SCM1612
Wi-Fi 6 and BLE 5 Low-Power SoC

MQTT Development Guide

Revision 1.0
Date 2024-09-02

Contact Information
Senscomm Semiconductor (www.senscomm.com)
Room 303, International Building, West 2 Suzhou Avenue,
SIP, Suzhou, China
For sales or technical support, please send email to
info@senscomm.com

Disclaimer and Notice

This document is provided on an “as-is” basis only. Senscomm reserves the right to make corrections, improvements and other changes to it or any specification contained herein without further notice.

All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

All third party’s information in this document is provided as is with NO warranties to its authenticity and accuracy.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners and are hereby acknowledged.

Version History

Version	Date	Description
1.0	2024-09-02	Updated build and CLI commands
0.1	2024-03-11	Initial draft

Table of Contents

1 Development Guide

This guide provides instructions for implementing applications that require MQTT client functionality using the SCM1612 SDK.

1.1 Overview

The SCM1612 SDK integrates the coreMQTT-Agent and underlying coreMQTT library for handling MQTT communication.

- API location:
  - `lib/mqtt/coreMQTT-Agent`
  - `lib/mqtt/coreMQTT`
- Demo location
  - `api/examples/protocols/mqtt`

1.2 Demo Instructions

Follow these steps to run the MQTT demo on the SCM1612 platform.

1.2.1 Set Up Build Configuration

1. Select the MQTT demo as the main application:
   $ make scm1612s_defconfig
   $ make menuconfig
2. Navigate to:
   `Applications -> Protocols Demo`
3. Select
   `Protocols Demo -> MQTT Demo`
4. Exit and save the configuration.

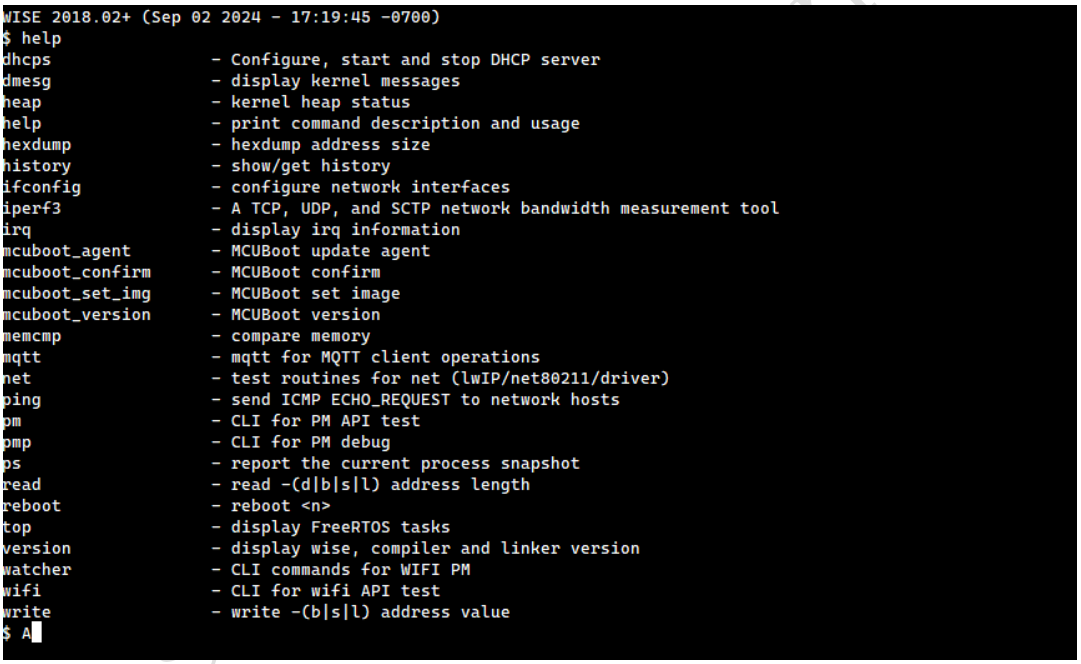
1.2.2 Set Up Wi-Fi Parameters

1. Open the configuration menu:
   $ make menuconfig
2. Navigate to
   `Applications -> Common -> include WI-FI Configuration`
3. Enter the required Wi-Fi parameters under:
   `DEMO WI-FI Configuration`
   (Use the Help menu for each item if needed.)
4. Exit and save the configuration.

1.2.3 Configure MQTT Client Parameters:

1. Open the configuration menu again:
   $ make menuconfig
2. Navigate to:
   `Applications -> MQTT demo`
3. Modify the MQTT client parameters as necessary.

4. Build the firmware image:wise-mcuboot.bin.
   $ make
5. Refer to the `SDK_Getting_Started_Guide` for instructions on how to download the generated wise-mcuboot.bin image and run it on the SCM1612 EVK.



1.2.4 Running the Demo

To run the MQTT demo, you'll need a separate MQTT client that will interact with the same MQTT server, test.mosquitto.org, either as a subscriber or publisher. For this example, the PC version of Eclipse mosquito has been used.

- The demo allows the use of **CLI commands** for interactive testing with the MQTT client.

```
WISE 2018.02+ (Sep 02 2024 - 17:19:45 -0700)
$ help
dhcps                - Configure, start and stop DHCP server
dmesg                - display kernel messages
heap                 - kernel heap status
help                 - print command description and usage
hexdump              - hexdump address size
history              - show/get history
ifconfig             - configure network interfaces
iperf3               - A TCP, UDP, and SCTP network bandwidth measurement tool
irq                  - display irq information
mcuboot_agent        - MCUBoot update agent
mcuboot_confirm      - MCUBoot confirm
mcuboot_set_img      - MCUBoot set image
mcuboot_version      - MCUBoot version
memcmp               - compare memory
mqtt                 - mqtt for MQTT client operations
net                  - test routines for net (lwIP/net80211/driver)
ping                 - send ICMP ECHO_REQUEST to network hosts
pm                   - CLI for PM API test
pmp                  - CLI for PM debug
ps                   - report the current process snapshot
read                 - read -(d|b|s|l) address length
reboot               - reboot <n>
top                  - display FreeRTOS tasks
version              - display wise, compiler and linker version
watcher              - CLI commands for WIFI PM
wifi                 - CLI for wifi API test
write                - write -(b|s|l) address value
$
$ mqtt
Usage: mqtt init url port secure(0|1) <ca_file> <client_cert_file> <client_key_file>
  or:  mqtt sub topic <qos(0|1|2)>
  or:  mqtt unsub topic
  or:  mqtt pub topic payload <qos(0|1|2)>
  or:  mqtt ping
$
$
```

1.2.5 Initialize an MQTT Client

The MQTT client can be initialized and started using the mqtt init CLI command. The command requires several parameters as described below:

Parameter	Value	(M)andatory / (O)ptional	Example
url	MQTT server's URL	M	test.mosquitto.org
port	Port number of the MQTT server	M	1883 (plaintext) 8883 (encrypted, unauthenticated) 8884 (encrypted, authenticated)
secure	0: plaintext TCP transport 1: TLS transport	M	

ca_file	Full path to CA certificate file	M if secure is 1	/path/to/ca.crt
client_cert_file	Full path to client certificate file	O if secure is 1	/path/to/client.crt
client_key_file	Full path to client key file	O if secure is 1	/path/to/client.key

1.2.5.1 Using plaintext TCP Transport

Below is an example of CLI command to initialize and start a MQTT client which will connect to a configured MQTT server upon plaintext TCP transport.

The Wi-Fi parameters should already be configured during the build process. When the mqtt init command runs, the SCM1612 device will automatically connect to the configured AP.

```
$ mqtt
Usage: mqtt init url port secure(0|1) <ca_file> <client_cert_file> <client_key_file>
  or:  mqtt sub topic <qos(0|1|2)>
  or:  mqtt unsub topic
  or:  mqtt pub topic payload <qos(0|1|2)>
  or:  mqtt ping
$
$ mqtt init test.mosquitto.org 1883 0
$
$ p
WIFI CONNECTED
I (99521) SCM_API: AP SSID: Xiaohu_ASUS
I (99522) SCM_API: AP BSSID: 50:eb:f8:19:88:a0
I (99522) SCM_API: AP CH: 11
I (99523) SCM_API: AP RSSI: -29
I (99524) SCM_API: AP Country : AA
I (99525) SCM_API: Status: CONNECTED

WIFI GOT IP
s
 PID   PR  STWM  S  %CPU+      TIME+   TASK
   1    3   532  X    0.0    0:00:01 init                          (0x21ce60-0x21de50, 0x21d91c)
  10    3   934  R    0.0    0:00:00 mqtt-agent                    (0x226410-0x227c00, 0x22787c)
   2    0   965  R    6.3    0:01:38 idle                          (0x2107d8-0x2117d0, 0x2116ec)
   4    3   459  B    0.0    0:00:00 knetd                         (0x21e210-0x21ee00, 0x21ecac)
   3    5   192  B   93.5    0:24:05 ksofttimerd                   (0x21034c-0x210740, 0x21066c)
   7    3   279  B    0.0    0:00:00 rt_msg                        (0x222870-0x222e60, 0x222d3c)
  11    3   528  B    0.0    0:00:00 wpa_supplicant                (0x227e40-0x229230, 0x228fcc)
   9    3   260  B    0.0    0:00:00 wise_event_loop_task          (0x225450-0x226040, 0x225efc)
   5    3   231  B    0.0    0:00:00 scm2020-wlan fast taskq       (0x220910-0x221300, 0x2211fc)
   8    7    48  B    0.0    0:00:00 ll                            (0x223470-0x2236e0, 0x2235dc)
   6    3   207  B    0.0    0:00:00 knet80211d/wlan0              (0x221f90-0x222580, 0x22247c)
$
```

1.2.5.2 Using TLS with Encryption Only

To connect via TLS, an appropriate CA certificate must be stored in the file system. For example, the SCM1612 MQTT client can connect to

test.mosquitto.org on port 8883 using the CA certificate (mosquitto.org.crt).

mosquitto

MQTT

This is test.mosquitto.org. It hosts a publicly available Eclipse Mosquitto MQTT server/broker. MQTT is a very lightweight protocol that uses a publish/subscribe model. This makes it suitable for "machine to machine" messaging such as with low power sensors or mobile devices.

For more information on MQTT, see http://mqtt.org/ or the Mosquitto MQTT man page.

If you are interested in your own hosted instance of Mosquitto you should look at the Cedalo offering. Cedalo are the company that sponsor the main development of Mosquitto.

The server

The server listens on the following ports:

- 1883 : MQTT, unencrypted, unauthenticated
- 1884 : MQTT, unencrypted, authenticated
- 8883 : MQTT, encrypted, unauthenticated
- 8884 : MQTT, encrypted, client certificate required
- 8885 : MQTT, encrypted, authenticated
- 8886 : MQTT, encrypted, unauthenticated
- 8887 : MQTT, encrypted, server certificate deliberately expired
- 8080 : MQTT over WebSockets, unencrypted, unauthenticated
- 8081 : MQTT over WebSockets, encrypted, unauthenticated
- 8090 : MQTT over WebSockets, unencrypted, authenticated
- 8091 : MQTT over WebSockets, encrypted, authenticated

The encrypted ports support TLS v1.3, v1.2 or v1.1 with x509 certificates and require client support to connect. For ports 8883 and 8884 you should use the certificate authority file (mosquitto.org.crt (PEM format), or mosquitto.org.der (DER format)) to verify the server connection. Ports 8081 and 8886 have a Lets Encrypt certificate, so you should use your system CA certificates or the appropriate Lets Encrypt CA certificate for verification.

Port 8884 requires clients to provide a certificate to authenticate

You are free to use it for any application, but please do not abuse or rely upon it for anything of importance. This server runs on an Intel Atom N2800, and as such is a low power device. It is not intended to demonstrate any performance characteristics.

You should also build your client to cope with the broker restarting.

If you have the mosquitto clients installed try:

- mosquitto_sub -h test.mosquitto.org -t "#" -u wildcard -v

Please don't publish anything sensitive, anybody could be listening.

Caveats

This server is provided as a service for the community to do testing, but it is also extremely useful for testing the server. This means that it will often be running unreleased or experimental code and may not be as stable as you might hope. It may also be slow - the broker often runs under valgrind or perf. Finally, not all of the features may be available all of the time, depending on what testing is being done. In particular, websockets and TLS support are the most likely to be unavailable.

In general you can expect the server to be up and to be stable though.

Get in touch

Come and discuss the Mosquitto project on Slack (go to the Mosquitto channel).

If you do publish things to this server on a regular basis, please get in touch to satisfy my curiosity - there are lots of topics that look interesting but I know nothing about. I'm **ral** on the libera.chat #mqtt irc channel, or see the mosquitto source for contact details..

Examples using this service

- Websockets $SYS tree for test.mosquitto.org
- Websockets $SYS tree for test.mosquitto.org (TLS)

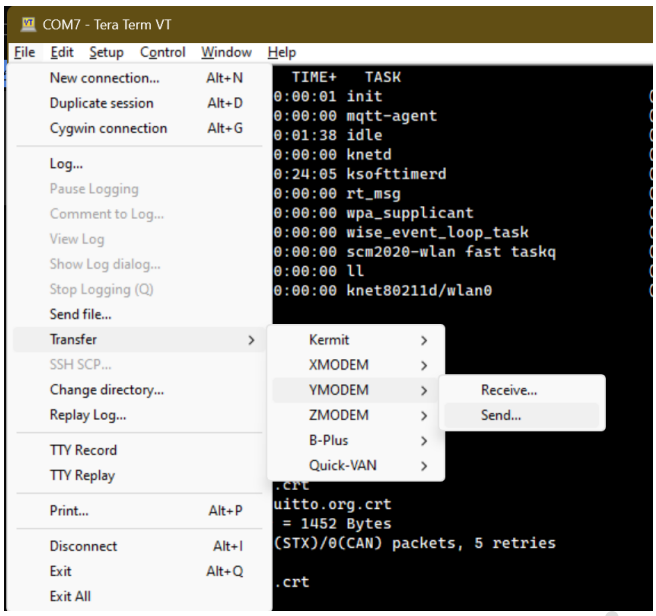
Steps:

1. Download the CA certificate from test.mosquitto.org to your PC.
2. Use the fs load CLI command to load the certificate file onto the target device's file system.

```
$ help fs
Usage: fs load <filename>
  or:  fs read <filename>
  or:  fs write <filename> <content>
  or:  fs rm <filename>
  or:  fs size <filename>
CLI for scm_fs operations
$
$ fs load /mqtt/mosquitto.org.crt
load local file to /mqtt/mosquitto.org.crt
CCC## Total Size = 0x000005ac = 1452 Bytes
xyzModem - CRC mode, 1(SOH)/2(STX)/0(CAN) packets, 5 retries
$
$ fs read /mqtt/mosquitto.org.crt
read /mqtt/mosquitto.org.crt
size: 1452
-----BEGIN CERTIFICATE-----
MIIEAzCCAuugAwIBAgIUBY1hlCGvdj4NhBXkZ/uLUZNILAwwDQYJKoZIhvcNAQEL
BQAwgZAxCzAJBgNVBAYTAkdCMRcwFQYDVQQIDA5Vbml0ZWQgS2luZ2RvbTEOMAwG
A1UEBwwFRGVyYnkxEjAQBgNVBAoMCU1vc3F1aXR0bzELMAkGA1UECwwCQ0ExFjAU
BgNVBAMMDW1vc3F1aXR0by5vcmcxHzAdBgkqhkiG9w0BCQEWEHJvZ2VyQGF0Y2hv
by5vcmcwHhcNMjAwNjA5MTEwNjM5WhcNMzAwNjA3MTEwNjM5WjCBkDELMAkGA1UE
BhMCR0IxFzAVBgNVBAgMDlVuaXRlZCBLaW5nZG9tMQ4wDAYDVQQHDAVEZXJieTES
MBAGA1UECgwJTW9zcXVpdHRvMQswCQYDVQQLDAJDQTEWMBQGA1UEAwwNbW9zcXVp
dHRvLm9yZzEfMB0GCSqGSIb3DQEJARYQcm9nZXJAYXRjaG9vLm9yZzCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAME0HKmIzfTOwkKLT3THHe+ObdizamPg
UZmD64Tf3zJdNeYGYn4CEXbyP6fy3tWc8S2boW6dzrH8SdFf9uo320GJA9B7U1FW
Te3xda/Lm3JFfaHjkWw7jBwcauQZjpGINHapHRlpiCZsquAthOgxW9SgDgYlGzEA
s06pkEFiMw+qDfLo/sxFKB6vQlFekMeCymjLCbNwPJyqyhFmPWwio/PDMruBTzPH
3cioBnrJWKXc3OjXdLGFJOfj7pP0j/dr2LH72eSvv3PQQFl90CZPFhrCUcRHSSxo
E6yjGOdnz7f6PveLIB574kQORwt8ePn0yidrTC1ictikED3nHYhMUOUCAwEAAaNT
MFEwHQYDVR0OBBYEFPVV6xBUFPiGKDyo5V3+Hbh4N9YSMB8GA1UdIwQYMBaAFPVV
6xBUFPiGKDyo5V3+Hbh4N9YSMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQEL
BQADggEBAGa9kS21N70ThM6/Hj9D7mbVxKLBjVWe2TPsGfbl3rEDfZ+OKRZ2j6AC
6r7jb4TZO3dzF2p6dgbrlU71Y/4K0TdzIjRj3cQ3KSm41JvUQ0hZ/c04iGDg/xWf
+pp58nfPAYwuerruPNWmlStWAXf0UTqRtg4hQDWBuUFDJTuWuuBvEXudz74eh/wK
sMwfu1HFvjy5Z0iMDU8PUDepjVolOCue9ashlS4EB5IECdSR2TItnAIiIwimx839
LdUdRudafMu5T5Xma182OC0/u/xRlEm+tvKGGmfFcN0piqVl8OrSPBgIlb+1IKJE
m/XriWr/Cq4h/JfB7NTsezVslgkBaoU=
-----END CERTIFICATE-----

$
```

3. The file transfer can be done using YMODEM protocol, depending on the terminal program in use. For Tera Term:
   - Go to Transfer -> YMODEM -> Send, then select the certificate file.



4. After transferring, initialize and start the MQTT client with TLS enabled:

```
$ mqtt
Usage: mqtt init url port secure(0|1) <ca_file> <client_cert_file> <client_key_file>
  or:  mqtt sub topic <qos(0|1|2)>
  or:  mqtt unsub topic
  or:  mqtt pub topic payload <qos(0|1|2)>
  or:  mqtt ping
$
$ mqtt init test.mosquitto.org 8883 1 /mqtt/mosquitto.org.crt
$
WIFI CONNECTED
I (55479) SCM_API: AP SSID: Xiaohu_ASUS
I (55480) SCM_API: AP BSSID: 50:eb:f8:19:88:a0
I (55480) SCM_API: AP CH: 11
I (55482) SCM_API: AP RSSI: -27
I (55483) SCM_API: AP Country : AA
I (55483) SCM_API: Status: CONNECTED

WIFI GOT IP

$ ps
 PID   PR  STWM  S  %CPU+      TIME+   TASK                                (0x21ce60-0x21de50, 0x21d91c)
   1    3   532  X    1.0    0:00:01 init                                (0x226a80-0x228270, 0x227e4c)
  10    3   707  R    0.7    0:00:01 mqtt-agent                          (0x2107d8-0x2117d0, 0x2116ec)
   2    0   957  R   40.0    0:00:56 idle                                (0x21e210-0x21ee00, 0x21ecac)
   4    3   451  B    0.0    0:00:00 knetd                               (0x21034c-0x210740, 0x21066c)
   3    5   165  B   57.9    0:01:21 ksofttimerd                         (0x222870-0x222e60, 0x222d3c)
   7    3   255  B    0.0    0:00:00 rt_msg                              (0x228290-0x229680, 0x22941c)
  11    3   519  B    0.0    0:00:00 wpa_supplicant                      (0x225460-0x226050, 0x225f0c)
   9    3   260  B    0.0    0:00:00 wise_event_loop_task                (0x220910-0x221300, 0x2211fc)
   5    3   231  B    0.0    0:00:00 scm2020-wlan fast taskq             (0x223c70-0x223ee0, 0x223ddc)
   8    7    48  B    0.0    0:00:00 ll                                  (0x221f90-0x222580, 0x22247c)
   6    3   207  B    0.1    0:00:00 knet80211d/wlan0                    (0x221f90-0x222580, 0x22247c)
$
```

1.2.5.3 Using TLS with Encryption and Client Authentication

For client authentication, you will need both a client certificate and a client key. These can be generated using tools like openssl, and test.mosquitto.org provides a guide on how to create them.

mosquitto

MQTT

This is test.mosquitto.org. It hosts a publicly available Eclipse Mosquitto MQTT server/broker. MQTT is a very lightweight protocol that uses a publish/subscribe model. This makes it suitable for "machine to machine" messaging such as with low power sensors or mobile devices.

For more information on MQTT, see http://mqtt.org/ or the Mosquitto MQTT man page.

If you are interested in your own hosted instance of Mosquitto you should look at the Cedalo offering. Cedalo are the company that sponsor the main development of Mosquitto.

The server

The server listens on the following ports:

- 1883 : MQTT, unencrypted, unauthenticated
- 1884 : MQTT, unencrypted, authenticated
- 8883 : MQTT, encrypted, unauthenticated
- 8884 : MQTT, encrypted, client certificate required
- 8885 : MQTT, encrypted, authenticated
- 8886 : MQTT, encrypted, unauthenticated
- 8887 : MQTT, encrypted, server certificate deliberately expired
- 8080 : MQTT over WebSockets, unencrypted, unauthenticated
- 8081 : MQTT over WebSockets, encrypted, unauthenticated
- 8090 : MQTT over WebSockets, unencrypted, authenticated
- 8091 : MQTT over WebSockets, encrypted, authenticated

The encrypted ports support TLS v1.3, v1.2 or v1.1 with x509 certificates and require client support to connect. For ports 8883 and 8884 you should use the certificate authority file (mosquitto.org.crt (PEM format), or mosquitto.org.der (DER format)) to verify the server connection. Ports 8081 and 8886 have a Lets Encrypt certificate, so you should use your system CA certificates or the appropriate Lets Encrypt CA certificate for verification.

Port 8884 requires clients to provide a certificate to authenticate their connection. You can generate your own certificate.

The configuration is available to view.

Authentication and topic access

You are free to use it for any application, but please do not abuse or rely upon it for anything of importance. This server runs on an Intel Atom N2800, and as such is a low power device. It is not intended to demonstrate any performance characteristics.

You should also build your client to cope with the broker restarting.

If you have the mosquitto clients installed try:

- mosquitto_sub -h test.mosquitto.org -t "#" -u wildcard -v

Please don't publish anything sensitive, anybody could be listening.

Caveats

This server is provided as a service for the community to do testing, but it is also extremely useful for testing the server. This means that it will often be running unreleased or experimental code and may not be as stable as you might hope. It may also be slow - the broker often runs under valgrind or perf. Finally, not all of the features may be available all of the time, depending on what testing is being done. In particular, websockets and TLS support are the most likely to be unavailable.

In general you can expect the server to be up and to be stable though.

Get in touch

Come and discuss the Mosquitto project on Slack (go to the Mosquitto channel).

If you do publish things to this server on a regular basis, please get in touch to satisfy my curiosity - there are lots of topics that look interesting but I know nothing about. I'm **ral** on the libera.chat #mqtt irc channel, or see the mosquitto source for contact details..

Examples using this service

- Websockets $SYS tree for test.mosquitto.org
- Websockets $SYS tree for test.mosquitto.org (TLS)

Keep the service running

Please sponsor this service so we can move to a more powerful

1. Generate the client certificate and key.

2. Load the client certificate and key onto the target device using the fs load CLI command, similar to the CA certificate.

```
$ fs load /mqtt/client.crt
load local file to /mqtt/client.crt
CC## Total Size = 0x0000053e = 1342 Bytes
xyzModem - CRC mode, 2(SOH)/2(STX)/0(CAN) packets, 4 retries
$
$ fs load /mqtt/client.key
load local file to /mqtt/client.key
CC## Total Size = 0x0000068b = 1675 Bytes
xyzModem - CRC mode, 1(SOH)/2(STX)/0(CAN) packets, 4 retries
$
$ fs read /mqtt/client.crt
read /mqtt/client.crt
size: 1342
-----BEGIN CERTIFICATE-----
MIIDsjCCApqgAwIBAgIBADANBgkqhkiG9w0BAQsFADCBkDELMAkGA1UEBhMCR0Ix
FzAVBgNVBAgMDlVuaXRlZCBLaW5nZG9tMQ4wDAYDVQQHDAVEZXJieTESMBAGA1UE
CgwJTW9zcXVpdHRvMQswCQYDVQQLDAJDQTEWMBQGA1UEAwwNbW9zcXVpdHRvLm9y
ZzEfMB0GCSqGSIb3DQEJARYQcm9nZXJAYXRjaG9vLm9yZzAeFw0yNDA5MDMxODAy
MzlaFw0yNDEyMDIxODAyMzlaMIGLMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2Fs
aWZvcm5pYTEPMA0GA1UEBwwGSXJ2aW5lMREwDwYDVQQKDAhTZW5zY29tbTEOMAwG
A1UECwwFSW9UU3cxDzANBgNVBAMMBlRob21hczEiMCAGCSqGSIb3DQEJARYTdGhv
bWFzQHNlbnNjb21tLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AN+4/05/PMoZALNAjj7Az1hylqeniQR1T5OR5ZBq7RJ89aXgdoVU629Q4tusvikg
WcX3WUZIrcb/PHo3OlaA+LxCnJdrNmoD9eDbsRXdy91oLwK/NVhqJRptozARYkAH
ohFsZ2rrDbOnGmGww4XbH/DvwjhiAyQeM4y3bGO+fcY5WxYieZCwwY8o+udjftKP
pGe5I5hZR/YXC0Twz9hVHgeMW8gkU40sLKG6n0BItRrxqh06yZwSfpdrFe06LKYP
NyoQkBPS1zIOrjfCjvj+Da+SdcAXCSkgTeWl/fdkFeXiVAj0/TdGAsbRlBMpG0/D
nLKtJuHB8yXRHxSOs5xt0XMCAwEAAaMBgwCQYDVR0TBAIwADALBgNVHQ8EBAMC
BeAwDQYJKoZIhvcNAQELBQADggEBACZV7M8PhMeob7vXVRq0QGjb/It/uGhCgPeX
GOjkOeZpsHKw5gMhroQDyrxj3y5fN8cM36vXNjR0XobBOlNBtzsjbehI+/B52zvt
RUI//kCF0+0Ltj/6/a9+w85xH7TdgS4qb1iO+wzJdVCS42loRmTbfWDBhG3Zks+J
oGS96D+hS4KPpZPG+mhIYB4TyUAe10pxJsXCVK8HW+7oLm2Lo78/QB8Xd7lDWPd1
ahCu/AM27PsVbPaU7hKzaOS12+C6frt9brk1H9nRr8vKQc8Z82EEUJkW0qyZH56T
Euk6Z5qBLUJvsj7gRRWM/ZLz6/47M0KDRVnG6g4AkmCePFM4+Ec=
-----END CERTIFICATE-----

$ fs read /mqtt/client.key
read /mqtt/client.key
size: 1675
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA37j/Tn88yhkAs0COPsDPWHKWp6eJBHVPk5HlkGrtEnz1peB2
hVTrb1Di26y+KSBZxfdZRkitxv88ejc6VoD4vEKcl2s2agP14NuxFd3L3WgvAr81
WGolGm2jMBFiQAeiEWxnausNs6caYbDDhdsf8O/COGIDJB4zjLdsY759xjlbFiJ5
kLDBjyj652N+0o+kZ7kjmFlH9hcLRPDP2FUeB4xbyCRTjSwsobqfQEi1GvGqHTrJ
nBJ+l2sV7Tospg83KhCQE9LXMg6uN8KO+P4Nr5J1wBcJKSBN5aX992QV5eJUCPT9
N0YCxtGUEykbT8Ocsq0m4cHzJdEfFI6znG3RcwIDAQABAoIBAB7FXSg4y+2oHraI
7IepEVvC9wG8Q1Y/pGBRstd4PX9LfKYCB4szMOawo2M/kTAq6O4XpUnLeUtjk7fj
nLyFJLEQIGWbM3LFdK4myWaOiRm82KpyDi5I+y11YvqbiX3xDtOCq37DMbFCDJjf
LSVeDDihYX9Ly87N8J0sJVZavwXiX9WRUeIiEIG338P7oCzUPuFWS4zY/py/raA/
lIMrmUwhYNIKDhhTW8HjNWZ2U42QGCRxek8loTFEkwDp1/auGZcpVd9DlG4lKvzQ
jIsdrLu9Jpb9lMGjE5GjMAhUB3a03TPoEnZrHHTAVTwh+d/CuLIieHfI8IXjvnPm
g+h1h2ECgYEA8Btmoa10aFRqlXGQ0UqbMUw0zMxIuUaLwfoL1MxltBdoJOg+5Q0B
eT5NIRGc+QpaHqueGDnYotNgacUbhBdp39Tp2WHfgwZpJdloO7Pp4cBdXP2asU9n
mnUDFPYLLP0qS0q7A66yLc88Bjrhjo9z+Ehkls91tcvkiqIsBcMkGLsCgYEA7of2
wtSGoB9pb/jvlbf/BdXKEJ2zMVAcZqkXoAfmMufxaF1JdBzzqJ4hzEraBtFE9xDH
KIiRb2vzYGY8SIyn8j+nggYk+yv/syPN/UiJlSVI6SgnYHMWccIfp00A/Q1nGJGL
h6JvDp+icrzeA6RntnQTqhQwZg2/Pkpe0nUbmqkCgYBGzt3aaiu9JL/16HLbtdPE
mwOrLcd4lBxdDR92Fv0bOhflYnRB2i8IEV5vlSEktG/VQaky3cRMaGezaYRu1PTN
JJ3+FIX0vuw9VDsP+EPN5oviA4weJBuaik1pXhH3p4VUhOpX8KntJoM+FxkzkIyT
uM2pR+8fuMxiMecnn9/CRwKBgFUO8vWi56btzKF7OaRACpbAF+A4/A9Xq+kH3z7X
0kTQ5Qr8SRc4w8kbMR4yivDnaxpXR02y9XmwGweDLXgJgFIoVMlO+5z9oZ7145yw
Yy9mLkvGX9RK9fP272avVOzn/J2MOR9S0xt53Wng46KU2876MZDyxhd+S3Yg+xM9
Y2yRAoGBALWi4sfIhrDmqDoRo6k0vkQ1LGRZCglb8Dj7pabSVUcevv0tT3rY4AJj
JSAKJDLzK97SATuw8ekhMsTm89g14fb+TYU0jsIXn7PY73sYs0wwq9bohAzGVF7F
uQhewWrMvTZQl2GWP+OJdifbaNIns+lhxz9BpfMxzzLHDpZ709eB
-----END RSA PRIVATE KEY-----

$
```

3. Initialize and start the MQTT client with both TLS and client authentication:

```
$ mqtt
Usage: mqtt init url port secure(0|1) <ca_file> <client_cert_file> <client_key_file>
  or:  mqtt sub topic <qos(0|1|2)>
  or:  mqtt unsub topic
  or:  mqtt pub topic payload <qos(0|1|2)>
  or:  mqtt ping
$ mqtt init test.mosquitto.org 8884 1 /mqtt/mosquitto.org.crt /mqtt/client.crt /mqtt/client.key
$
$ ps
 PID   PR  STWM  S  %CPU+      TIME+   TASK
   1    3   523  X    2.6    0:00:22 init                          (0x21ce60-0x21de50, 0x21d91c)
  15    3   715  R    0.2    0:00:02 mqtt-agent                    (0xa0002f10-0xa0004700, 0xa0003c7c)
   2    0   957  R   87.1    0:12:26 idle                          (0x2107d8-0x2117d0, 0x2116ec)
   3    5   165  B    9.5    0:01:21 ksofttimerd                   (0x21034c-0x210740, 0x21066c)
   4    3   451  B    0.0    0:00:00 knetd                         (0x21e210-0x21ee00, 0x21ecac)
   6    3   207  B    0.0    0:00:00 knet80211d/wlan0              (0x221f90-0x222580, 0x22247c)
   7    3   255  B    0.0    0:00:00 rt_msg                        (0x222870-0x222e60, 0x222d3c)
   5    3   231  B    0.1    0:00:01 scm2020-wlan fast taskq       (0x220910-0x221300, 0x2211fc)
  11    3   519  B    0.0    0:00:00 wpa_supplicant                (0x228290-0x229680, 0x22941c)
   9    3   260  B    0.0    0:00:00 wise_event_loop_task          (0x225460-0x226050, 0x225f0c)
   8    7    48  B    0.0    0:00:00 ll                            (0x223c70-0x223ee0, 0x223ddc)
$
$
```

1.2.6 Testing Subscription to a Specific Topic:

To test subscribing to a specific topic, follow these steps:

1. **Subscribe to a topic** using the MQTT client's CLI command:

```
$ mqtt
Usage: mqtt init url port secure(0|1) <ca_file> <client_cert_file> <client_key_file>
  or:  mqtt sub topic <qos(0|1|2)>
  or:  mqtt unsub topic
  or:  mqtt pub topic payload <qos(0|1|2)>
  or:  mqtt ping
$ mqtt sub senscomm/light1
$
```

2. **Publish a message** to the same topic from a separate MQTT client running on your PC (e.g., using Eclipse Mosquitto):

```
thomas@Thomas-Gram22:~$ mosquitto_pub -h test.mosquitto.org -t senscomm/light1 -m on
thomas@Thomas-Gram22:~$
```

3. **Check the message** received by the SCM1612 MQTT client. The message should be displayed in the terminal.

```
$ mqtt
Usage: mqtt init url port secure(0|1) <ca_file> <client_cert_file> <client_key_file>
  or:  mqtt sub topic <qos(0|1|2)>
  or:  mqtt unsub topic
  or:  mqtt pub topic payload <qos(0|1|2)>
  or:  mqtt ping
$ mqtt sub senscomm/light1
$
$ I (1162076) MQTT_APP: Got Message:on published
I (1162077) MQTT_APP:  on topic:senscomm/light1.
```

1.2.7 Testing Message Publication to a Specific Topic

To test publishing a message to a specific topic, follow these steps:

1. **Subscribe to a topic** from your PC's MQTT client:

```
thomas@Thomas-Gram22:~$ mosquitto_sub -h test.mosquitto.org -t senscomm/light2
```

2. **Publish a message** to the same topic using the SCM1612 MQTT client's CLI command:

```
$
$ mqtt pub senscomm/light2 keep_on
$
```

3. **Check the message** received on the PC side, which should display the message published by the SCM1612 client.

```
thomas@Thomas-Gram22:~$ mosquitto_sub -h test.mosquitto.org -t senscomm/light2
keep_on
```