



# Application Note

**BRT\_AN\_041**

## FT90X MQTT Application

**Version 1.0**

**Issue Date: 2018-09-19**

In an increasingly connected world, more and more devices are going online. To enable online connectivity typically requires an MCU with Ethernet that can communicate with existing infrastructure. To demonstrate the principle, this Application Note describes an MQTT client that can connect to a broker and publish messages and subscribe to topics.

Use of Bridgetek devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold Bridgetek harmless from any and all damages, claims, suits or expense resulting from such use.

**Bridgetek Pte Ltd (BRTChip)**  
178 Paya Lebar Road, #07-03, Singapore 409030  
Tel: +65 6547 4827 Fax: +65 6841 6071  
Web Site: <http://www.brtchip.com>  
Copyright © Bridgetek Pte Ltd

---

## **Table of Contents**

<b>1 Introduction .....</b>	<b>4</b>
<b>1.1 Overview .....</b>	<b>4</b>
1.1.1 Part 1 .....	4
1.1.2 Part 2 .....	4
<b>1.2 Open Source Software.....</b>	<b>5</b>
<b>1.3 Scope .....</b>	<b>5</b>
1.3.1 Features .....	5
1.3.2 Possible Enhancements.....	5
<b>2 Project Overview .....</b>	<b>6</b>
<b>2.1 Sources Folder.....</b>	<b>6</b>
<b>2.2 Iwip Folder .....</b>	<b>6</b>
<b>2.3 mbedTLS Folder .....</b>	<b>7</b>
<b>3 Software Implementation.....</b>	<b>8</b>
<b>3.1 Brokers.....</b>	<b>8</b>
<b>3.2 Setup Part 1 .....</b>	<b>9</b>
<b>3.3 Setup Part 2 .....</b>	<b>9</b>
3.3.1 Creating Certificates.....	9
3.3.2 Using the Certificates (Mosquitto) .....	9
3.3.3 Using the Certificates (Application) .....	10
<b>4 Operation .....</b>	<b>11</b>
<b>5 Additional MQTT Brokers .....</b>	<b>12</b>
<b>6 Importing into the FT9XX Toolchain .....</b>	<b>13</b>
<b>7 Contact Information .....</b>	<b>14</b>
<b>Appendix A– References .....</b>	<b>15</b>
<b>Document References .....</b>	<b>15</b>
<b>Acronyms and Abbreviations.....</b>	<b>15</b>
<b>Appendix B – List of Tables &amp; Figures .....</b>	<b>16</b>
<b>List of Tables.....</b>	<b>16</b>
<b>List of Figures .....</b>	<b>16</b>

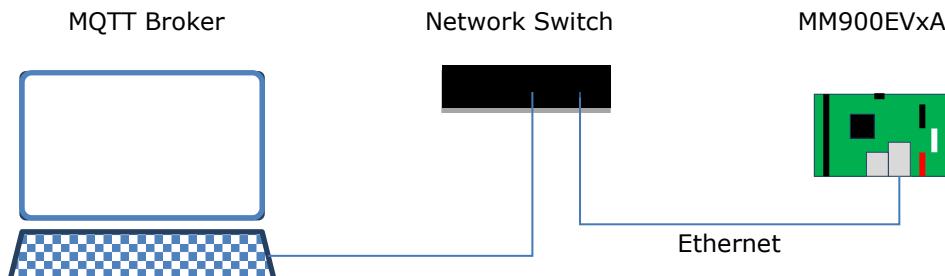
---

---

**Appendix C– Revision History ..... 17**

## 1 Introduction

This Application Note describes a network node implemented with an FT9XX device. The network node communicates with an MQTT broker using the MQTT protocol. It will periodically publish messages and act on messages for topics to which it has subscribed to from the broker.



**Figure 1 - Block Diagram**

The FT90X runs lwIP (lightweight IP stack) and its included ‘mqtt’ application. Together these components allow for the MQTT protocol to be used for communication and control of the application. Later in the application note secure sockets using mbedTLS are discussed in relation to MQTT.

The document should be read in association with the two example code projects provided in the references section.

Part 1 is an implementation of MQTT made with standard network sockets with no encryption or authentication.

Part 2 uses TLS sockets to encrypt and authenticate the connection between the device and the broker.

### 1.1 Overview

#### 1.1.1 Part 1

This document describes the design and implementation of the FT90X MQTT Application. Part 1 of the application allows a user to:

- Connect the device to a LAN.
- Implement an MQTT client and connect to an MQTT broker.
- Subscribe to a topic and publish messages to a topic.

This document is intended to demonstrate the networking capabilities of the FT90X family of microcontrollers by demonstrating a simple network-connected Internet of Things (IoT) device.

#### 1.1.2 Part 2

An extension of this application note is the additional capability to:

- Implement a TLS layer to securely connect to the MQTT broker.

## 1.2 Open Source Software

Third-party open source code is used to implement this application note:

- Printf – tinyprintf
- lwIP – lightweight IP stack
- mbedTLS – Secure sockets (used in Part 2 only).

Links to resources for these libraries are in [Appendix A – References](#).

## 1.3 Scope

Part 1 of this application implements a simple MQTT client.

Part 2 demonstrates communication *via* TLS.

### 1.3.1 Features

The basic building blocks of an IoT (Internet of Things) client project are provided. The client can connect to a standard MQTT broker and perform the ‘publish’ and ‘subscribe’ operations.

### 1.3.2 Possible Enhancements

This application note can be seen as a start for an IoT project. Some example enhancements could be:

- Send real world data from sensors to broker. A sensor could be connected to the I<sup>2</sup>C or SPI busses and report real-time data.
- Receive messages on a topic to perform a real world action. A topic could be used to control GPIO which drives an actuator or motor.
- Connect to a remote MQTT broker via the Internet.

## 2 Project Overview

The project files for the application are divided into the following folders.

Folder	Description
Source	Application source code and abstraction files.
Includes	Application specific header files.
Certificates	Certificate files for mbedTLS. (Part 2 only)
lib	Library files.
lib\FreeRTOS	FreeRTOS library
lib\lwip	lwIP library. Minimum of Version 2.1.0 RC1 is required.
lib\mbedtls	mbedtls library. (Part 2 only)
lib\tinyprintf	tinyprintf library.

**Table 1 - Project Files Overview**

Version 2.5.0 or later of the FT9XX Toolchain is required.

### 2.1 Sources Folder

The main part of the application is found in the “Sources” folder. This is split into two sections and has 2 source code files.

- For Part 1 code the “main.c” file is generally responsible for setting up the FT90X device, initialising FreeRTOS, and setting up and handling application specific callbacks for the lwIP MQTT library.
- The “main.c” file in Part 2 is responsible for setting up the FT90X device, initialising FreeRTOS, loading certificates and configuring the mbedtls library, and setting up and handling application specific callbacks for the lwIP MQTT library.
- The second file “net.c” implements an interface between the lwIP library and the main code.
- 

The other file in this folder is:

- “crt0.S” a modified startup file (in FT9XX assembly language) to allow the application to write to a protected section of FlashROM on the device.

Files in these folders use the “Includes” folder for application specific header files.

### 2.2 lwip Folder

This folder holds the lwIP library which handles networking for the application note. The code has been ported to FT90X and supports reading the file system from program memory rather than RAM. Version 2.1.0 of lwIP contains the minimum version which supports the TLS library and MQTT functions required.

The lwIP configuration file for this application is in “Includes/lwipopts.h”. The main options enabled are:

NO_SYS	0	Enable RTOS mode to use FreeRTOS methods.
MEM_LIBC_MALLOC	1	Enable libc like malloc functions. The names of the malloc/free equivalent functions in FreeRTOS are set.
MEMP_MEM_MALLOC	1	Use malloc instead of static accelerators.
LWIP_SOCKET	1	Enable socket interface.
LWIP_COMPAT_SOCKETS	1	Enable Unix compatible socket interface.
LWIP_NETCONN	1	Enable network sequential API.
LWIP_RAW	0	Disable raw access to network (use sockets) .
LWIP_UDP	1	Enable UDP packets.
LWIP_TCP	1	Enable TCP packets.
LWIP_DNS	1	Enable DNS. Specify a small table size as we only ever look up one host name.
LWIP_DHCP	1	Enable DHCP. This allows us to automatically configure the network settings.
MEM_SIZE	10*1024	Specify an absolute maximum size for any requested mallocs as a safeguard.
TCP_MSS	1460	Maximum segment size for TCP.
LWIP_ARP	1	Enable ARP.

The settings made here may require to be altered for production systems. For instance the ETHARP\_TRUST\_IP\_MAC flag is set on. This will allow local hosts to spoof MAC and therefore IP addresses easily.

## 2.3 mbedtls Folder

The embedded library for TLS sockets is included. mbedtls 2.11.0 is included in the package. The configuration file for this library is in "Includes/mbedtls\_config.h". The configuration has a minimum number of options to allow the application to connect to an MQTT broker via a secure port.

MBEDTLS_SSL_TLS_C	Enable TLS.
MBEDTLS_SSL_CLI_C	TLS client mode.
MBEDTLS_CIPHER_C	Add cipher layer.
MBEDTLS_MD_C	Add message digest layer.
MBEDTLS_ENTROPY_C	Generate platform-specific entropy.
MBEDTLS_CTR_DRBG_C	AES-256 random number generator.
MBEDTLS_SSL_PROTO_TLS1_2	Support TLS v1.2.
MBEDTLS_KEY_EXCHANGE_RSA_ENABLED	Enable RSA ciphersuites.
MBEDTLS_RSA_C	Enable RSA public key cryptosystem.
MBEDTLS_OID_C	Enable OID database.
MBEDTLS_PKCS1_V15	Enable support for PKCS#1 v1.5 operations.
MBEDTLS_BIGNUM_C	Enable the multi-precision integer library.
MBEDTLS_X509_CRT_PARSE_C	Enable X.509 certificate parsing.
MBEDTLS ASN1_PARSE_C	Enable the generic ASN1 parser.
MBEDTLS_PK_PARSE_C	Enable the generic public ( <u>asymmetric</u> ) key parser.
MBEDTLS PEM_PARSE_C	Enable PEM encoding.
MBEDTLS_BASE64_C	Enable the BASE64 module.
MBEDTLS_SHA1_C	Enable the SHA1 cryptographic hash algorithm.
MBEDTLS_SHA256_C	Enable the SHA-224 and SHA-256 cryptographic hash algorithms.
MBEDTLS_AES_C	Enable the AES block cipher.
MBEDTLS_CIPHER_MODE_CBC	Enable Cipher Block Chaining mode (CBC) for symmetric ciphers.
MBEDTLS_PLATFORM_MEMORY	Allow mbedtls to use non-library versions of malloc/free. These are mapped to the FreeRTOS versions through a stub function.

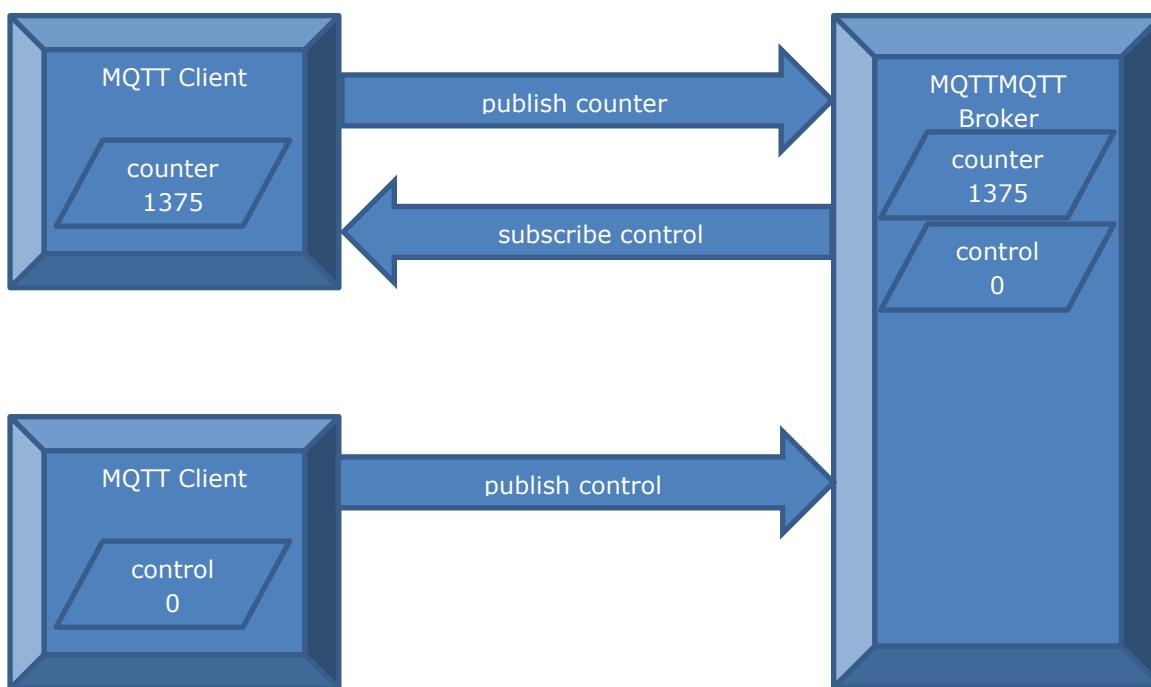
### 3 Software Implementation

The lwIP MQTT application provides the main part of the application note code. This resides in the lwIP library and callback functions are found in "main.c" to provide the desired functionality.

The IP address of the broker is hardcoded in the source code in main.c. There are various methods for provisioning MQTT clients to connect to brokers and this is out of the scope of this demonstration.

The MQTT application receives messages from a thread in the main.c code and sends them to the connected broker. The topic name and message text are specified in the call to the MQTT application. A callback receives messages to which the application has subscribed.

The function of the demonstration code is a counter which increments every 1000 ms. The topic for this message is "counter". A message is published each time the counter increments. A schematic for the MQTT system is shown in Figure 2.



**Figure 2 - MQTT Schematic for Application**

A receiver in the application will subscribe to a topic called "control". Once a message is received on that topic then the counter will be reset with either zero, for a zero length message, or the value of the string received in the message body. The library function "strtol" (string to unsigned long) is used to decode the number passed in the message. Another MQTT client is used to send the "control" topic message, the broker will update the MQTT client with the control message.

#### 3.1 Brokers

There are many MQTT compatible brokers available. In this example the Eclipse Mosquitto broker (<https://mosquitto.org/>) is used for testing. In Part 1 and Part 2 the broker is installed locally on a private IPv4 network. The address of the broker is given as a local private IP address.

## 3.2 Setup Part 1

The setup for Mosquitto in Part 1 does not require any changes from the default Mosquitto installation. However, a sample mosquitto.conf file is included in the application note code which can be used for Part 1 and Part 2.

## 3.3 Setup Part 2

The TLS connection from the client to the local broker requires certificates for the broker and client. The openSSL program, or an equivalent, is required to create the certificates and keys. The steps below allow files to be made for both the broker and the client. These are self-signed certificates for testing and demonstration purposes only.

Note that the value of "Common Name" used when filling out information requested by the openSSL utility is the network name or IP address of the MQTT broker.

### 3.3.1 Creating Certificates

**Step 1:** Make a key pair for the Certificate Authority (CA).

This will be used to create certificates for the server and client.

```
openssl genrsa -des3 -out ca.key 2048
```

This creates a key pair file "ca.key" for the Certificate Authority.

**Step 2:** Create the certificate for the CA. The key pair is used to make a certificate file for the Certificate Authority.

```
openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
```

The openSSL will ask questions about the data to be included in the certificate. This is filled with the details from your organisation that will appear in the certificates.

**Note:** 1826 days is 5 years (including one leap year).

The Certificate Authority certificate "ca.crt" is created.

**Step 3:** Create a key pair to be used by the MQTT broker.

```
openssl genrsa -out server.key 2048
```

A new key pair for the server "server.key" will be generated.

**Step 4:** Create a certificate request file.

```
openssl req -new -out server.csr -key server.key
```

The command asks for input again. "Common Name" is again the network name of the server.

**Step 5:** Sign the certificate request file.

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 365
```

This generates the server certificate file "server.crt".

### 3.3.2 Using the Certificates (Mosquitto)

The files created above should be moved to a directory where they can be accessed by the mosquitto.exe. Suggested location is within the Mosquitto directory in the Program Files area.

e.g. "C:\Program Files (x86)\mosquitto\certs"

In the Mosquitto configuration file "mosquitto.conf" these files are referenced by the *cafile*, *certfile* and *keyfile* lines in a *listener* section.

```
listener 8883
cafile C:\Program Files (x86)\mosquitto\certs\ca.crt
certfile C:\Program Files (x86)\mosquitto\certs\server.crt
keyfile C:\Program Files (x86)\mosquitto\certs\server.key
```

A sample mosquitto.conf file is included in the application note code along with sample certificate files for a random private IP address of 172.16.3.31.

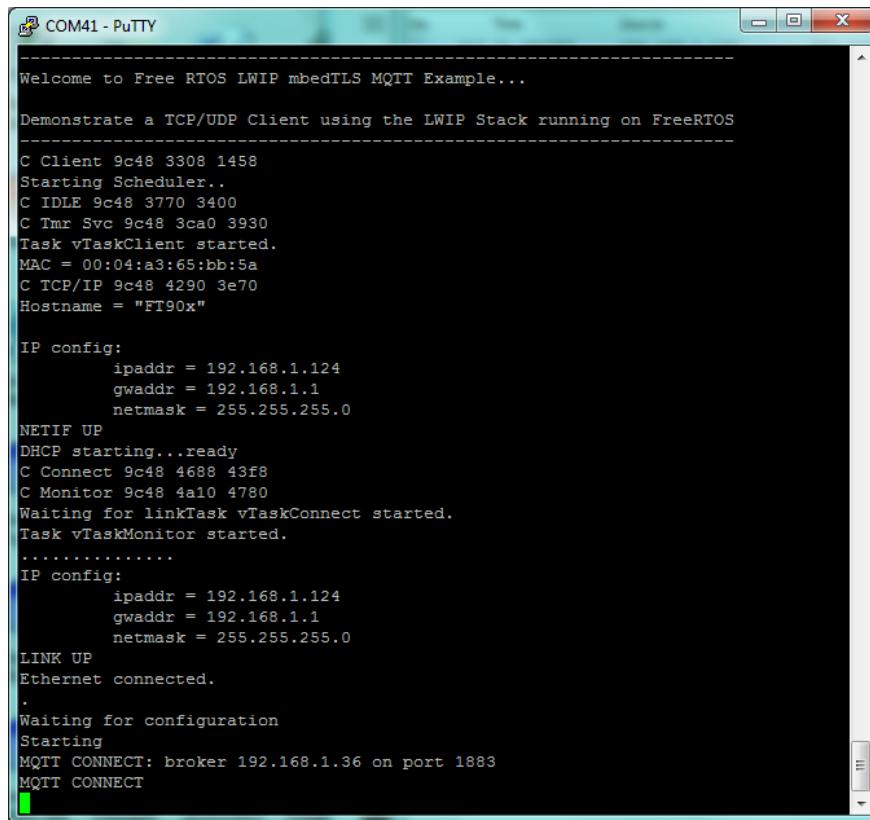
### **3.3.3 Using the Certificates (Application)**

The "ca.crt" file used in the Mosquitto configuration is also used in the source code of the application in the "Certificates" folder. There is a Builder which calls certificate.mk in the Certificates folder. This converts the text of the certificate file into a form used by the application.

The mbedTLS code loads the certificate and authenticates with the broker.

## 4 Operation

Once the network is configured then the device will attempt to connect to the broker.



```

COM41 - PuTTY
-----
Welcome to Free RTOS LWIP mbedTLS MQTT Example...
-----
Demonstrate a TCP/UDP Client using the LWIP Stack running on FreeRTOS
-----
C Client 9c48 3308 1458
Starting Scheduler..
C IDLE 9c48 3770 3400
C Tmr Svc 9c48 3ca0 3930
Task vTaskClient started.
MAC = 00:04:a3:65:bb:5a
C TCP/IP 9c48 4290 3e70
Hostname = "FT90x"

IP config:
    ipaddr = 192.168.1.124
    gwaddr = 192.168.1.1
    netmask = 255.255.255.0

NETIF UP
DHCP starting...ready
C Connect 9c48 4688 43f8
C Monitor 9c48 4a10 4780
Waiting for linkTask vTaskConnect started.
Task vTaskMonitor started.
.....
IP config:
    ipaddr = 192.168.1.124
    gwaddr = 192.168.1.1
    netmask = 255.255.255.0

LINK UP
Ethernet connected.
.
Waiting for configuration
Starting
MQTT CONNECT: broker 192.168.1.36 on port 1883
MQTT CONNECT

```

**Figure 3 - FT90X UART Output when Connecting to MQTT Broker**

The application will report the progress of its configuration and connection to the broker on the UART port at 115200 baud. This is shown in Figure 3. After the connection is complete ("MQTT CONNECT" message) then it will subscribe to the "control" topic and begin sending messages to the "counter" topic.

A mosquito\_sub.exe command can be used to monitor the messages sent by the device to the broker.

```
mosquitto_sub -t counter -h 172.16.3.31
```

Additionally, counter values can be changed by publishing to the "control" topic.

```
mosquitto_pub -t control -h 172.16.3.31 -m 1234
```

Using the mosquito.conf file supplied the broker can be accessed either through the unsecure port 8883 or the secure port 8883.

When accessing the secure port Mosquitto utilities need access to the certificate files to properly authenticate with the server.

```
mosquitto_sub -t counter --ca-file ca.crt -p 8883 -h 172.16.3.31
mosquitto_pub -t control --ca-file ca.crt -p 8883 -h 172.16.3.31 -m 1234
```

## 5 Additional MQTT Brokers

There are publically accessible MQTT brokers available for testing. The Mosquitto project runs one such broker. The details of the broker are available at <https://test.mosquitto.org/>.

If the LAN is able to reach the MQTT ports on the internet, then this application will be able to connect and run MQTT with remote brokers in the cloud.

Replace the value of MQTT\_BROKER in main.c with the name of the broker, and MQTT\_BROKER\_PORT with the port number.

Certificates for using TLS for connections to remote brokers will be provided – in Part 2 replace ca.crt with the provided certificate.

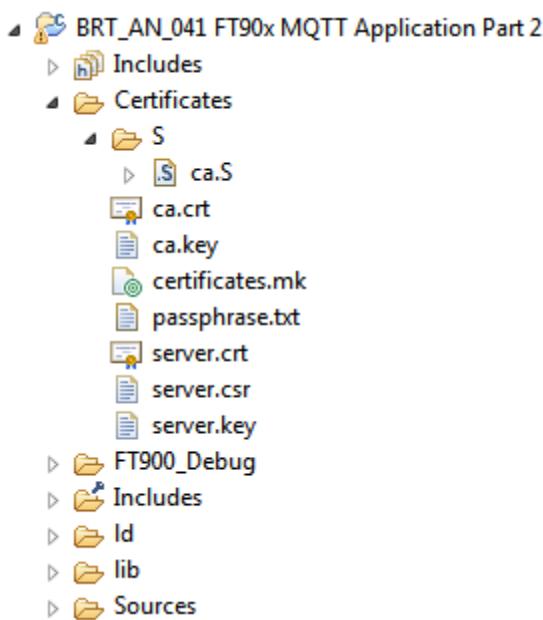
## 6 Importing into the FT9XX Toolchain

The firmware found at the following link can be easily imported into the [FT9XX Toolchain](#):

<http://brtchip.com/SoftwareExamples-ft90x/>

Once installed, select File --> Import --> General --> Existing Projects into Eclipse, and point to the downloaded and extracted project directory.

The project will appear in Eclipse Project Explorer as shown in **Error! Reference source not found..**



**Figure 4 - Eclipse Project Structure**

## 7 Contact Information

### Head Quarters – Singapore

Bridgetek Pte Ltd  
178 Paya Lebar Road, #07-03  
Singapore 409030  
Tel: +65 6547 4827  
Fax: +65 6841 6071

E-mail (Sales) [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com)  
E-mail (Support) [support.apac@brtchip.com](mailto:support.apac@brtchip.com)

### Branch Office – Taipei, Taiwan

Bridgetek Pte Ltd, Taiwan Branch  
2 Floor, No. 516, Sec. 1, Nei Hu Road, Nei Hu District  
Taipei 114  
Taiwan, R.O.C.  
Tel: +886 (2) 8797 5691  
Fax: +886 (2) 8751 9737

E-mail (Sales) [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com)  
E-mail (Support) [support.apac@brtchip.com](mailto:support.apac@brtchip.com)

### Branch Office - Glasgow, United Kingdom

Bridgetek Pte. Ltd.  
Unit 1, 2 Seaward Place, Centurion Business Park  
Glasgow G41 1HH  
United Kingdom  
Tel: +44 (0) 141 429 2777  
Fax: +44 (0) 141 429 2758

E-mail (Sales) [sales.emea@brtchip.com](mailto:sales.emea@brtchip.com)  
E-mail (Support) [support.emea@brtchip.com](mailto:support.emea@brtchip.com)

### Branch Office – Vietnam

Bridgetek VietNam Company Limited  
Lutaco Tower Building, 5th Floor, 173A Nguyen Van  
Troi,  
Ward 11, Phu Nhuan District,  
Ho Chi Minh City, Vietnam  
Tel : 08 38453222  
Fax : 08 38455222

E-mail (Sales) [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com)  
E-mail (Support) [support.apac@brtchip.com](mailto:support.apac@brtchip.com)

### Web Site

<http://brtchip.com/>

### Distributor and Sales Representatives

Please visit the Sales Network page of the [Bridgetek Web site](#) for the contact details of our distributor(s) and sales representative(s) in your country.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Bridgetek Pte Ltd (BRTChip) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested Bridgetek devices and other materials) is provided for reference only. While Bridgetek has taken care to assure it is accurate, this information is subject to customer confirmation, and Bridgetek disclaims all liability for system designs and for any applications assistance provided by Bridgetek. Use of Bridgetek devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless Bridgetek from any and all damages, claims, suits or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. Bridgetek Pte Ltd, 178 Paya Lebar Road, #07-03, Singapore 409030. Singapore Registered Company Number: 201542387H.

## Appendix A – References

### Document References

[FT900/901/902/903 Datasheet](#)

[FT905/906/907/908 Datasheet](#)

[FT930/931/933 Datasheet](#)

[MM900EVxA datasheet](#)

[AN\\_324 FT9XX User Manual](#)

Eclipse Mosquitto <https://mosquitto.org/>

OpenSSL <https://www.openssl.org/>

IwIP <https://savannah.nongnu.org/projects/lwip/>

mbedTLS <https://tls.mbed.org/>

BRT\_AN\_041 FT90X MQTT Application - Source Code V1.0

### Acronyms and Abbreviations

Terms	Description
CA	Certificate Authority
IP	Internet Protocol
IC	Integrated Circuit
IoT	Internet of Things
MCU	Microcontroller Unit
TLS	Transport Layer Security
lwIP	lightweight IP
MQTT	Message Queuing Telemetry Transport
TLS	Transport Layer Security
ROM	Read Only Memory
RTOS	Real Time Operating System
SPI	Serial Peripheral Interface
SSL	Secure Socket Layer
UART	Universal Asynchronous Receiver / Transmitter

---

## Appendix B – List of Tables & Figures

### List of Tables

Table 1 - Project Files Overview .....	6
----------------------------------------	---

### List of Figures

Figure 1 - Block Diagram .....	4
Figure 2 - MQTT Schematic for Application .....	8
Figure 3 - FT90X UART Output when Connecting to MQTT Broker .....	11
Figure 4 - Eclipse Project Structure .....	13

## Appendix C– Revision History

Document Title: BRT\_AN\_041 FT90x MQTT Application  
Document Reference No.: BRT\_000240  
Clearance No.: BRT#132  
Product Page: <http://brtchip.com/ft93x/> & <http://brtchip.com/ft900/>  
Document Feedback: [Send Feedback](#)

Revision	Changes	Date
1.0	Initial version	2018-09-19