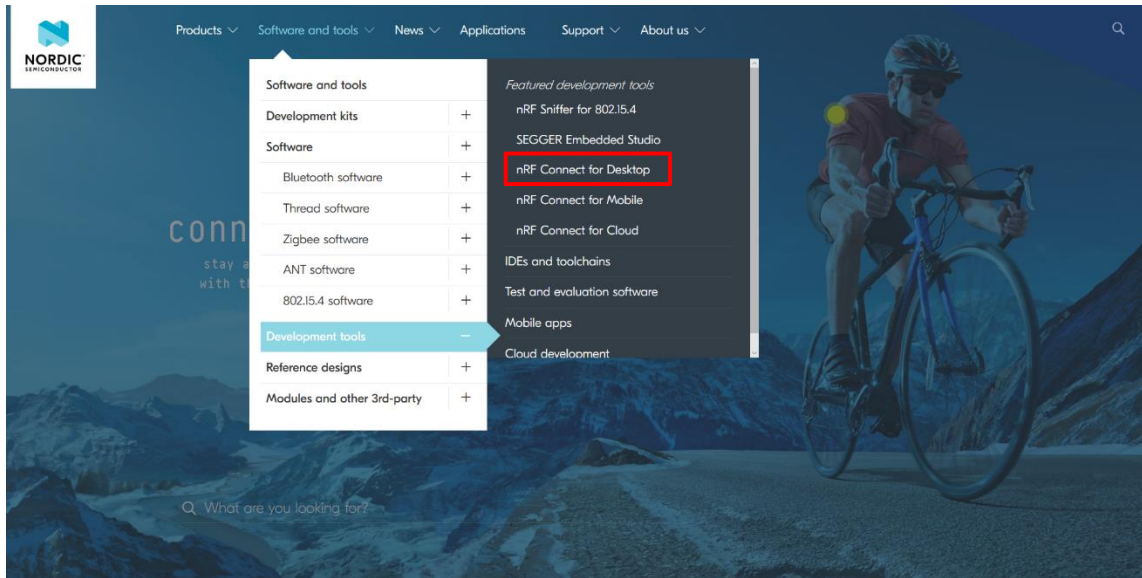


Fig5.2: nRF Connect for Desktop

4- Choose your platform, **Download** and **Run** the latest version of nRF Connect for Desktop.

A screenshot of the nRF Connect for Desktop application interface. The interface is divided into two main sections. The left section is titled 'Highlights' and contains two sub-sections: 'Bluetooth Low Energy' and 'Cellular IoT with LTE-M and NB-IoT'. The right section is titled 'Download latest version, nRF Connect for Desktop' and features a large blue circular icon with a white downward arrow. Below the icon is a red-bordered button labeled 'Choose platform' with a dropdown arrow. The background of the application shows a log window with various system messages and a list of connected devices on the right side.

**Log**

15:40:47.281121	Bluetooth	Scan started
15:40:47.281121	Bluetooth	Connecting to device
15:45:46.6270	Bluetooth	Connect cancelled
15:45:48.8686	Bluetooth	Scan started
15:45:53.7860	Bluetooth	Connecting to device
15:46:01.7240	Bluetooth	Connected to device D0:D3:A7:4C:D0:D9
15:46:01.9480	Bluetooth	Attribute value read, handle: 0x03, value (0x) 4E:4F:72:64:69:63:5F:48:52:4D

**Highlights**

**Bluetooth Low Energy**

- Easy-to-use cross platform application for Bluetooth LE connectivity testing
- Supports auto detection of connected Nordic kit and automatic FW uploading
- Supports LE Security introduced in Bluetooth 4.2
- Up to 8 concurrent Bluetooth LE connections
- Max 8 concurrent central connections
- Max 1 peripheral connection
- Scans for Bluetooth LE devices
- Parses advertisement data
- Shows RSSI value
- Connects to any connectable Bluetooth LE device
- Discovers and parses services and characteristics

**Cellular IoT with LTE-M and NB-IoT**

- Easy-to-use cross platform application for connectivity testing using LTE-M and NB-IoT
- Getting started assistant for installing the development tools and SDK for nRF9160
- Supports auto detection of connected Nordic kit
- LTE Link Monitor for use of AT commands with the nRF91 Series
- Shows connection status with cellular network
- Shows RSSI values

**Download latest version, nRF Connect for Desktop**

Choose platform ▾

Fig 5.3: Platform Choosing



5- After you download it on your desktop go to **Settings** and **check for updates**.

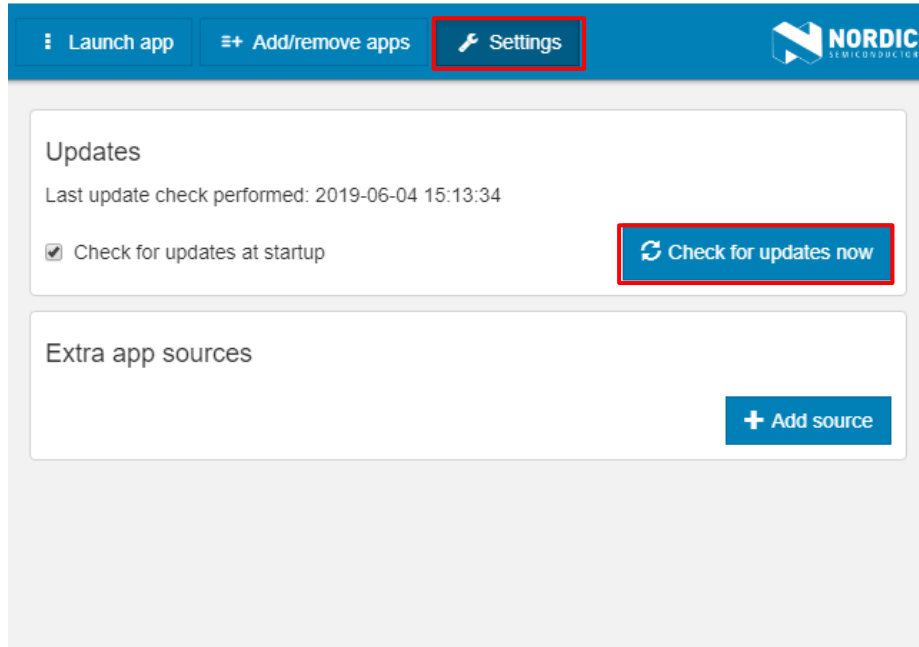


Fig5.4: Settings

6- Go to **Add/Remove Apps**, install **Getting Started Assistant** and **Programmer**.

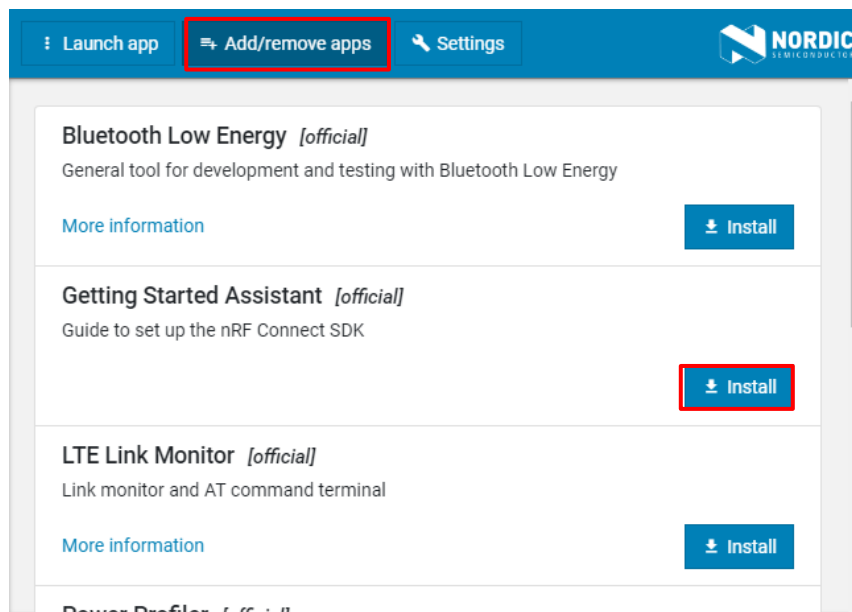


Fig 5.5: Getting Started Assistant

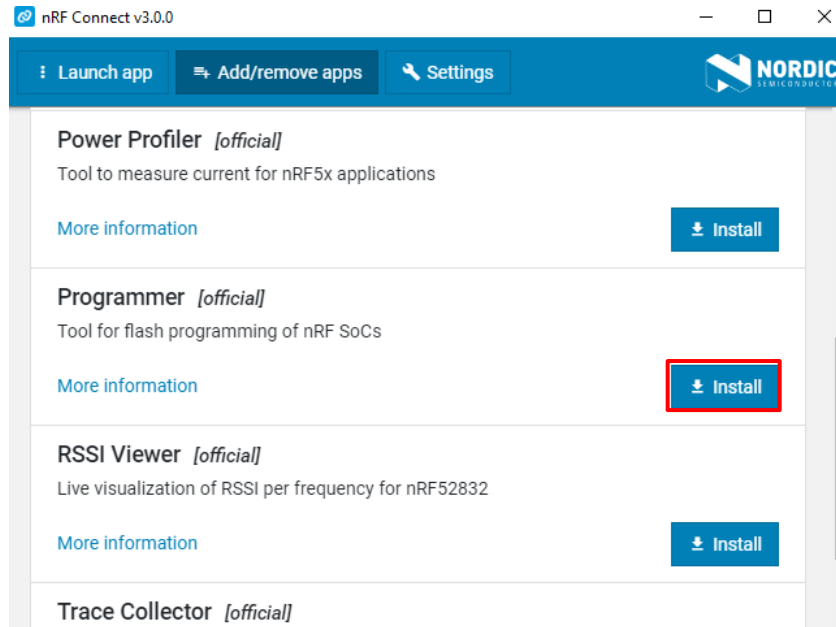


Fig 5.6: Programmer Install

7- Go to **launch** app, and launch **Getting Started Assistant**.

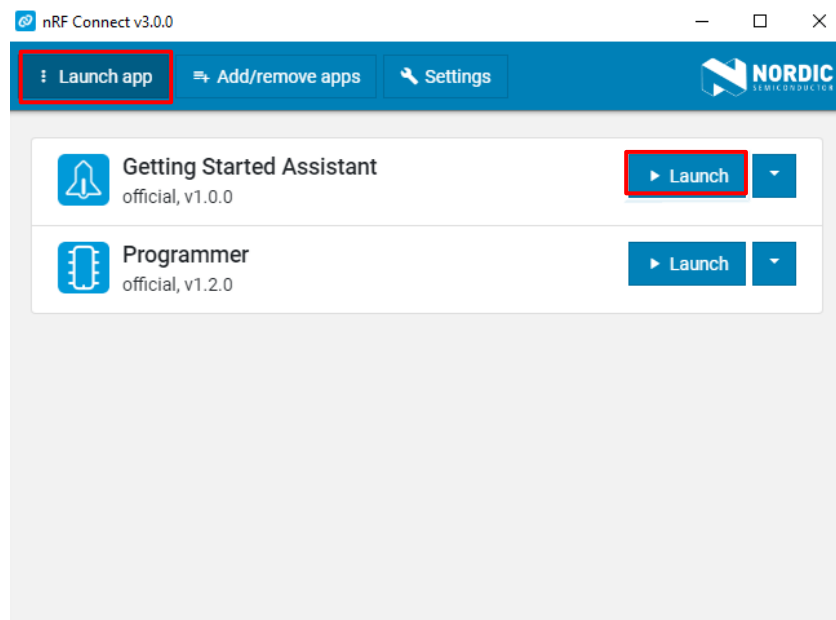


Fig 5.7: Launch Apps

8- Follow the steps from 1 to 4 to finish installation.

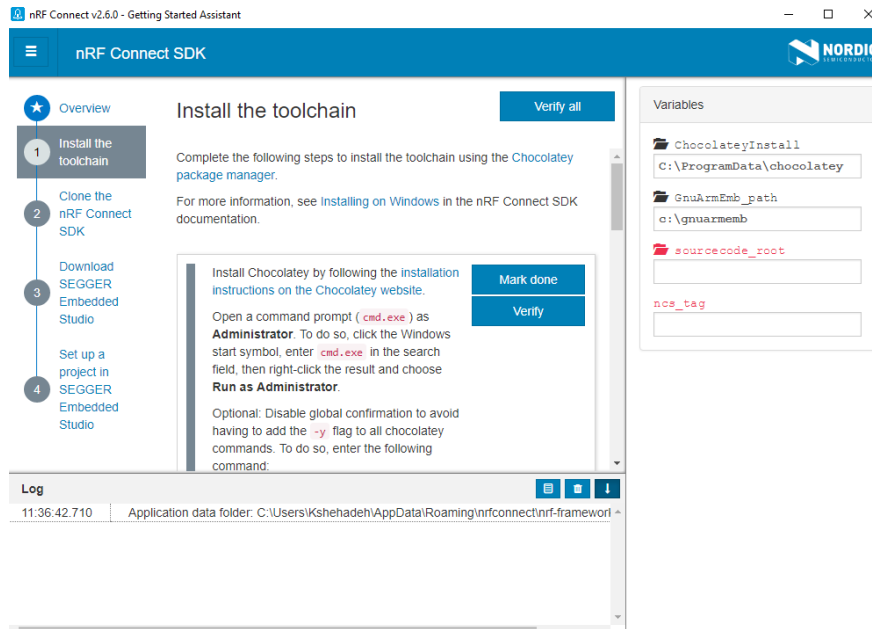


Fig 5.8: Install the toolchain

9- The first step starts with opening your **Command Prompt**, make a right click, and choose **Run as administrator**.

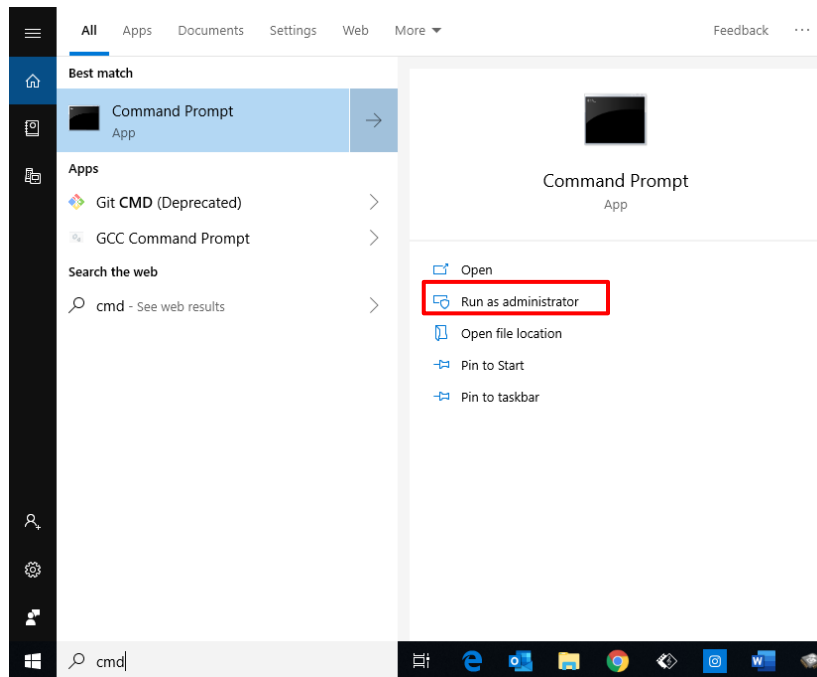


Fig 5.9: Command Prompt

10- Enter commands as described.

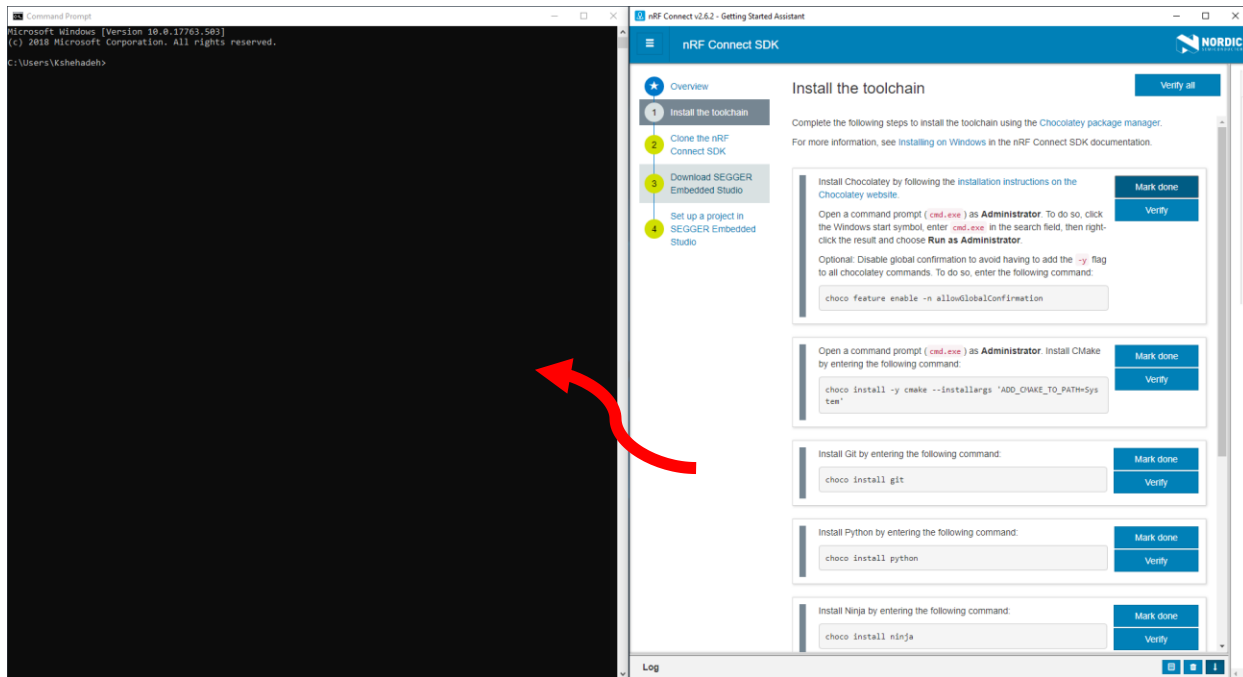


Fig 5.10: Installing Steps

Note: At the end of **Step 1** create a new folder (**c:\gnuarmemb**) and install GNU ARM Embedded toolchain into it as mentioned.

11- Verify all commands in step one and move to the next step.

The screenshot shows the 'nRF Connect SDK' documentation page. The title is 'Clone the nRF Connect SDK'. A 'Verify all' button is highlighted with a red box in the top right. The left sidebar shows a progress indicator with four steps: 1. Install the toolchain, 2. Clone the nRF Connect SDK (highlighted with a red arrow), 3. Download SEGGER Embedded Studio, and 4. Set up a project in SEGGER Embedded Studio. The main content area contains three sections, each with a 'Mark done' and 'Verify' button:

- Section 1:** To manage the combination of repositories and versions, the nRF Connect SDK uses west. Install it by entering the following command:  

```
pip3 install west
```
- Section 2:** Initialize west and clone the nRF Connect SDK manifest repository  
`nrf:`  
*Note: If you already cloned the nRF Connect SDK repositories and want to continue using these clones, skip this step and see [Updating your existing clones to use west](#) instead.*  

```
cd <sourcecode_root>
mkdir ncs
cd ncs
west init -m https://github.com/NordicPlayground/fw-nrfconnect-nrf
west update
```
- Section 3:** Decide if you want to work with a tagged release of the nRF Connect SDK or with the latest state of development.  
*Note: The latest state is not necessarily tested. For a higher degree of*

Fig 5.11: Verify Commands

12- In **step 3** be sure to download the Segger Embedded Studio using the link attached in step 3.

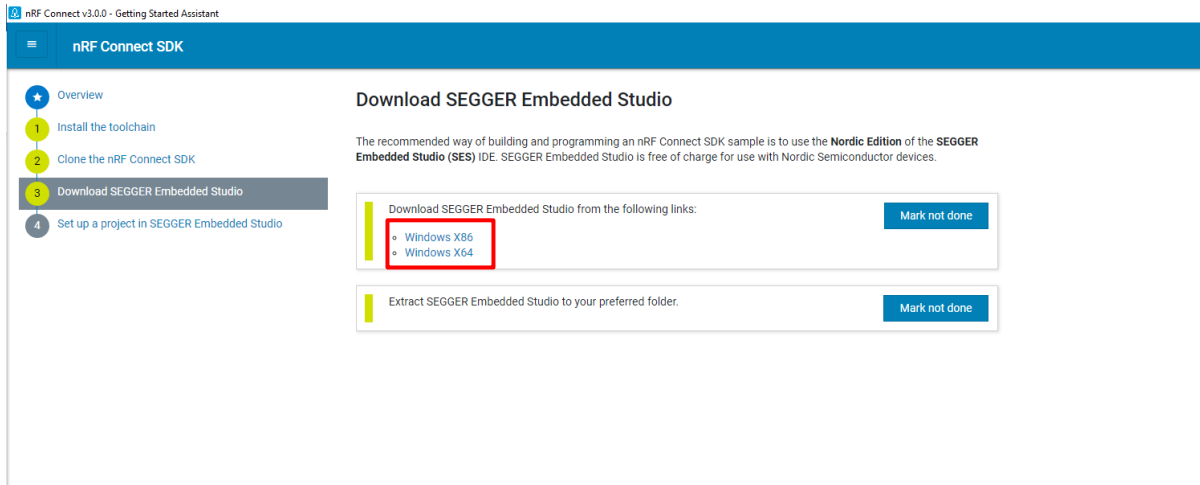
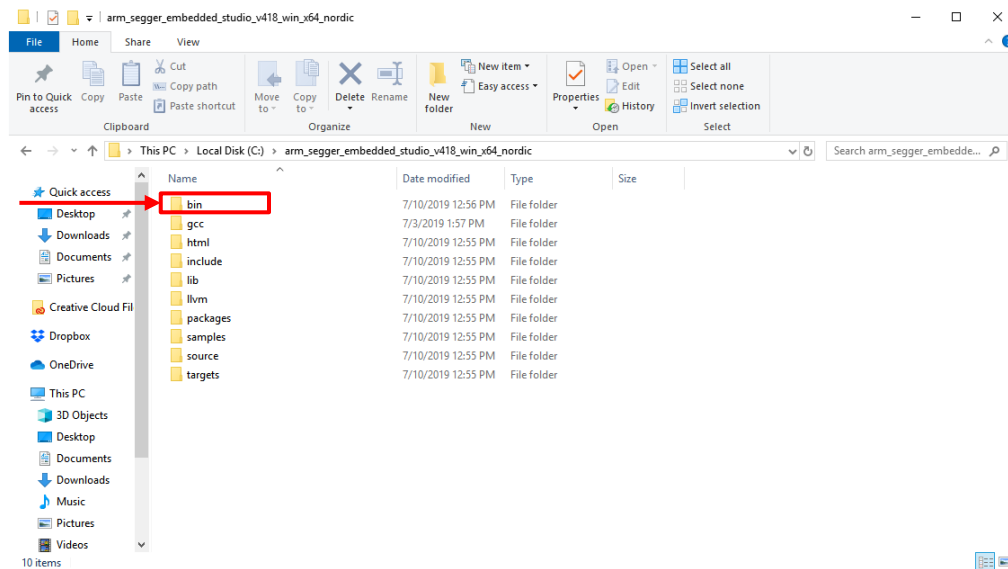


Fig 5.12: Segger Embedded Studio in Step 3

13- After downloading and extracting Segger Embedded Studio files (Step 3) go to **bin**, make a shortcut of **Segger Embedded Studio** and move it to your desktop.



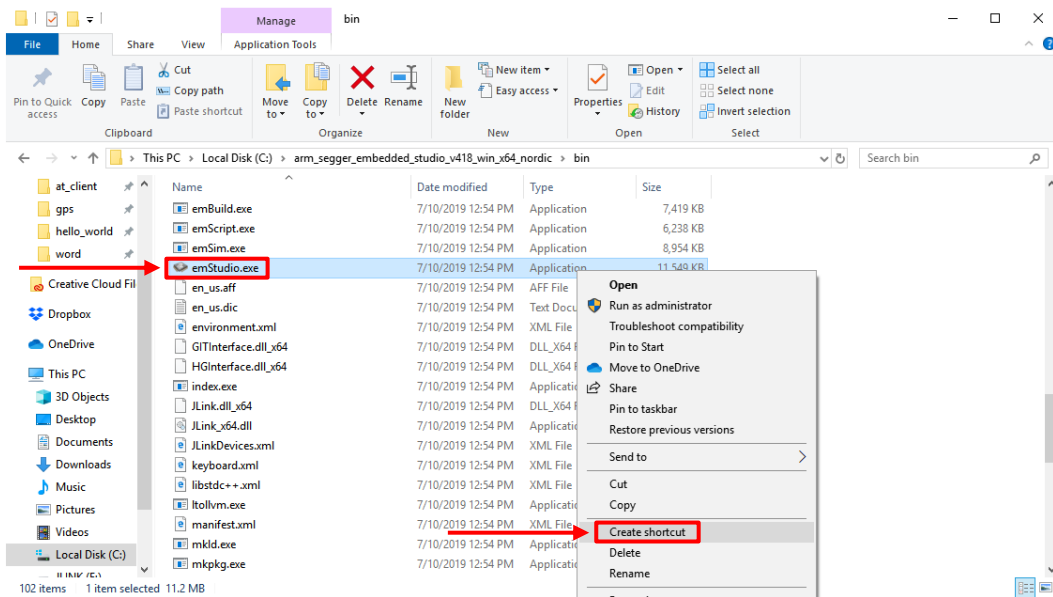


Fig 5.13: Segger Studio shortcut

14- Open the Segger Studio and complete the next two steps as described.

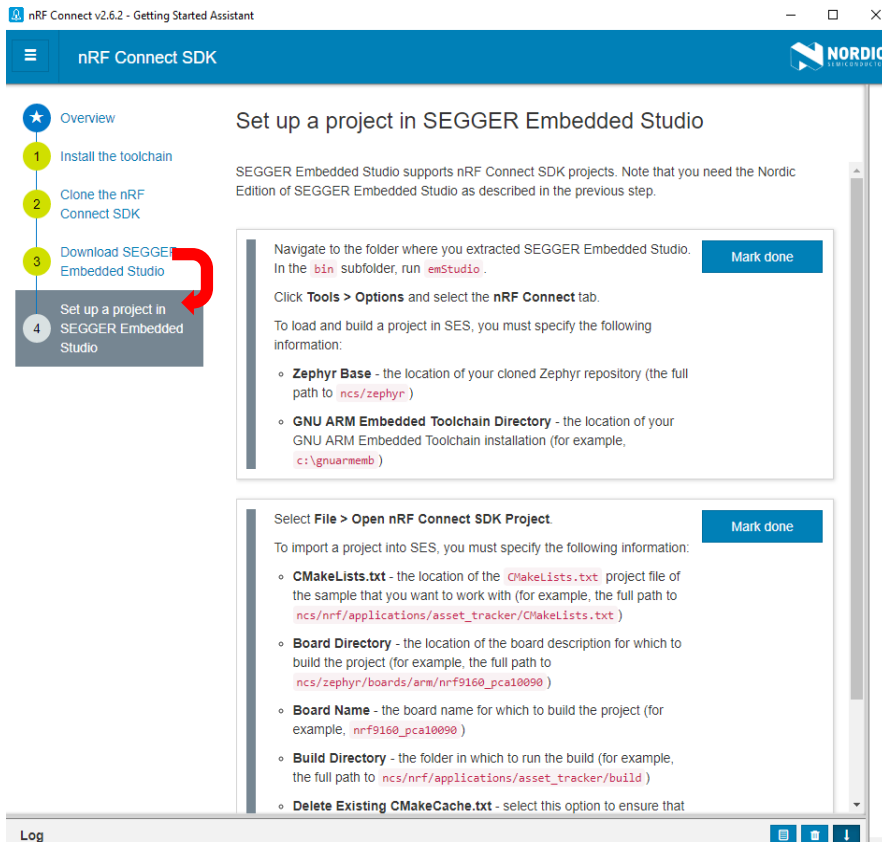


Fig 5.14: Installation Steps

15- Open SEGGER Embedded Studio which was installed in step 3.

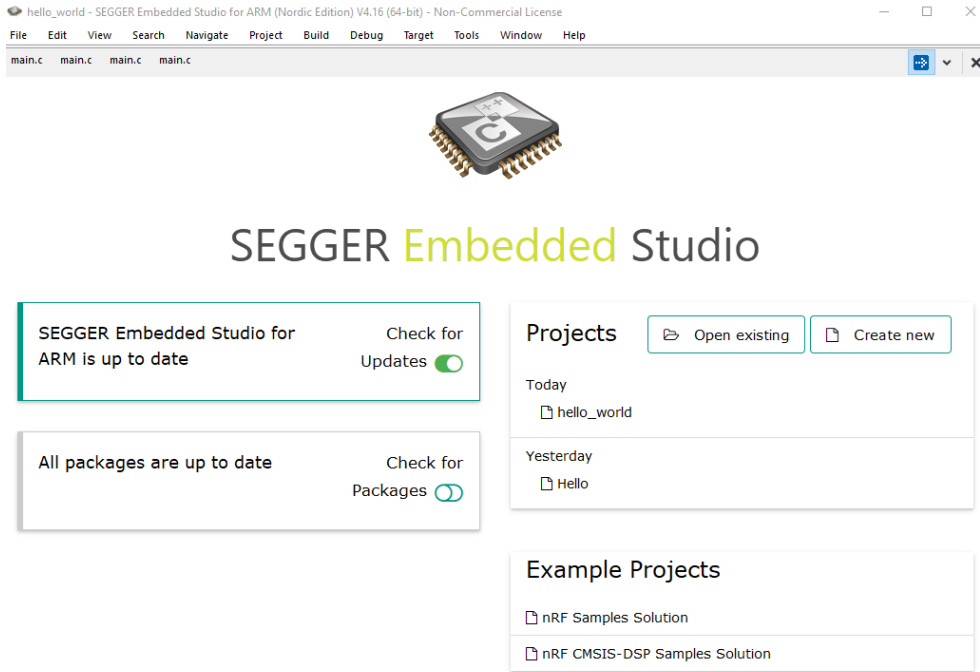


Fig 5.15:Segger Embedded Studio



16- As mentioned in Step 4 click **Tools > Options** and select the **nRF Connect** tab.

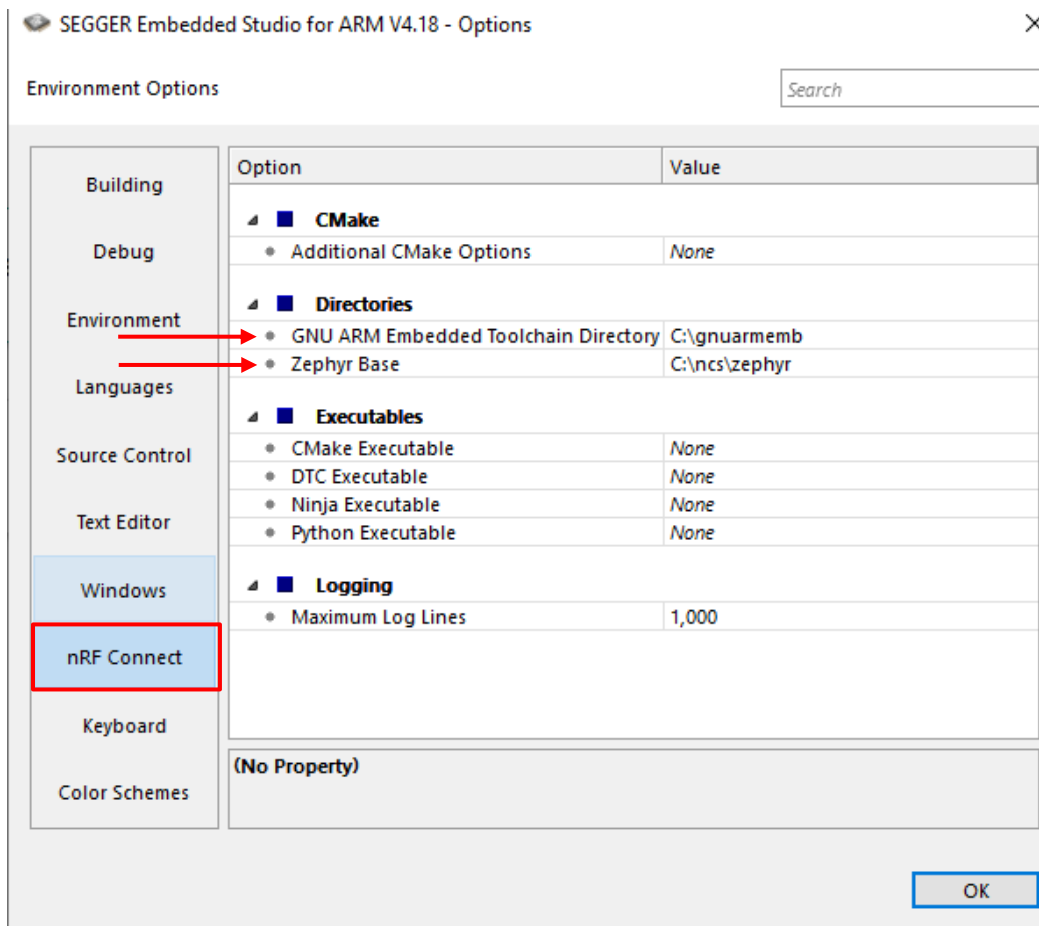
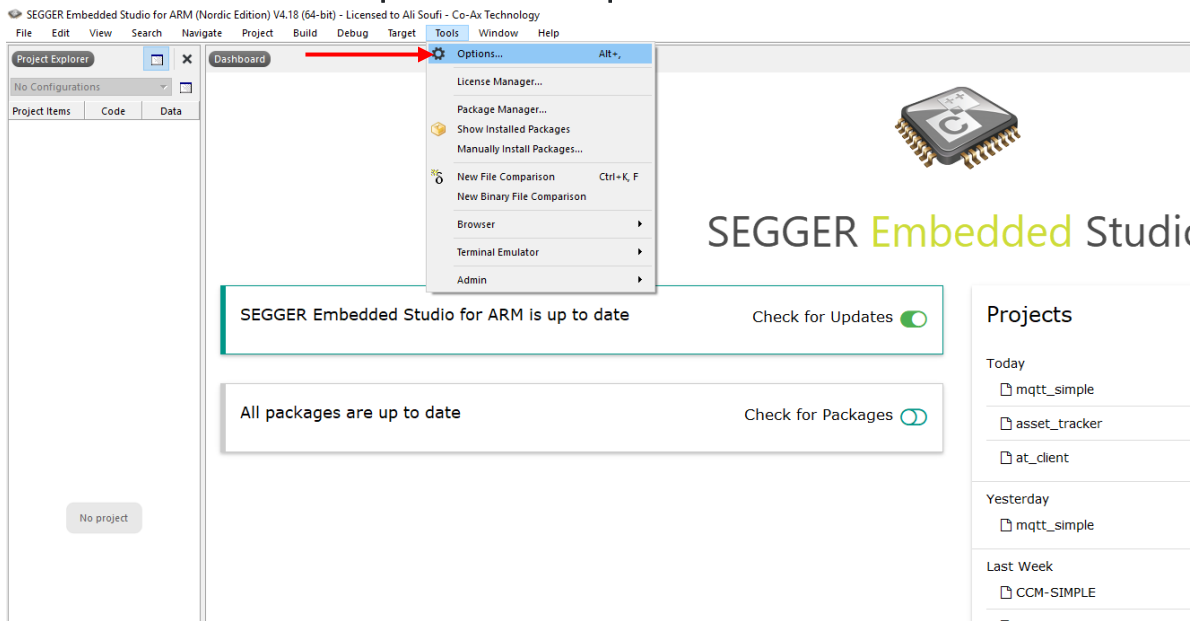


Fig 5.16: Project Setup

17- Go to **File > Open nRF connect SDK project.**

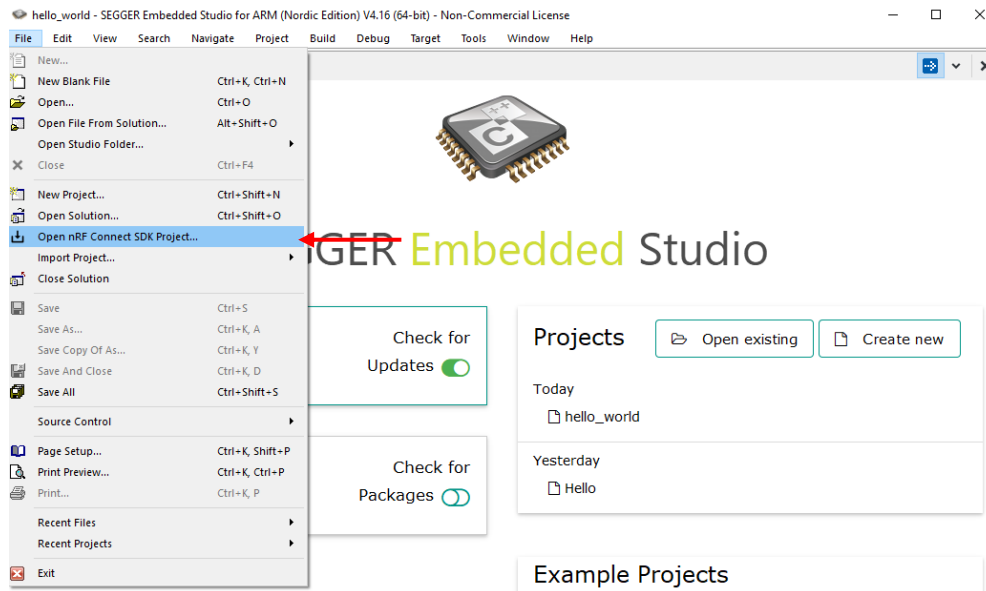


Fig 5.17: Open nRF Project

18- Set CMakeLists.txt, Board Directory, Board Name and Build Directory. **Notice that you will be able to run this program only before erasing your nRF9160 or reprogram it.**

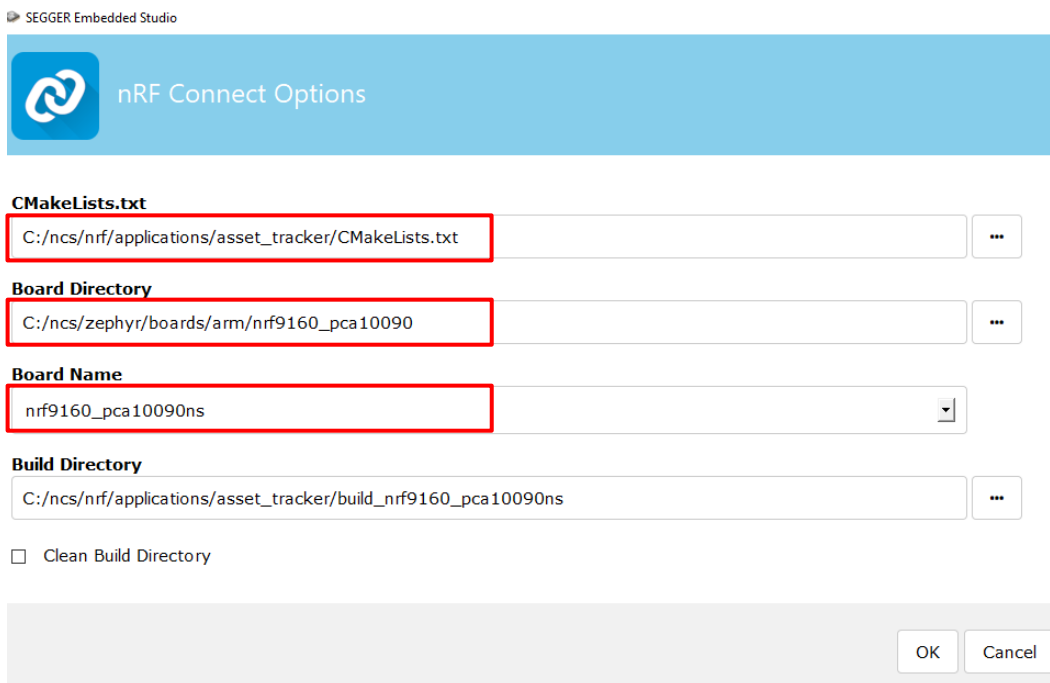


Fig 5.18: Connect Options

## 19- Plug the nRF9160 DK to your computer and go to **Build > Build Solution**

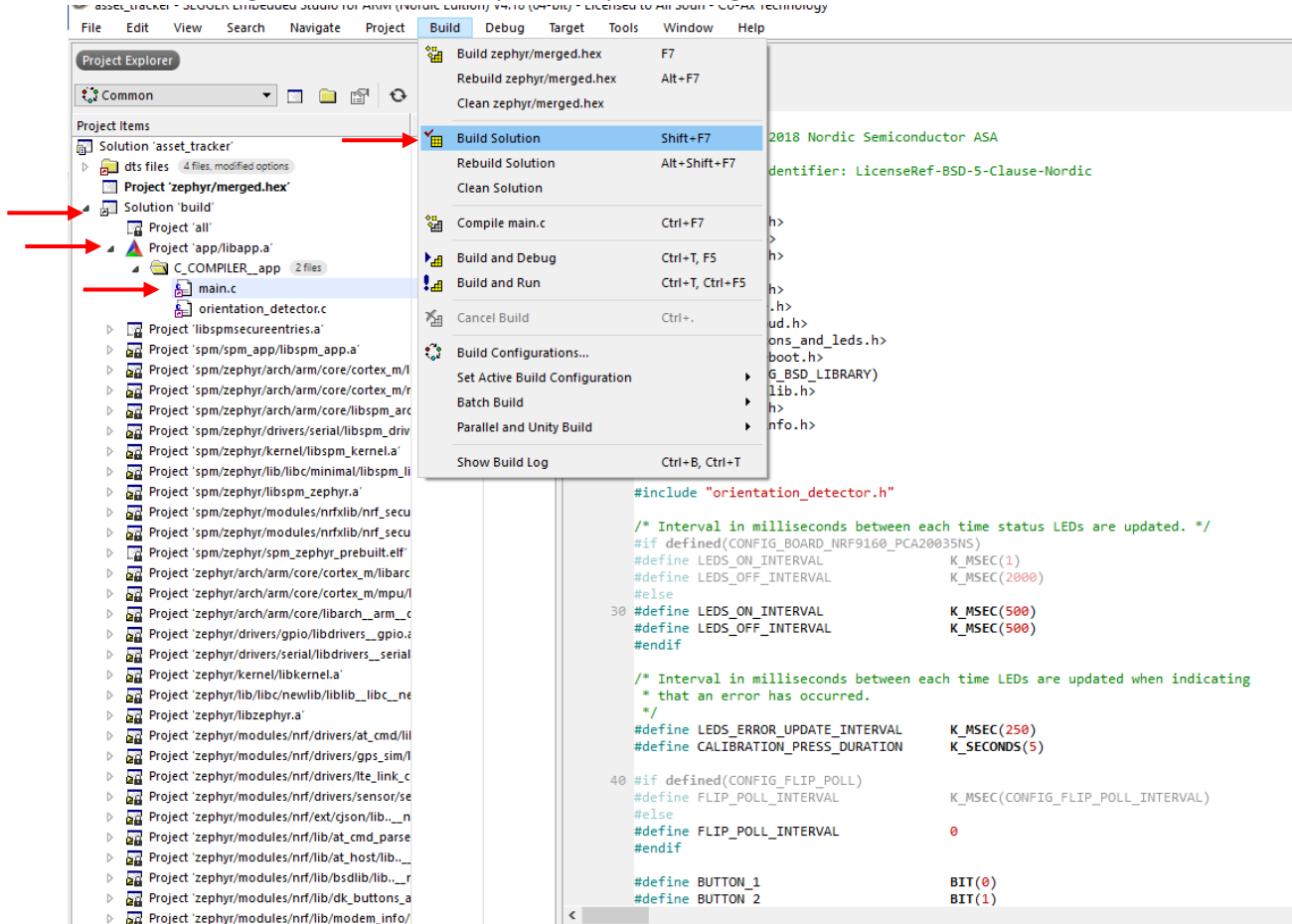


Fig 5.19: Build Solution

The sample will be compiled when you click Build Solution, but the nRF9160 board will not be programmed yet.

20- Before start programming you need to update **Modem Firmware**. Go back to [Nordic semiconductor](https://www.nordicsemi.com) website > Software and tools > Development kits > nRF9160 DK.

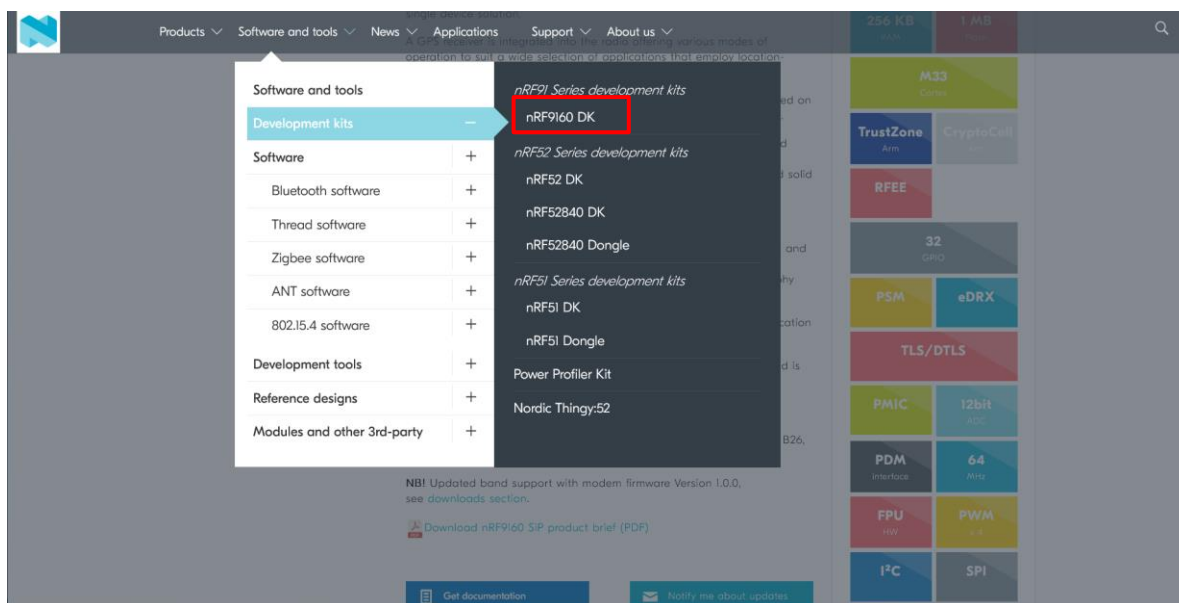


Fig 5.20: Modem Update

## 21- Download the latest version of the Modem Firmware.

Overview **Downloads** Get started

### Feature Brief

for nRF9160 LTE-M/NB-IoT/GPS Modem firmware

→

**LTE Modem images** are precompiled binaries, signed and encrypted by Nordic Semiconductor.

Features:

- Precompiled binary, signed and encrypted
- Update modem FW via PC tool [nRF Connect for Desktop](#)
- Secure boot with image authentication
- LTE Rel-13 Cat-M1 (LTE-M/eMTC)
- LTE Rel-13 Cat-NB1 (NB-IoT)
- Type B half duplex (HD), frequency division duplex (FDD)
- Cat-M1 operation is enabled on E-UTRA Bands 1, 2, 3, 4, 5, 8, 12, 13, 14, 17, 18, 19, 20, 25, 26, 28 and 66.
- Cat-NB1 operation is enabled on E-UTRA Bands 1, 2, 3, 4, 5, 8, 12, 13, 17, 19, 20, 25, 26, 28 and 66.
- Power saving
  - Power Save Mode (PSM)
  - Idle-DRX and Connected Mode-DRX, DRX/extended-DRX in both
  - Independent clock and sleep state control
- Interface to Application CPU
  - AT-command Interface for control
  - Socket Interface for Data
  - Modem production test support
  - Antenna Tuner per band Configurability with limited MPI-RFFE support
- Integrated TLS(1.2)/DTLS(1.2) and TCP/UDP/IPV4/IPV6 Dual Stack
  - Storage of TLS and Cloud credentials
- SMS PDU Mode
- Differential FOTA support enables small upgrade images
- Support for SIM ATK and remote provisioning via Bearer Independent Protocol
  - eSIM support
- GPS L1 C/A receiver during LTE PSM mode
  - Single shot, fixed interval and continuous tracking modes

**Note:** it is **not** possible to downgrade to previous modem firmware releases after applying this upgrade.

This release is aligned with v1.0 of nRF91 AT command [reference guide](#).

---

**nRF9160 LTE-M/NB-IoT/GPS Modem Firmware**

and Modem DFU Tool

→

Selected version  
1.0.0  
[mfw\\_nrf9160\\_1.0.0.zip](#)

**Download file**

Older version ▾

Fig 5.21: Modem Firmware Download

22- Go back to the nRF Connect, click launch other App to launch **Programmer**.

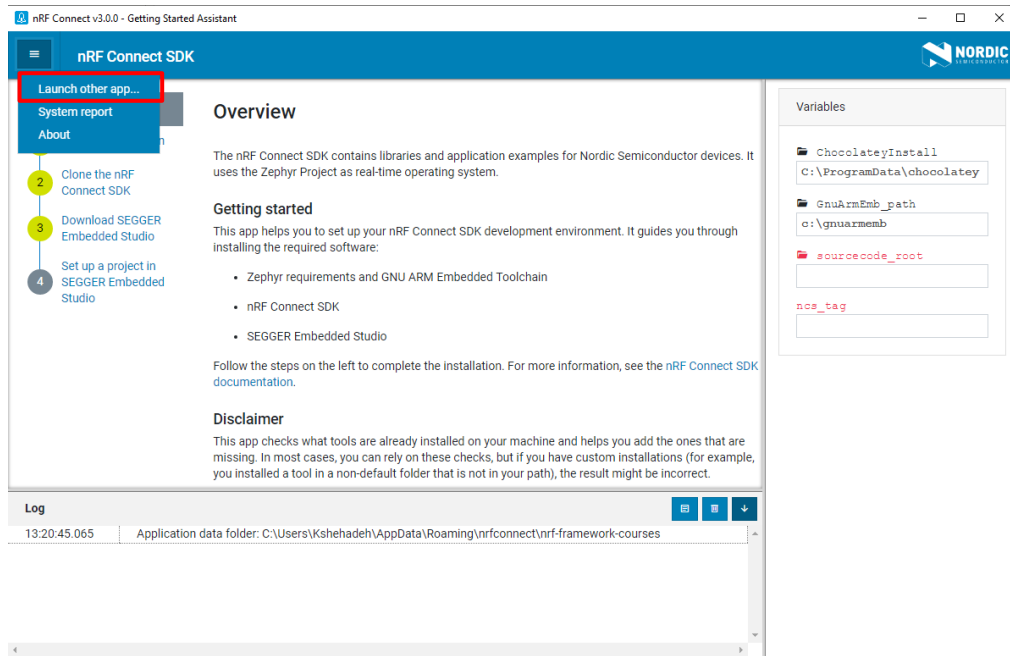


Fig 5.22: Other App Launch

23- Launch **Programmer**.

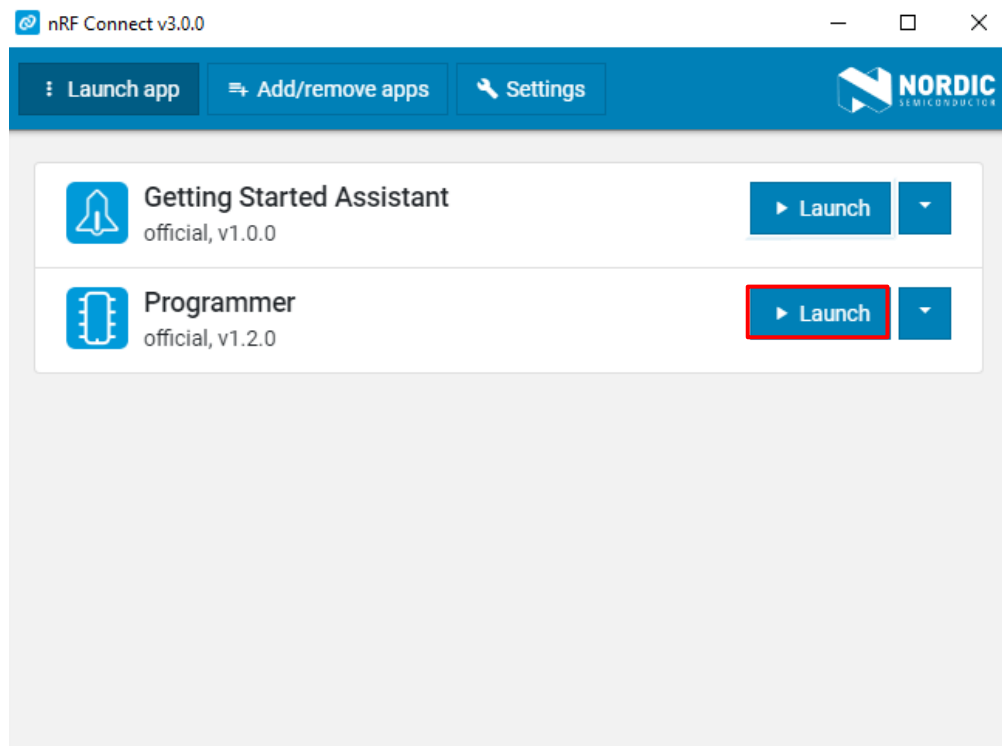


Fig 5.23: Programmer Launch

24- Select your device and update modem firmware when its ready (The nRF9160 has to be connected to your computer).

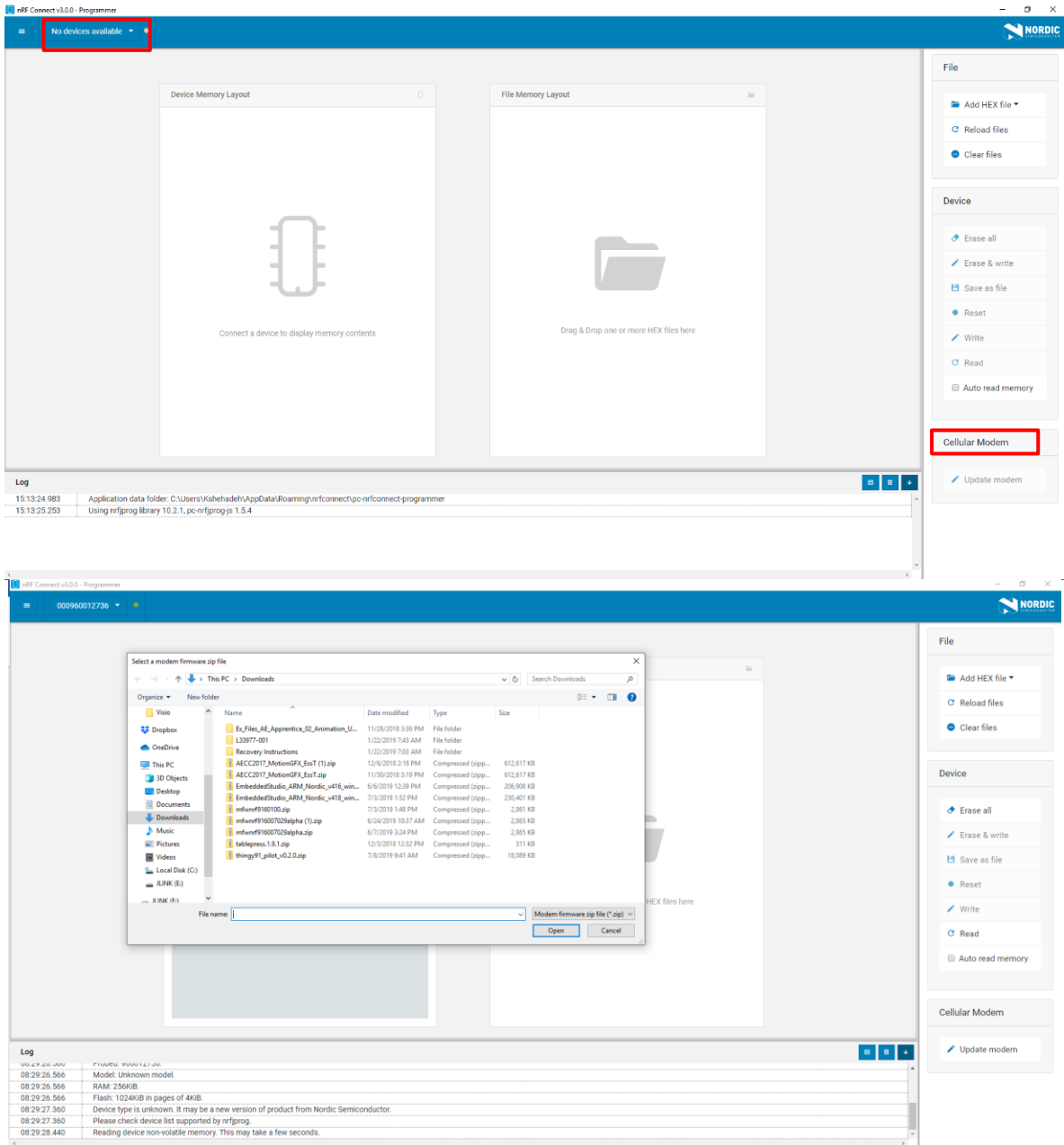


Fig 5.24: Modem firmware Update

## 25- Open the modem firmware zip file and click **write**.

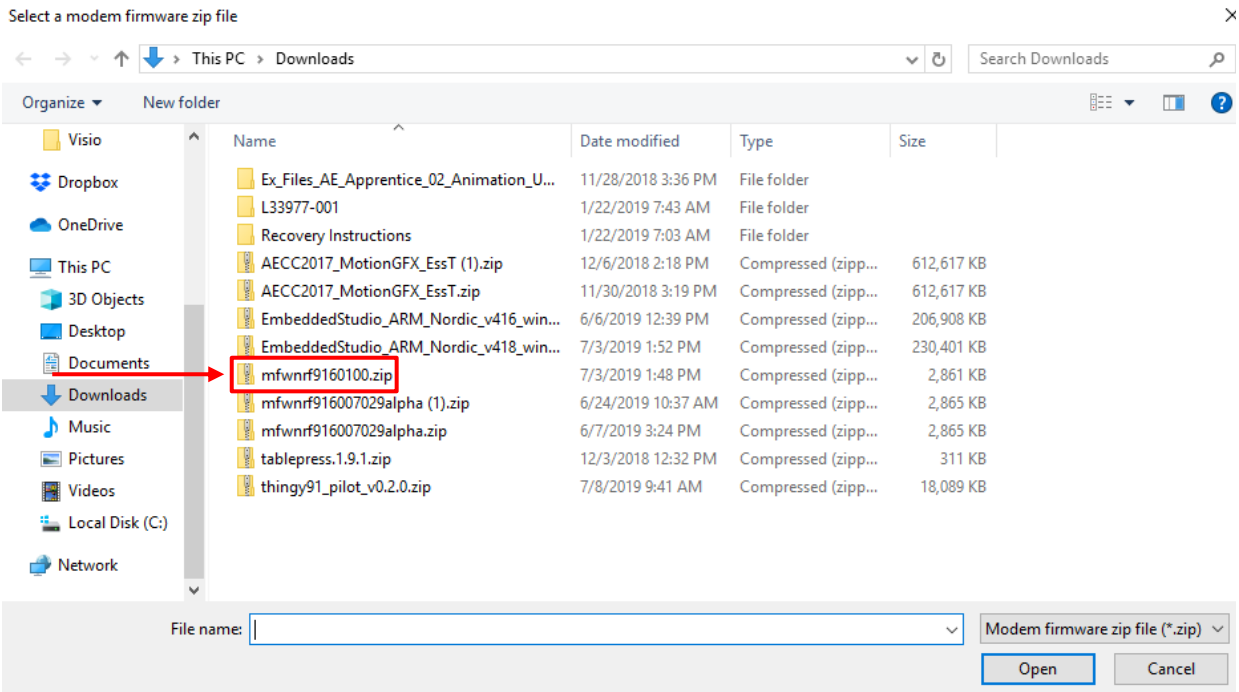


Fig 5.25: Modem firmware zip file

**Note:** To activate your eSIM attached with the nRF9061 Dev kit follow the direction [here \(nordicsemi.com/GetStarted\)](https://www.nordicsemi.com/GetStarted)

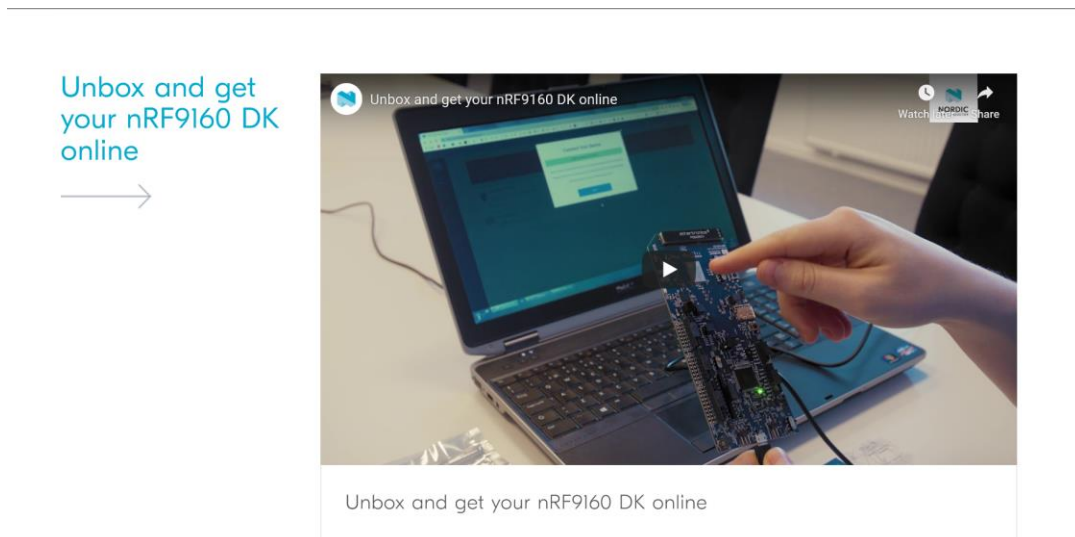


Fig 5.26: Getting Started Video



## 26- After updating the firmware go back to Segger Studio, select **Target > Connect J-Link**

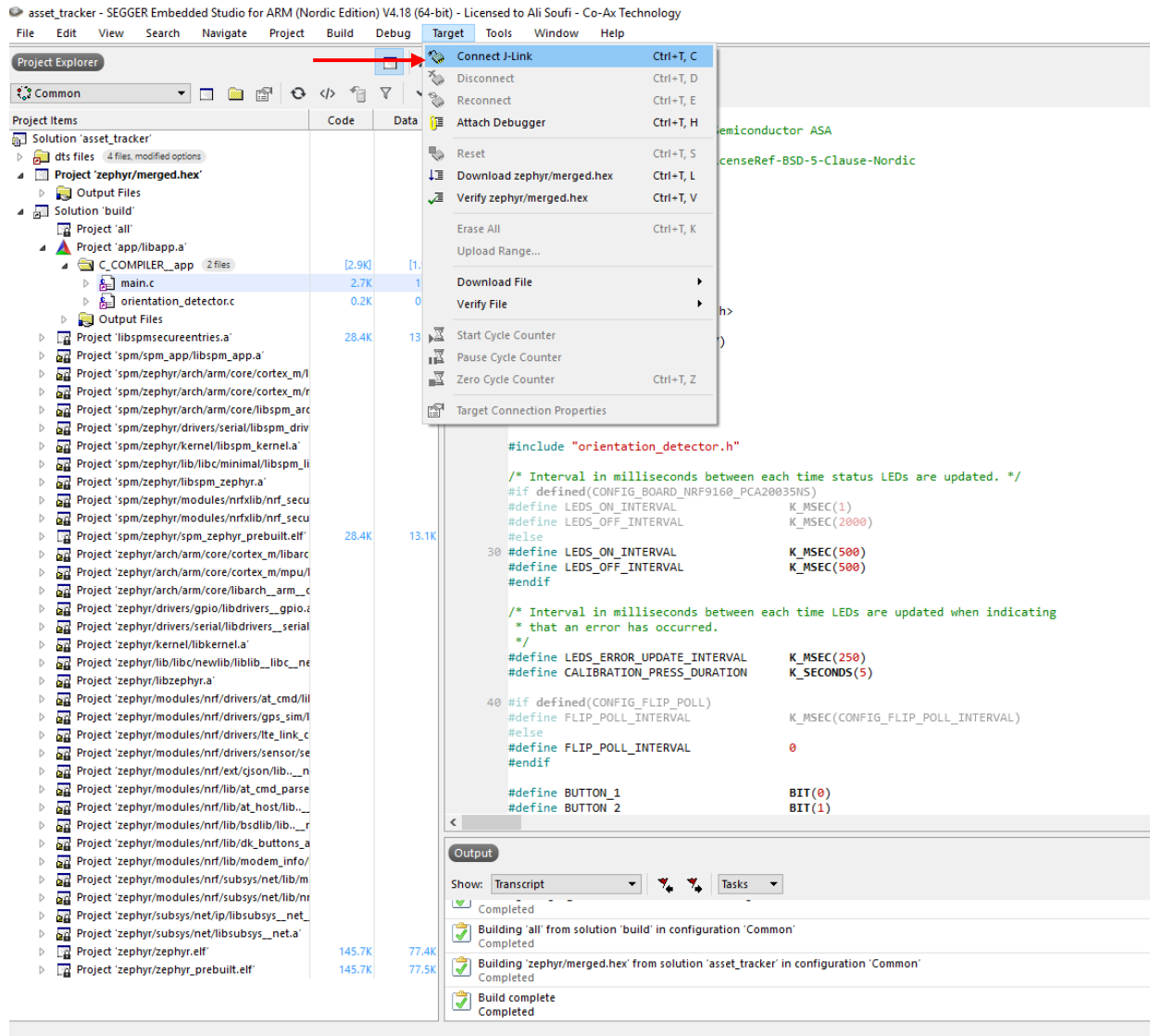
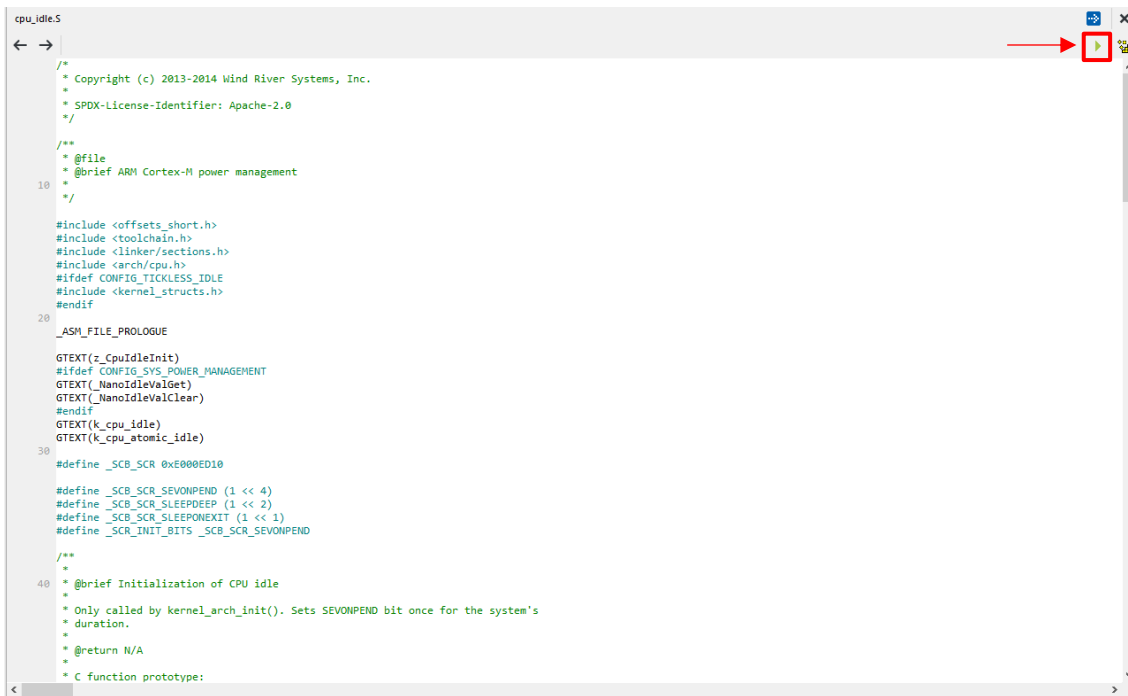


Fig 5.27: Connect J-Link



Fig 5.28: J-Link Connecting

## 27- Click on the green arrow to program the board



```
cpu_idle.S
/*
 * Copyright (c) 2013-2014 Wind River Systems, Inc.
 * SPDX-License-Identifier: Apache-2.0
 */

/**
 * @file
 * @brief ARM Cortex-M power management
 */

#include <offsets_short.h>
#include <toolchain.h>
#include <linker/sections.h>
#include <arch/cpu.h>
#ifdef CONFIG_TICKLESS_IDLE
#include <kernel_structs.h>
#endif

20
_ASM_FILE_PROLOGUE

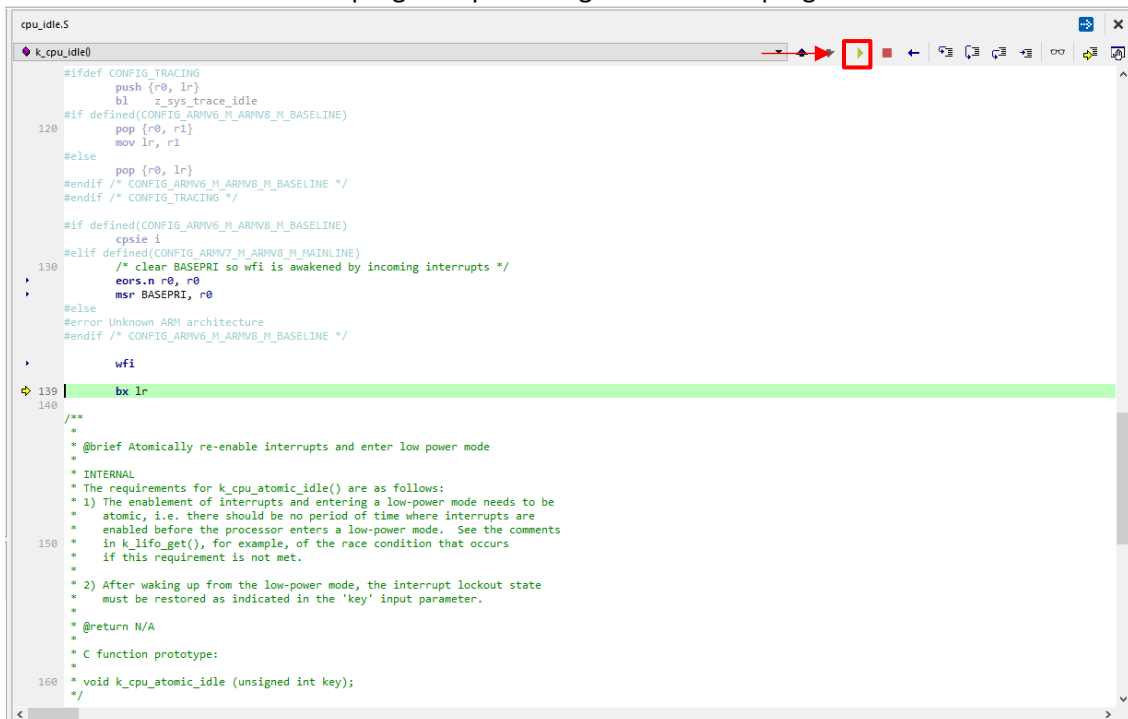
GTEXT(z_cpuIdleInit)
#ifdef CONFIG_SVS_POWER_MANAGEMENT
GTEXT(_NanoIdleLevelGet)
GTEXT(_NanoIdleLevelClear)
#endif
GTEXT(k_cpu_idle)
GTEXT(k_cpu_atomic_idle)

30
#define _SCB_SCR 0xE00ED10

#define _SCB_SCR_SEVONPEND (1 << 4)
#define _SCB_SCR_SLEEPEEP (1 << 2)
#define _SCB_SCR_SLEEPEXIT (1 << 1)
#define _SCB_INIT_BITS _SCB_SCR_SEVONPEND

/**
 *
 * @brief Initialization of CPU idle
 *
 * Only called by kernel_arch_init(). Sets SEVONPEND bit once for the system's
 * duration.
 *
 * @return N/A
 *
 * C function prototype:
 */
```

## 28- After the board programs press it again to run the program.



```
cpu_idle.S
k_cpu_atomic_idle
#ifdef CONFIG_TRACING
push {r0, lr}
bl z_sys_trace_idle
#endif
120 #if defined(CONFIG_ARMV6_M_ARMV8_M_BASELINE)
pop {r0, r1}
mov lr, r1
#else
pop {r0, lr}
#endif /* CONFIG_ARMV6_M_ARMV8_M_BASELINE */
#endif /* CONFIG_TRACING */

#ifdef CONFIG_ARMV6_M_ARMV8_M_BASELINE
cpsie i
130 #elif defined(CONFIG_ARMV7_M_ARMV8_M_MAINLINE)
/* clear BASEPRI so wfi is awakened by incoming interrupts */
eor.n r0, r0
msr BASEPRI, r0
#else
#error Unknown ARM architecture
#endif /* CONFIG_ARMV6_M_ARMV8_M_BASELINE */

wfi
139 bx lr
140
/**
 *
 * @brief Atomically re-enable interrupts and enter low power mode
 *
 * INTERNAL
 * The requirements for k_cpu_atomic_idle() are as follows:
 * 1) The enablement of interrupts and entering a low-power mode needs to be
 * atomic, i.e. there should be no period of time where interrupts are
 * enabled before the processor enters a low-power mode. See the comments
 * in k_lifo_get(), for example, of the race condition that occurs
 * if this requirement is not met.
 *
 * 2) After waking up from the low-power mode, the interrupt lockout state
 * must be restored as indicated in the 'key' input parameter.
 *
 * @return N/A
 *
 * C function prototype:
 * void k_cpu_atomic_idle (unsigned int key);
 */
```

Fig 5.29: Programming

---

29- To connect your nRF9160 to the the nRF Connect for Cloud follow the directions [here](#).

30- To monitor your program, you need to have a console application. We are using **Tera Term**.  
in case you don't have a console application you can download Tera Term by clicking [here](#) .

31- Go to Tera Term

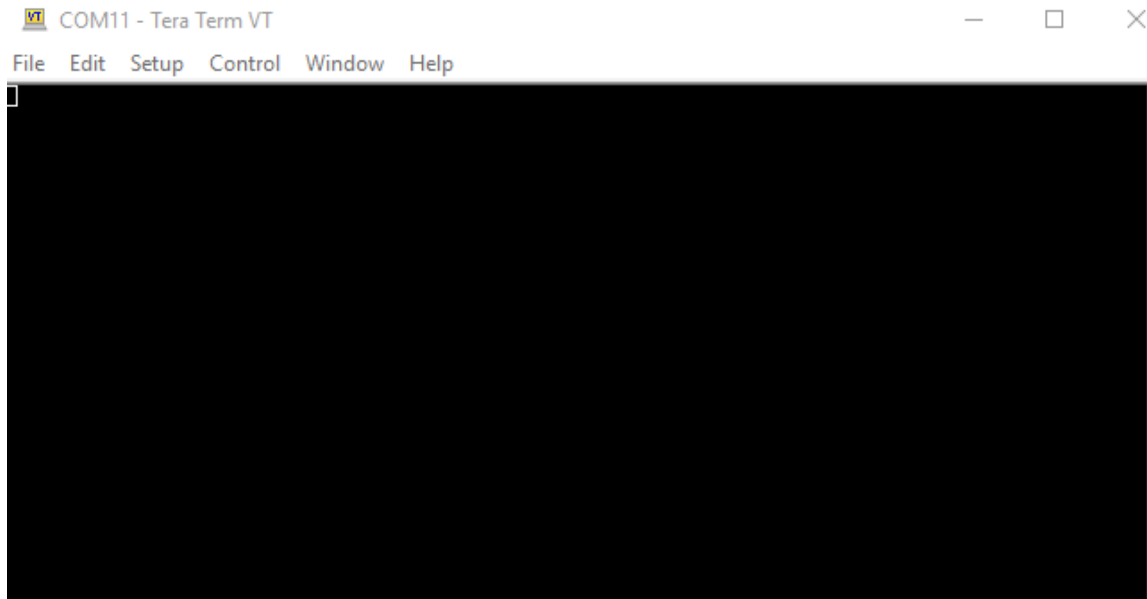


Fig 5.30: Tera Term

32- Go to **Setup>>Serial Port** and set your serial port settings, to figure the correct port between the 3 serial ports, try all of them and message will appear when it's right port.

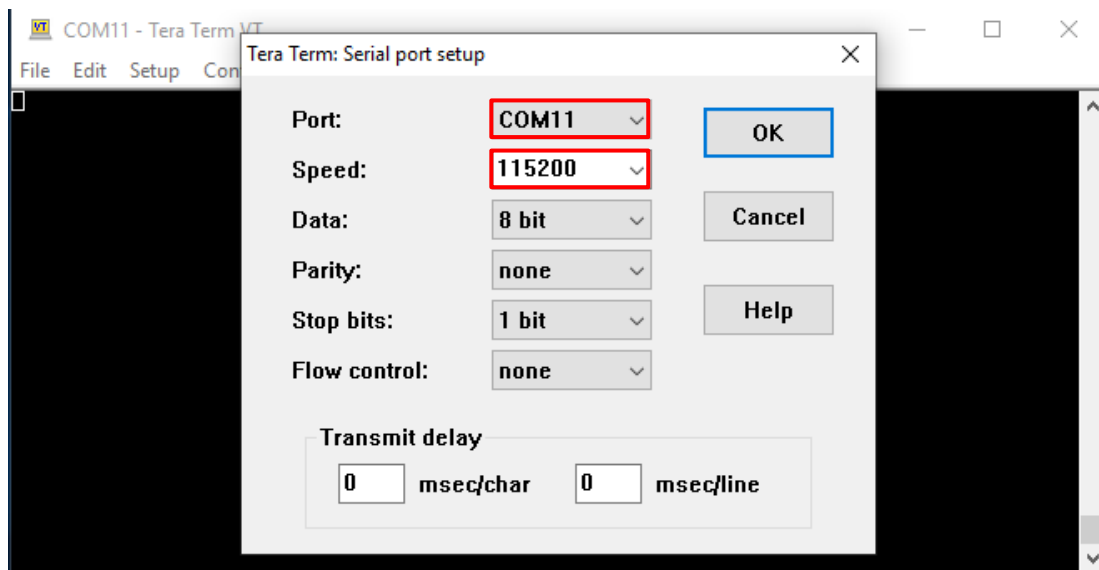
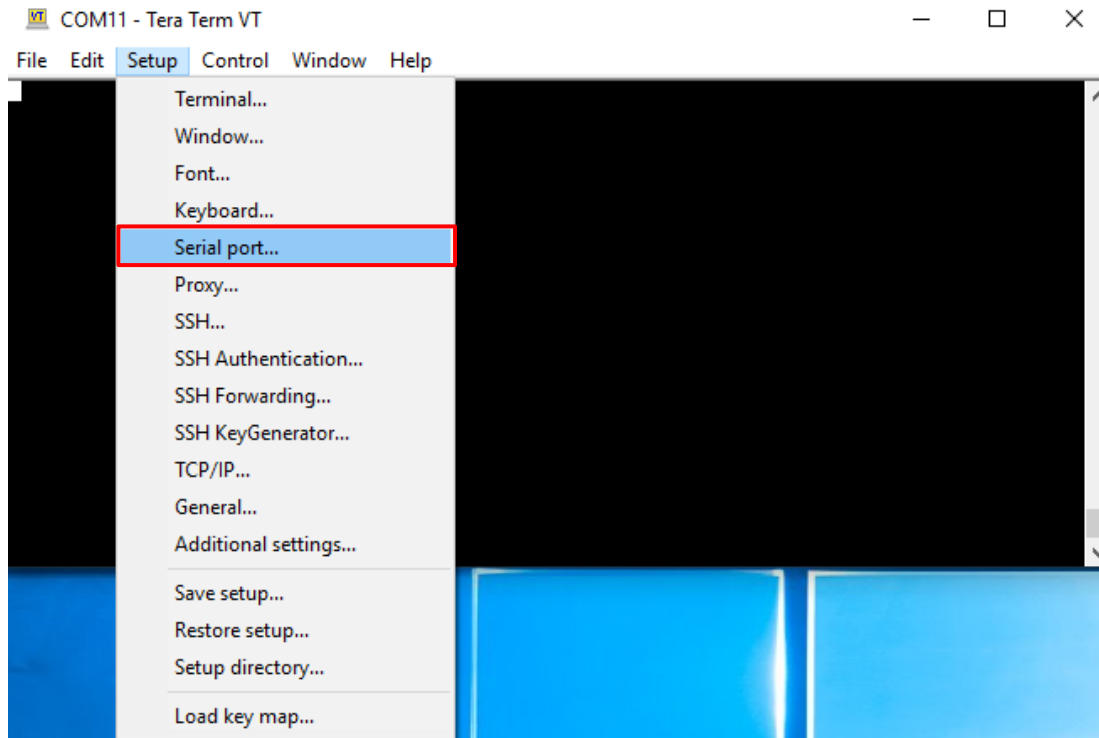


Fig 5.31: Serial Port Setup

To program SCM-1 you need to connect SCM-1 to the nRF9160. Connect the **p1** header on the cellular module to the NORDIC Nrf9160 development kit **p15** header. When power is applied to the module you will be able to program and debug, as if it were the onboard chip.

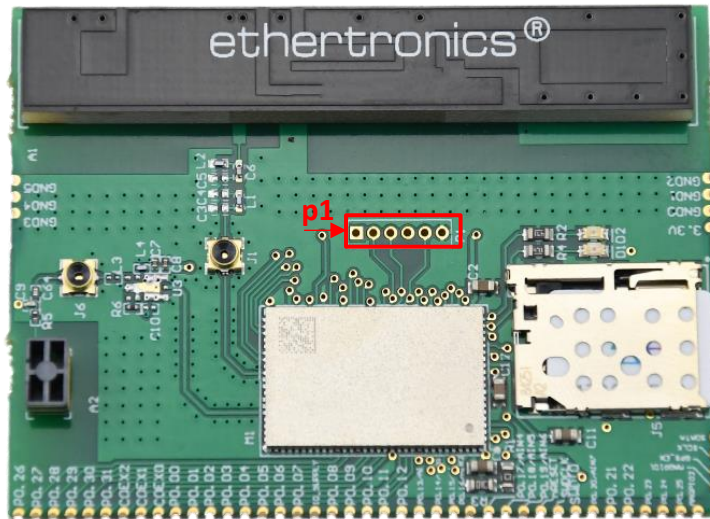


Fig5.32: SCM-1

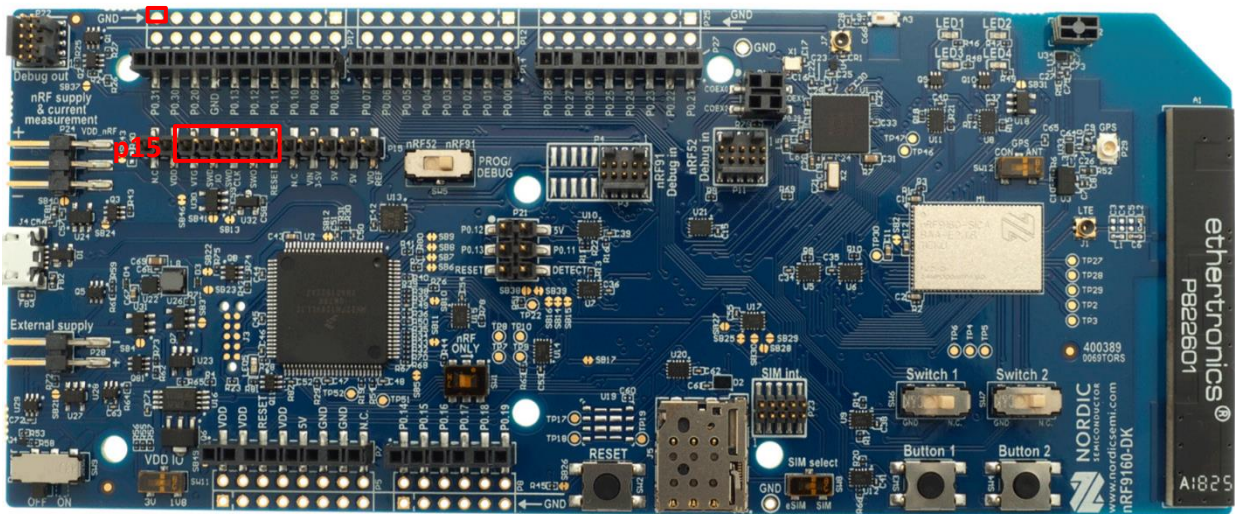


Fig5.33: nRF9160 DK

## 5.3 Programming

Examples and tutorials:

The following code is a simple sample (GPIO configuration). It toggles two LEDs and one other pin within a delay of 1 sec.

```
#include <zephyr.h>
#include <device.h>
#include <gpio.h>

/* 1000 msec = 1 sec */
#define SLEEP_TIME 1000

void main(void)
{
    int cnt = 0;
    struct device *dev;

    dev = device_get_binding("GPIO_0");
    /* Set LED pin as output */
    gpio_pin_configure(dev, 3, GPIO_DIR_OUT); //p0.03 == LED2
    gpio_pin_configure(dev, 4, GPIO_DIR_OUT); //p0.04 == LED3
    gpio_pin_configure(dev, 17, GPIO_DIR_OUT); //p0.17

    while (1) {
        /* Set pin to HIGH/LOW every 1 second */
        gpio_pin_write(dev, 3, cnt % 2); //p0.03 == LED2
        gpio_pin_write(dev, 4, cnt % 2); //p0.04 == LED3
        gpio_pin_write(dev, 17, cnt % 2); //p0.17 Toggling pin 17
        cnt++;
        k_sleep(SLEEP_TIME);
    }
}
```

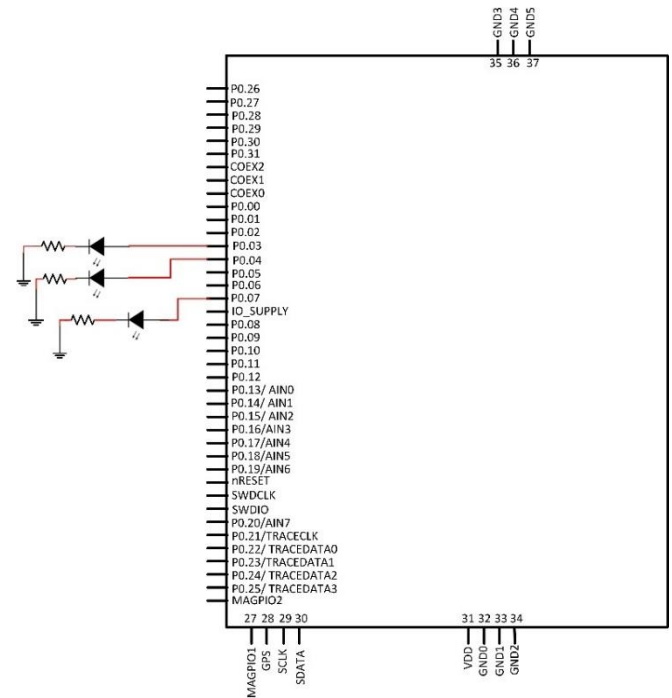


Fig 5.34: Simple sample with the SCM-1 wiring diagram

nRF connect DK provides samples that shows the use of different features.

## General Zephyr Samples

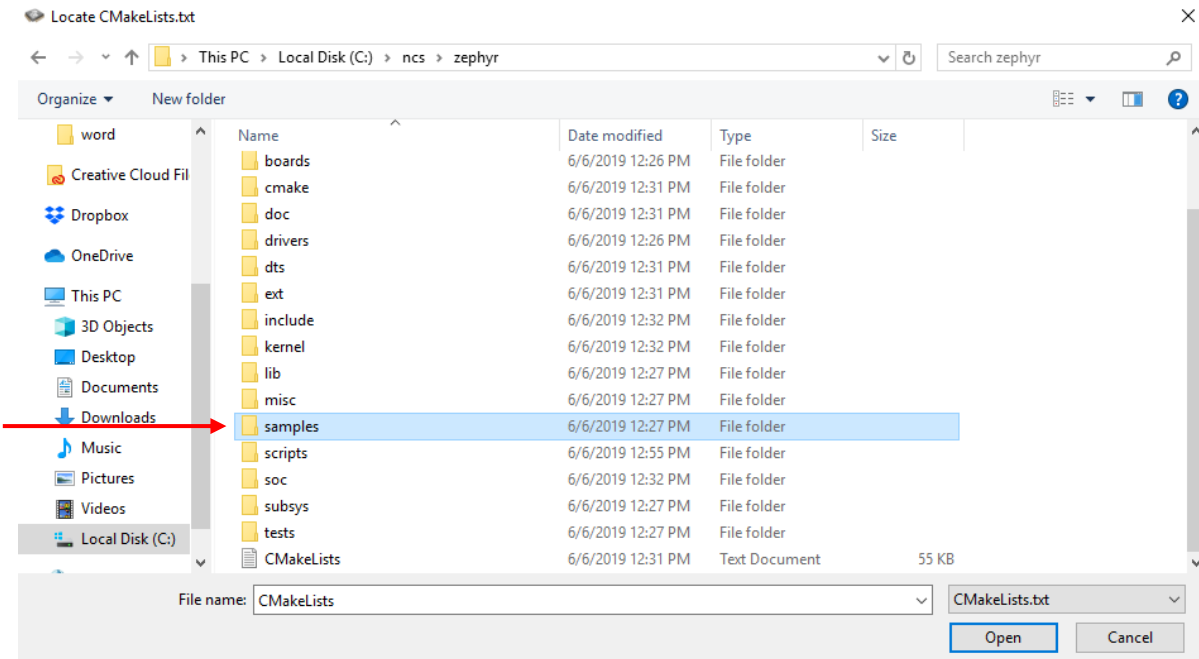


Fig 5.35: General Samples

## nRF Specific Samples

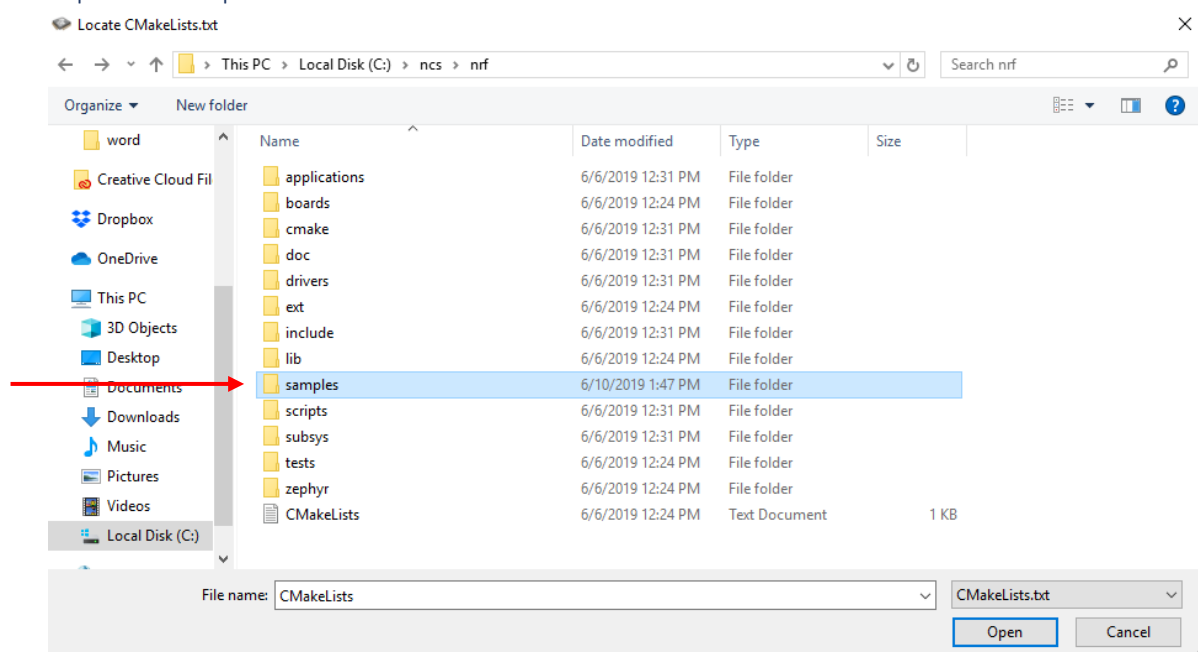


Fig 5.36: Specific Samples

### MQTT Simple Sample:

The MQTT simple sample is one of the Nordic specific examples. To open and test this sample using Segger Studio follow the directions.

#### 1- Open SEGGER Embedded Studio.

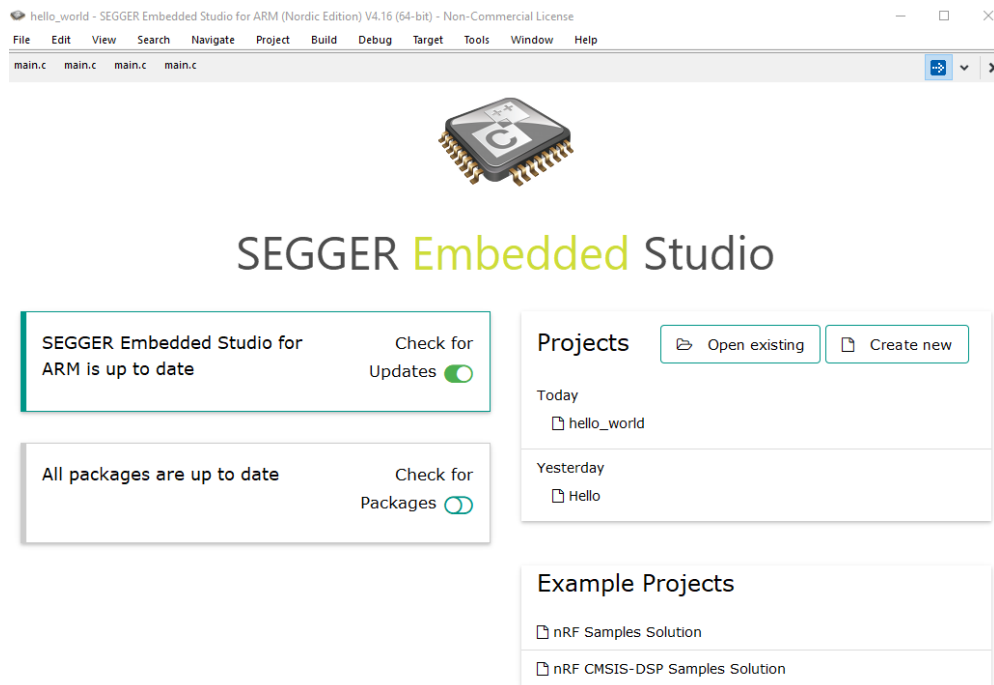


Fig 5.37 Segger Embedded Studio Dashboard

#### 2- Go to File > Open nRF connect SDK project.

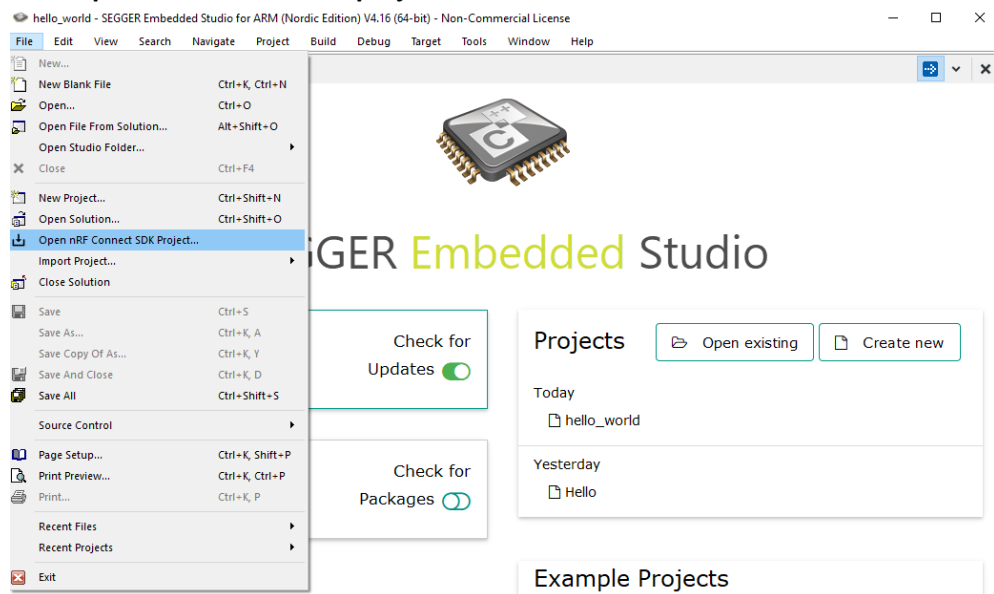


Fig 5.38 Open nRF connect SDK Project



### 3- Set CMakeLists.txt, Board Directory, Board Name and Build Directory.

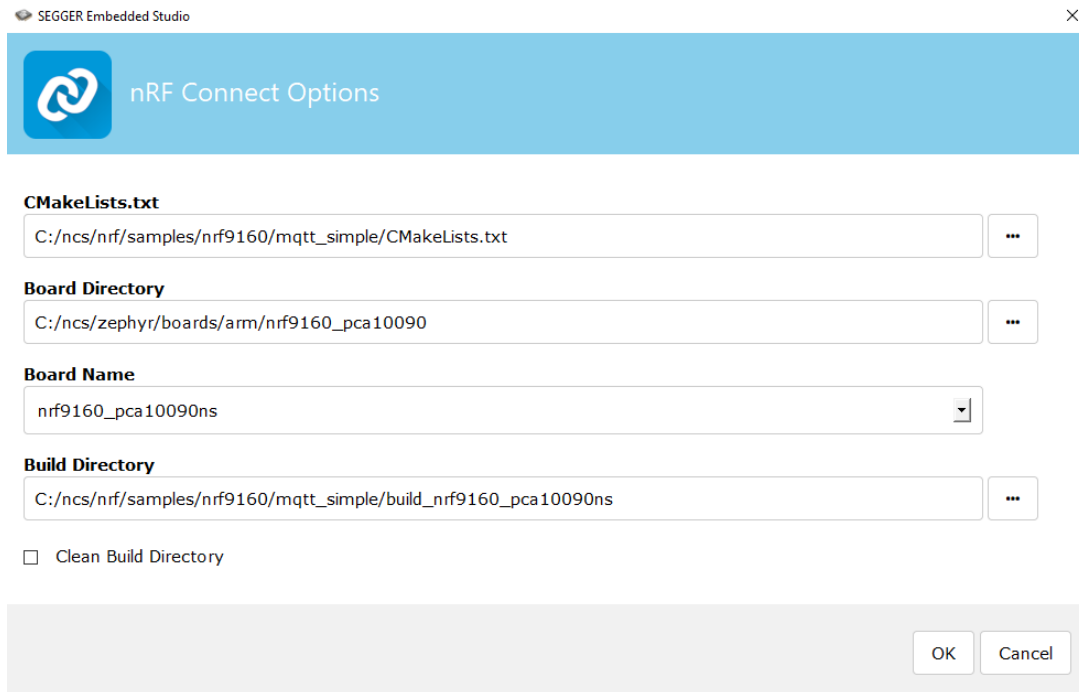


Fig 5.39: nRF Connect Options

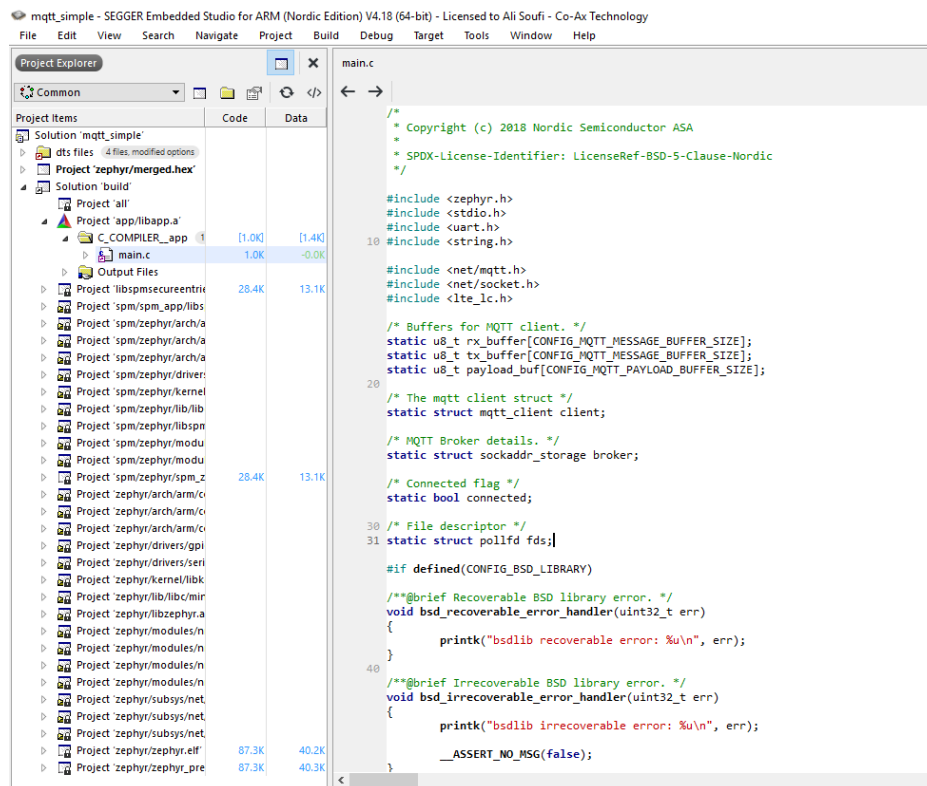


Fig 5.40: MQTT Simple Sample

- 4- To set the broker, broker port, client id, subscribe topic and publish topic go to **Project>>Configure nRF Connect SDK Project**

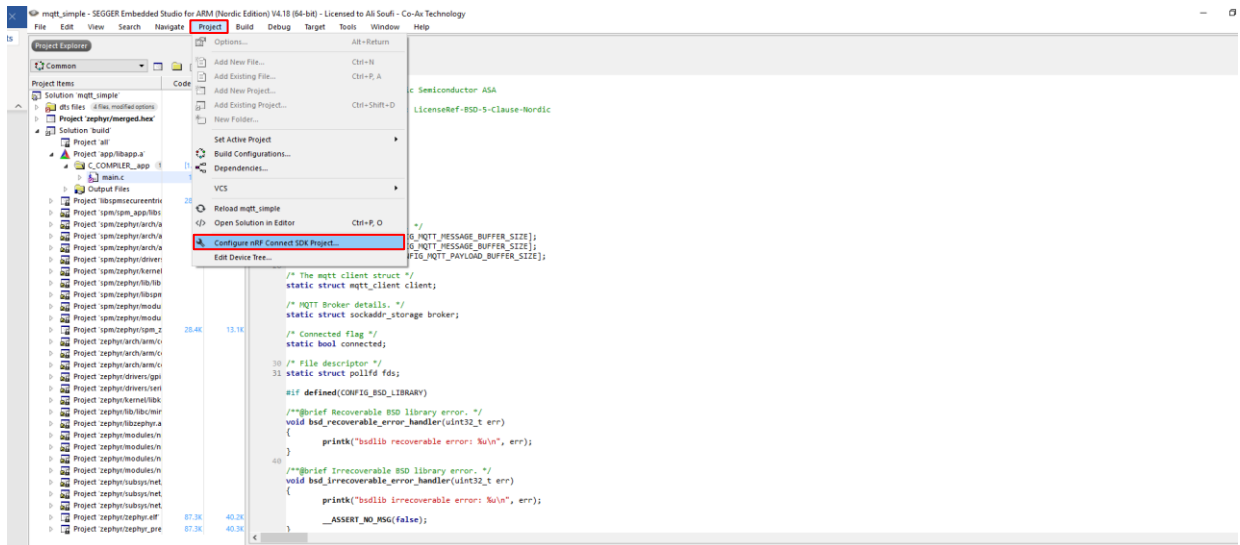


Fig 5.41: Configure nRF Connect SDK Project

- 5- Choose **menuconfig**

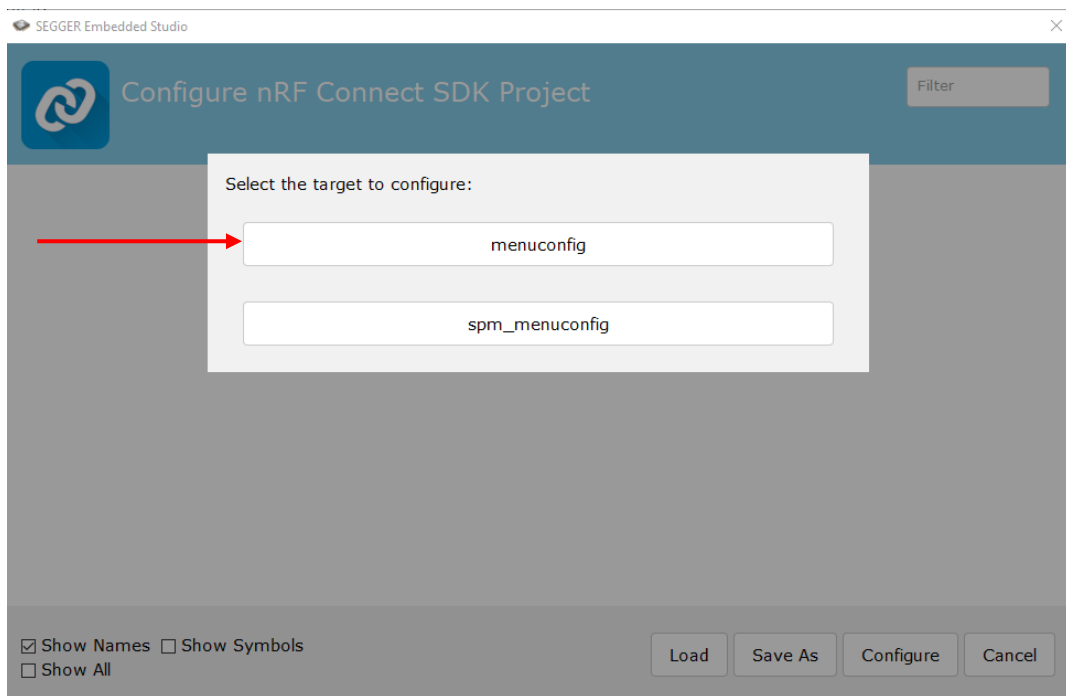


Fig 5.42: menuconfig

- 6- Change the broker hostname to the broker you need to connect. For testing issues, you can choose any public broker as mosquitto, eclipse or Hivemq, then choose **configure**.

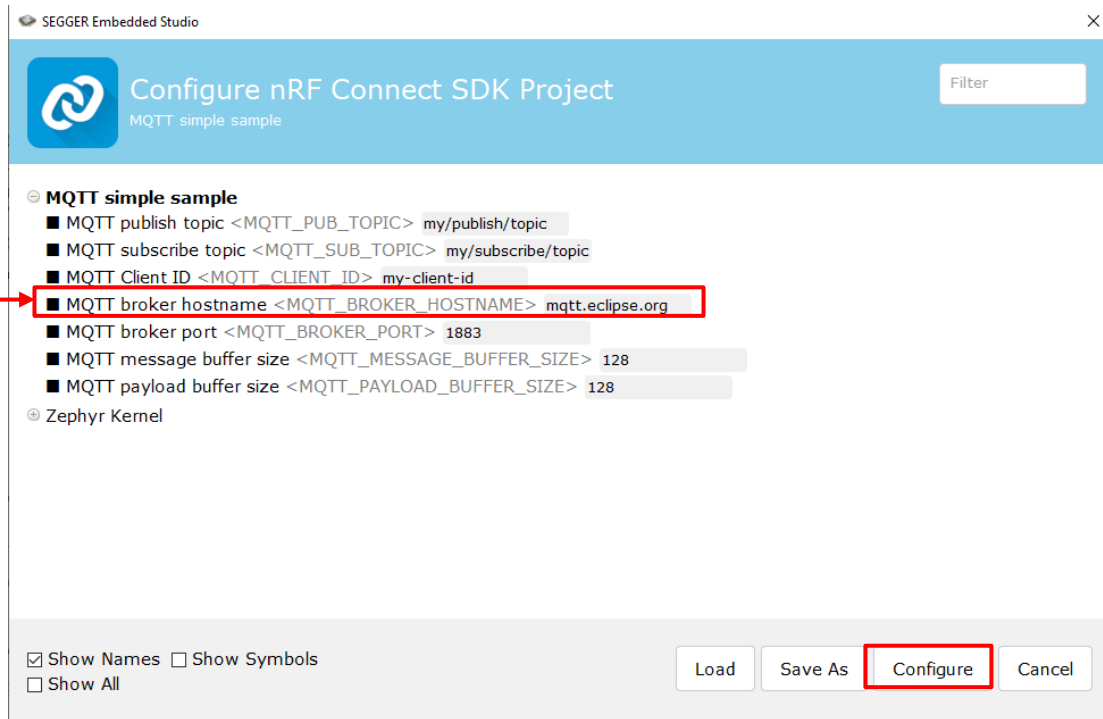


Fig 5.43: menuconfig

- 7- Plug the nRF9160 DK to your computer and go to **Build > Build Solution**.

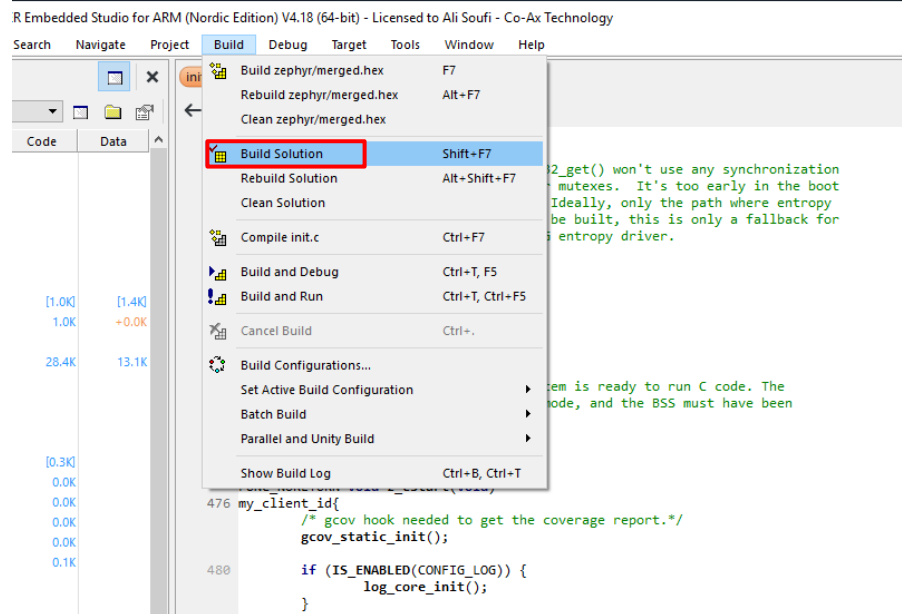


Fig 5.44: Build Solution

- Go to **Target > Connect J-Link**, when its connected go back to **Target** and choose **Erase All** (Be sure to erase the board every time you change your code).

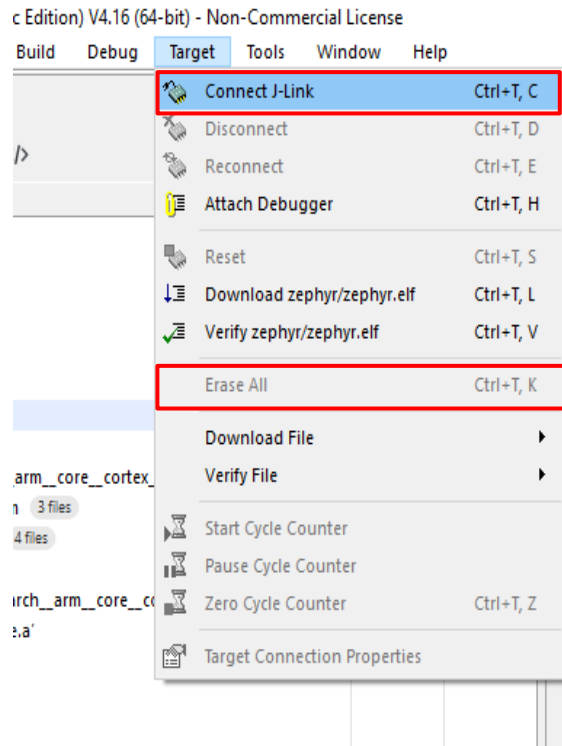


Fig 5.45: Connect J-Link

- Click on the green arrow to program.

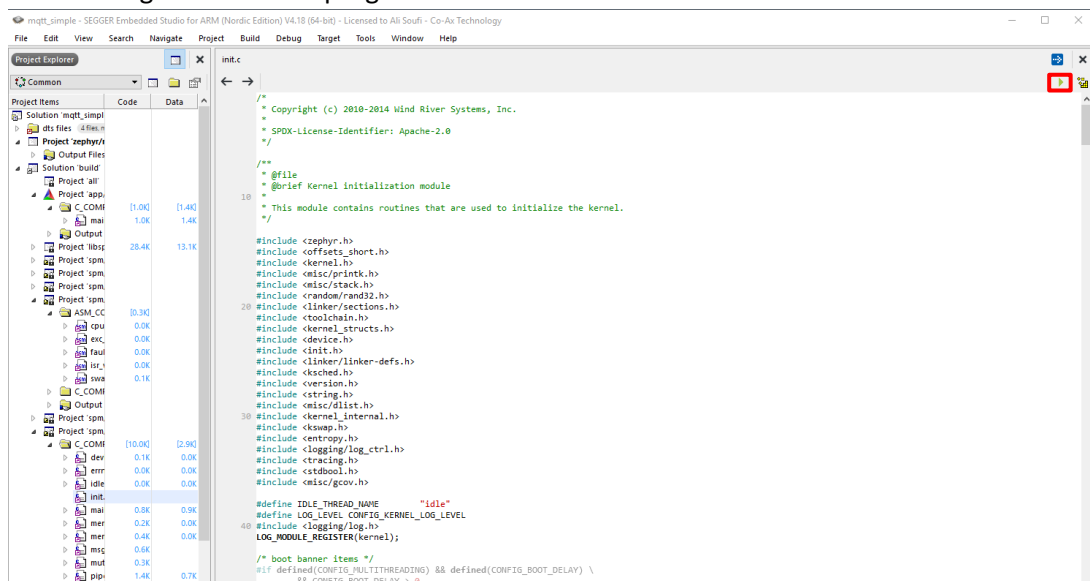


Fig 5.46: Start Programming

10- Run the program and use Tera Term for testing and monitoring.

11- In order to test publishing and subscribing with other client you can use any MQTT App, in this example we used **MQTT Box**.

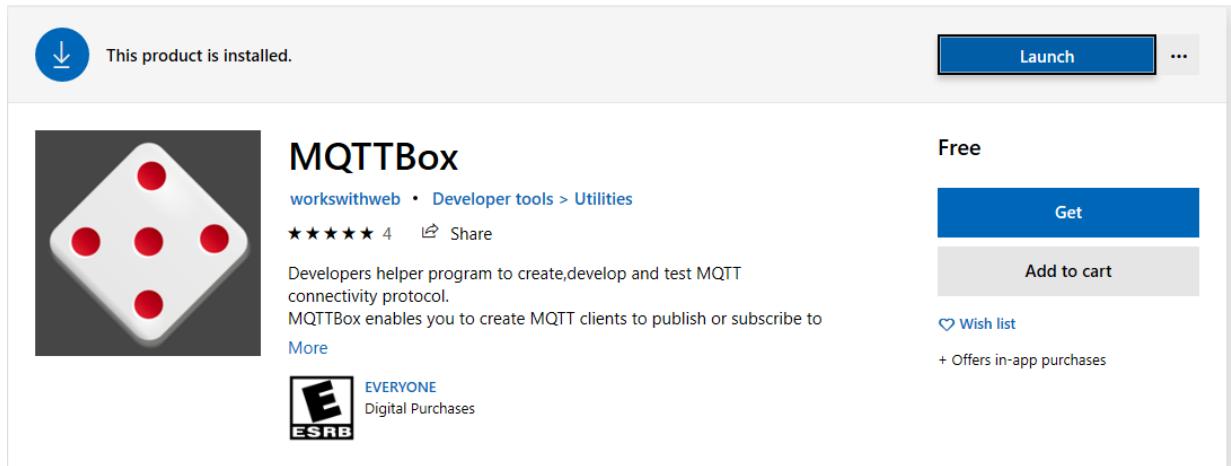


Fig 5.47: MQTT Box App

12- Pick a client name, change the protocol to mqtt/tcp and change the host broker to the broker you are using, then go to **save**.

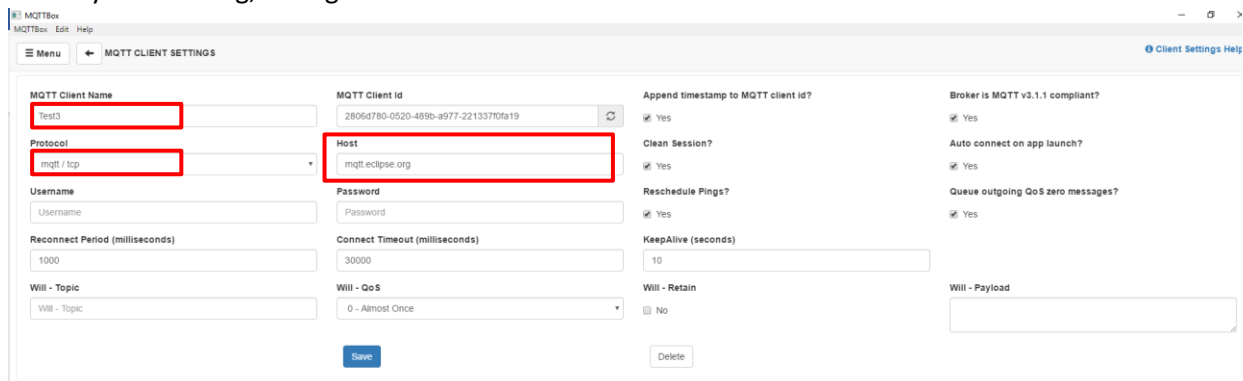


Fig 5.48: MQTT Box App Settings

### 13- Use your subscribe and publish topics to test publishing and subscribing.

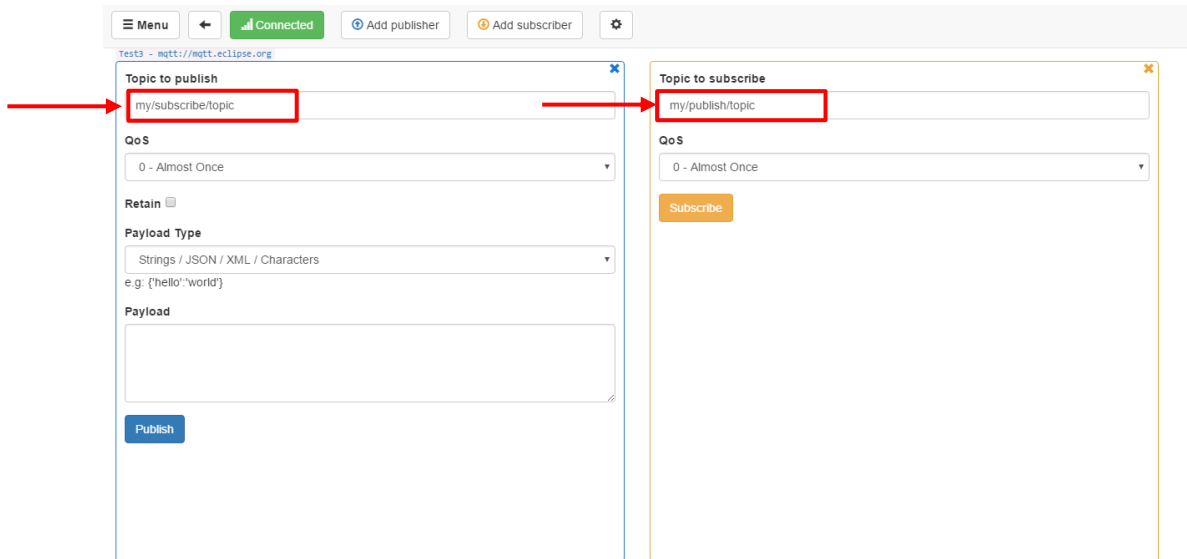


Fig 5.49: MQTT Client Settings

Be sure to use your publishing topic to publish your message on the other side(client)and your subscribing topic to receive a message from the other side(client). Figures 5.47 and 5.48 show the messages that both clients get in case of publishing or subscribing.

To Publish a message in this example you need to modify the example code by adding a publish function. We used this function to publish our test message

**data\_publish(&client, MQTT\_QOS\_1\_AT\_LEAST\_ONCE, "test", strlen("test"));**

```
void mqtt_evt_handler(struct mqtt_client *const c,
                     const struct mqtt_evt *evt)
{
    int err;

    switch (evt->type) {
    case MQTT_EVT_CONNACK:
        if (evt->result != 0) {
            printk("MQTT connect failed %d\n", evt->result);
            break;
        }

        connected = true;
        printk("[%s:%d] MQTT client connected!\n", __func__, __LINE__);
        subscribe();

        data_publish(&client, MQTT_QOS_1_AT_LEAST_ONCE, "test", strlen("test"));
    }

    break;
}
```

If you don't edit the code it will connect, and you will be able to receive messages but without publishing.



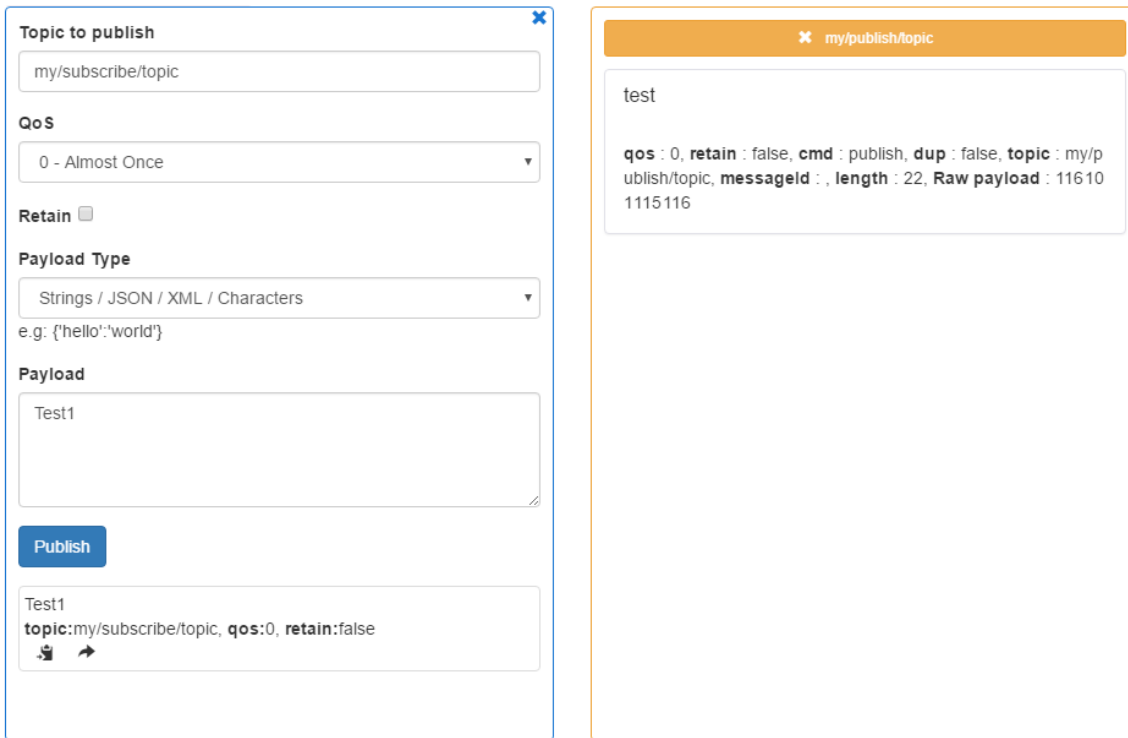


Fig 5.50: MQTT Client Settings

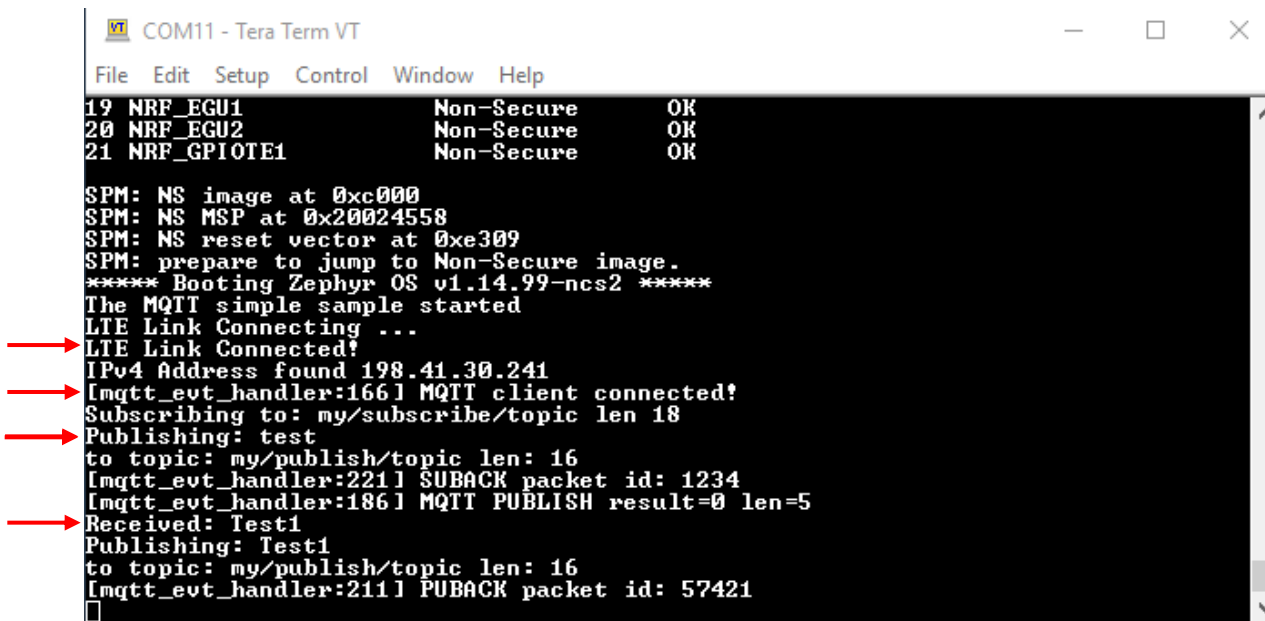


Fig 5.51: Tera Term

---

### *AWS MQTT/TLS Sample:*

The AWS MQTT/TLS sample connects an AWS MQTT broker securely. The code of this sample is attached with this document at the end of this section.

To connect your nRF9160 to an AWS broker you need to have an AWS account. If you don't have an AWS account already go to [AWS IoT Console](#) and create one, then follow the direction below to setup the account.

- 1- After creating an AWS account, the first thing you need to do is creating a policy, so select **Secure>>Policies>> Create a Policy**.

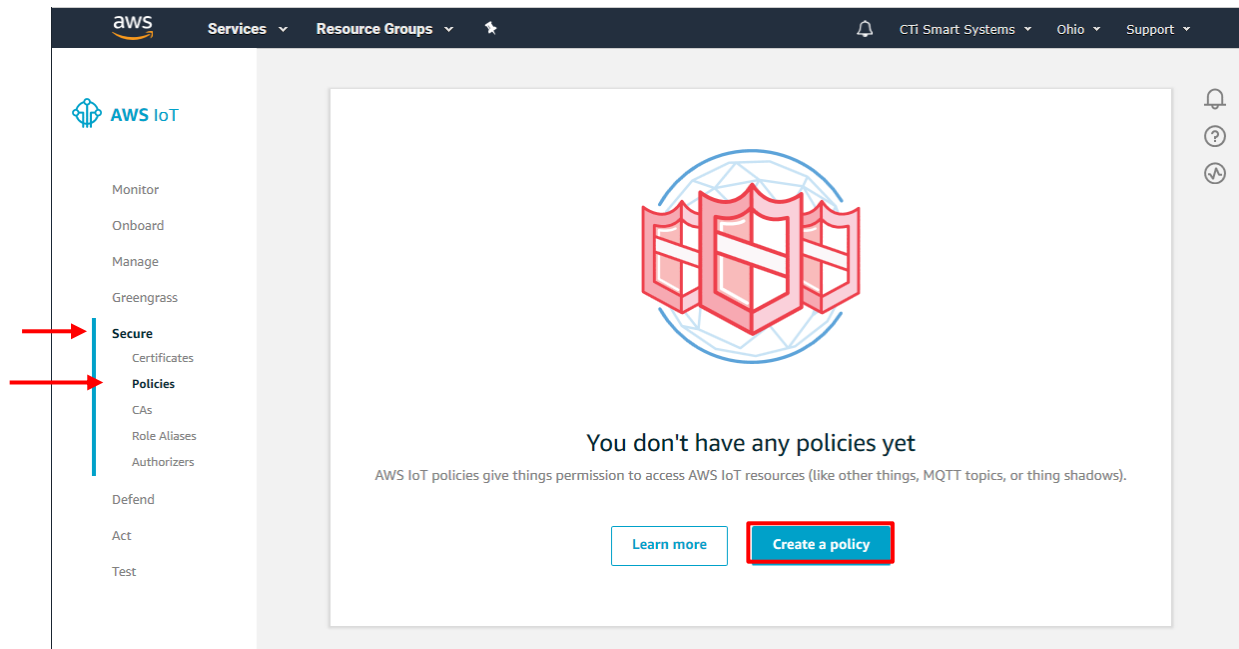


Fig 5.52: Creating a policy



---

2- Add the policy statements you need, and if you are not sure what policies are needed for your device, you can use the one in the figure.

**Create a policy**

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the [AWS IoT Policies documentation page](#).

Name  
Policy1

**Add statements**  
Policy statements define the types of actions that can be performed by a resource. **Advanced mode**

Action	Resource ARN	Effect
iot:*	*	<input checked="" type="checkbox"/> Allow <input type="checkbox"/> Deny

Add statement

Create

Fig 5.53: create a policy

The \* in figure 5.53 **means** that you are using **all** available resources.

- 2- After creating a policy, you need to create a thing that matches your client id. Select **Manage>>Things>>Register a thing**, then choose **Create a single thing**.

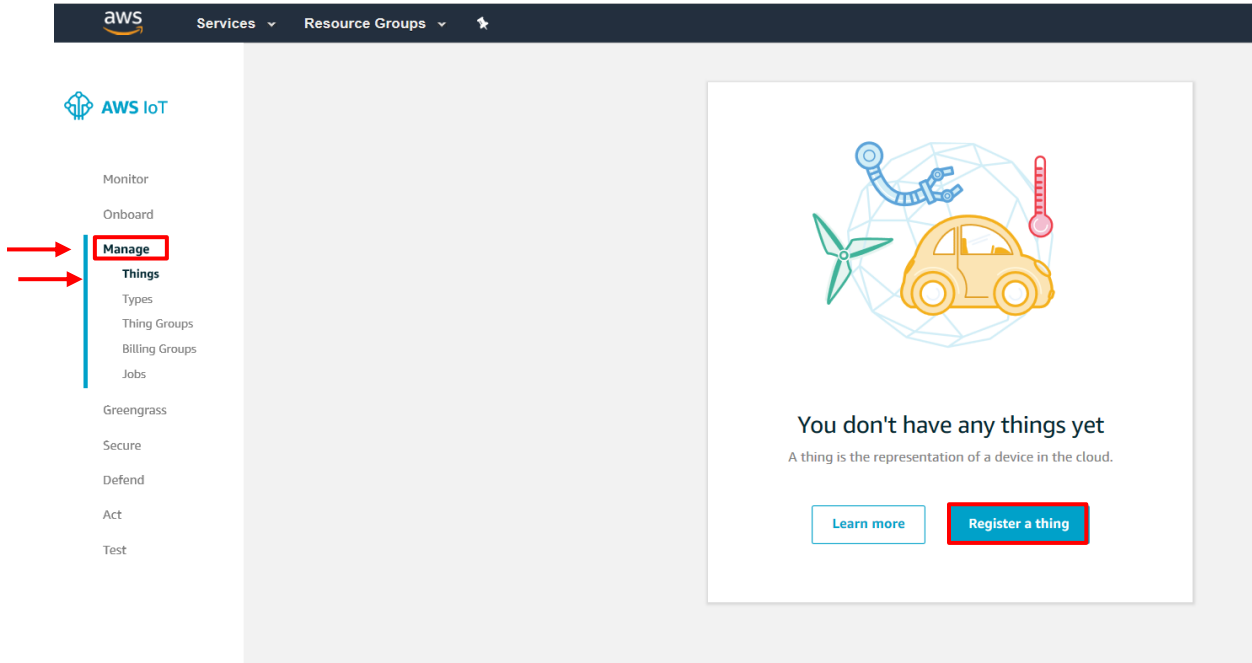


Fig 5.54: AWS IoT

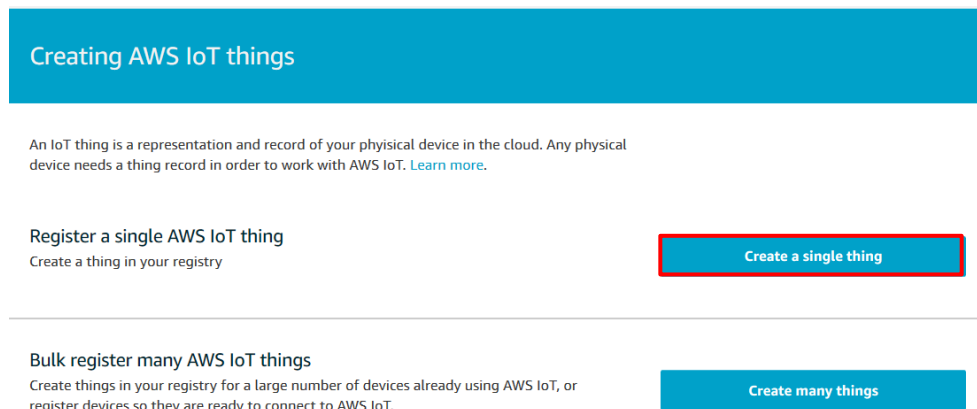


Fig 5.55: Creating AWS IoT Thing

- 
- 3- Add your device to the thing registry by writing your client id as a thing name.

CREATE A THING

STEP 1/3

## Add your device to the thing registry

This step creates an entry in the thing registry and a thing shadow for your device.

Name

my-client-id

Apply a type to this thing

Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

No type selected Create a type

Add this thing to a group

Adding your thing to a group allows you to manage devices remotely using jobs.

Thing Group

Groups / Create group Change

Set searchable thing attributes (optional)

Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key Value

Provide an attribute key, e.g. Manufacturer Provide an attribute value, e.g. Acme-Corporation Clear

Add another

Show thing shadow ▾

Cancel Back Next

Fig 5.56: Creating AWS IoT Thing

#### 4- You need to create a certificate for your thing.

CREATE A THING STEP 2/3

### Add a certificate for your thing

A certificate is used to authenticate your device's connection to AWS IoT.

**One-click certificate creation (recommended)**  
This will generate a certificate, public key, and private key using AWS IoT's certificate authority. [Create certificate](#)

---

**Create with CSR**  
Upload your own certificate signing request (CSR) based on a private key you own. [Create with CSR](#)

---

**Use my certificate**  
Register your CA certificate and use your own certificates for one or many devices. [Get started](#)

---

**Skip certificate and create thing**  
You will need to add a certificate to your thing later before your device can connect to AWS IoT. [Create thing without certificate](#)

Fig 5.57: Adding a certificate

#### 5- Download and Activate the certificate and the keys.

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	51aedab585.cert.pem	<a href="#">Download</a>
A public key	51aedab585.public.key	<a href="#">Download</a>
A private key	51aedab585.private.key	<a href="#">Download</a>

You also need to download a root CA for AWS IoT:  
A root CA for AWS IoT [Download](#)

[Activate](#)

Cancel Done [Attach a policy](#)

Fig 5.58: Certificates Download

- 
- 6- Before attaching policy, you need to download an **AWS root CA**, Click Download, and click on one of Amazon root certificates then copy the certificate and save it where you have the keys and the client certificate. We used **Amazon Root CA 1**.

**Server Authentication**

Server certificates allow your devices to verify that they're communicating with AWS IoT and not another server impersonating AWS IoT. Service certificates must be copied onto your device and referenced when devices connect to AWS IoT. For more information, see the [AWS IoT Device SDKs](#).

AWS IoT server certificates are signed by one of the following CA certificates:

**VeriSign Endpoints (legacy)**

- RSA 2048 bit key: [VeriSign Class 3 Public Primary G5 root CA certificate](#)

**Amazon Trust Services Endpoints (preferred)**

- RSA 2048 bit key: [Amazon Root CA 1](#).
- RSA 4096 bit key: Amazon Root CA 2 - Reserved for future use.
- ECC 256 bit key: [Amazon Root CA 3](#).
- ECC 384 bit key: Amazon Root CA 4 - Reserved for future use.



Fig 5.59: Amazon Root CA

- 7- You can go back and click Attach Policy.

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

**In order to connect a device, you need to download the following:**

A certificate for this thing	51aedab585.cert.pem	<a href="#">Download</a>
A public key	51aedab585.public.key	<a href="#">Download</a>
A private key	51aedab585.private.key	<a href="#">Download</a>

**You also need to download a root CA for AWS IoT:**

A root CA for AWS IoT [Download](#)

[Deactivate](#)

[Cancel](#)

[Done](#)

[Attach a policy](#)

Fig 5.60: Attach a Policy

8- Select your policy and click **Register Thing**

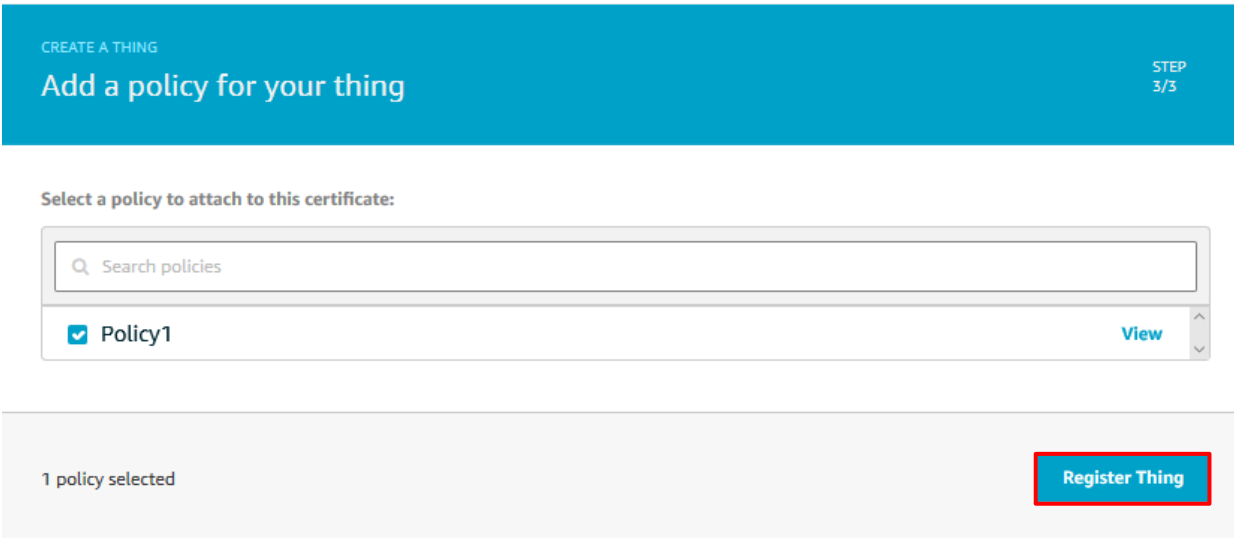


Fig 5.61: Adding your policy

9- You need to add the certificates you downloaded to your device configuration. Go to **CC-MSIMPLE-TLS>>src>>certificates.h**

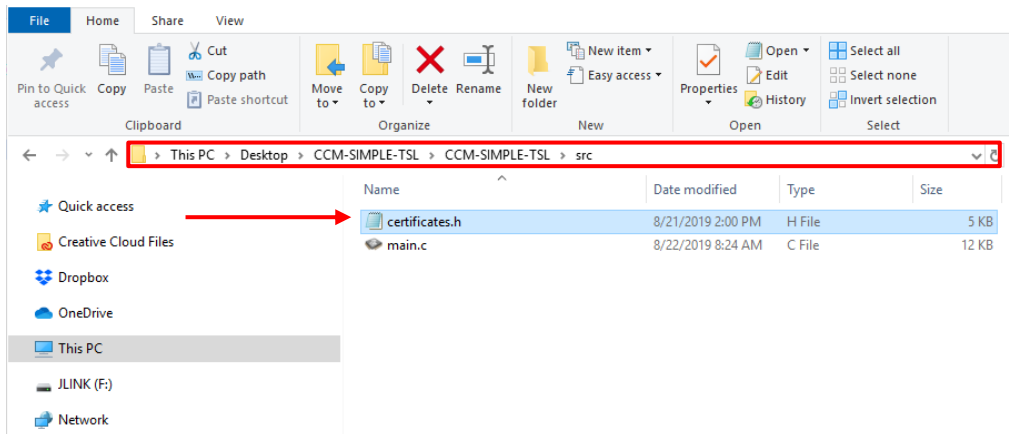


Fig 5.62: Certificates.h directory

10- Copy the **client certificate**, **private key** and the **Amazon root CA** to **certificates.h** as strings then save. **You don't need to use the public key here.**

```
certificates.h - Notepad
File Edit Format View Help
/*
 * Copyright (c) 2018 Nordic Semiconductor ASA
 *
 * SPDX-License-Identifier: BSD-5-Clause-Nordic
 */

#define CLIENT_ID "nRF9160-DK"

#define CA_CERTIFICATE \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDQTCCAIegAwIBAgITBeyfz5w/jAoS4vB41kPw1jZbyjANBgkqhkiG9w0BAQsF\n" \
"ADASPDQwCQYDVQQGEwVUuzEPMAAGA1UEChMGQGN1heeb9uPWRxufwYDVQQDEwBBBwF6\n" \
"l24gIw9vKBDQSAwPB4XOTE1MDUyNjA0PDwPFoKDTMAMEeNzAwPDwPFowOTEI\n" \
"lWkGA1UEBHFVPMwDzANBgNVBAoTBkFtYXNjYXN0eS5kaW8uY29udGVudjE2\n" \
"l3QgQDEgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHwKXj\n" \
"ca0HgFB0W7Y14h29j1o91ghYP10hAeVrAIthtOgQ3pOsqTQ9ro8vo3bSPgHFz2H\n" \
"906IIBc+6zF1tRn45W1w3te5dJgdY26k/oI2peVKVuRF4Fn9tBb6d9qczU5L/qr\n" \
"JFAGbHrQgLKw+a/sRwPUDgH3KXHVj4urtlp+UhrPCbu1HheB4wJUCAdmahRMe6\n" \
"VQuJwSH55Nz/BegwLX0tdHA114gk957EhM67c4cXBj3GKLHD+rcdqsq88p8kD111\n" \
"93FcRw/6pKCyZ1K+1A4b9v7LkIbxcceVDF34GfID5yHI9Y/QCB/IIDEgEw+OyQm\n" \
"jgSub3rIggBCAwEAaAHCMEAwDwYDVR8TAQH/BALwAwEB/zAQBgNVHQ8BAF8EBAPC\n" \
"AYYwqYDVR8QB8BYEFIQVzIUW7LwP13QuCFmcx7IQTgoIPMAGCSqGSIb3DQEBCwIA\n" \
"AAIBAQCYBjdaQZChGsV2U5ggN1PD+uYou6r41KS1p0B/G/wkjiUuByKX9rbxwDZ\n" \
"USPCCjJwCXP16T531HTFIU3+U6adT+CC2q3eHZERxh1bI1BjJt/wsv0tadQ1wJs\n" \
"lwgD563pYaAcbvXy8P4y7Vu33Pq00HeeE6V/Uq2V8v1T09GLXFvKW13bYKBU90v\n" \
"e/uqJ3VtPNTBQcPHRn8j+dkPSHCa2XV4cdFyQzR1b1d2wg3c3wApzyKZFo6IQ6XJ\n" \
"5MsI+ymRQ+HDKXJ1oa1dXgJAKK642M4wrt8VBob2x3NDd22hwLnoQdeXeGADbkpy\n" \
"nqXRfboQno2sG4q5WTP4685QvvG5\n" \
"-----END CERTIFICATE-----\n" \

#define CLIENT_PUBLIC_CERTIFICATE \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDNjCCAKGwAwIBAgIVAPPh3JNS8rp1a5vd714a/A751EerPMAGCSqGSIb3DQE3B\n" \
"CuIAMEBx5z87BgnVBAwMQkFtYXpvc18X2Mlgl2VydmljZDpGtz1BbWf6b24uY29u\n" \
"IE1uYy4gTD1TZWf8dGx11FNUPVdhc2hpbnw8b24gQz1VUzAeFw8OTAAMPjIwODI\n" \
"lHjBafw88OTEyPzEyPzUSNT1aPB4xHDAaHgNVBAPPE8FKUy87b1QgQ2Vydg1waf\n" \
"dlGaggEIPMAGCSqGSIb3DQEBAQA4I8DwAgggEKAoIBAQDSr1zY9n31VDx/eS/2\n" \
"TrUP6suhAzEcsvpfhL3v0GSCO22nx3PAb9aPC4/818Pg1z68P83y65AHj+acv85\n" \
"yXKv1obE3+tb0wrbko891zbr1s5BFH7E+v235CLYNAn31HEyt1DY7rMe+Q9vCBE\n" \
"TwDccgcPxY1eL2o+c4ek8gVTUrtCw5y7oNe20719Lh3QYF+w5RQ02k2G1u0LQbT\n" \
"FKRRlgeSet3zbY7kzsvG7FSKcT+4s2w4cxKkwydFMAnkxjxtu05LEp1gPVoTX+fi\n" \
"-----\n" \
```

Fig 5.63: Certificates.h file

11- Go back to your AWS account, click on the thing you created and select Interact. The thing shadow is the host broker that you will need to add to your configuration.

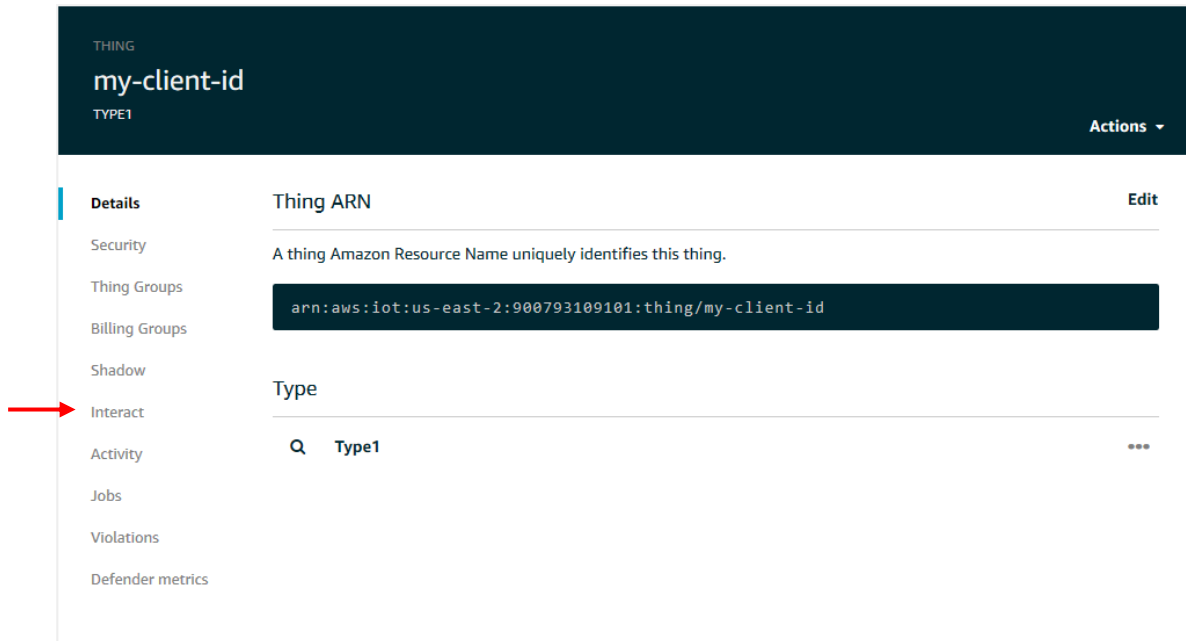


Fig 5.64: Thing Details

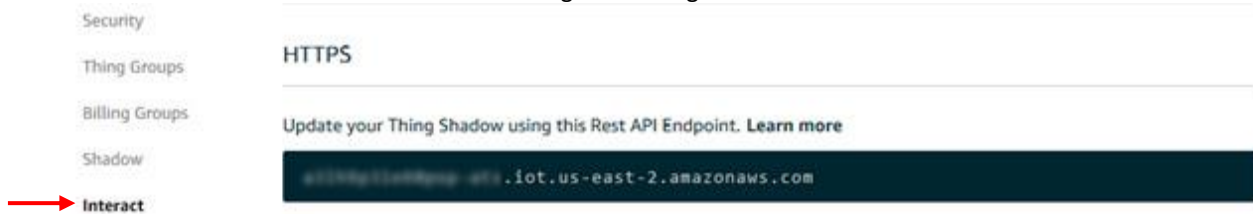


Fig 5.65: Host Broker

12- Open your **Segger embedded studio**, select file, **Open nRF connect SDK Project** and choose our **zip file**.



- 13- Select **Project**>>**Configure nRF Connect SDK Project**>>**menuconfig**>>**MQTT simple sample**.
- 14- Use your new Host Broker, then choose publish and subscribe topics, and configure.

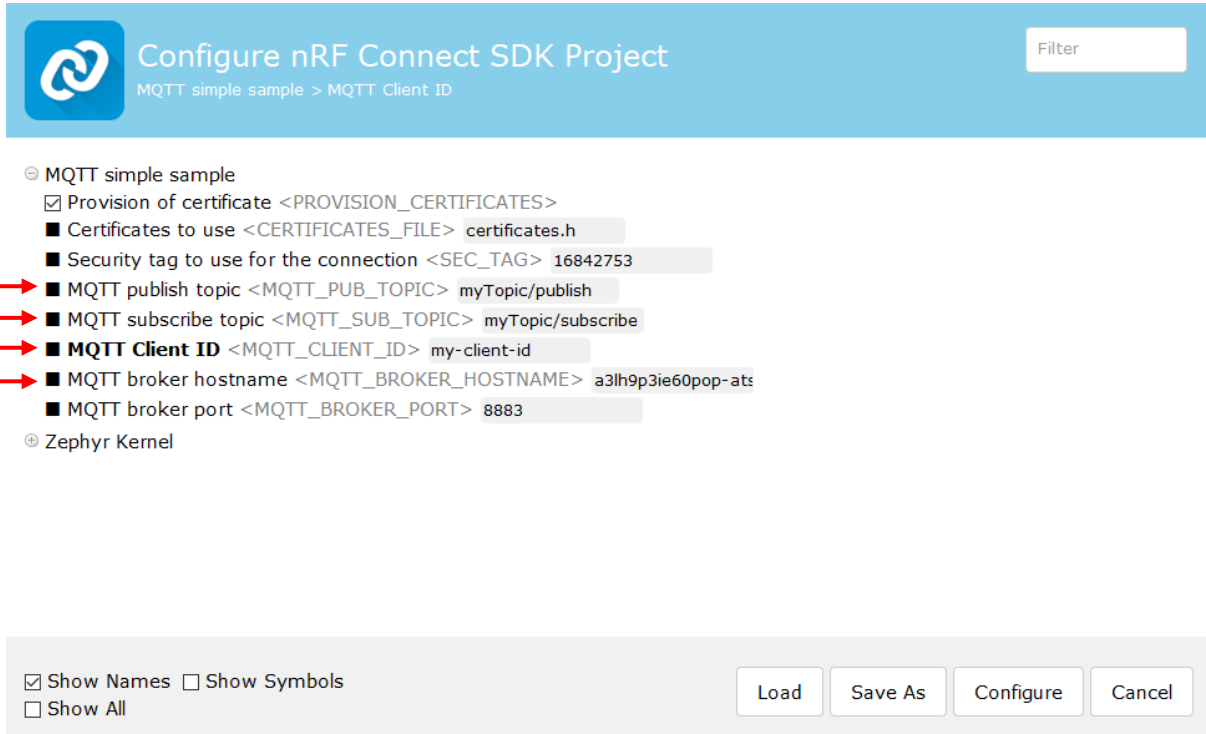


Fig 5.66: Configure nRF Connect SDK Project

- 15- Plug the nRF9160 DK to your computer and go to **Build > Build Solution**.
- 16- Go to **Target > Connect J-Link**, when its connected go back to **Target** and choose **Erase All** (Be sure to erase the board every time you need to reprogram it).
- 17- Program and run your program by clicking the green arrow.
- 18- To make sure that your nRF9160 is connected to your AWS account, open TeraTerm. It should look like figure 5.67.

```

COM11 - Tera Term VT
File Edit Setup Control Window Help
The MQTT simple sample started
Deleting certs sec_tag: 16842753
nrf_inbuilt_key_delete(16842753, 0) => result=0
Deleting certs sec_tag: 16842753
nrf_inbuilt_key_delete(16842753, 1) => result=0
Deleting certs sec_tag: 16842753
nrf_inbuilt_key_delete(16842753, 2) => result=0
Deleting certs sec_tag: 16842753
nrf_inbuilt_key_delete(16842753, 3) => result=2
Deleting certs sec_tag: 16842753
nrf_inbuilt_key_delete(16842753, 4) => result=2
Write ca certs sec_tag: 16842753
Write private cert sec_tag: 16842753
Write public cert sec_tag: 16842753
LTE Link Connecting ...
LTE Link Connected!
IPv4 Address found 0x6441dd12
[mqtt_evt_handler:146] MQTT client connected!
Subscribing to: myTopic/subscribe len 17
Publish: Test
to topic: myTopic/publish len: 15
[mqtt_evt_handler:191] SUBACK packet id: 1234
[mqtt_evt_handler:181] PUBACK packet id: 4151

```

Fig 5.67: Tera Term

19- To test publishing and subscribing using AWS MQTT Client, go back to your **AWS account**, select **Activity>>MQTT Client**.

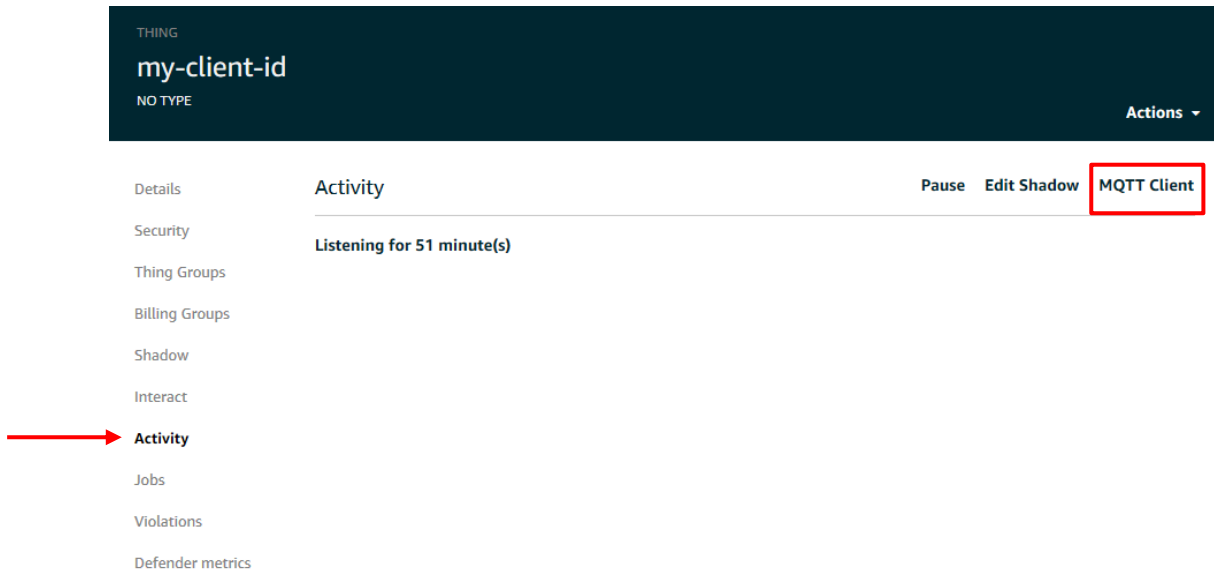


Fig 5.68: Activity

20- Choose a publish and subscribe topics based on your publish and subscribe topics in the **menuconfig (Segger Studio)**. **Remember** that you publish to a subscribe topic and you subscribe to a publish topic.

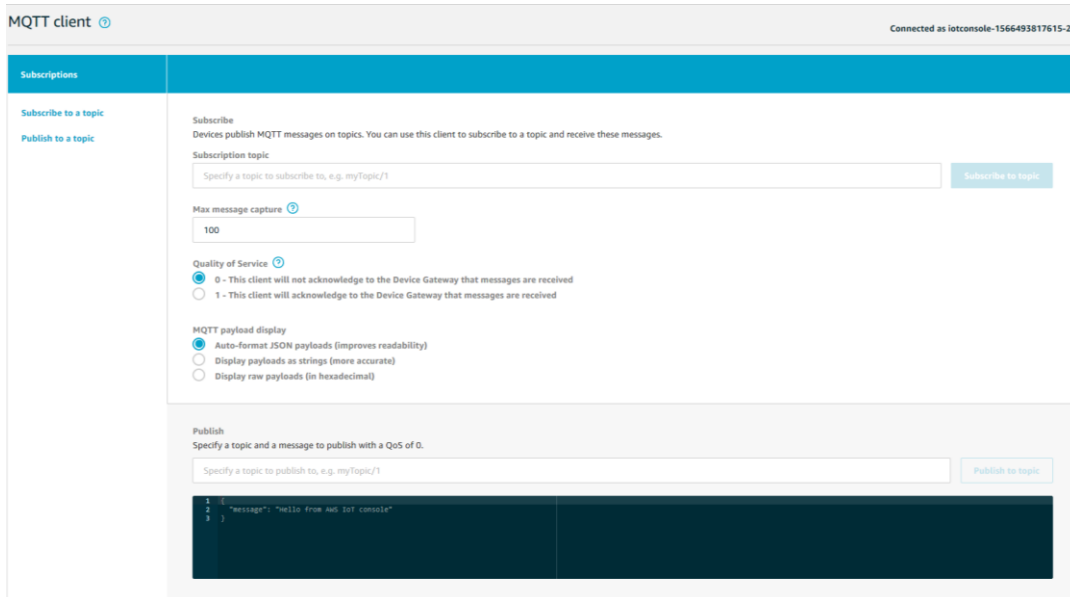


Fig 5.69: Subscribe and publish topics

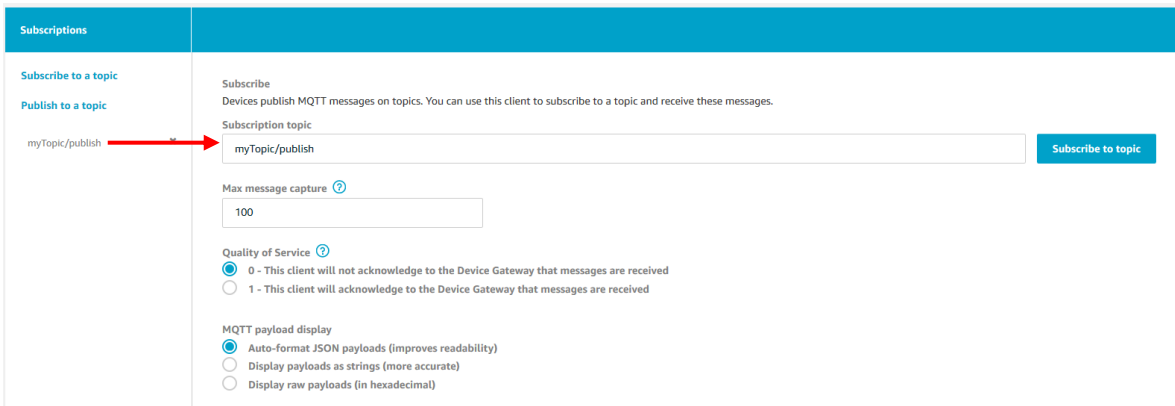


Fig 5.70: Subscription Topic

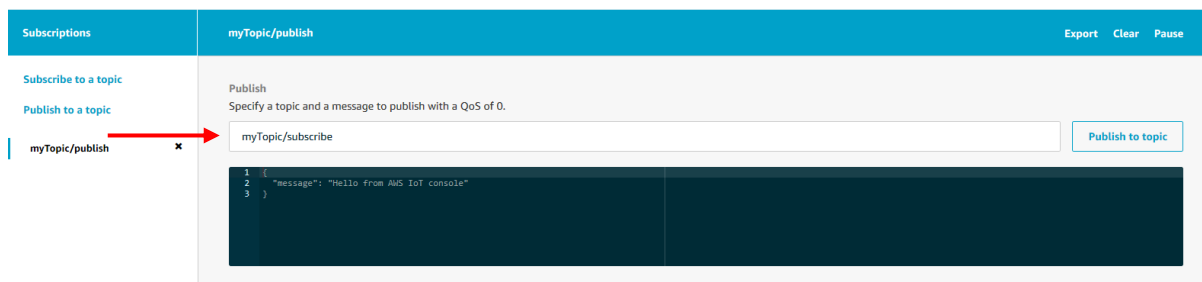


Fig 5.71: Publish to Topic

---

21- To Publish a Test message press on Button 2 on your nRF9160 and you will see the message published in your AWS account.

myTopic/publish Export Clear Pause

Publish  
Specify a topic and a message to publish with a QoS of 0.

myTopic/subscribe Publish to topic

```
1 {  
2   "message": "Hello from AWS IoT console"  
3 }
```

myTopic/publish Aug 27, 2019 9:41:11 AM -0400 Export Hide

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

Test

Fig 5.72: Test message

[Download the code here](#)