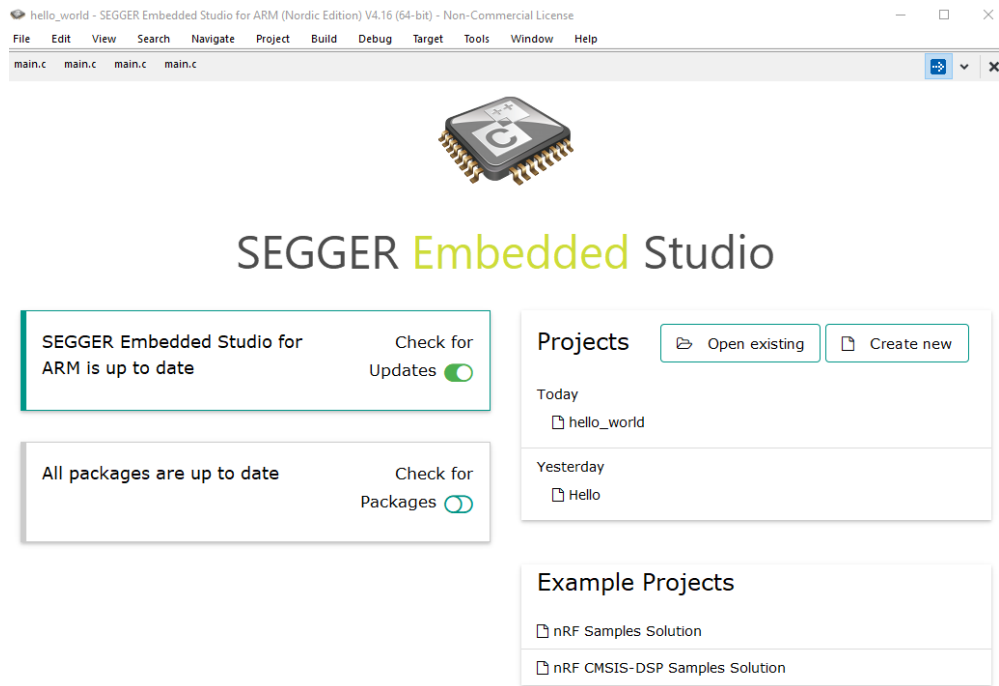


MQTT Simple Sample:

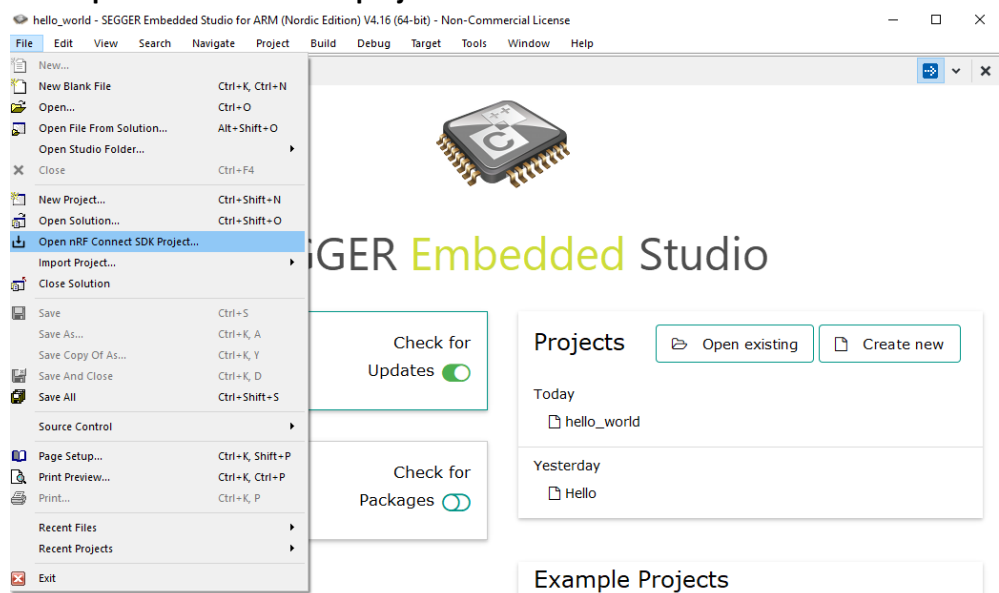
The MQTT simple sample is one of the Nordic specific examples. To open and test this sample using Segger Studio follow the directions.

1- Open SEGGER Embedded Studio.



Segger Embedded Studio Dashboard

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



Open nRF connect SDK Project

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



nRF Connect Options

CMakeLists.txt

C:/ncs/nrf/samples/nrf9160/mqtt_simple/CMakeLists.txt



Board Directory

C:/ncs/zephyr/boards/arm/nrf9160_pca10090



Board Name

nrf9160_pca10090ns



Build Directory

C:/ncs/nrf/samples/nrf9160/mqtt_simple/build_nrf9160_pca10090ns



Clean Build Directory

OK

Cancel

nRF Connect Options

Project Explorer

- Common
- Project Items
- Solution 'mqtt_simple'
 - dts files 4 files, modified options
 - Project 'zephyr/merged.hex'
 - Solution 'build'
 - Project 'all'
 - Project 'app/libapp.a'
 - C_COMPILER_app 1 [1.0K] [1.4K]
 - main.c 1.0K -0.0K
 - Output Files
 - Project 'libspmsecureintrin' 28.4K 13.1K
 - Project 'spm/spm_app/libs'
 - Project 'spm/zephyr/arch/a'
 - Project 'spm/zephyr/arch/a'
 - Project 'spm/zephyr/arch/a'
 - Project 'spm/zephyr/driver'
 - Project 'spm/zephyr/kernel'
 - Project 'spm/zephyr/lib/lib'
 - Project 'spm/zephyr/lib/spn'
 - Project 'spm/zephyr/modu'
 - Project 'spm/zephyr/modu'
 - Project 'spm/zephyr/spm_z' 28.4K 13.1K
 - Project 'zephyr/arch/arm/c'
 - Project 'zephyr/arch/arm/c'
 - Project 'zephyr/arch/arm/c'
 - Project 'zephyr/drivers/gpi'
 - Project 'zephyr/drivers/seri'
 - Project 'zephyr/kernel/libk'
 - Project 'zephyr/lib/lib/mir'
 - Project 'zephyr/libzephyr.a'
 - Project 'zephyr/modules/n'
 - Project 'zephyr/modules/n'
 - Project 'zephyr/modules/n'
 - Project 'zephyr/subsys/net'
 - Project 'zephyr/subsys/net'
 - Project 'zephyr/subsys/net'
 - Project 'zephyr/zephyr.elf' 87.3K 40.2K
 - Project 'zephyr/zephyr_pre' 87.3K 40.3K

main.c

```

/*
 * Copyright (c) 2018 Nordic Semiconductor ASA
 *
 * SPDX-License-Identifier: LicenseRef-BSD-5-Clause-Nordic
 */

#include <zephyr.h>
#include <stdio.h>
#include <uart.h>
#include <string.h>

#include <net/mqtt.h>
#include <net/socket.h>
#include <lte_lc.h>

/* Buffers for MQTT client. */
static u8_t rx_buffer[CONFIG_MQTT_MESSAGE_BUFFER_SIZE];
static u8_t tx_buffer[CONFIG_MQTT_MESSAGE_BUFFER_SIZE];
static u8_t payload_buf[CONFIG_MQTT_PAYLOAD_BUFFER_SIZE];

/* The mqtt client struct */
static struct mqtt_client client;

/* MQTT Broker details. */
static struct sockaddr_storage broker;

/* Connected flag */
static bool connected;

/* File descriptor */
static struct pollfd fds;

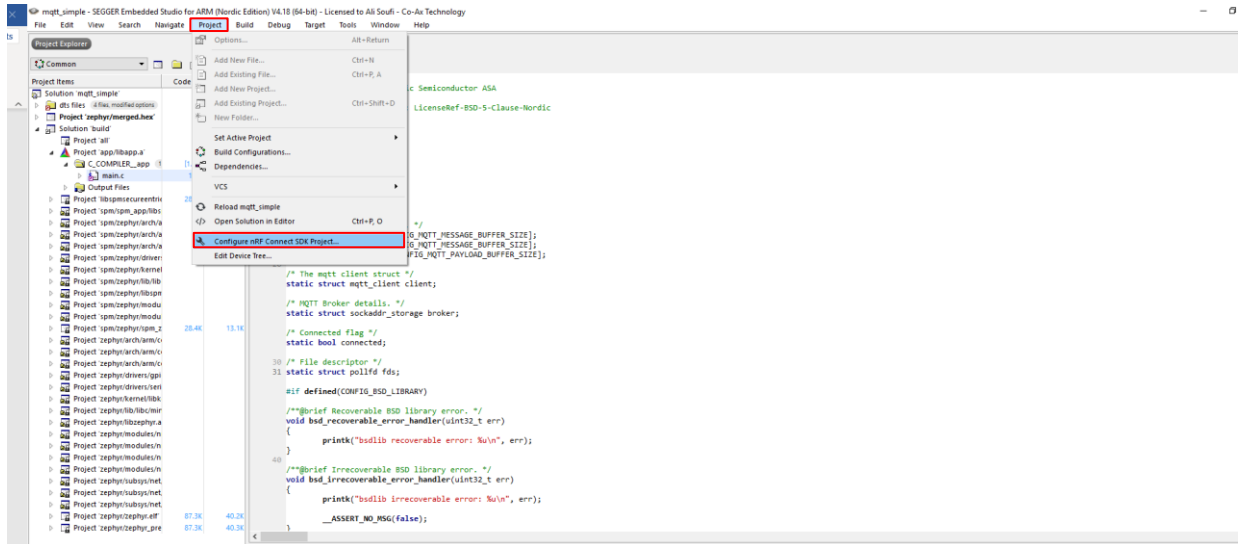
#if defined(CONFIG_BSD_LIBRARY)
/**@brief Recoverable BSD library error. */
void bsd_recoverable_error_handler(uint32_t err)
{
    printk("bsdlib recoverable error: %u\n", err);
}

/**@brief Irrecoverable BSD library error. */
void bsd_irrecoverable_error_handler(uint32_t err)
{
    printk("bsdlib irrecoverable error: %u\n", err);
}

__ASSERT_NO_MSG(false);
    
```

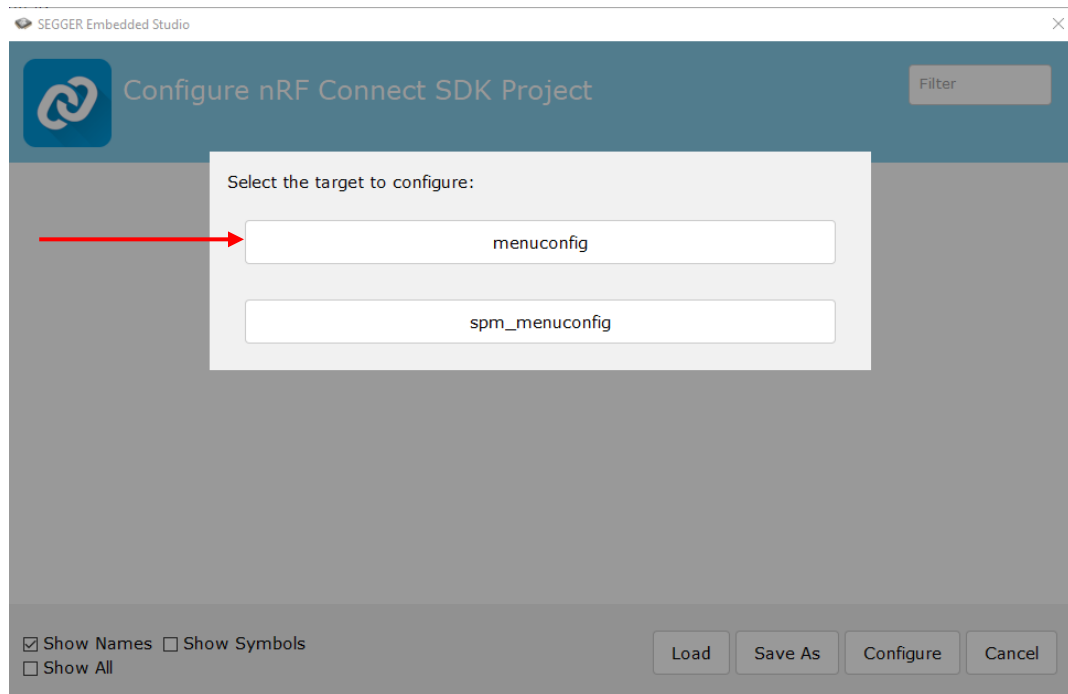
MQTT Simple Sample

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



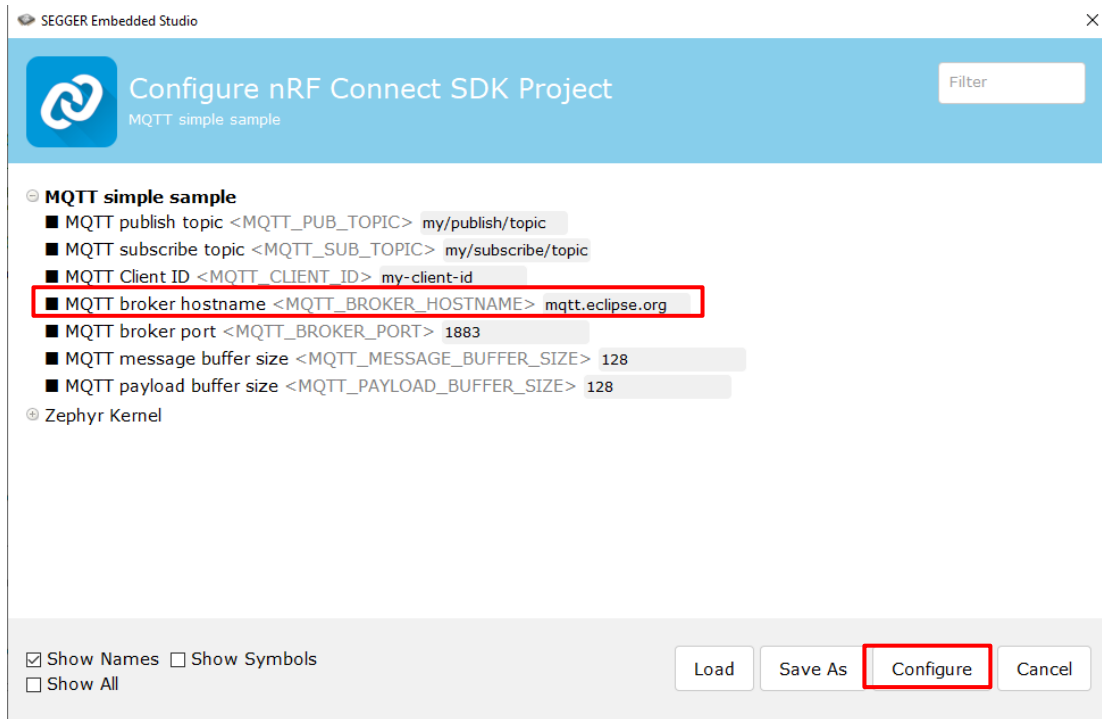
Configure nRF Connect SDK Project

- 5- Choose **menuconfig**



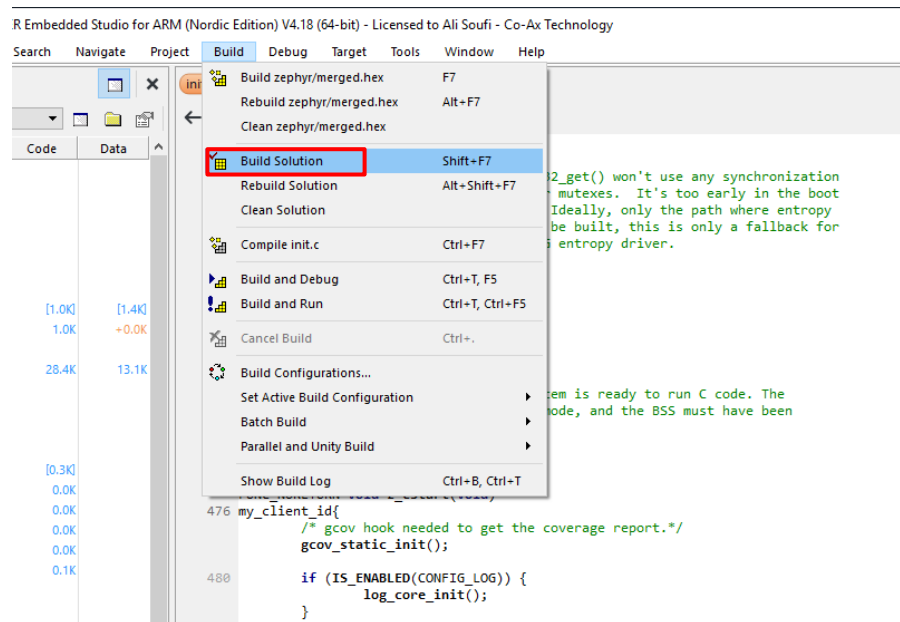
menuconfig

- 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**.



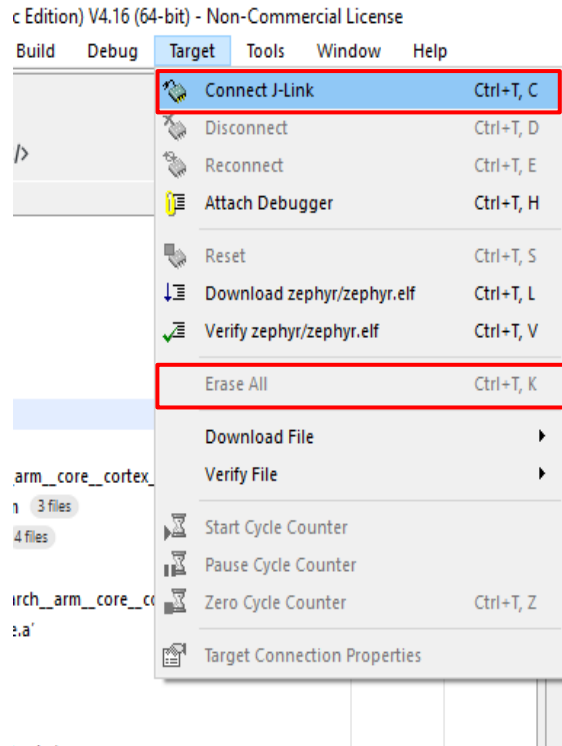
menuconfig

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



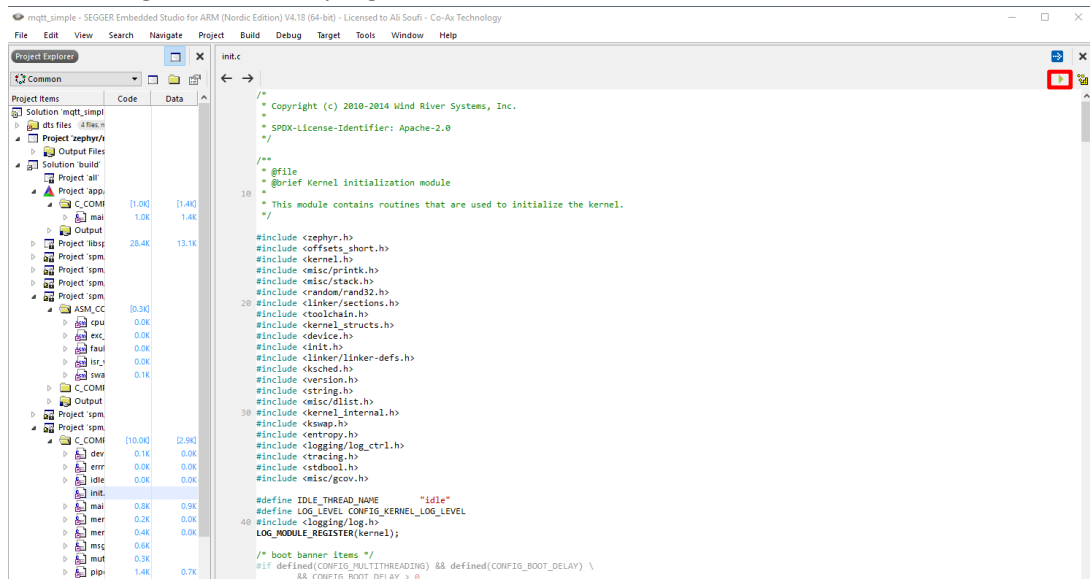
Build Solution

8- 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).



Connect J-Link


9- Click on the green arrow to program.




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**.

 This product is installed.
 [Launch](#) ⋮



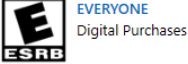
MQTTBox

workswithweb · [Developer tools > Utilities](#)

★★★★★ 4 [Share](#)

Developers helper program to create, develop and test MQTT connectivity protocol.
MQTTBox enables you to create MQTT clients to publish or subscribe to

[More](#)



Free

Get

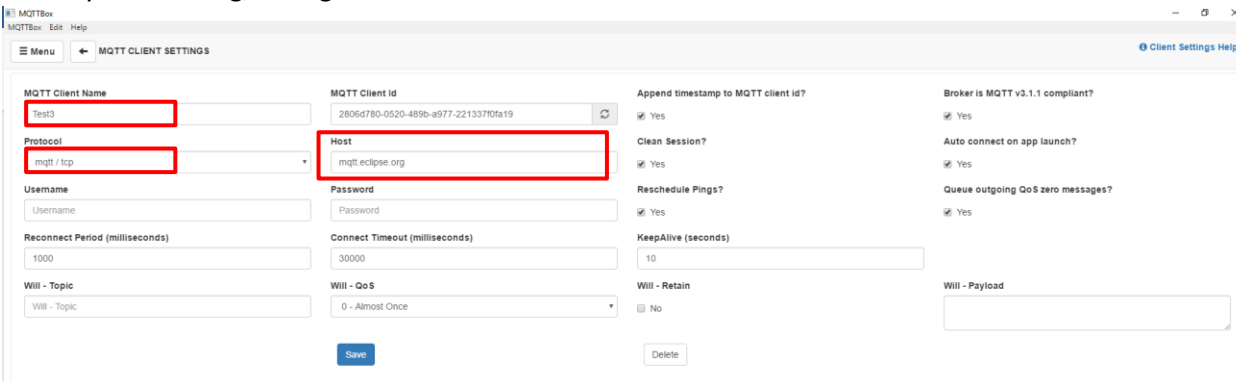
Add to cart

[Wish list](#)

+ Offers in-app purchases

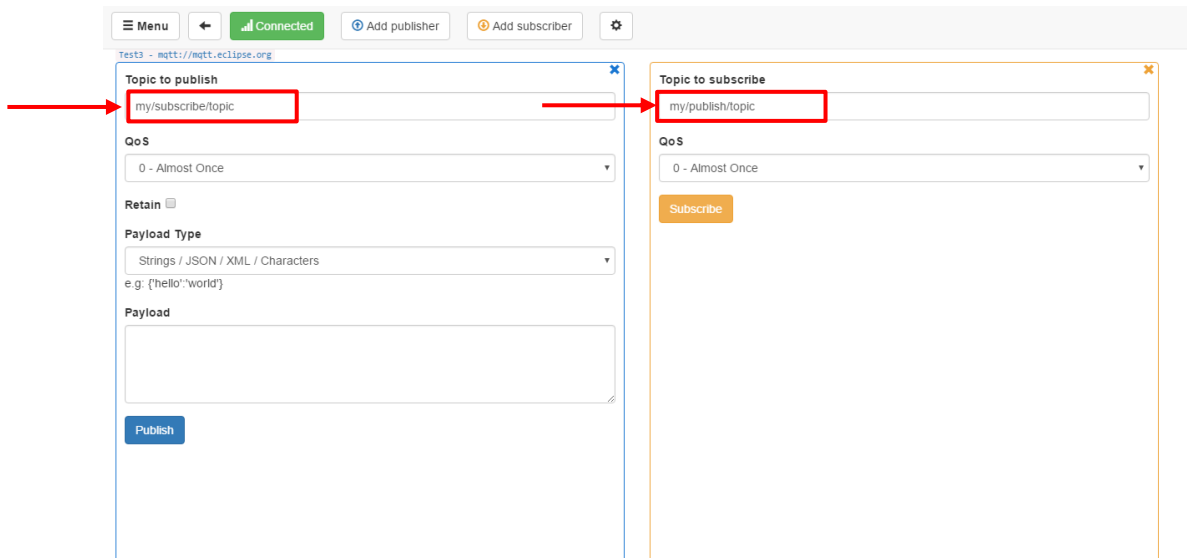
MQTT Box App

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**.



MQTT Box App Settings

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



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;

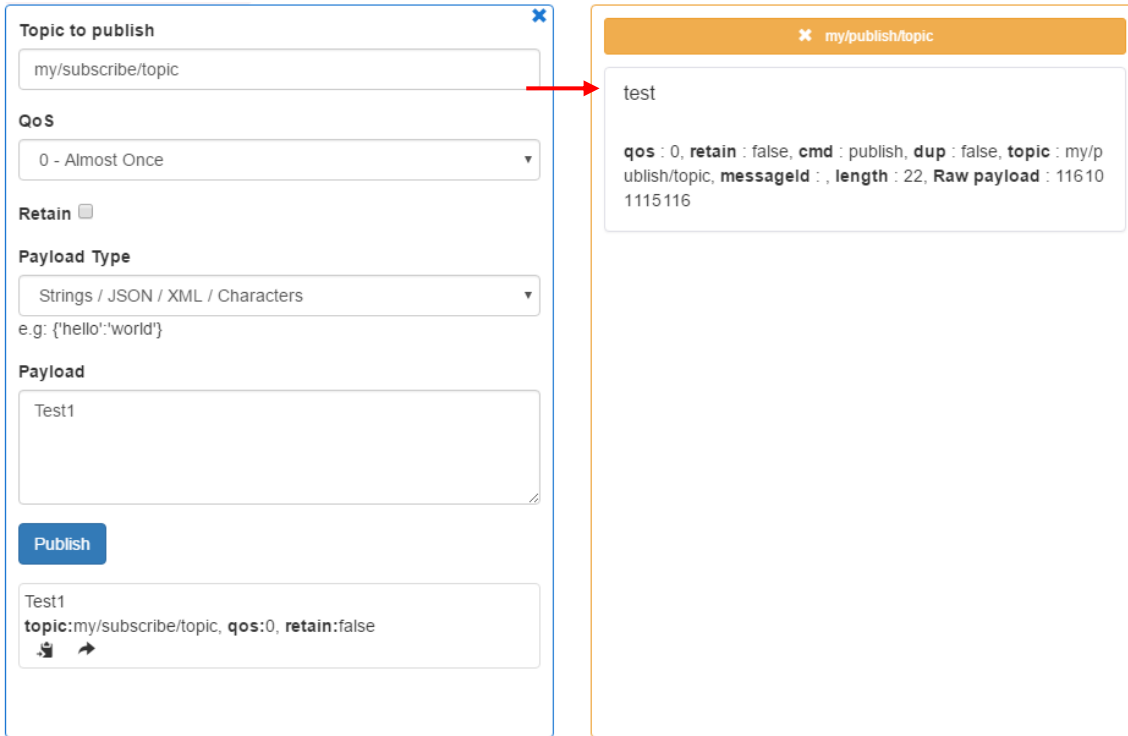
    switch (evt->type) {
    case MQTT_EVT_CONNACK:
        if (evt->result != 0) {
            printk("MQTT connect failed %d\n", evt->result);
            break;
        }

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

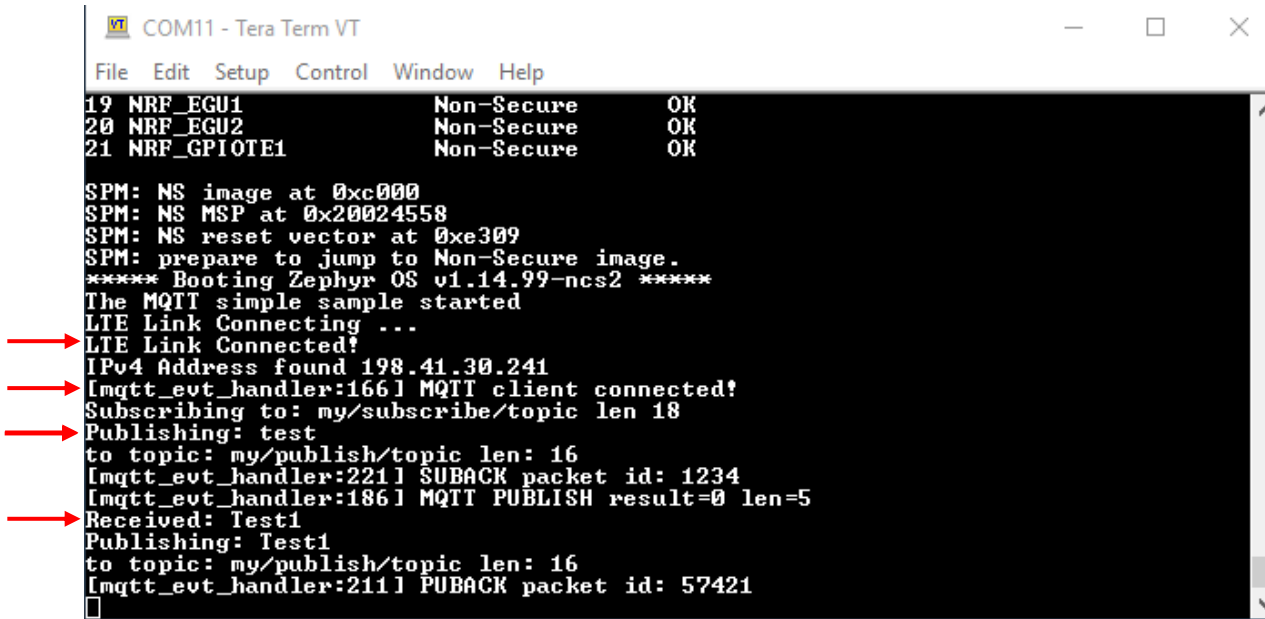
        data_publish(&client, MQTT_QOS_1_AT_LEAST_ONCE, "test", strlen("test"));
    }

    break;
}
```

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



MQTT Client Settings



Tera Term