User Guide for Secured IoT Connectivity Kit

For AWS Cloud Connectivity only

Contents

1 Overview

The PSoC 62 Wi-Fi BT Prototyping Kit (CY8CPROTO-062-4343W) is a low-cost hardware platform that enables design and debug of PSoC 6 MCUs with FreeRTOS and AWS IoT integration. The Cypress PSoC 62 is purpose-built for the IoT, delivering the industry's lowest power, most flexibility, and built-in security for the IoT. It delivers a dual-core platform with a 150-MHz Arm® Cortex®-M4 as the primary application processor and an 100-MHz Arm Cortex-M0+ as the secondary processor for low-power operation. The kit comes with a PSoC 62 MCU with 2MB of Flash / 1MB of SRAM memory, industry-leading CapSense® for touch buttons and slider, on-board debugger/programmer with KitProg3, MicroSD card interface, 512-Mb Quad-SPI NOR flash, PDM microphone, and a thermistor. For wireless connectivity, including dual-band Wi-Fi, it also provides an on-board Murata LBEE5KL1DX module built on the Cypress CYW4343W Wi-Fi / Bluetooth single-chip solution.

The PSOC can be paired with OPTIGA™ Trust M to enable protection of sensitive security tokens on the device, such as X.509 certificates and private keys.

2 Hardware Setting

The hardware setting to run aws FreeRTOS:

a) PSoC6x (based on ARM CORTEX M4) – using Infineon's PSoC62 Wi-Fi BT Prototyping Kit [https://www.cypress.com/documentation/development-kitsboards/psoc-6-wi-fi-bt-prototyping-kit-cy8cproto-062-4343w]

b) OPTIGA™ Trust M – using Infineon's Shield2GO Cloud Security OPTIGA™ Trust M

[https://www.infineon.com/cms/en/product/evaluation-boards/s2go-cloud-optiga-m/]

Or OPTIGA™ Trust M – using Infineon's OPTIGA™ Trust M Shield2GO

[https://www.infineon.com/cms/en/product/evaluation-boards/s2go-security-optiga-m/]

c) PSoC62 adaptor board V1.2

d) Mini CO2 module

[https://www.infineon.com/cms/en/product/evaluation-boards/eval_pasco2_miniboard/]

e) S2GO PRESSURE DPS310

[https://www.infineon.com/cms/en/product/evaluation-boards/s2go-pressure-dps310/]

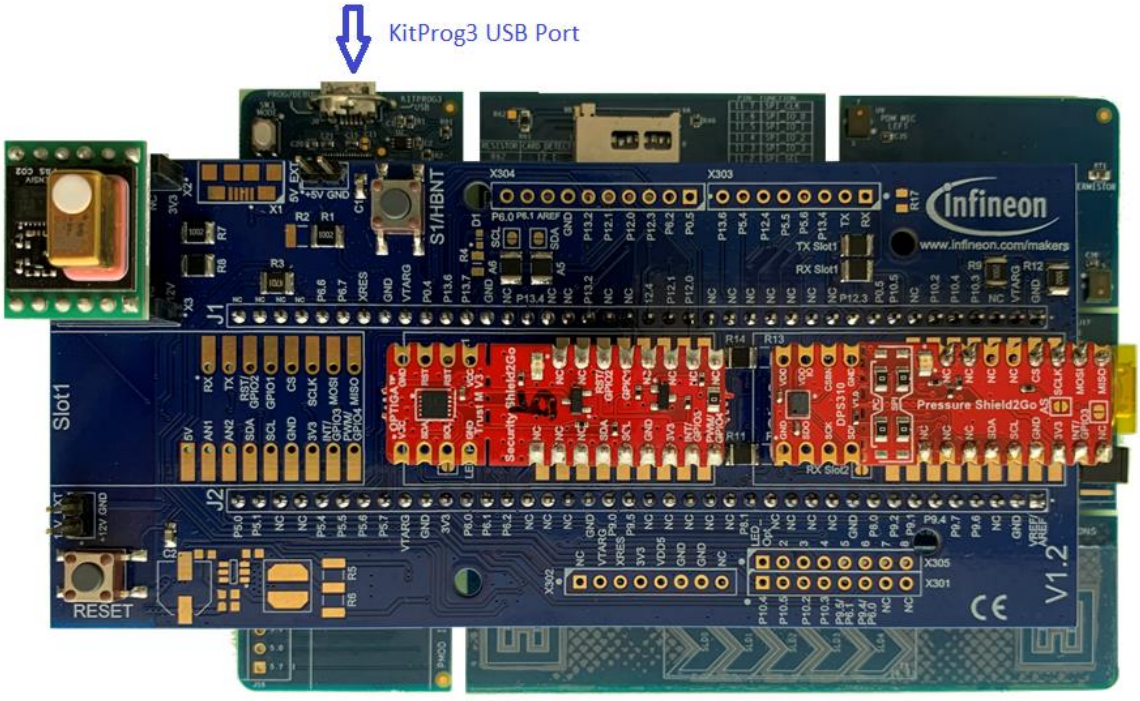


Figure1 Secure IoT Connectivity Kit

Connect the KitProg3 USB port to computer

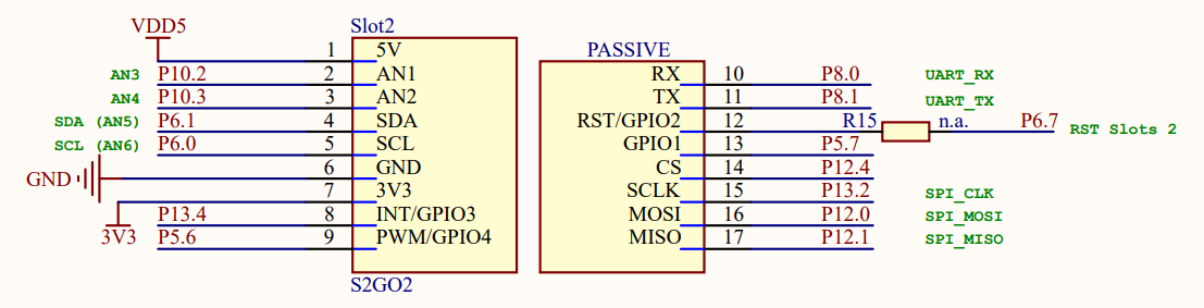


Figure2 Hardware connection for Trust M and Psoc62 WIFI BT prototyping kits

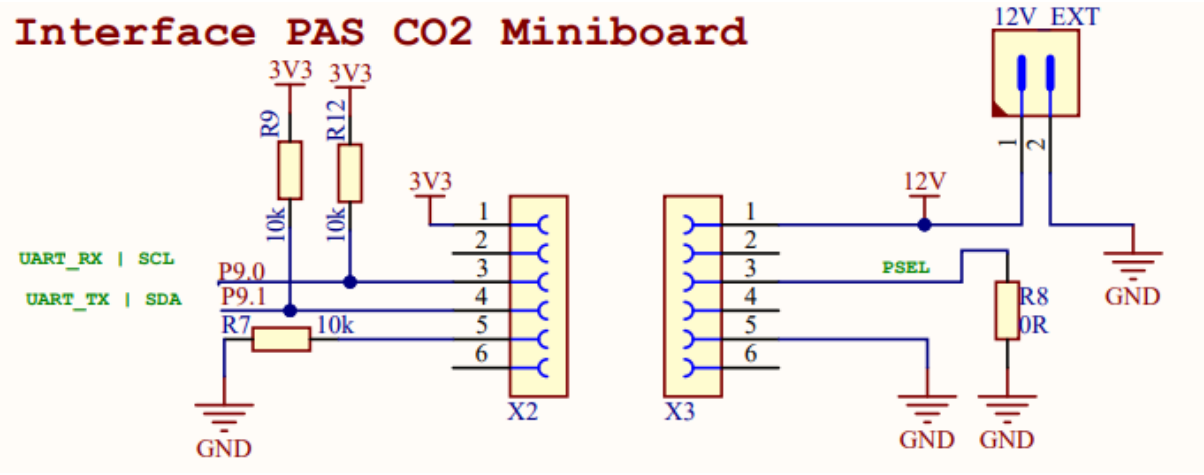


Figure3 Hardware connection for CO2 Miniboard and Psoc62 WIFI BT prototyping kits

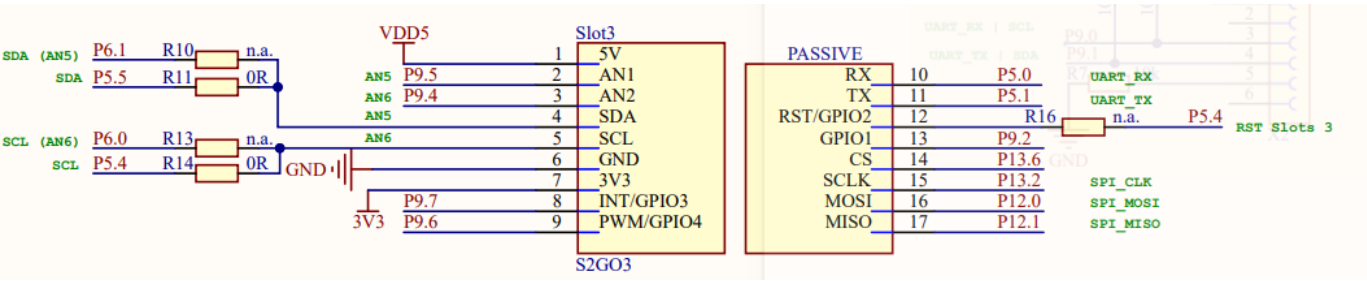


Figure4 Hardware connection for DPS310/DPS368 and Psoc62 WIFI BT prototyping kits

3 Set up the Development Environment

3.1 Install ModusToolbox

FreeRTOS works with either a CMake or Make build flow. You can use ModusToolbox for your Make build flow. You can use the Eclipse IDE delivered with ModusToolbox.The Eclipse IDE is compatible with the Windows, macOS, and Linux operating systems.

Before you begin, download and install the latest ModusToolbox software. For more information, see the ModusToolbox Installation Guide.

The versions currently supported are ModusToolbox 2.2 and ModusToolbox 2.3.

4 Setup your AWS account and Permissions

To create an AWS account, see Create and Activate an AWS Account.

To add an IAM user to your AWS account, see IAM User Guide. To grant your IAM user account access to AWS IoT and FreeRTOS, attach the following IAM policies to your IAM user account:

- AmazonFreeRTOSFullAccess
- AWSIoTFullAccess

To attach the AmazonFreeRTOSFullAccess policy to your IAM user

1. Browse to the IAM console, and from the navigation pane, choose **Users**.
2. Enter your user name in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter **AmazonFreeRTOSFullAccess**, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

To attach the AWSIoTFullAccess policy to your IAM user

1. Browse to the IAM console, and from the navigation pane, choose **Users**.
2. Enter your user name in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter **AWSIoTFullAccess**, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

For more information about IAM and user accounts, see IAM User Guide.

For more information about policies, see IAM Permissions and Policies.

5 Import the project into ModusToolbox

1. Open the Eclipse IDE for ModusToolbox and choose, or create, a workspace
2. In ModusToolbox, choose **File**, and then choose **Import**. Click **ModusToolbox Application Import**, then choose **Next**.
3. In the **Import Eclipse IDE for ModusToolbox Project** window, choose **Browse**, In the directory where you unzipped your FreeRTOS download, the demo project is located in located at \projects\cypress\CY8CPROTO_062_4343W\mtb\aws_demos
4. Choose **Finish** to import the project into your workspace
5. The aws_demos project should be imported into your workspace.

6 Create AWS IoT Thing on AWS IOT console

1. Browse to the AWS IoT console.
2. In the navigation pane, choose **Manage**, and then choose **Things**.
3. If you do not have any IoT things registered in your account, the **You don't have any things yet** page is displayed. If you see this page, choose **Register a thing**. Otherwise, choose **Create**.
4. On the **Creating AWS IoT things** page, choose **Create a single thing**.
5. On the **Add your device to the thing registry** page, enter a name for your thing, and then choose **Next**.
6. On the **Create a Certificate** Page, choose **Use my certificate**
7. Select **Next**
8. Then on next page(**Register existing device certificates**), click **Select Certificate** and point to the location in local machine where certificate.pem(or deviceCert.pem) is stored
   NOTE: *For the certificate export/generation, Please refer to section 7.1.2 or 7.2.5.Please skip Register CA if you are using pre-provision certificate.*
9. Click on **Activate all** button and then click on **Register certificates**
10. You will be redirected to Certificates section
11. Go to **Secure\ Policies** section, click on create button, give a name and click on advanced mode
12. Then paste the following policy and click on **create**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iot:*"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

13. Go back to **Secure\ Certificate** section
14. Open the certificate provisioned, click on **Actions** and click **Attach Policy**
15. Find the policy name that you created and click on **Attach**

7 Configure and run the FreeRTOS

7.1 Using the pre-provisioned key and certificate

7.1.1 Initial Configuration

Perform the steps in Configuring the FreeRTOS Demos, but skip the last step (that is, don't do *To format your AWS IoT credentials*). The net result should be that the demos/include/aws_clientcredential.h file has been updated with settings, but the demos/include/aws_clientcredential_keys.h file has not.

The OPTIGA Trust M comes pre-provisioned with a certificate signed with Infineon's OPTIGA(TM) Trust M CA and a private key, and can make use of the AWS IoT Multi-Account Registration feature. To retrieve the certificate from the OPTIGA Trust M, and register it with AWS IoT using Multi-Account Registration, follow the instructions below.

7.1.2 Read out the pre-provision cert from Trustm

NOTE: *if you do have Shield2Go Security OPTIGA™ Trust M and there is no QR code on your package follow the steps mentioned [here](Register the certificate with AWS IoT), to retrieve your certificate and continue with the "Register the certificate with AWS IoT" step*

Download the certificate from the OPTIGA™ Trust M with Infineon Toolbox

If you have the Shield2Go Cloud Security OPTIGA™ Trust M variant, you can obtain the certificate that is pre-loaded on the secure element with the Infineon Toolbox. Follow the steps below to retrieve your certificate:

1. Under **First Steps** choose **please register**.
2. If you have a QR Code, choose **Scan QR code** and follow the instructions. Otherwise, if you have the Serial number, choose **Enter Serial Number** and enter the Serial Number on the next page.
3. After inputting your serial number or QR code, the Shield2Go Cloud Security OPTIGA™ Trust M should appear in the browser. Choose **View Details**.
4. The public X.509 certificate that is present on your device should be displayed. Copy or download this certificate to your local environment.

Export the certificate from the OPTIGA™ Trust M using CLI tools

If you have Shield2Go OPTIGA™ Trust M without serial number or QR code, you can use our CLI tools to export the certificate inside 0xE0E0.

7.1.3 Configure PKCS#11 on FreeRTOS

The certificate is already pre-loaded on the OPTIGA™ Trust M, you will not need to add the certificate to the FreeRTOS code. Instead, you will specify the PKCS#11 labels for the OPTIGA™ Trust M's certificate and private key slots.

Open the file \vendors\cypress\boards\CY8CPROTO_062_4343W\aws_demos\config_files\iot_pkcs11_config.h

a. Set pkcs11configLABEL_DEVICE_PRIVATE_KEY_FOR_TLS to 0xE0F0
b. Set pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS to 0xE0E0
c. Set pkcs11configLABEL_JITP_CERTIFICATE to 0xE0E0
d. Open the file demos/include/aws_clientcredential_keys.h.
e. Ensure keyJITR_DEVICE_CERTIFICATE_AUTHORITY_PEM, keyCLIENT_CERTIFICATE_PEM, and keyCLIENT_PRIVATE_KEY_PEM are set to ""

7.1.4 Build the application

a. From the **Quick Panel**, select **Build aws_demos Application**.
b. Choose **Project** and choose **Build All**.

Make sure the project compiles without errors.

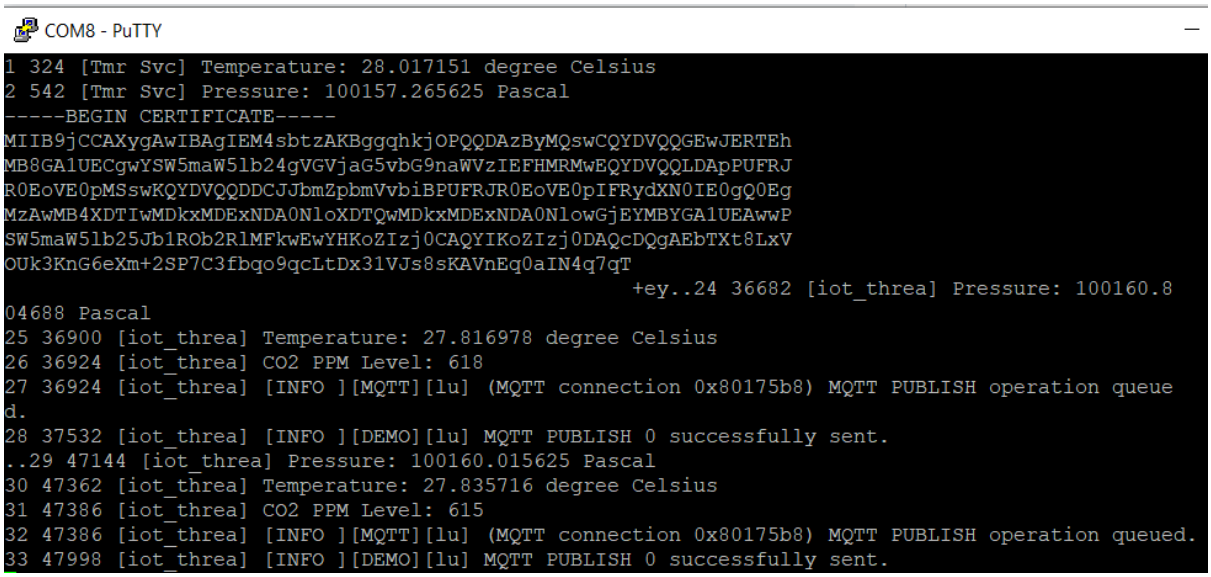
7.1.5 Set up a serial connection

a. Connect the kit to your host computer shown in Chapter 2.
b. The USB Serial port for the kit is automatically enumerated on the host computer. Identify the port number. In Windows, you can identify it using the **Device Manager** under **Ports** (COM & LPT).
c. Start a serial terminal and open a connection with the following settings:

- Baud rate: 115200
- Data: 8 bit
- Parity: None
- Stop bits: 1
- Flow control: None

7.1.6 Run the FreeRTOS demo project

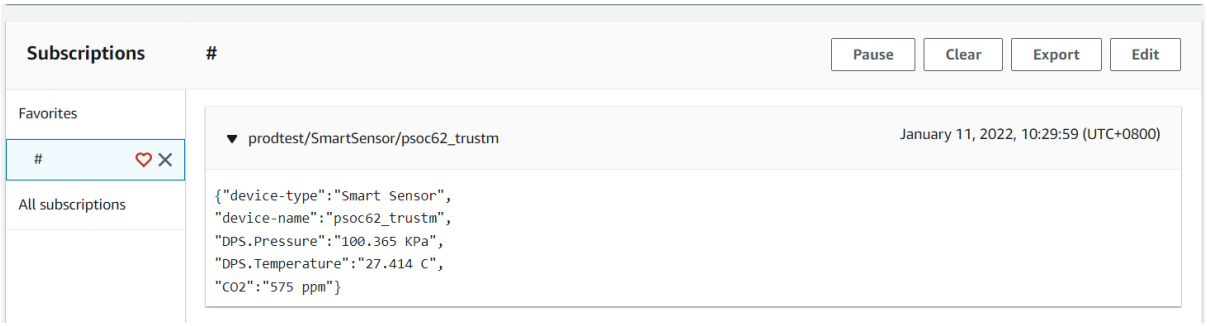
a. Select the project `aws_demos` in the workspace.
b. From the **Quick Panel**, select **aws_demos Program (KitProg3)**. This programs the board and the demo application starts running after the programming is finished.
c. You can view the status of the running application in the serial terminal. The following figure shows a part of the terminal output.



You can use the MQTT client in the AWS IoT console to monitor the messages that your device sends to the AWS Cloud. You might want to set this up before the device runs the demo project.

To subscribe to the MQTT topic with the AWS IoT MQTT client

1. Sign in to the AWS IoT console.
2. In the navigation pane, choose **Test** to open the MQTT client.
3. In **Subscription topic**, enter **iotdemo/topic/#**, and then choose **Subscribe to topic**.



7.2 Using your own key and Certificate(Using PSoC62 kits)

To generate your own device key pair and certificate securely, please follow the instructions below:

7.2.1 Key slot selection

There are three key slots to select for key generation and certificate store, you can choose the key slot you want to use by changing the setting below:
Open the file:
\vendors\cypress\boards\CY8CPROTO_062_4343W\aws_demos\config_files\iot_pkcs11_config.h

1. Set pkcs11configLABEL_DEVICE_PRIVATE_KEY_FOR_TLS to 0xE0F1(or 0xE0F2)
2. Set pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS to 0xE0E1(or 0xE0E2)
3. Set pkcs11configLABEL_JITP_CERTIFICATE to 0xE0E1(or 0xE0E2)

7.2.2 Demo Project Configuration

In the project, open the file aws_dev_mode_key_provisioning.c and change the definition of keyprovisioningFORCE_GENERATE_NEW_KEY_PAIR, which is set to zero by default, to one:

#define keyprovisioningFORCE_GENERATE_NEW_KEY_PAIR 1

Then build and run the demo project and continue to the next step.

7.2.3 Public Key Extraction

Since the device has not yet been provisioned with a private key and client certificate, the demo will fail to authenticate to AWS IoT. However, the Hello World MQTT demo starts by running developer-mode key provisioning, resulting in the creation of a private key if one was not already present. Example output on the serial console is given below:

```
7 910 [IP-task] Device public key, 91 bytes:
3059 3013 0607 2a86 48ce 3d02 0106 082a
8648 ce3d 0301 0703 4200 04cd 6569 ceb8
1bb9 1e72 339f e8cf 60ef 0f9f b473 33ac
6f19 1813 6999 3fa0 c293 5fae 08f1 1ad0
41b7 345c e746 1046 228e 5a5f d787 d571
dcb2 4e8d 75b3 2586 e2cc 0c
```

Copy the six lines of key bytes into a file called DevicePublicKeyAsciiHex.txt. Then use the command-line tool "xxd" to parse the hex bytes into binary:

xxd -r -ps DevicePublicKeyAsciiHex.txt DevicePublicKeyDer.bin

Use "openssl" to format the binary encoded (DER) device public key as PEM:

openssl ec -inform der -in DevicePublicKeyDer.bin -pubin -pubout -outform pem -out DevicePublicKey.pem

7.2.4 Disable Macro

Next step is to disable the temporary key generation setting you enabled above. Otherwise, the device will create another key pair, and you will have to repeat the previous steps:

#define keyprovisioningFORCE_GENERATE_NEW_KEY_PAIR 0

NOTE: *Remember to disable this marco and flash into PSoC62 before power off this kits to avoid recreating new keypair.*

7.2.5 Public Key Infrastructure Setup

Follow the instructions in Registering Your CA Certificate to create a certificate hierarchy for your device lab certificate. Stop before executing the sequence described in the section *Creating a Device Certificate Using Your CA Certificate*.

In this case, the device will not be signing the certificate request (that is, the Certificate Service Request or CSR) because the X.509 encoding logic required for creating and signing a CSR has been excluded from the FreeRTOS demo projects to reduce ROM size. Instead, for lab testing purposes, create a private key on your workstation and use it to sign the CSR.

openssl genrsa -out tempCsrSigner.key 2048
openssl req -new -key tempCsrSigner.key -out deviceCert.csr

Once your Certificate Authority has been created and registered with AWS IoT, use the following command to issue a client certificate based on the device CSR that was signed in the previous step:

openssl x509 -req -in deviceCert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out deviceCert.pem -days 500 -sha256 -force_pubkey DevicePublicKey.pem

Even though the CSR was signed with a temporary private key, the issued certificate can only be used with the actual device private key. The same mechanism can be used in production if you store the CSR signer key in separate hardware, and configure your certificate authority so that it only issues certificates for requests that have been signed by that specific key. That key should also remain under the control of a designated administrator.

7.2.6 Certificate Import

With the certificate issued, the next step is to import it into your device.

In the **aws_clientcredential_keys.h** file in your project, set

the **keyCLIENT_CERTIFICATE_PEM** macro to be the contents of deviceCert.pem

To format the certificate for aws_clientcredential_keys.h

1. In a browser window, open /tools/certificate_configuration/CertificateConfigurator.html.
2. Under **Certificate PEM file**, choose deviceCert.pem.
3. Choose **Generate and save aws_clientcredential_keys.h**
   Copy the below part into **aws_clientcredential_keys.h** in /demos/include.

```
#define keyCLIENT_CERTIFICATE_PEM \
"-----BEGIN CERTIFICATE-----\n"\
"MIIBzjCCAXQCFC8f9vGQDp2auJOO/ObU/vb44pdlMAoGCCqGSM49BAMCMHcxCzAJ\n"\
"BgNVBAYTAkRFMSEwHwYDVQQKDBhJbmZpbmVvbiBUZWNobm9sb2dpZXMgQUcxEzAR\n"\
"BgNVBAsMCk9QVElHQShUTSkxMDAuBgNVBAMMJ0luZmluZW9uIE9QVElHQShUTSkg\n"\
"VHJ1c3QgTSBUZXN0IENBIDAwMDAeFw0yMTA3MDUwMTQ5MTJaFw0yMjExMTcwMTQ5\n"\
"MTJaMFwxCzAJBgNVBAYTAlNHMQswCQYDVQQIDAJTRzELMAkGA1UEBwwCU0cxDTAL\n"\
"BgNVBAoMBElGQVAxDDAKBgNVBAsMA0NTUzEWMBQGA1UEAwwNVHJ1c3RtM0w2ZTBl\n"\
"MjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABI88UH4de94o0rsCVDD5/4Ys5iIj\n"\
"Y8/9hWtRjrAKGErO0FBm9VgjcWIESDWIysmLeySV+BAkC0RIk6YgcOgmGgAwCgYI\n"\
"KoZIzj0EAwIDSAAwRQIhANZ7Eeik3s0B16sQ1fXC5Yx7Ai2ysDoGPXBbfquyy7oI\n"\
"AiBFGSj7UKSrBM0G+oI6F9w9Hr7qTLu8iwHSX8PDJklnVg==\n"\
"-----END CERTIFICATE-----"
```

Disable this macrco in original **aws_clientcredential_keys.h** in /demos/include and save this file.

#define keyCLIENT_CERTIFICATE_PEM ""

Repeat the steps described in section 7.1.4-7.1.6 to build and run the project.

7.2.7 Device Authorization

Import deviceCert.pem into the AWS IoT registry as described in Use Your Own Certificate. You must create a new AWS IoT thing, attach the PENDING certificate and a policy to your thing, then mark the certificate as ACTIVE. All of these steps can be performed manually in the AWS IoT console. Once the new client certificate is ACTIVE and associated with a thing and a policy, the connection to the AWS IoT MQTT broker will succeed.

Note: *Once the device has been successfully connected to AWS IoT for the first time,* ***Set*** *the* ***keyCLIENT_CERTIFICATE_PEM*** *macro to ""in the* ***aws_clientcredential_keys.h***

Follow 7.1.4-7.1.6 to build and run the demo

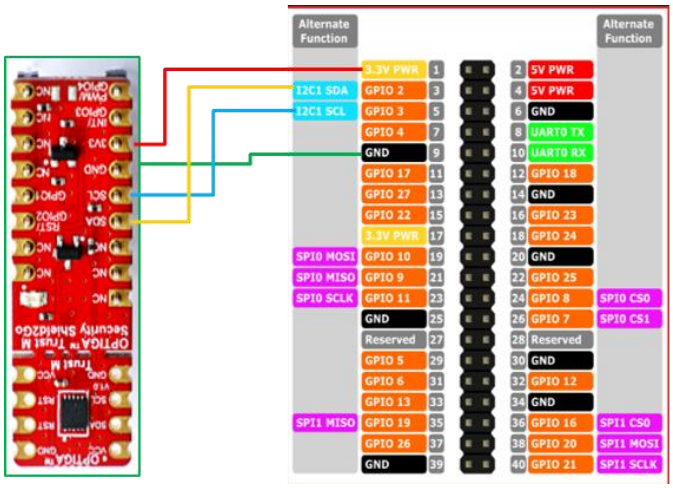
7.3 Using your own key and Certificate (using raspberry pi to auto provision to aws)

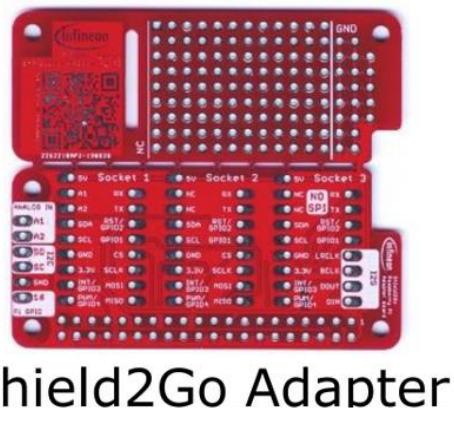
There is another easy way to auto register trustm to aws using your own key and Certificate.

7.3.1 Prepare the hardware

Hardware platforms and boards:

- Raspberry PI 3/4 on Linux kernel >= 4.19
- OPTIGA™ Trust M
- Shield2Go Adapter for Raspberry Pi





7.3.2 Prepare the software build Toolchain

Go to: https://github.com/Infineon/linux-optiga-trust-m to clone the code and build the library following the instruction here: https://github.com/Infineon/linux-optiga-trust-m/blob/development_v3/ex_aws-iot-device-sdk-embedded-C-1.1.2/README.md

For first time setup, follow instructions up to section "3.1 Policy Registration on AWS IOT console"

7.3.3 Auto provision to aws

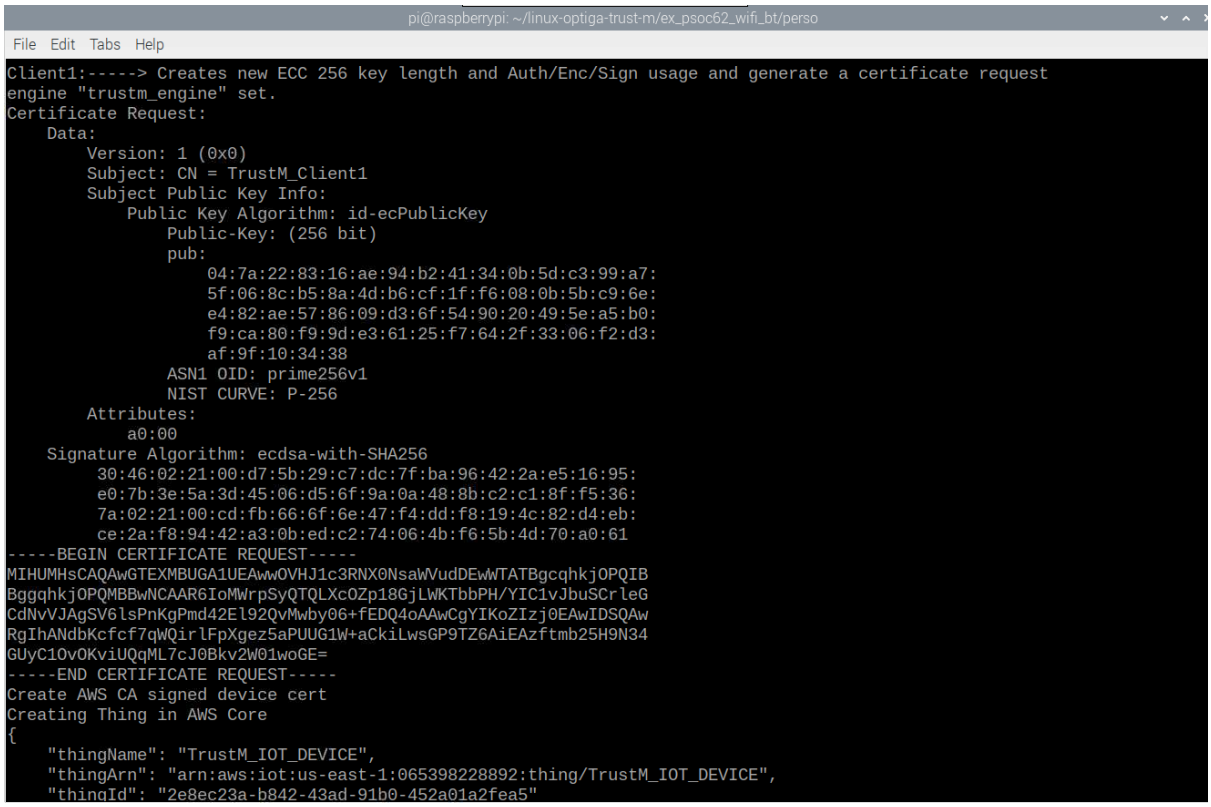
Change the policy name in the psoc62_auto_provision_to_aws.sh script located in **linux-optiga-trust-m/ ex_psoc62_wifi_bt/perso** folder to match to the policy name you created in AWS IoT console

Note: Make sure the policy is already created in AWS IOT console and the policy name matches the name inside psoc62_auto_provision_to_aws.sh script before running the script.

Running the script:

pi@raspberrypi:~/linux-optiga-trust-m/ex_psoc62_wifi_bt/perso $ sh psoc62_auto_provision_to_aws.sh

The output is shown as below as example:



7.3.4 Key slot selection in PSoC62 project

There are three key slots to select for key generation and certificate store, you can choose the key slot you want to use by changing the setting below:
Open the file:
\vendors\cypress\boards\CY8CPROTO_062_4343W\aws_demos\config_files\iot_pkcs11_config.h

1. Set pkcs11configLABEL_DEVICE_PRIVATE_KEY_FOR_TLS to 0xE0F1(or 0xE0F2)
2. Set pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS to 0xE0E1(or 0xE0E2)
3. Set pkcs11configLABEL_JITP_CERTIFICATE to 0xE0E1(or 0xE0E2)

Note:

1. Make sure the key slot you use is same with the one you use in the psoc62_auto_provision_to_aws.sh script located in **linux-optiga-trust-m/ ex_psoc62_wifi_bt/perso** folder
2. Make sure the thing name you use is same with the one you use in the psoc62_auto_provision_to_aws.sh script located in **linux-optiga-trust-m/ ex_psoc62_wifi_bt/perso** folder

Power off and Plug out OPTIGA™ Trust M Shield2GO board from Raspberry Pi and plug it into PSoC62 platform.

Follow 7.1.4-7.1.6 to build and run the demo.

8 References

1. OPTIGA™ Trust M Solution Reference Manual V3.15