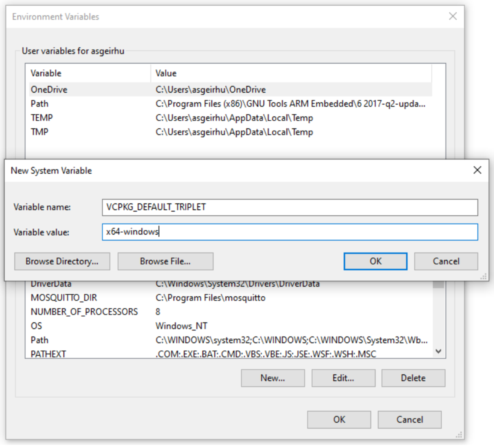
C++ server:

1. Download and install Visual Studio 19 for Windows, x64.
2. Download and run vcpkg (git bash was used in this example):
    - Git clone https://github.com/microsoft/vcpkg.git (in the C:\ folder was used in this example)
    - cd vcpkg
    - To configure vcpkg to install x64 packages, go into windows environmental variables and add a new system variable, VCPKG_DEFAULT_TRIPLET=x64-windows as following:



Environment Variables

User variables for asgeirhu

Variable	Value
OneDrive	C:\Users\asgeirhu\OneDrive
Path	C:\Program Files (x86)\GNU Tools ARM Embedded\6 2017-q2-upda...
TEMP	C:\Users\asgeirhu\AppData\Local\Temp
TMP	C:\Users\asgeirhu\AppData\Local\Temp

New System Variable

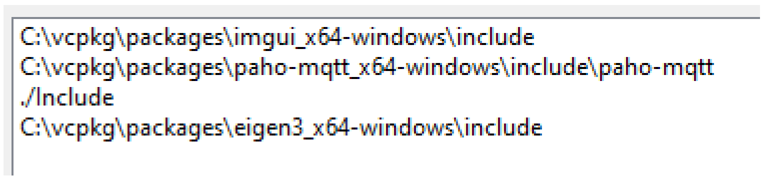
Variable name: VCPKG_DEFAULT_TRIPLET

Variable value: x64-windows

Variable	Value
DriverData	C:\Windows\System32\Drivers\DriverData
MOSQUITTO_DIR	C:\Program Files\mosquitto
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wb...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

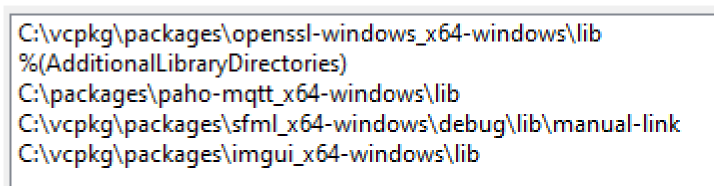
    - Now run ./bootstrap-vcpkg.bat (must be run every time a new library is to be installed)
    - Install the following packages, make sure it's the x64 versions being installed:
        - ./vcpkg install boost
        - ./vcpkg install sfml
        - ./vcpkg install imgui
        - ./vcpkg install paho-mqtt
        - ./vcpkg install thor
        - ./vcpkg install eigen3
        - /vcpkg install spdlog
    - Now run ./vcpkg integrate install
3. Open the project in visual studio.
   Configure settings for the project to find the packages:
   Make sure to use the correct path, if you install vcpkg in another folder than C:\
   In visual studio: Project → SLAM-application properties

- Configuration properties → C/C++ → General → Additional Include Directories



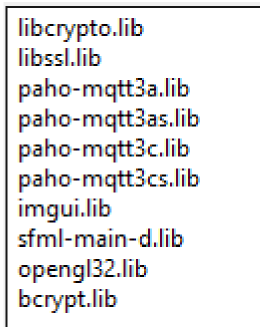
```
C:\vcpkg\packages\imgui_x64-windows\include
C:\vcpkg\packages\paho-mqtt_x64-windows\include\paho-mqtt
./Include
C:\vcpkg\packages\eigen3_x64-windows\include
```

- Configuration properties → Linker → General → Additional Library Directories



```
C:\vcpkg\packages\openssl-windows_x64-windows\lib
%(AdditionalLibraryDirectories)
C:\packages\paho-mqtt_x64-windows\lib
C:\vcpkg\packages\sfml_x64-windows\debug\lib\manual-link
C:\vcpkg\packages\imgui_x64-windows\lib
```

- Configuration properties → Linker → Input → Additional Dependencies



```
libcrypto.lib
libssl.lib
paho-mqtt3a.lib
paho-mqtt3as.lib
paho-mqtt3c.lib
paho-mqtt3cs.lib
imgui.lib
sfml-main-d.lib
opengl32.lib
bcrypt.lib
```

4. Now press debug → start without debugging and the program should start without any errors.

Testing the Server

Read ''Mullins'' for information about which parameters can be tunes and what effects it has on the server.

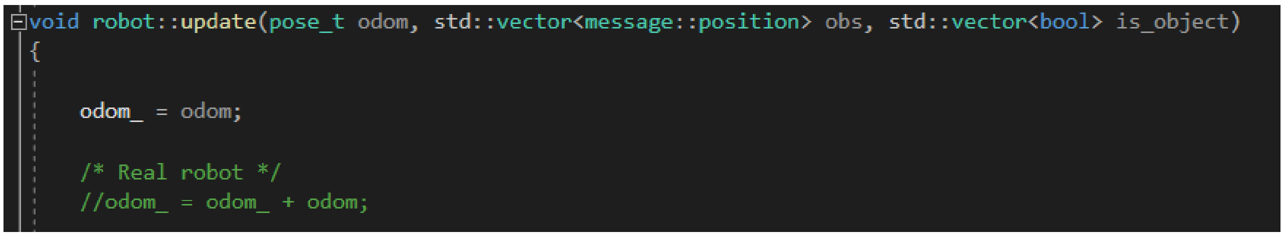
Testing the simulator

1. Start the server by pressing "Run" or CTRL+F5 in visual Studio. A terminal window and GUI should open
2. Press the "Robot simulation" panel and check off "Simulation On". A new window should appear
3. Press the new window, "New Robot connection", give an initial position to the robot, for example (0,0,0), and press "done". A blue robot should appear in the map.
4. Press the "clicks" column and choose "set target". The waypoint of the robot can now be set by right clicking.
5. Press the "Robots" column and choose "Set auto simulation path" to make the robot navigate the entire map.
6. To make the robot start running right click anywhere on the map to set its waypoint. The robot will first navigate the entire map, ending in the lower right corner, and then end up at the chosen waypoint.
7. The messages between the robot and server can be found in the MQTT column. The output from the server can be found in the console as well as in a log file in the log folder of the project. Every time the server runs, a new log file will be created. The log file will contain any message logged in the server, while the console window will not show the TRACE log messages. Read "Hunshamar" for more information about logging.

Testing with a real robot

For testing with real robots, first the thread network, with border router, must be set up as explained in the next tutorial.

1. For real robots one line of code must be changed. In "Robot.cpp" the update-functions odometry -updates must be added together. The reason being the messages from the robots are sending odometry-changes consecutively, followed by only 0 while it scans. The line is commented out in the code, so uncomment it.



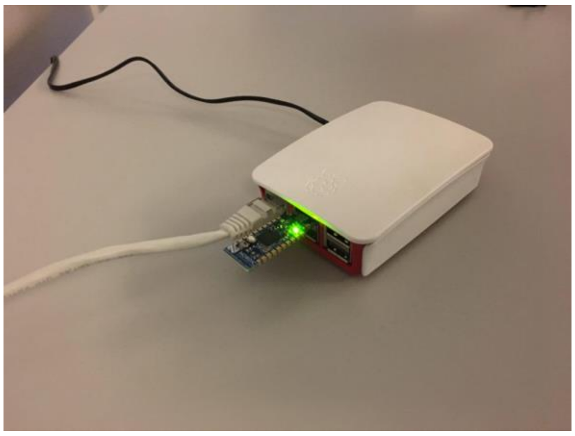
```
void robot::update(pose_t odom, std::vector<message::position> obs, std::vector<bool> is_object)
{


    odom_ = odom;

    /* Real robot */
    //odom_ = odom_ + odom;
```

2. Same procedure as with simulator, but without activating the simulation. "Auto simulation path" has no purpose here. Use right click or manual coordinates to control the server.

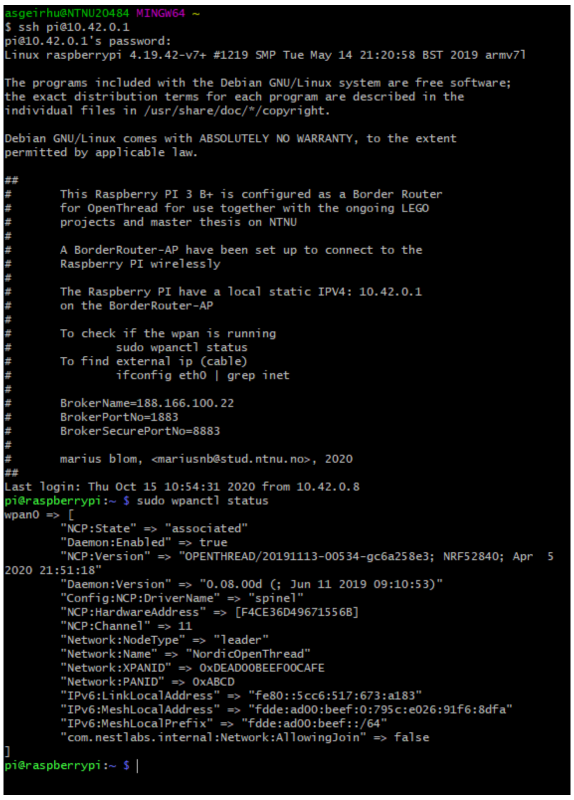
Setting up border router:

Following this guide:
https://infocenter.nordicsemi.com/index.jsp?topic=%2Fsdk_tz_v4.1.0%2Fthread_border_router.html

1. Connect the raspberry pi to power and ethernet and make sure the SD card with the correct software is inserted.
2. Connect the nrf52840 dongle to the raspberry Pi USB port as shown in the picture.
   The dongle should be flashed with the nrf52840_xxaa_mbr_pca_10059.hex file from the ncp example in the nRF5 SDK for Thread and Zigbee in the SDK/examples/thread/ncp/ftd/usb/hex folder.
   https://www.nordicsemi.com/Software-and-tools/Software/nRF5-SDK-for-Thread-and-Zigbee
   The dongle can be flashed using the nrf connect -> programmer software.
3. A green LED on the dongle should start blinking and then light up green as in the picture below. If not, there is probably an issue with the dongle. Try reflashing it.



4.
5. Connect the computer to the internet through ethernet or connect to the BorderRouter-AP network through wifi (SSID: BorderRouter-AP, password: 12345678). Open a terminal and type “SSH pi@10.42.1.0 “ and enter “legoproject” as password to access the linux terminal of the raspberry pi.
6. Enter “sudo wpanctl status”. Should look like this. If not, reflash the dongle or search online for a solution.

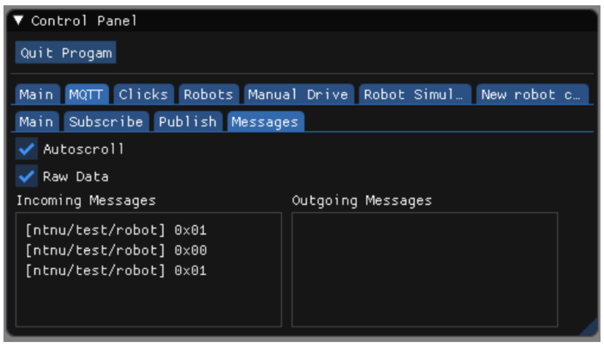


```
asgeirhu@NTNU20484 MINGW64 ~
$ ssh pi@10.42.0.1
pi@10.42.0.1's password:
Linux raspberrypi 4.19.42-v7+ #1219 SMP Tue May 14 21:20:58 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

##
#       This Raspberry PI 3 B+ is configured as a Border Router
#       for OpenThread for use together with the ongoing LEGO
#       projects and master thesis on NTNU
#
#       A BorderRouter-AP have been set up to connect to the
#       Raspberry PI wirelessly
#
#       The Raspberry PI have a local static IPV4: 10.42.0.1
#       on the BorderRouter-AP
#
#       To check if the wpan is running
#               sudo wpanctl status
#       To find external ip (cable)
#               ifconfig eth0 | grep inet
#
#       BrokerName=188.166.100.22
#       BrokerPortNo=1883
#       BrokerSecurePortNo=8883
#
#       marius blom, <mariusnb@stud.ntnu.no>, 2020
##
Last login: Thu Oct 15 10:54:31 2020 from 10.42.0.8
pi@raspberrypi:~ $ sudo wpanctl status
wpan0 => [
        "NCP:State" => "associated"
        "Daemon:Enabled" => true
        "NCP:Version" => "OPENTHREAD/20191113-00534-gc6a258e3; NRF52840; Apr  5
2020 21:51:18"
        "Daemon:Version" => "0.08.00d (; Jun 11 2019 09:10:53)"
        "Config:NCP:DriverName" => "spinel"
        "NCP:HardwareAddress" => [F4CE36D49671556B]
        "NCP:Channel" => 11
        "Network:NodeType" => "leader"
        "Network:Name" => "NordicOpenThread"
        "Network:XPANID" => 0xDEAD00BEEF00CAFE
        "Network:PANID" => 0xABCD
        "IPv6:LinkLocalAddress" => "fe80::5cc6:517:673:a183"
        "IPv6:MeshLocalAddress" => "fdde:ad00:beef:0:795c:e026:91f6:8dfa"
        "IPv6:MeshLocalPrefix" => "fdde:ad00:beef::/64"
        "com.nestlabs.internal:Network:AllowingJoin" => false
]
pi@raspberrypi:~ $ |
```

Test the network

1. To test the openthread network, flash a Nordic nrf52840 development board with the thread example code from nRF5 SDK for Thread and Zigbee as explained in the next tutorial. Either the mqttsn_client_publisher og mqttsn_client_subscriber example.
2. The board should successfully connect to the thread-network, indicated by a led on the board. In the publisher example the board will publish content by pressing a button on the board.

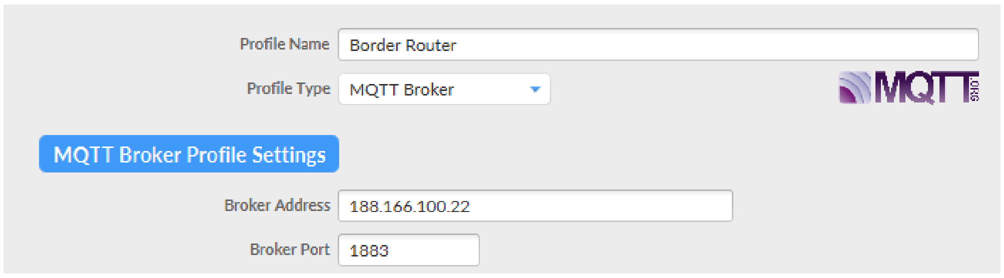
   If subscribing to the correct topic on the server, the data from the board should show up on the MQTT-tab on the server.

   ✓ Autoscroll
   ✓ Raw Data

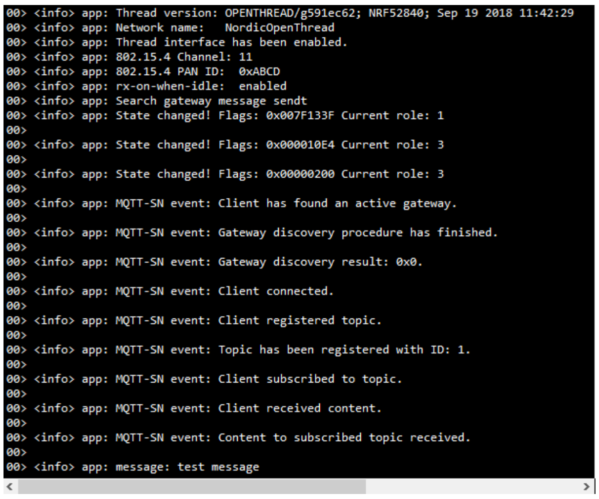
   Incoming Messages
   ```
   [ntnu/test/robot] 0x01
   [ntnu/test/robot] 0x00
   [ntnu/test/robot] 0x01
   ```
   Outgoing Messages

   If the server is not responding, test using a MQTT client on the PC. For example MQTT.fx. Connect with the following credentials

Profile Name	Border Router
Profile Type	MQTT Broker
Broker Address	188.166.100.22
Broker Port	1883

3. You can read the logging from the program by using J-link RTT viewer

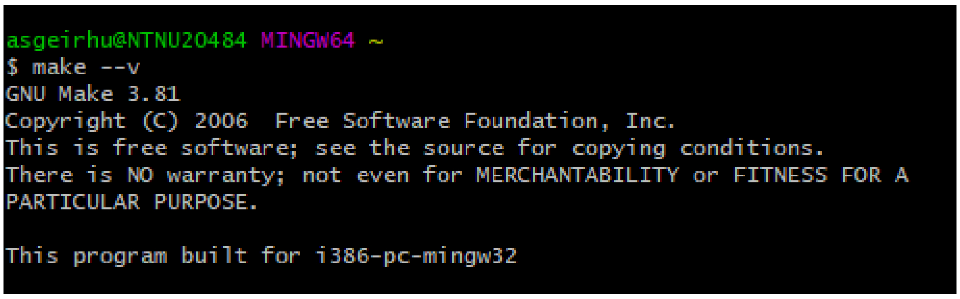
   

```
00> <info> app: Thread version: OPENTHREAD/g591ec62; NRF52840; Sep 19 2018 11:42:29
00> <info> app: Network name:   NordicOpenThread
00> <info> app: Thread interface has been enabled.
00> <info> app: 802.15.4 Channel: 11
00> <info> app: 802.15.4 PAN ID:  0xABCD
00> <info> app: rx-on-when-idle:  enabled
00> <info> app: Search gateway message sendt
00> <info> app: State changed! Flags: 0x007F133F Current role: 1
00>
00> <info> app: State changed! Flags: 0x000010E4 Current role: 3
00>
00> <info> app: State changed! Flags: 0x00000200 Current role: 3
00>
00> <info> app: MQTT-SN event: Client has found an active gateway.
00>
00> <info> app: MQTT-SN event: Gateway discovery procedure has finished.
00>
00> <info> app: MQTT-SN event: Gateway discovery result: 0x0.
00>
00> <info> app: MQTT-SN event: Client connected.
00>
00> <info> app: MQTT-SN event: Client registered topic.
00>
00> <info> app: MQTT-SN event: Topic has been registered with ID: 1.
00>
00> <info> app: MQTT-SN event: Client subscribed to topic.
00>
00> <info> app: MQTT-SN event: Client received content.
00>
00> <info> app: MQTT-SN event: Content to subscribed topic received.
00>
00> <info> app: message: test message
```

Flashing code from the Nordic nRF5 SDK to the board using make

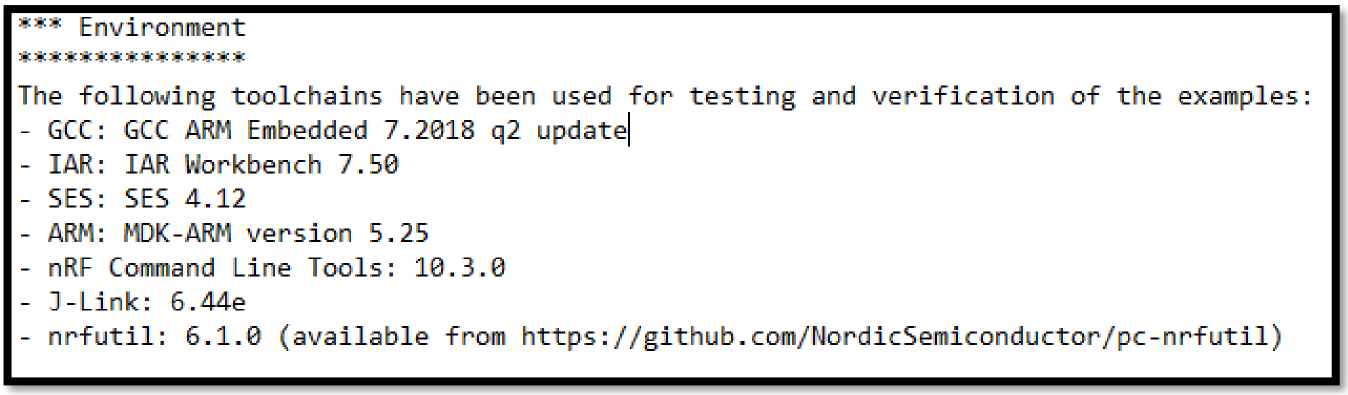
This tutorial will show how to flash over thread example code from the Nordic SDK to nrf52840 DK using GNU Make and Nordic toolchain. You can either use windows built in CMD terminal, git bash or any other terminal.

1. Install a text editor, for example VS Code.
2. Download the Nordic SDK for Thread and Zigbee https://www.nordicsemi.com/Software-and-tools/Software/nRF5-SDK-for-Thread-and-Zigbee/Download#infotabs and place the folder anywhere, for example the desktop.
3. Install GNU Make for windows http://gnuwin32.sourceforge.net/packages/make.htm
   Download the Setup with complete package. Follow the setup. After the setup is add the folder with the make.exe to the path environmental variables of windows so it can be run from the terminal.
   a. Go to environmental variables
   b. Click path in "User variables for <user>" and press edit
   c. Press "New" and "Browse"
   d. Find the folder with make.exe, should be in C:\Program Files (x86)\GnuWin32\bin and add the folder.
   e. Press OK in both windows.
   f. Check if Make is working by running "make –v" in any terminal



```
asgeirhu@NTNU20484 MINGW64 ~
$ make --v
GNU Make 3.81
Copyright (C) 2006  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

This program built for i386-pc-mingw32
```

4. Install GNU Arm embedded Toolchain.
   https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads
   To find the correct version check the release_notes.txt inside the documentation folder of the SDK. Scroll down to "Environment" and find the correct version of GCC



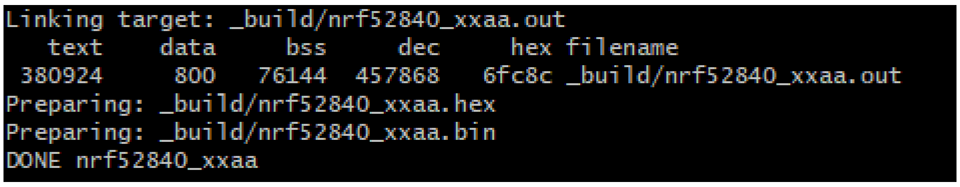
```
*** Environment
***************
The following toolchains have been used for testing and verification of the examples:
- GCC: GCC ARM Embedded 7.2018 q2 update
- IAR: IAR Workbench 7.50
- SES: SES 4.12
- ARM: MDK-ARM version 5.25
- nRF Command Line Tools: 10.3.0
- J-Link: 6.44e
- nrfutil: 6.1.0 (available from https://github.com/NordicSemiconductor/pc-nrfutil)
```

   If you want to use another version of GCC, change the Makefile.windows file in the SDK/components/toolchain/gcc folder.

   After installing the toolchain, make sure to check off "Add path to environmental variable" before finishing. If not, this must be done manually.
5. To flash the code over, nrf command line tools must be installed.
   https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Command-Line-Tools/Download
   After installing, the bin folder in the install path must me manually added to environmental variables to be

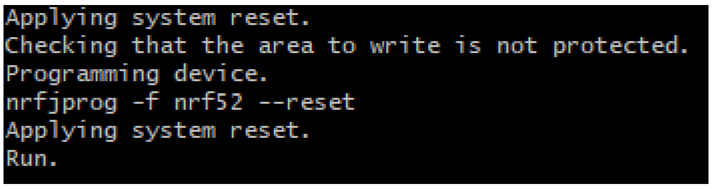
run from the command line. Add the folder **C:\Program Files (x86)\Nordic Semiconductor\nrf-command-line-tools\bin** to the environmental variable in the same manner as you did with GNU Make.

6. Now the code should be ready to be compiled and flashed using make.
   Open the mqttsn_client_publisher example in the **examples/thread** folder in the SDK. The NRF52840 DK uses the PCA10056 folder. This example uses no softdevices, so inside the **blank** folder open the **armgcc** folder, which should contain a makefile. Open a terminal inside the folder by either writing “cmd” in the address field of the folder, or right clicking the folder and press “Git Bash Here” if you have Git Bash terminal installed.
7. Write “Make” in the terminal and the program should be compiled successfully. This can take a while the first time.



```
Linking target: _build/nrf52840_xxaa.out
   text    data     bss     dec     hex filename
 380924     800   76144  457868   6fc8c _build/nrf52840_xxaa.out
Preparing: _build/nrf52840_xxaa.hex
Preparing: _build/nrf52840_xxaa.bin
DONE nrf52840_xxaa
```

8. Make sure the NRf52840 DK is connected with USB to the computer and write “make flash” in the terminal to flash the code over.



```
Applying system reset.
Checking that the area to write is not protected.
Programming device.
nrfjprog -f nrf52 --reset
Applying system reset.
Run.
```

9. Now the code should be running successfully on the board.
10. For development, you can change the code in the main.c file in the example. For using other libraries etc. changes must be made to the makefile.

Programming nRF5 SDK using SEGGER Embedded Studio

1. Install Segger Embedded Studio
   https://www.segger.com/products/development-tools/embedded-studio/
2. Open the EMPROJECT File in the **example/pca10056/blank/ses** folder in the SDK in segger embedded studio.
3. For more information, check the Nordic youtube tutorial on segger embedded studio
   https://www.youtube.com/watch?v=YZouRE_Ol8g&list=PLx_tBuQ_KSqGHmzdEL2GWEOeix-S5rgTV