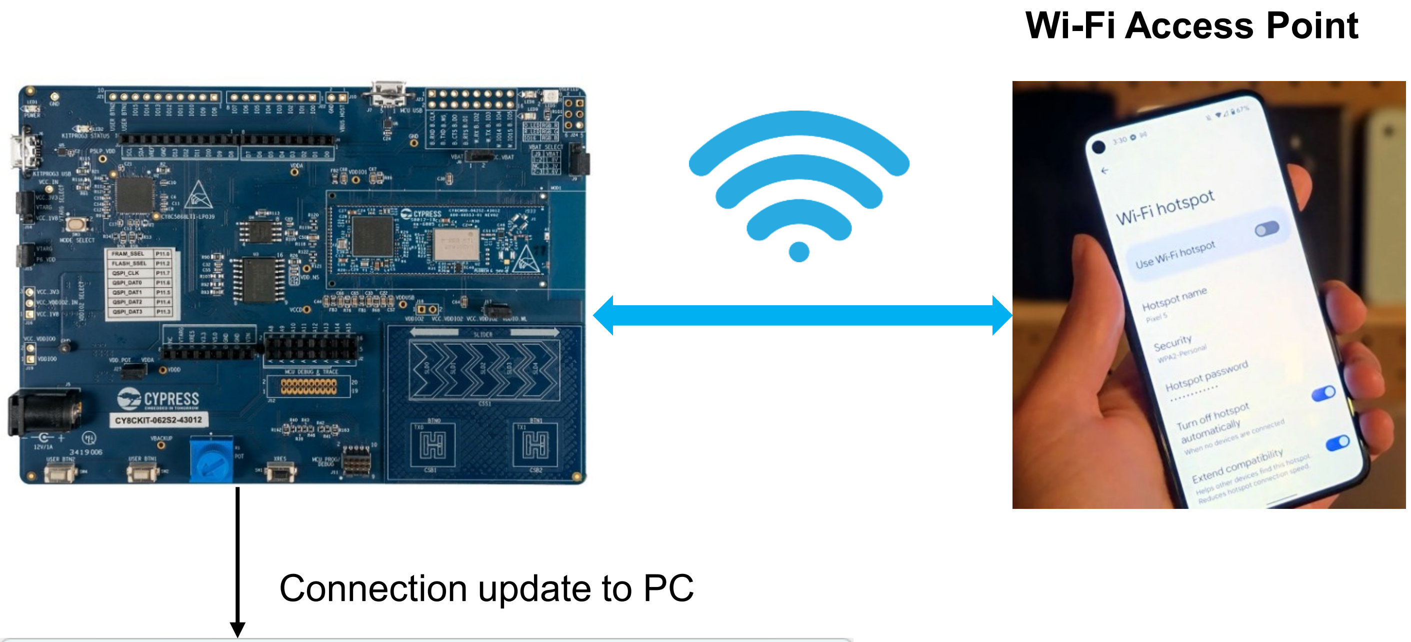
PSoC6 WIFI Hands On Training

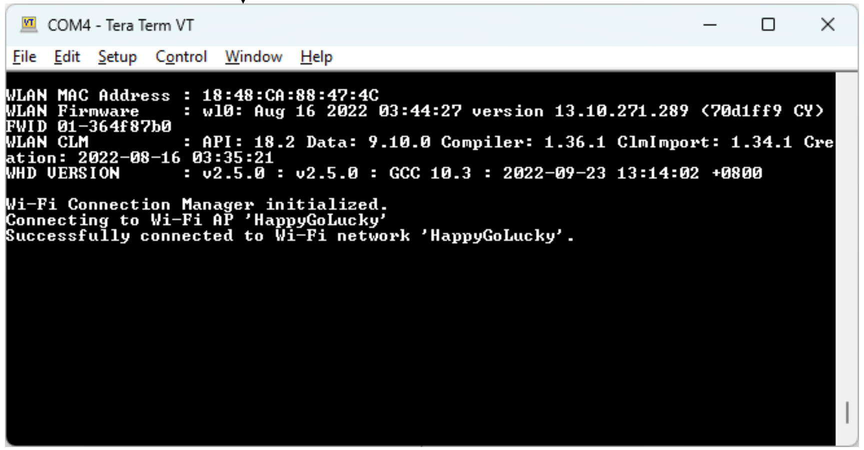
restricted

Connect to WPA2 Wi-Fi network

Objective

- Create an App from an empty template that attaches to a WPA2 AES PSK network, have LED1 turn on for success and blink for a failure.
- Objective is to understand what are the libraries and APIs needed for a simple Wi-Fi connection.





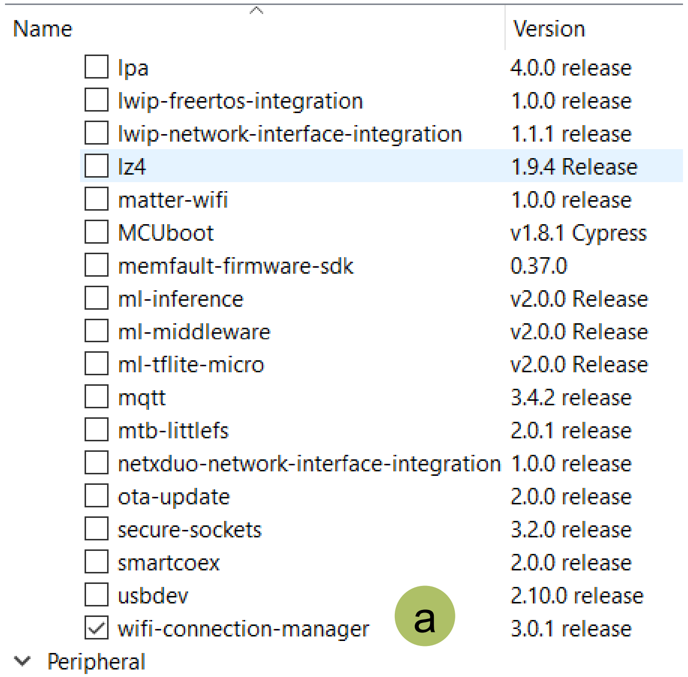
```
WLAN MAC Address : 18:48:CA:88:47:4C
WLAN Firmware    : wl0: Aug 16 2022 03:44:27 version 13.10.271.289 (70d1ff9 CY)
FWID 01-364f87b0
WLAN CLM         : API: 18.2 Data: 9.10.0 Compiler: 1.36.1 ClmImport: 1.34.1 Cre
ation: 2022-08-16 03:35:21
WHD VERSION      : v2.5.0 : v2.5.0 : GCC 10.3 : 2022-09-23 13:14:02 +0800

Wi-Fi Connection Manager initialized.
Connecting to Wi-Fi AP 'HappyGoLucky'
Successfully connected to Wi-Fi network 'HappyGoLucky'.
```

Create App and add Library

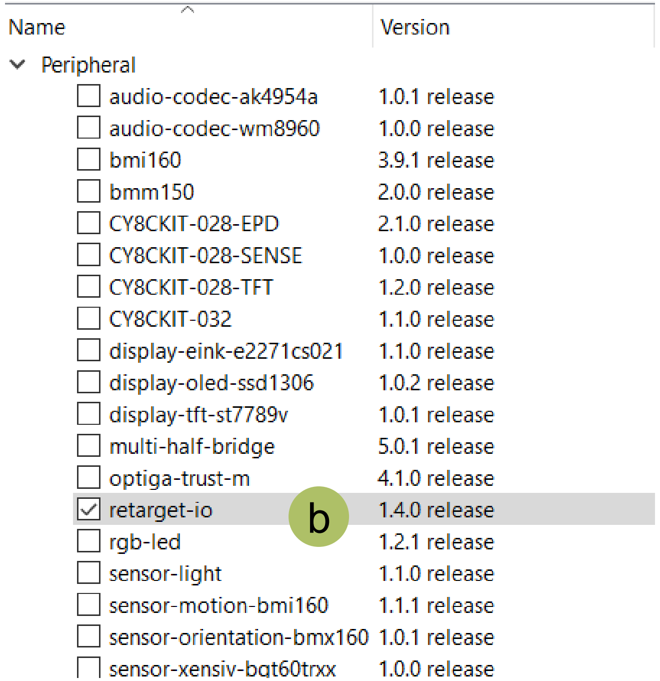
1. Create a new application named **WiFi_Connect** based on the **Empty_PSoC6_App** template.

2. Open the Library manager and add

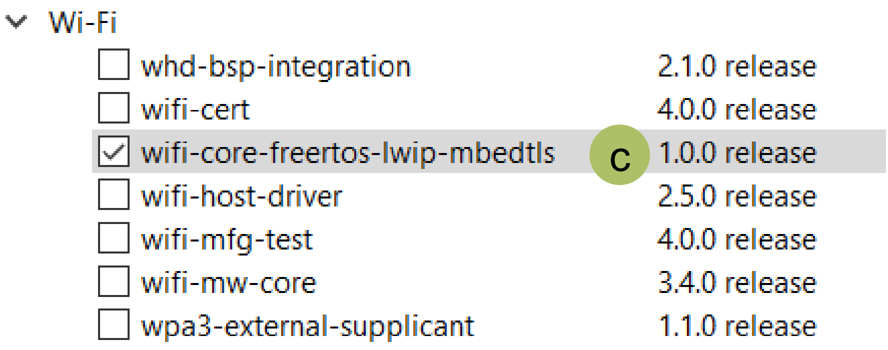
   i. Middleware >> **wifi-connection-manager**
   ii. Peripheral >> **retarget-io libraries**
   iii. Wi-Fi >> **wifi-core-freertos-lwip-mbedtls**.



Name	Version
□ lpa	4.0.0 release
□ lwip-freertos-integration	1.0.0 release
□ lwip-network-interface-integration	1.1.1 release
□ lz4	1.9.4 Release
□ matter-wifi	1.0.0 release
□ MCUboot	v1.8.1 Cypress
□ memfault-firmware-sdk	0.37.0
□ ml-inference	v2.0.0 Release
□ ml-middleware	v2.0.0 Release
□ ml-tflite-micro	v2.0.0 Release
□ mqtt	3.4.2 release
□ mtb-littlefs	2.0.1 release
□ netxduo-network-interface-integration	1.0.0 release
□ ota-update	2.0.0 release
□ secure-sockets	3.2.0 release
□ smartcoex	2.0.0 release
□ usbdev	2.10.0 release
☑ wifi-connection-manager	3.0.1 release
Peripheral	



Name	Version
Peripheral	
□ audio-codec-ak4954a	1.0.1 release
□ audio-codec-wm8960	1.0.0 release
□ bmi160	3.9.1 release
□ bmm150	2.0.0 release
□ CY8CKIT-028-EPD	2.1.0 release
□ CY8CKIT-028-SENSE	1.0.0 release
□ CY8CKIT-028-TFT	1.2.0 release
□ CY8CKIT-032	1.1.0 release
□ display-eink-e2271cs021	1.1.0 release
□ display-oled-ssd1306	1.0.2 release
□ display-tft-st7789v	1.0.1 release
□ multi-half-bridge	5.0.1 release
□ optiga-trust-m	4.1.0 release
☑ retarget-io	1.4.0 release
□ rgb-led	1.2.1 release
□ sensor-light	1.1.0 release
□ sensor-motion-bmi160	1.1.1 release
□ sensor-orientation-bmx160	1.0.1 release
□ sensor-xensiv-bgt60trxx	1.0.0 release

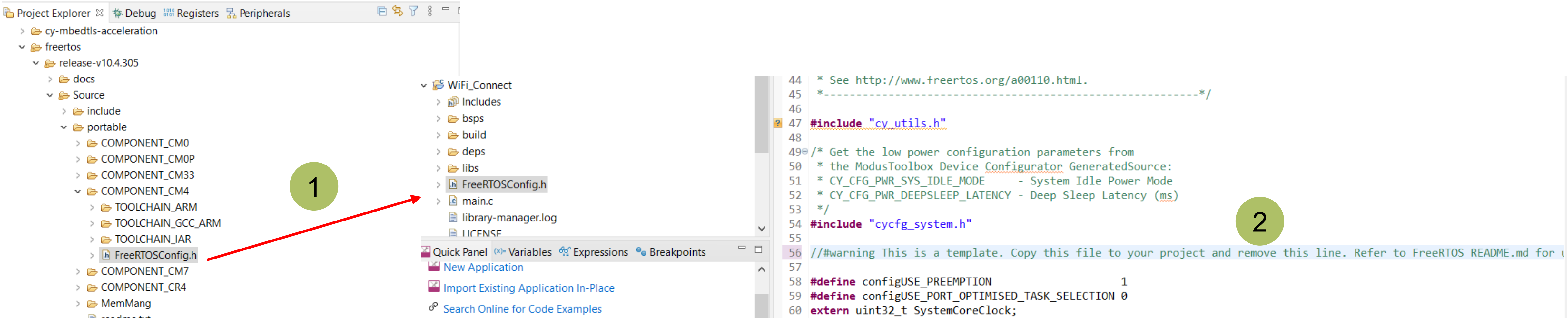


Wi-Fi	
□ whd-bsp-integration	2.1.0 release
□ wifi-cert	4.0.0 release
☑ wifi-core-freertos-lwip-mbedtls	1.0.0 release
□ wifi-host-driver	2.5.0 release
□ wifi-mfg-test	4.0.0 release
□ wifi-mw-core	3.4.0 release
□ wpa3-external-supplicant	1.1.0 release

Copy and Edit files

3. Copy **FreeRTOSConfig.h** from **mtb_shared/freertos/release-vX.X.X/Source/portable/COMPONENT_CM4** to your project root directory.

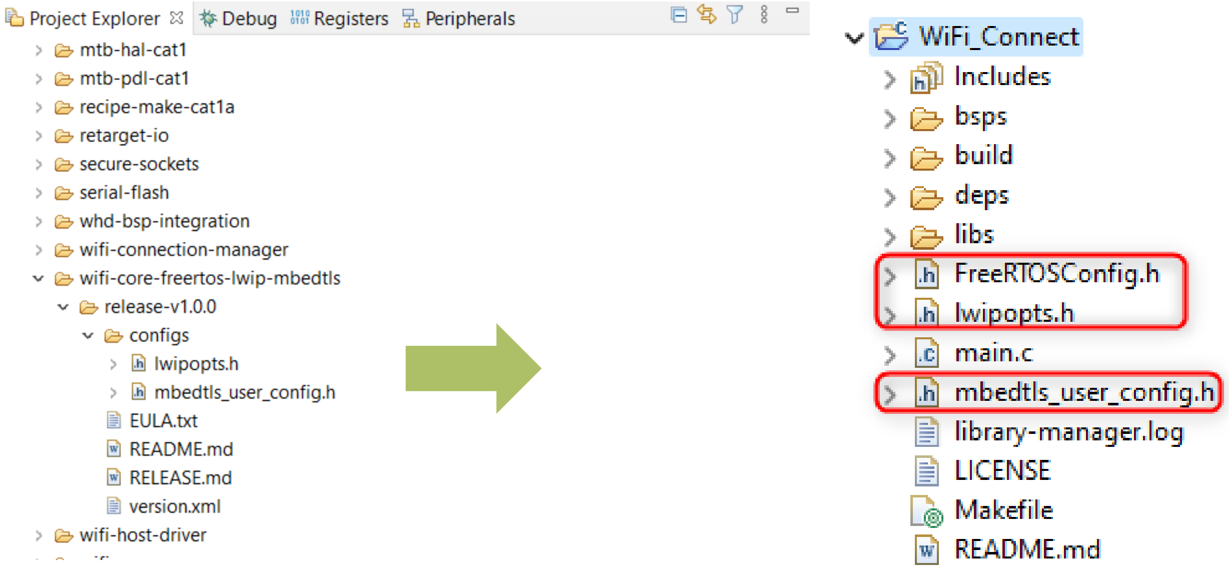
4. Open this file and **delete** the line that starts with #warning.



```
 * See http://www.freertos.org/a00110.html.
 *----------------------------------------------------------*/

#include "cy_utils.h"

/* Get the low power configuration parameters from
 * the ModusToolbox Device Configurator GeneratedSource:
 * CY_CFG_PWR_SYS_IDLE_MODE      - System Idle Power Mode
 * CY_CFG_PWR_DEEPSLEEP_LATENCY - Deep Sleep Latency (ms)
 */
#include "cycfg_system.h"

//#warning This is a template. Copy this file to your project and remove this line. Refer to FreeRTOS README.md for

#define configUSE_PREEMPTION                    1
#define configUSE_PORT_OPTIMISED_TASK_SELECTION 0
extern uint32_t SystemCoreClock;
```

Copy and Edit files

5. Copy the files from the **mtb_shared/wifi-core-freertos-lwip-mbedtls/release-vX.X/configs** directory to your root project directory. The files are:
   i. lwipopts.h
   ii. mbedtls_user_config.h

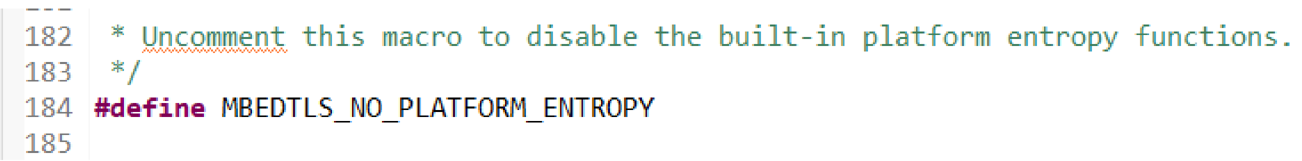


```
Project Explorer   Debug   Registers   Peripherals
> mtb-hal-cat1
> mtb-pdl-cat1
> recipe-make-cat1a
> retarget-io
> secure-sockets
> serial-flash
> whd-bsp-integration
> wifi-connection-manager
v wifi-core-freertos-lwip-mbedtls
  v release-v1.0.0
    v configs
      > lwipopts.h
      > mbedtls_user_config.h
      EULA.txt
      README.md
      RELEASE.md
      version.xml
> wifi-host-driver
```

```
v WiFi_Connect
  > Includes
  > bsps
  > build
  > deps
  > libs
  > FreeRTOSConfig.h
  > lwipopts.h
  > main.c
  > mbedtls_user_config.h
    library-manager.log
    LICENSE
    Makefile
    README.md
```

6. Open the **mbedtls_user_config.h** file and verify that the following line is not commented out:

#define MBEDTLS_NO_PLATFORM_ENTROPY

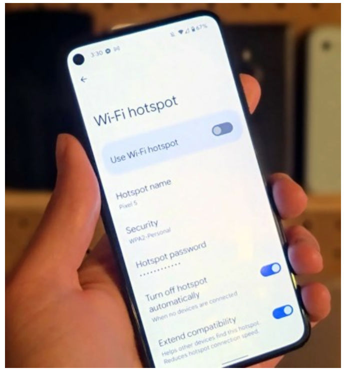


```
 * Uncomment this macro to disable the built-in platform entropy functions.
 */
#define MBEDTLS_NO_PLATFORM_ENTROPY

```

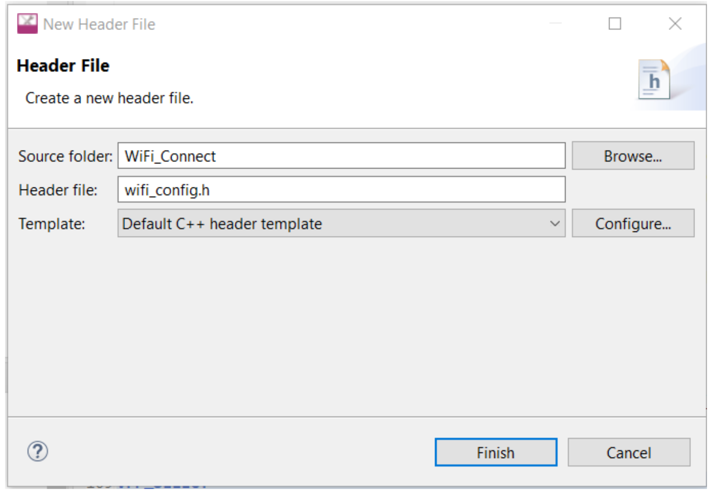
Create a file wifi_config.h

7. Copy the following code into a new header file called wifi_config.h

8. Edit WIFI_SSID and WIFI_PASSWORD to join to your AP (mobile hot spot)
   i. #define WIFI_SSID “MyHotSpot”
   ii. #define WIFI_PASSWORD “MyPassword”



```
#ifndef WIFI_CONFIG_H_
#define WIFI_CONFIG_H_

#include "cy_wcm.h"
/* SSID of the Wi-Fi Access Point to which the MQTT client connects.
*/
#define WIFI_SSID "xxxxxxxx"
/* Passkey of the above mentioned Wi-Fi SSID. */
#define WIFI_PASSWORD "xxxxxxxx"
/* Security type of the Wi-Fi access point. See 'cy_wcm_security_t'
structure
* in "cy_wcm.h" for more details. */
#define WIFI_SECURITY CY_WCM_SECURITY_WPA2_AES_PSK
/* Maximum Wi-Fi re-connection limit. */
#define MAX_WIFI_CONN_RETRIES (10u)
/* Wi-Fi re-connection time interval in milliseconds. */
#define WIFI_CONN_RETRY_INTERVAL_MS (2000)

#endif /* WIFI_CONFIG_H_ */
```


New Header File

Header File

Create a new header file.

Source folder: WiFi_Connect

Header file: wifi_config.h

Template: Default C++ header template

Configure the makefile

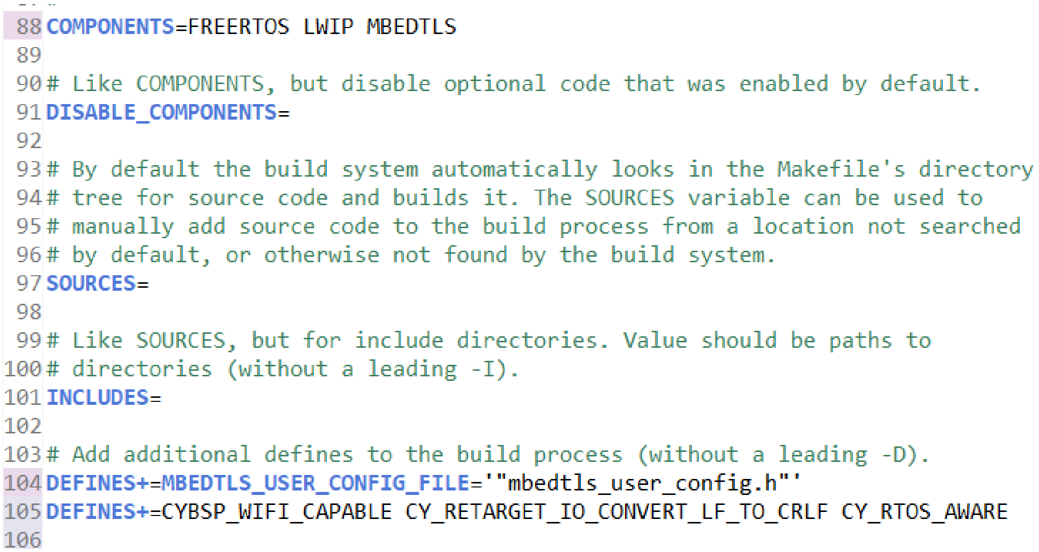
9. Add the following lines to your project's Makefile:

COMPONENTS=FREERTOS LWIP MBEDTLS

DEFINES+=MBEDTLS_USER_CONFIG_FILE=‘”mbedtls_user_config.h”’

DEFINES+=CYBSP_WIFI_CAPABLE CY_RETARGET_IO_CONVERT_LF_TO_CRLF CY_RTOS_AWARE

Note: There are blank COMPONENTS and DEFINES lines in the file that you can modify.



```
COMPONENTS=FREERTOS LWIP MBEDTLS

# Like COMPONENTS, but disable optional code that was enabled by default.
DISABLE_COMPONENTS=

# By default the build system automatically looks in the Makefile's directory
# tree for source code and builds it. The SOURCES variable can be used to
# manually add source code to the build process from a location not searched
# by default, or otherwise not found by the build system.
SOURCES=

# Like SOURCES, but for include directories. Value should be paths to
# directories (without a leading -I).
INCLUDES=

# Add additional defines to the build process (without a leading -D).
DEFINES+=MBEDTLS_USER_CONFIG_FILE='"mbedtls_user_config.h"'
DEFINES+=CYBSP_WIFI_CAPABLE CY_RETARGET_IO_CONVERT_LF_TO_CRLF CY_RTOS_AWARE

```

Main code

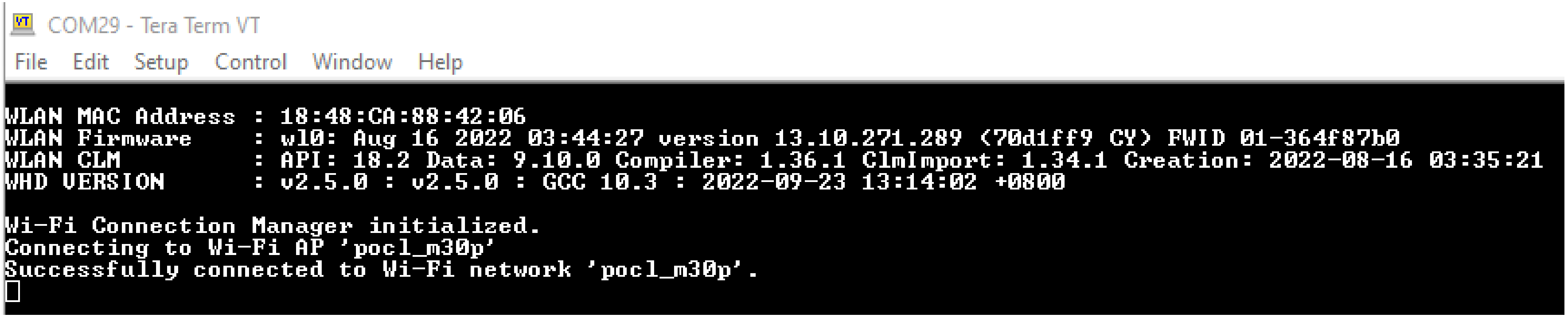
10. Add the following code into main.c

```c
/*******************************************************************************
* Header Files
*******************************************************************************/
#include "cyhal.h"
#include "cybsp.h"
#include "FreeRTOS.h"
#include "task.h"
#include "wifi_config.h"
#include "cy_retarget_io.h"
#include "cy_wcm.h"


/*******************************************************************************
* Function Definitions
*******************************************************************************/
void wifi_connect(void *arg)
{
   cy_rslt_t result;
   cy_wcm_connect_params_t connect_param;
   cy_wcm_ip_address_t ip_address;
   uint32_t retry_count;
   /* Configure the interface as a Wi-Fi STA (i.e. Client) and initialize the WCM. */
   cy_wcm_config_t config = {.interface = CY_WCM_INTERFACE_TYPE_STA};
   cy_wcm_init(&config);
   printf("\nWi-Fi Connection Manager initialized.\n");
   /* Configure the connection parameters for the Wi-Fi interface. */
   memset(&connect_param, 0, sizeof(cy_wcm_connect_params_t));
   memcpy(connect_param.ap_credentials.SSID, WIFI_SSID, sizeof(WIFI_SSID));
   memcpy(connect_param.ap_credentials.password, WIFI_PASSWORD, sizeof(WIFI_PASSWORD));
   connect_param.ap_credentials.security = WIFI_SECURITY;
   /* Connect to the Wi-Fi AP. */
   for (retry_count = 0; retry_count < MAX_WIFI_CONN_RETRIES; retry_count++)
   {
      printf("Connecting to Wi-Fi AP '%s'\n", connect_param.ap_credentials.SSID);
      result = cy_wcm_connect_ap(&connect_param, &ip_address);
      if (result == CY_RSLT_SUCCESS)
      {
         printf("Successfully connected to Wi-Fi network '%s'.\n",connect_param.ap_credentials.SSID);
         break;
      }
   }
   for(;;) {
   //Enter code to handle LED
   }
}
```

```c
int main(void)
{
   cy_rslt_t result;
   /* Initialize the device and board peripherals */
   result = cybsp_init() ;
   if (result != CY_RSLT_SUCCESS)
   {
      CY_ASSERT(0);
   }
   /* Initialize retarget-io to use the debug UART port. */
   cy_retarget_io_init(CYBSP_DEBUG_UART_TX, CYBSP_DEBUG_UART_RX, CY_RETARGET_IO_BAUDRATE);
   __enable_irq();
   printf("\x1b[2J\x1b[;H\n"); /* ANSI ESC sequence to clear screen. */
   /* Create the WiFi connection task. */
   xTaskCreate(wifi_connect, "wifi_connect_task", 1024, NULL, 5, NULL);
   vTaskStartScheduler(); /* Never Returns */

   for (;;)
      {
      }
}
```

Built and Run the program

- Build and run the program
- Open Tera Term, 115200 baud rate, turn on mobile hot spot
- Device will try up to 10 times to connect to network.
- UART terminal display “Successfully connected to Wi-Fi network ‘my_network’ if device connects to network.



```
WLAN MAC Address : 18:48:CA:88:42:06
WLAN Firmware    : wl0: Aug 16 2022 03:44:27 version 13.10.271.289 (70d1ff9 CY) FWID 01-364f87b0
WLAN CLM         : API: 18.2 Data: 9.10.0 Compiler: 1.36.1 ClmImport: 1.34.1 Creation: 2022-08-16 03:35:21
WHD VERSION      : v2.5.0 : v2.5.0 : GCC 10.3 : 2022-09-23 13:14:02 +0800

Wi-Fi Connection Manager initialized.
Connecting to Wi-Fi AP 'pocl_m30p'
Successfully connected to Wi-Fi network 'pocl_m30p'.
```

Secure TCP connection

Secure TCP connection

Using a Secure TCP client and server example application to understand secure TCP connection implementation.

1. Find a partner. One to create new application “Wi-F Secure TCP server” and the other “Wi-Fi Secure TCP client”

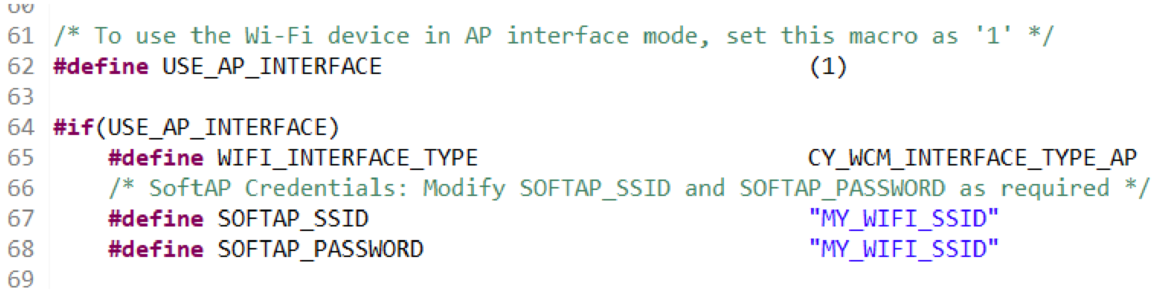
2. For Secure TCP server, edit the following in network_credential.h
   i. set #define USE_AP_INTERFACE to 1
   ii. edit #define SOFTAP_SSID
   iii. Edit #define SOFTAP_PASSWORD

   Note: For TCP Server, please use a unique SSID. TCP Server also acts as access point.

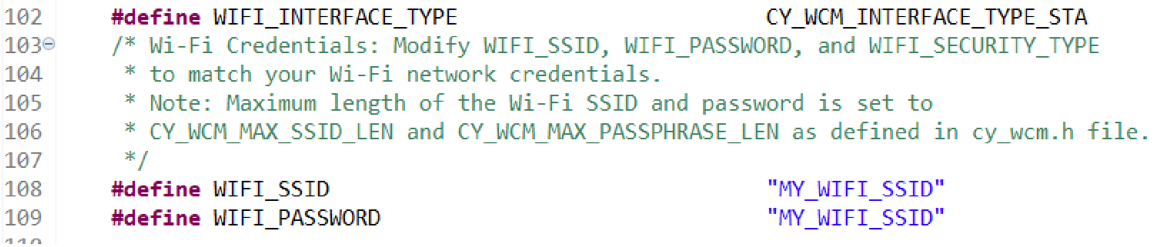
3. For Secure TCP client, edit the following in network_credential.h
   i. edit #define WIFI_SSID
   ii. Edit #define WIFI_PASSWORD to be the same as TCP server

   Note: For TCP client, please match SSID to server

4. Build and program the boards. Open tera term (115200 baud rate)



```
/* To use the Wi-Fi device in AP interface mode, set this macro as '1' */
#define USE_AP_INTERFACE                                         (1)

#if(USE_AP_INTERFACE)
    #define WIFI_INTERFACE_TYPE                                  CY_WCM_INTERFACE_TYPE_AP
    /* SoftAP Credentials: Modify SOFTAP_SSID and SOFTAP_PASSWORD as required */
    #define SOFTAP_SSID                                          "MY_WIFI_SSID"
    #define SOFTAP_PASSWORD                                      "MY_WIFI_SSID"

```

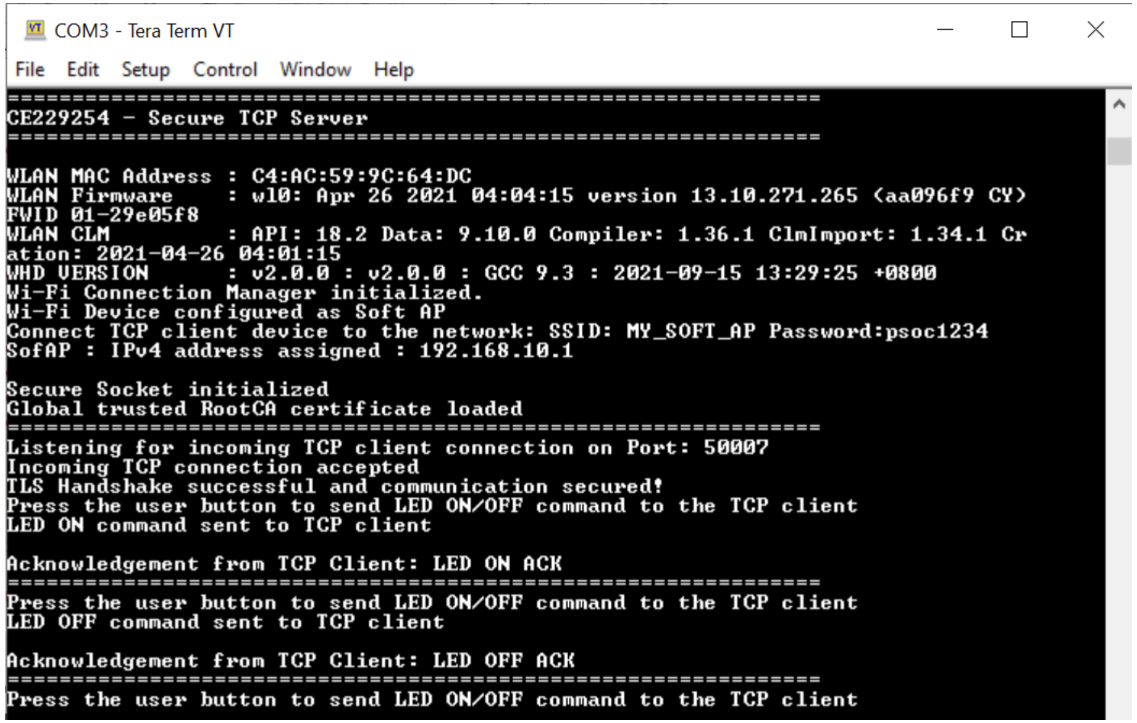


```
    #define WIFI_INTERFACE_TYPE                                  CY_WCM_INTERFACE_TYPE_STA
    /* Wi-Fi Credentials: Modify WIFI_SSID, WIFI_PASSWORD, and WIFI_SECURITY_TYPE
     * to match your Wi-Fi network credentials.
     * Note: Maximum length of the Wi-Fi SSID and password is set to
     * CY_WCM_MAX_SSID_LEN and CY_WCM_MAX_PASSPHRASE_LEN as defined in cy_wcm.h file.
     */
    #define WIFI_SSID                                            "MY_WIFI_SSID"
    #define WIFI_PASSWORD                                        "MY_WIFI_SSID"
```

Results

TCP Server

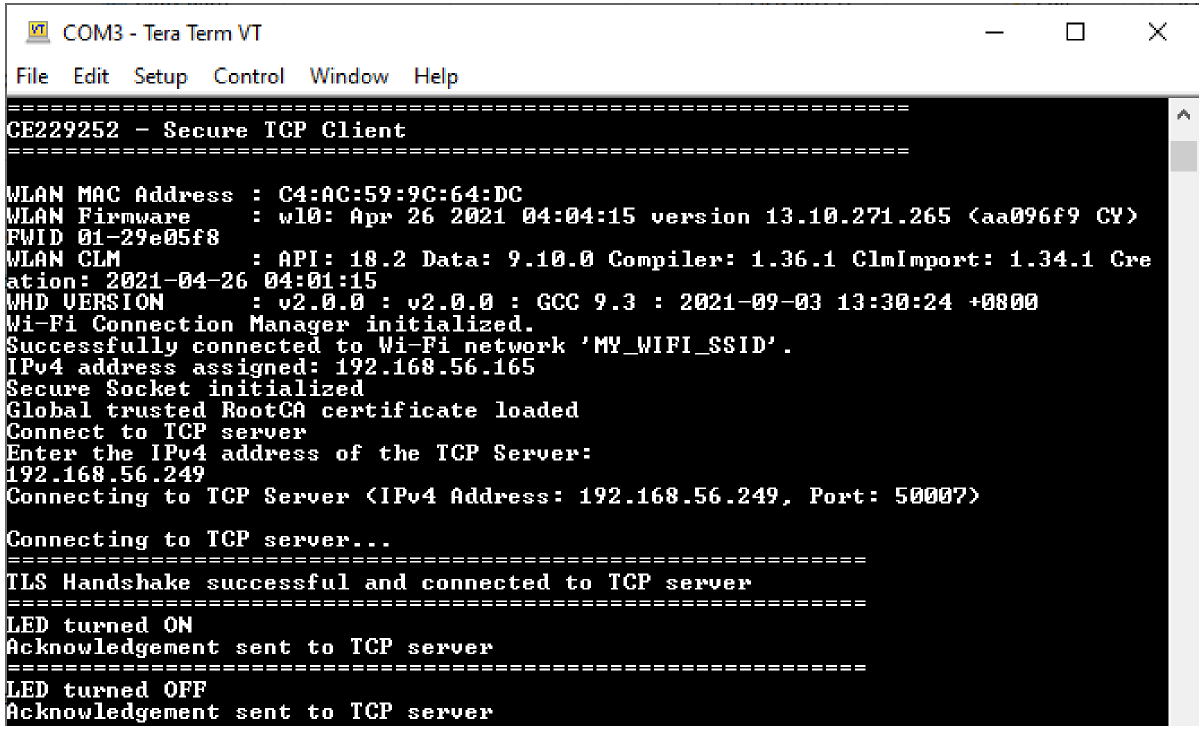
- Press the user button (CYBSP_USER_BTN) to send LED ON/OFF command to client.
- Each user button press will issue the LED ON or LED OFF commands alternately. The client in turn sends an acknowledgement message back to the server.



```
================================================================
CE229254 - Secure TCP Server
================================================================

WLAN MAC Address : C4:AC:59:9C:64:DC
WLAN Firmware    : wl0: Apr 26 2021 04:04:15 version 13.10.271.265 (aa096f9 CY)
FWID 01-29e05f8
WLAN CLM         : API: 18.2 Data: 9.10.0 Compiler: 1.36.1 ClmImport: 1.34.1 Cr
ation: 2021-04-26 04:01:15
WHD VERSION      : v2.0.0 : v2.0.0 : GCC 9.3 : 2021-09-15 13:29:25 +0800
Wi-Fi Connection Manager initialized.
Wi-Fi Device configured as Soft AP
Connect TCP client device to the network: SSID: MY_SOFT_AP Password:psoc1234
SofAP : IPv4 address assigned : 192.168.10.1

Secure Socket initialized
Global trusted RootCA certificate loaded
================================================================
Listening for incoming TCP client connection on Port: 50007
Incoming TCP connection accepted
TLS Handshake successful and communication secured!
Press the user button to send LED ON/OFF command to the TCP client
LED ON command sent to TCP client

Acknowledgement from TCP Client: LED ON ACK
================================================================
Press the user button to send LED ON/OFF command to the TCP client
LED OFF command sent to TCP client

Acknowledgement from TCP Client: LED OFF ACK
================================================================
Press the user button to send LED ON/OFF command to the TCP client
```

TCP Client

- Enter TPC Server address
- On receiving LED ON/OFF command from server, client will ON/OFF LED and send acknowledgement message back to server.

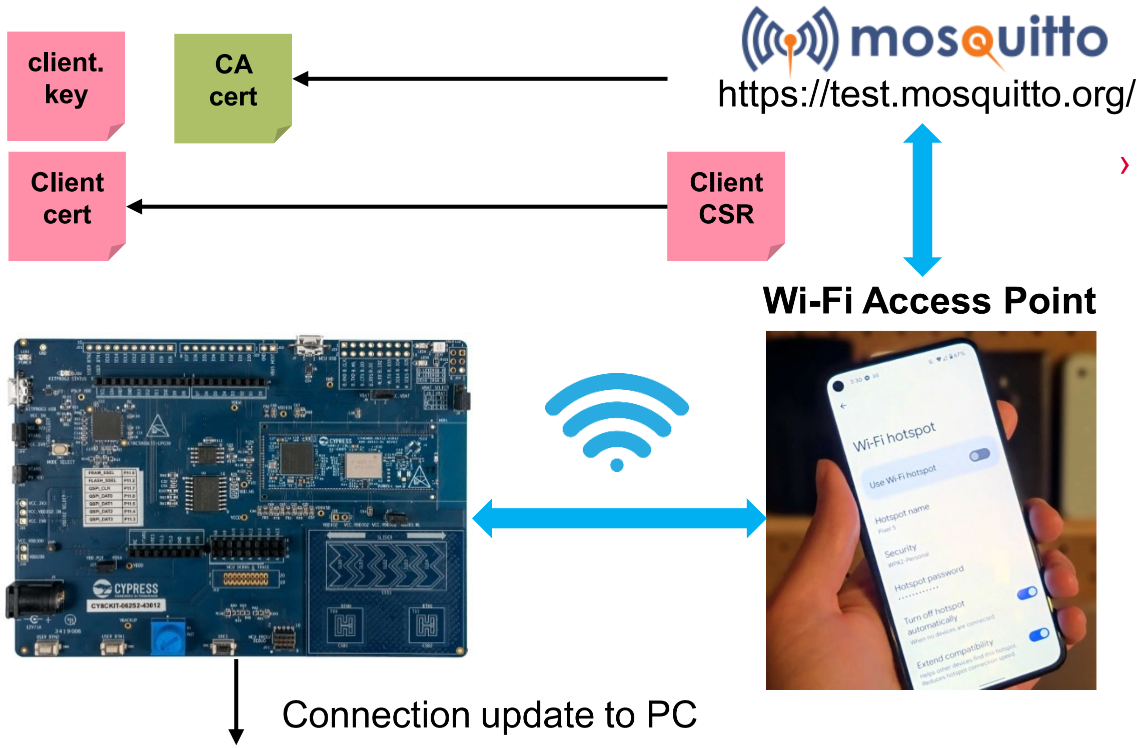


```
================================================================
CE229252 - Secure TCP Client
================================================================

WLAN MAC Address : C4:AC:59:9C:64:DC
WLAN Firmware    : wl0: Apr 26 2021 04:04:15 version 13.10.271.265 (aa096f9 CY)
FWID 01-29e05f8
WLAN CLM         : API: 18.2 Data: 9.10.0 Compiler: 1.36.1 ClmImport: 1.34.1 Cre
ation: 2021-04-26 04:01:15
WHD VERSION      : v2.0.0 : v2.0.0 : GCC 9.3 : 2021-09-03 13:30:24 +0800
Wi-Fi Connection Manager initialized.
Successfully connected to Wi-Fi network 'MY_WIFI_SSID'.
IPv4 address assigned: 192.168.56.165
Secure Socket initialized
Global trusted RootCA certificate loaded
Connect to TCP server
Enter the IPv4 address of the TCP Server:
192.168.56.249
Connecting to TCP Server (IPv4 Address: 192.168.56.249, Port: 50007)

Connecting to TCP server...
============================================================
TLS Handshake successful and connected to TCP server
============================================================
LED turned ON
Acknowledgement sent to TCP server
============================================================
LED turned OFF
Acknowledgement sent to TCP server
```

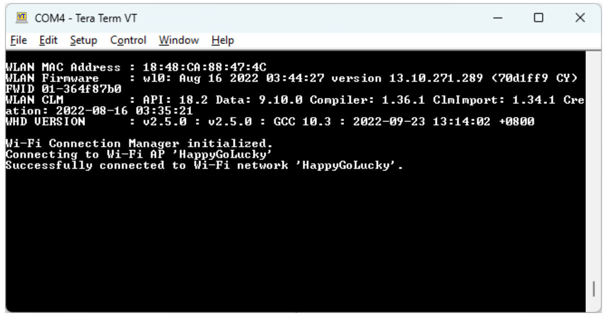
MQTT via Mosquitto broker

Objective

- Understand the concept of public key cryptography
- To generate key pairs and certificate signing request
- Review the code for MQTT connection.



- Return command to toggle LED when button 1 is pressed



```
WLAN MAC Address : 18:48:CA:88:47:4C
WLAN Firmware    : wl0: Aug 16 2022 03:44:27 version 13.10.271.289 (70d1ff9 CY)
FWID 01-364f87b0
WLAN CLM         : API: 18.2 Data: 9.10.0 Compiler: 1.36.1 ClmImport: 1.34.1 Cre
ation: 2022-08-16 03:35:21
WHD VERSION      : v2.5.0 : v2.5.0 : GCC 10.3 : 2022-09-23 13:14:02 +0800

Wi-Fi Connection Manager initialized.
Connecting to Wi-Fi AP 'HappyGoLucky'
Successfully connected to Wi-Fi network 'HappyGoLucky'.
```

Edit wifi_config.h

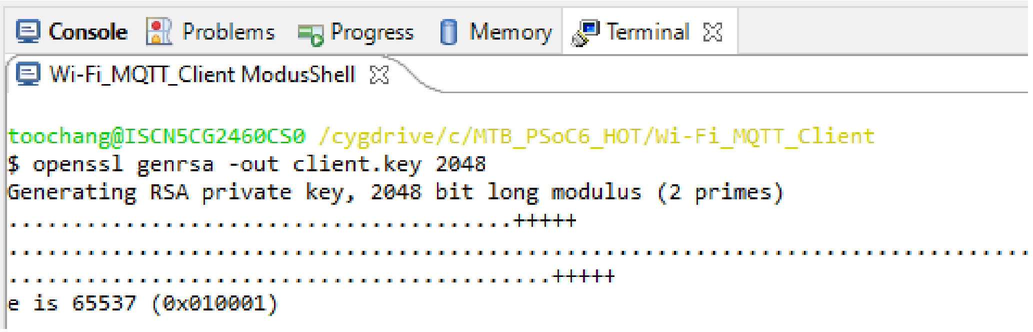
1. Create a new WiFi_MQTT_Client application

2. Use your mobile hotspot as access point. Edit “**wifi_config.h**” to match your mobile SSID setting
   i. #define WIFI_SSID “mobile_ssid”
   ii. #define WIFI_PASSWORD “password”

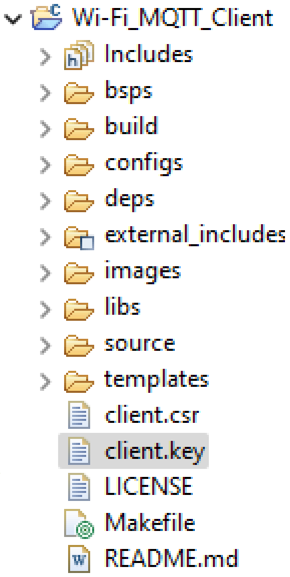
Generate Client key

3. Run the following commands with Terminal to generate the Client key (Client.key).

   i. openssl genrsa –out client.key 2048



```
toochang@ISCN5CG2460CS0 /cygdrive/c/MTB_PSoC6_HOT/Wi-Fi_MQTT_Client
$ openssl genrsa -out client.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
......................................+++++
..................................................................................
...........................................+++++
e is 65537 (0x010001)
```



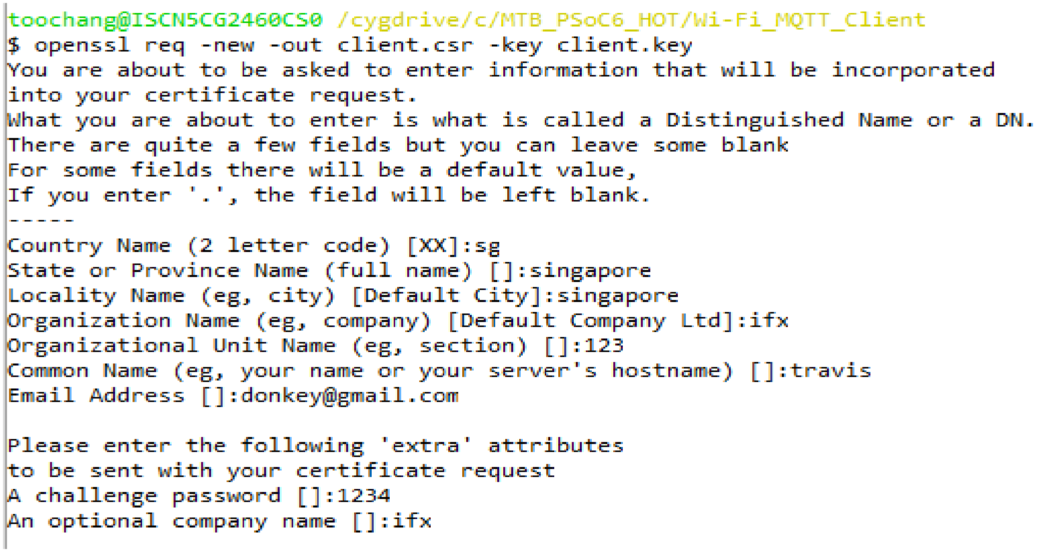
- Wi-Fi_MQTT_Client
  - Includes
  - bsps
  - build
  - configs
  - deps
  - external_includes
  - images
  - libs
  - source
  - templates
  - client.csr
  - client.key
  - LICENSE
  - Makefile
  - README.md

Generate Client Certificate

Generate the Client certificate (client.csr) with the following commands

ii. openssl req –out client.csr –key client.key –new

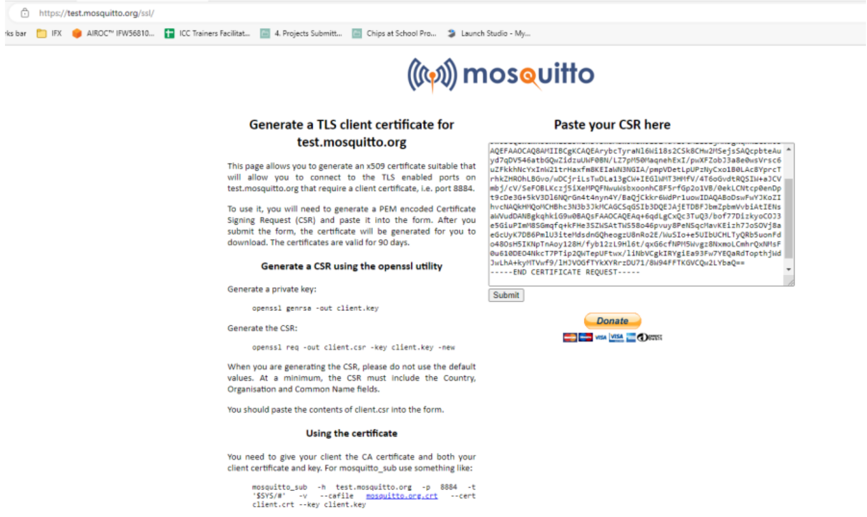
iii. Enter info required for the certificate signing request



```
toochang@ISCN5CG2460CS0 /cygdrive/c/MTB_PSoC6_HOT/Wi-Fi_MQTT_Client
$ openssl req -new -out client.csr -key client.key
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:sg
State or Province Name (full name) []:singapore
Locality Name (eg, city) [Default City]:singapore
Organization Name (eg, company) [Default Company Ltd]:ifx
Organizational Unit Name (eg, section) []:123
Common Name (eg, your name or your server's hostname) []:travis
Email Address []:donkey@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:1234
An optional company name []:ifx
```

4. Open client.csr, copy and paste content to https://test.mosquitto.org/ssl/

i. Save the generated client certificate into the project root folder



Generate a TLS client certificate for test.mosquitto.org

This page allows you to generate an x509 certificate suitable that will allow you to connect to the TLS enabled ports on test.mosquitto.org that require a client certificate, i.e. port 8884.

To use it, you will need to generate a PEM encoded Certificate Signing Request (CSR) and paste it into the form. After you submit the form, the certificate will be generated for you to download. The certificates are valid for 90 days.

Generate a CSR using the openssl utility

Generate a private key:

```
openssl genrsa -out client.key
```

Generate the CSR:

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

When you are generating the CSR, please do not use the default values. At a minimum, the CSR must include the Country, Organisation and Common Name fields.

You should paste the contents of client.csr into the form.

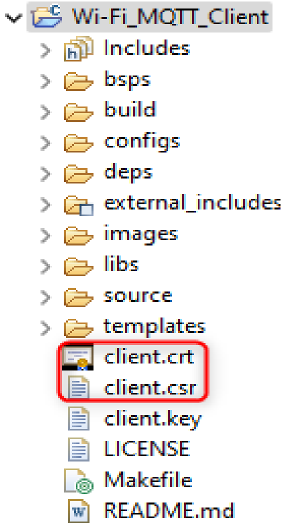
Using the certificate

You need to give your client the CA certificate and both your client certificate and key. For mosquitto_sub use something like:

```
mosquitto_sub -h test.mosquitto.org -p 8884 -t '$SYS/#' -v --cafile mosquitto.org.crt --cert client.crt --key client.key
```

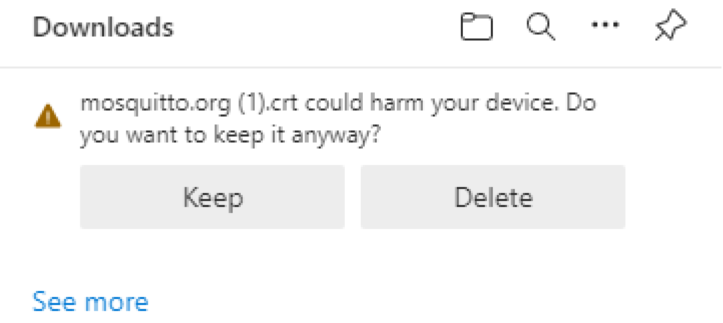
Paste your CSR here

AQEFAAOCAQ8AMIIBCgKCAQEArybcTyraNl6Wi18s2CSk8CHw2MSejsSAQcpbteAu
yd7qDV546atbGQwZidzuUWF0BN/LZ7pM50MaqnehExI/pwXFZobJ3a8e0wsVrsc6
uZFkkhNcYxInW21trHaxfm8KEIaWN3NGIA/pmpVDetLpUPzNyCxo1B0LAc8YprcT
rhkZHROhLBGvo/wDCjriLsTwDLa13gCW+IEGlWMT3HMfV/4T6oGvdtRQSIW+aJCV
mbj/cV/SeFOBLKczj5iXeMPQFNwuWsbxoonhC8F5rfGp2o1VB/0ekLCNtcp0enDp
t9cDe3G+5kV3Dl6NQrGn4t4nyn4Y/BaQjCkkr6WdPr1uowIDAQABoDswFwYJKoZI
hvcNAQkHMQoMCHBhc3N3b3JkMCAGCSqGSIb3DQEJAjETDBFJbmZpbmVvbiAtIENs
aWVudDANBgkqhkiG9w0BAQsFAAOCAQEAq+6qdLgCxQc3TuQ3/bof77DizkyoCOJ3
e5GiuPImM8SGmqfq+kFHe3SZWSAtTWS58o46pvuy8PeNSqcMavKEizh7JoSOVj8a
eGcUyK7DB6PmlU3iteMdsdnGQheogzU8nRo2E/WuSIo+e5UIbUCHLTyQRb5uonFd
o48OsH5IKNpTnAoy128H/fyb12zL9Hl6t/qxG6cfNPM5Wvgz8NxmoLCmhrQxNMsF
0u610DEO4NkcT7PTip2QWTepUFtwx/liNbVCgkIRYgiEa93Fw7YEQaRdTopthjWd
JwLhA+kyMTVwf9/lHJVOGfTYkXYRrzDU71/8W94FFTKGVCQw2LYbaQ==
-----END CERTIFICATE REQUEST-----

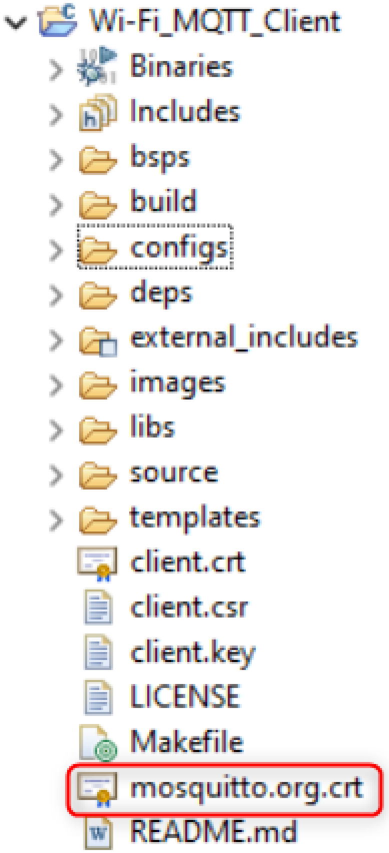


Download CA certificate

5. Download CA certificate from http://test.mosquitto.org/ssl/mosquitto.org.crt and save it to project root folder

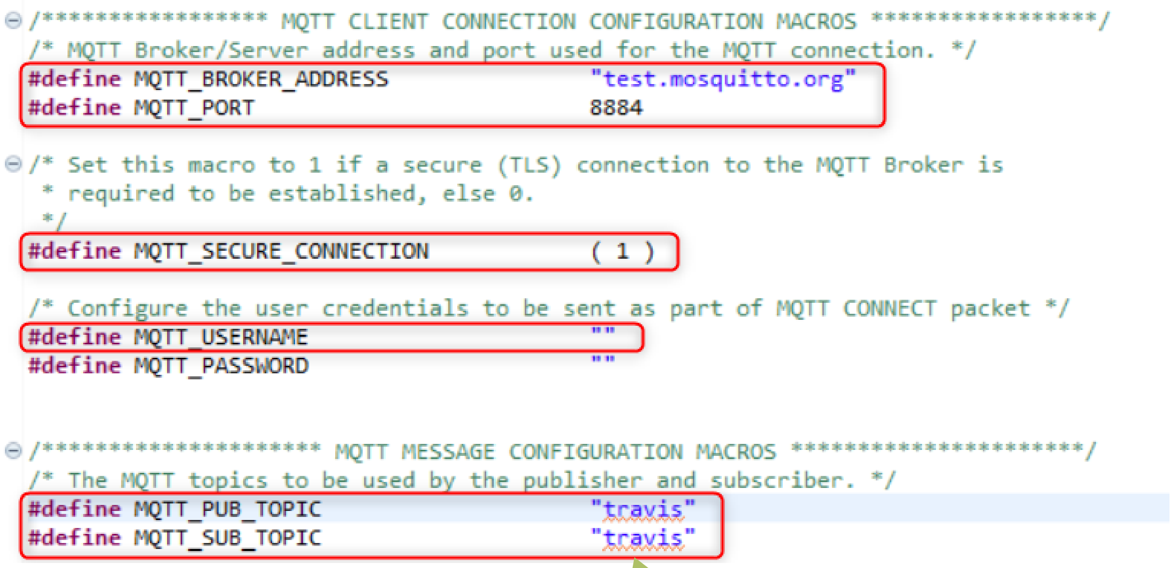


mosquitto.org (1).crt could harm your device. Do you want to keep it anyway?



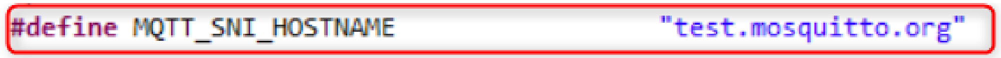
Edit file mqtt_client.h

6. Edit the following in “configs/mqtt_client.h”
   i. MQTT_BROKER_ADDRESS to “test.mosquitto.org”
   ii. Set the macros MQTT_PORT to 8884
   iii. MQTT_SECURE_CONNECTION to 1
   iv. MQTT_USERNAME to “”
   v. MQTT_PUB_TOPIC to “unique_topic”
   vi. MQTT_SUB_TOPIC to “unique_topic”
   vii. MQTT_SNI_HOSTNAME to “test.mosquitto.org”



```
/***************** MQTT CLIENT CONNECTION CONFIGURATION MACROS *****************/
/* MQTT Broker/Server address and port used for the MQTT connection. */
#define MQTT_BROKER_ADDRESS                            "test.mosquitto.org"
#define MQTT_PORT                                      8884

/* Set this macro to 1 if a secure (TLS) connection to the MQTT Broker is
 * required to be established, else 0.
 */
#define MQTT_SECURE_CONNECTION                         ( 1 )

/* Configure the user credentials to be sent as part of MQTT CONNECT packet */
#define MQTT_USERNAME                                  ""
#define MQTT_PASSWORD                                  ""


/******************** MQTT MESSAGE CONFIGURATION MACROS *********************/
/* The MQTT topics to be used by the publisher and subscriber. */
#define MQTT_PUB_TOPIC                                 "travis"
#define MQTT_SUB_TOPIC                                 "travis"
```

Use your own name

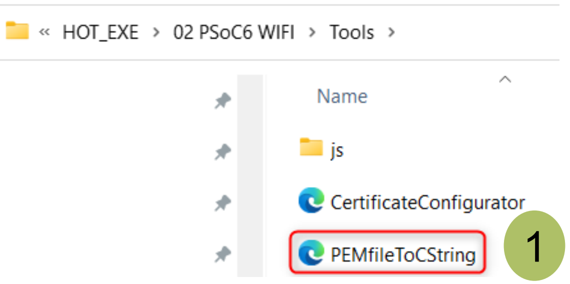


```
#define MQTT_SNI_HOSTNAME                              "test.mosquitto.org"
```

Edit file mqtt_client.h

7. Open the PEMfileToCstring

8. Use PEM to C String Conversion Tool to generate data format and copy to appropriate define in “**configs/mqtt_client_config.h**”
   1. client.crt -> #define CLIENT_CERTIFICATE
   2. client.key -> #define CLIENT_PRIVATE_KEY
   3. mosquitto.org.crt -> #define ROOT_CA_CERTIFICATE

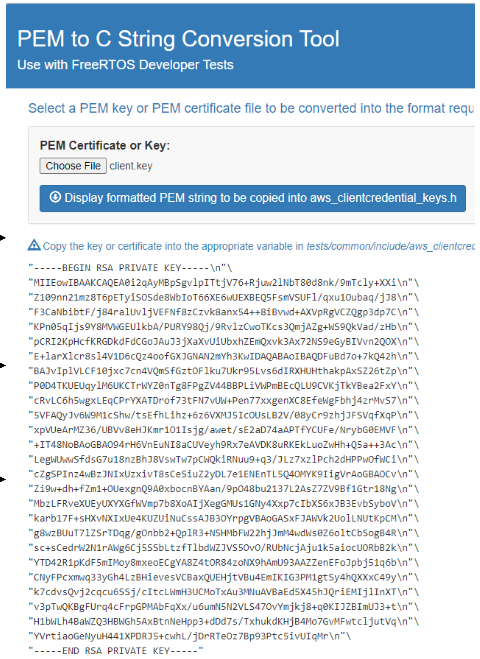


2

client.crt

client.key

mosquitto.org.crt



PEM to C String Conversion Tool

Use with FreeRTOS Developer Tests

Select a PEM key or PEM certificate file to be converted into the format requ

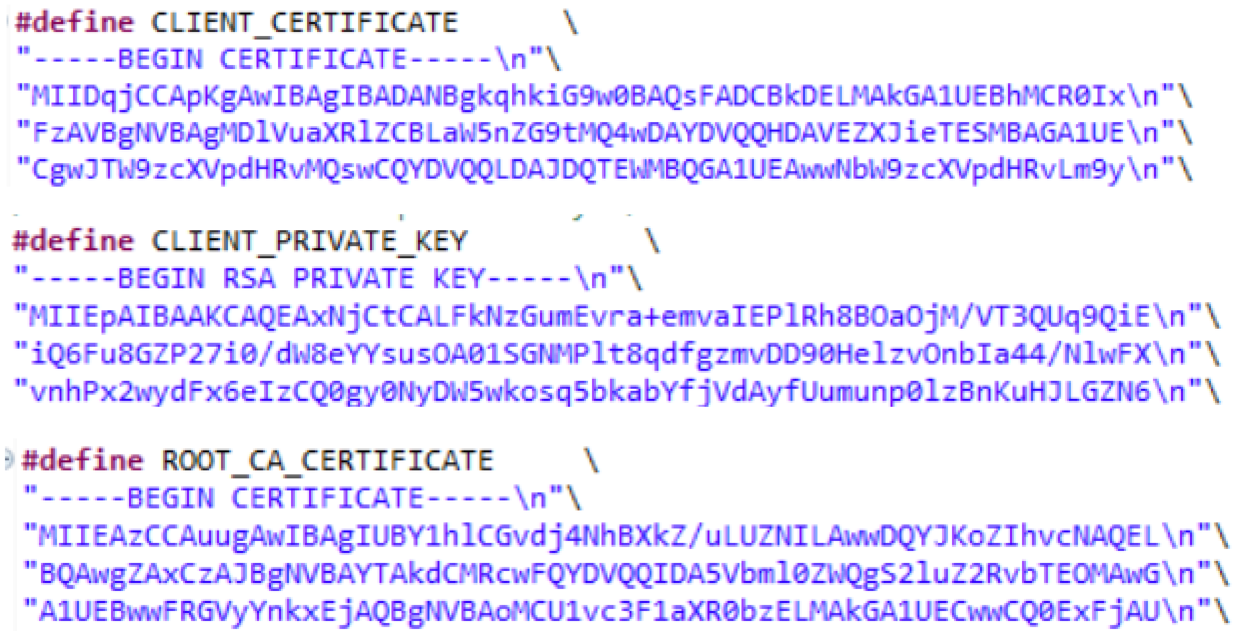
PEM Certificate or Key:

Choose File client.key

Display formatted PEM string to be copied into aws_clientcredential_keys.h

Copy the key or certificate into the appropriate variable in *tests/common/include/aws_clientcred*

```
"-----BEGIN RSA PRIVATE KEY-----\n"\
"MIIEowIBAAKCAQEA0i2qAyMBpSgvlpITtjV76+Rjuw2lNbT80d8nk/9mTcly+XXi\n"\
"Z109nn21mz8T6pETyiSOSde8WbIoT66XE6wUEXBEQ5FsmVSUFl/qxu1Oubaq/jJ8\n"\
"F3CaNbibtF/j84ralUvljVEFNf8zCzvk8anx54++8iBvwd+AXVpRgVCZQgp3dp7C\n"\
"KPn05qIjs9Y8MVWGEUlkbA/PURY98Qj/9RvlzCwoTKcs3QmjAZg+WS9QkVad/zHb\n"\
"pCRI2KpHcfKRGDkdFdCGoJAuJ3jXaXvUiUbxhZEmQxvk3Ax72NS9eGyBIVvn2QOX\n"\
"E+larXlcr8sl4V1D6cQz4oofGXJGNAN2mYh3KwIDAQABAoIBAQDFuBd7o+7kQ42h\n"\
"BAJvIplVLCF10jxc7cn4VQmSfGztOFlku7Ukr95Lvs6dIRXHUHthakpAxSZ26tZp\n"\
"P0D4TKUEUqylM6UKCTrWYZ0nTg8FPgZV44BBPLiVWPmBEcQLU9CVKjTkYBea2FxY\n"\
"cRvLC6h5wgxLEqCPrYXATDrof73tFN7vUW+Pen77xxgenXC8EfeWgFbhj4zrMvS7\n"\
"5VFAQyJv6W9M1cShw/tsEfhLihz+6z6VXMJ5IcOUsLB2V/08yCr9zhjJFSVqfXqP\n"\
"xpVUeArMZ36/UBVv8eHJKmr1O1Isjg/awet/sE2aD74aAPTfYCUFe/NrybG0EMVF\n"\
"+IT48NoBAoGBAO94rH6VnEuNI8aCUVeyh9Rx7eAVDK8uRKEkLuoZwHh+Q5a++3Ac\n"\
"LegWUwwSfdsG7u18nzBhJ8VswTw7pCWQkiRNuu9+q3/JLz7xzlPch2dHPPwOfWCi\n"\
"cZgSPInz4wBzJNIxUzxivT8sCeSiuZ2yDL7e1ENEnTL5Q4OMYK9IigVrAoGBAOCv\n"\
"Zi9w+dh+fZm1+OUexgnQ9A0xbocnBYAan/9pO48bu2137L2AsZ7ZV9Bf1Gtr18Ng\n"\
"MbzLFRveXUEyUXYXGfWVmp7b8XoAIjXegGMUs1GNy4Xxp7cIbXS6xJB3EvbSyboV\n"\
"karb17F+sHXvNXIxUe4KUZUiNuCssAJB3OYrpgVBAoGASxFJAWVk2UolLNUtKpCM\n"\
"g8wzBUuT7lZSrTDqg/gOnbb2+QplR3+N5HMbFW22hjJmM4wdWs0Z6oltCbSogB4R\n"\
"sc+sCedrW2N1rAWg6Cj5SSbLtzfTlbdWZJVS5OvO/RUbNcjAju1k5aiocUORbB2k\n"\
"YTD42R1pKdF5mIMoy8mxeoECgYA8Z4tOR84zoNX9hAmU93AAZZenEFoJpbj51q6b\n"\
"CNyFPcxmwq33yGh4LzBHievesVCBaxQUEHjtVBu4EmIKIG3PM1gtSy4hQXXxC49y\n"\
"k7cdvsQvj2cqcu6SSj/cItcLWmH3UCMoTxAu3MNuAVBaEd5X45hJQriEMIjlInXT\n"\
"v3pTwQKBgFUrq4cFrpGPMAbFqXx/u6umN5N2VLS47OvYmjkj8+q0KIJZBImUJ3+t\n"\
"H1bWLh4BaWZQ3HBWGh5AxBtnNeHpp3+dDd7s/TxhukdKHjB4Mo7GvMFwtcljutVq\n"\
"YVrtiaoGeNyuH441XPDRJ5+cwhL/jDrRTeOz7Bp93Ptc5ivUIqMr\n"\
"-----END RSA PRIVATE KEY-----"
```



```
#define CLIENT_CERTIFICATE      \
"-----BEGIN CERTIFICATE-----\n"\
"MIIDqjCCApKgAwIBAgIBADANBgkqhkiG9w0BAQsFADCBkDELMAkGA1UEBhMCR0Ix\n"\
"FzAVBgNVBAgMDlVuaXRlZCBLaW5nZG9tMQ4wDAYDVQQHDAVEZXJieTESMBAGA1UE\n"\
"CgwJTW9zcXVpdHRvMQswCQYDVQQLDAJDQTEWMBQGA1UEAwwNbW9zcXVpdHRvLm9y\n"\

#define CLIENT_PRIVATE_KEY          \
"-----BEGIN RSA PRIVATE KEY-----\n"\
"MIIEpAIBAAKCAQEAxNjCtCALFkNzGumEvra+emvaIEPlRh8BOaOjM/VT3QUq9QiE\n"\
"iQ6Fu8GZP27i0/dW8eYYsusOA01SGNMPlt8qdfgzmvDD90HelzvOnbIa44/NlwFX\n"\
"vnhPx2wydFx6eIzCQ0gy0NyDW5wkosq5bkabYfjVdAyfUumunp0lzBnKuHJLGZN6\n"\

#define ROOT_CA_CERTIFICATE      \
"-----BEGIN CERTIFICATE-----\n"\
"MIIEAzCCAuugAwIBAgIUBY1hlCGvdj4NhBXkZ/uLUZNILAwwDQYJKoZIhvcNAQEL\n"\
"BQAwgZAxCzAJBgNVBAYTAkdCMRcwFQYDVQQIDA5Vbml0ZWQgS2luZ2RvbTEOMAwG\n"\
"A1UEBwwFRGVyYnkxEjAQBgNVBAoMCU1vc3F1aXR0bzELMAkGA1UECwwCQ0ExFjAU\n"\
```

Build and run the program

1. Build and program. Open Tera Term, baud rate 115200

2. Press user button

3. The GPIO interrupt service routine (ISR) notifies the publisher task.

4. The publisher task publishes a message on a topic.

5. The MQTT broker sends back the message to the MQTT client because it is also subscribed to the same topic.

6. When the message is received, the subscriber task turns the LED ON or OFF. As a result, the user LED toggles every time the user presses the button.

Part of your life. Part of tomorrow.