# AZURE IOT on CEC1x02 Development Board

|  |  |  |
| --- | --- | --- |
| ***Platform*** | ***Device*** | ***Language*** |
| devBoardCEC1x02 | CEC1x02 | C |

# Run a simple C sample on Microchip devBoard1x02

## Table of Contents

* Introduction
* Step 1: Prerequisites
* Step 2: Prepare your device
* Step 3: Build and run the Sample
* Next Steps

## Introduction

**About this document**

This document describes how to connect Microchip DevBoard1x02 board along with Wifi Clicker Board (ATWINC1500) to Microsoft Azure IoT Hub, by using the X.509 based MQTT application sample, using Azure IoT Device SDK.This multi-step process includes:

* Configuring Azure IoT Hub
* Registering devBoard1x02 to Azure IoT Hub
* Build and deploy Azure IoT SDK on devBoard1x02

## Step 1: Prerequisites

You should have the following items ready before beginning the process

### Development Environment

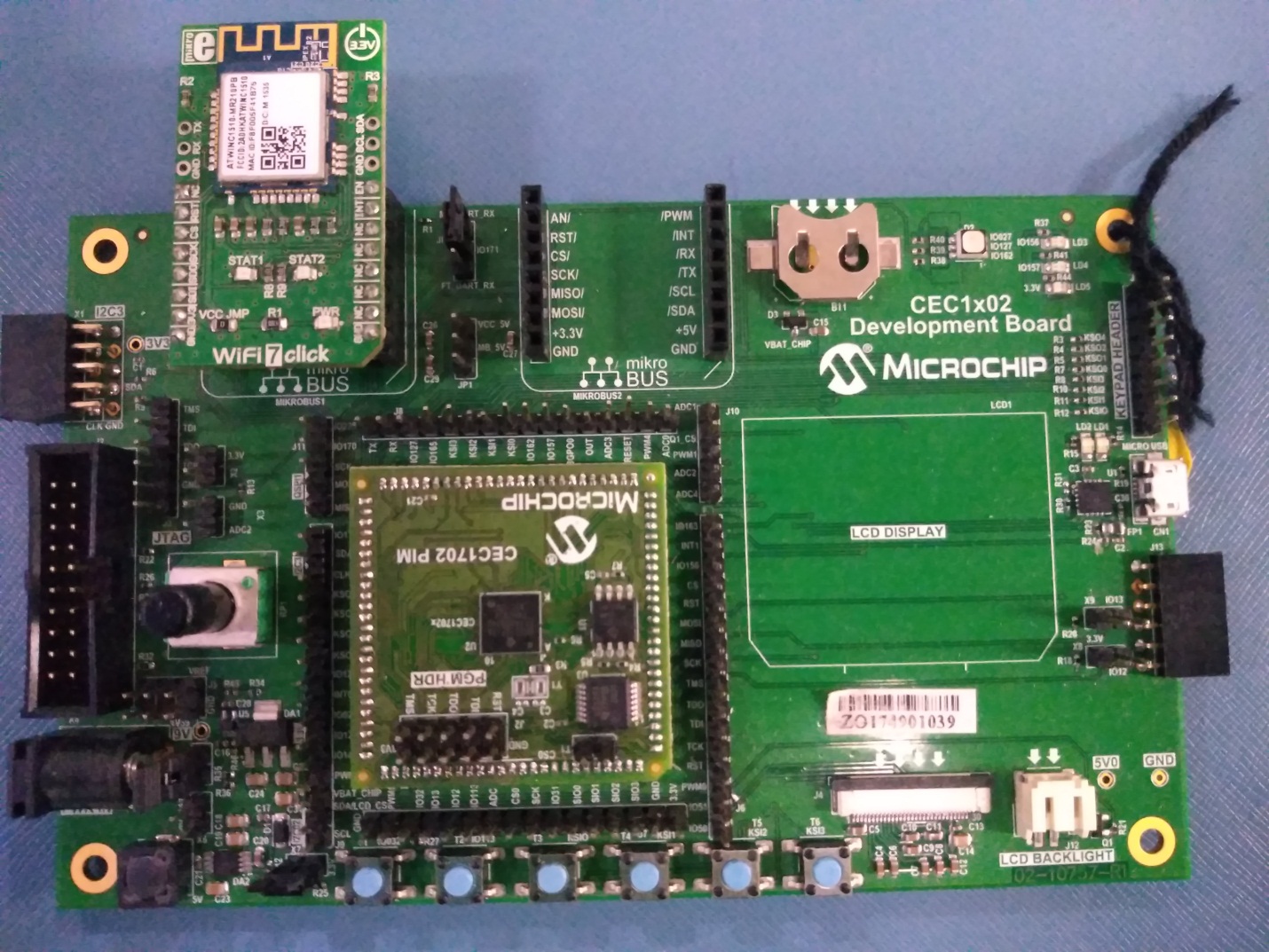
* ***Keil µVision IDE*** - For project build, downloading and debugging code
* ***mikroProg Suite For ARM*** – For loading spi image in flash
* ***ComXDBG.exe*** (provided with package) or any other serial terminal installed in your PC – for viewing trace messages from devBoard1x02 board
* [Setup your IoT hub](https://github.com/neeraj-khanna/azure-iot-device-ecosystem/blob/master/setup_iothub.md)
* [Provision your device and get its credentials](https://github.com/neeraj-khanna/azure-iot-device-ecosystem/blob/master/manage_iot_hub.md)

### 1.2 Hardware components

* CEC1x02 devBoard board
* ATWINC1510 Clicker Board

## Step 2: Prepare your Device

Attach the ATWINC1510 Clicker board on mikroBUS slot 1 on CEC1702 devBoard board as shown in the figure



## Step 3: Build and Run the sample

### 3.1 Build SDK and sample code

1. Download the [devBoard\_Azure\_IoT\_build](http://www.microchip.com/SWLibraryWeb/product.aspx?product=CEC1702_AZURE_IOT) package. This package contains all the required drivers to use with the CEC1x02 devBoard board with Winc1500 clicker board, together with pre-integrated Microsoft Azure IoT C SDK.

The project uses mbedTLS as the TLS stack, which has been added to the project as a library:

devBoard\_Azure\_IoT\Src\APP\framework\mbedTLS\libmbedtls240.lib

1. Unzip the package and open the Keil project file

devBoard\_Azure\_IoT\MDK-ARM\riotDemo.uvprojx

1. The MQTT application is **x509\_client\_sample.** It uses MQTT as the transport for communicating to the AZURE IOT hub.

devBoard\_Azure\_IoT\Src\APP\apps\x509\_client\_sample.c

The certificates and keys required for authentication is generated through DICE (Device Identification Composition Engine)

For testing purpose the UDS (Unique Device Secret) for the device, uses a test value hardcorded in dps\_hsm\_riot.c

static unsigned char g\_uds\_seed[DICE\_UDS\_LENGTH] = {

0x54, 0x10, 0x5D, 0x2E, 0xCD, 0x07, 0xF9, 0x01,

0x99, 0xB3, 0x95, 0xC7, 0x42, 0x61, 0xA0, 0x8C,

0xFF, 0x27, 0x1A, 0x0D, 0xF6, 0x6F, 0x1F, 0xE0,

0x00, 0x34, 0xBB, 0x11, 0xF7, 0x98, 0x9A, 0x12 };

devBoard\_Azure\_IoT\Src\APP\framework\azure\dps\_client\adapters\dps\_hsm\_riot.c

The thumbprint for the generated X.509 certificate using the test UDS is

69CF2777F616BA27926E8BA7D947B8E4622841A4

Later when you want to use the UDS value from efuse, you can uncomment the macro USE\_TEST\_UDS in dps\_hsm\_riot.c.

Please refer [Secure your IoT deployment](https://github.com/Microsoft/azure-docs/blob/master/includes/iot-secure-your-deployment.md) and [Control Access to IoT Hub](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-security#supported-x509-certificates) for more details on X.509 security.

In our case, the X.509 certificate is generated for the device through DICE and RIoT process. See [Device Identity with DICE and RIoT: Keys and Certificates](https://www.microsoft.com/en-us/research/publication/device-identity-dice-riot-keys-certificates/) for more details.

The DICE, RIoT and Azure application code are built together as one firmware image; they are organized separately in memory by using linker options. See riotDemo.sct for the linker file memory layouts.

devBoard\_Azure\_IoT\MDK-ARM\riotDemo.sct

1. Update your iothub host url and device id in **x509\_client\_sample.c**

#define IOTHUB\_CONNECTION\_STRING "azure-iothub-mchp-1.azure-devices.net"

#define IOTHUB\_DEVICE\_ID "dice-device-1"

1. Wifi Configuration - Currently the code is configured to connect through WPA-PSK. The SSID and password are set statically in ***winc1500\_connect.c***

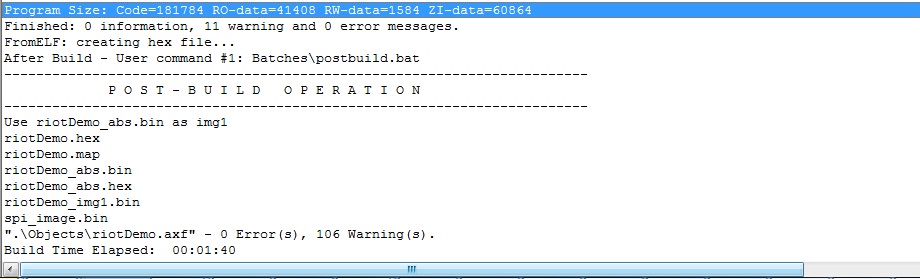
devBoard\_Azure\_IoT\Src\APP\platform\winc\winc1500\_connect.c

#define CONN\_SSID "TP-LINK\_6934"

#define CONN\_PSK\_PWD "03893708"

Modify the above values to match your wifi router.

1. Build Output



Compiler: Keil uVision V5.23.0.0

The spi\_image (spi\_image.bin) is created as part of the post-build process. The output files are placed in: devBoard\_Azure\_IoT\MDK-ARM\targets

### 3.2 Build Firmware into Flash

The CEC1x02 firmware application is stored in the external SPI flash device.

Use the mikroProg Suite For ARM to program the SPI in the CEC1x02 PIM.

### 3.3 Connect and send messages to Azure IoT Hub

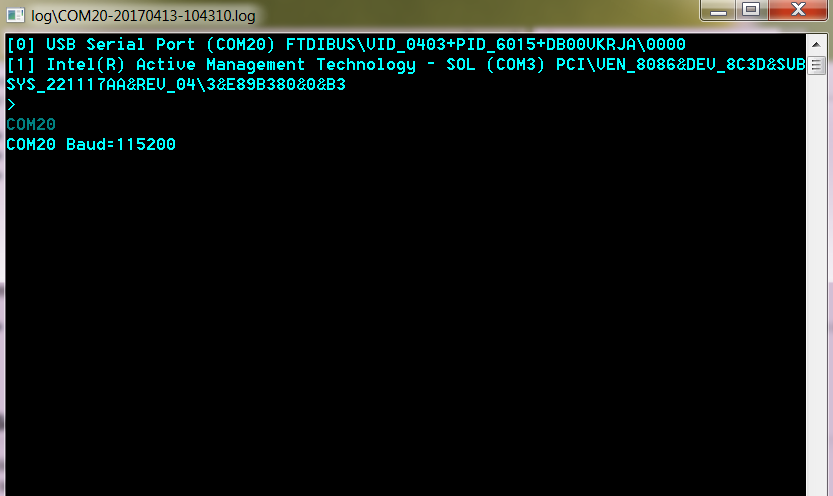
To view the UART traces:

1. connect USB cable between devBoard board and Windows host
2. After driver installation, start ComXDBG.exe

devBoard\_Azure\_IoT\utilities\ ComEDBG \ ComXDBG.exe

1. Select FTDIBUS COM port

For example; for the below options we would enter 0



1. You should be able to view UART traces from the devBoard board

Alternatively, you can use your serial terminal (e.g. TeraTerm for Windows) with the following parameters:

* BaudRate : 115200
* Data : 8-bit
* Parity : None
* Stop : 1 bit
* Flow Control : None

Sample log

[20:39:04.932] AZURE IoT Node

[20:39:04.932] Microchip CEC1702 devBoard 01

[20:39:04.932] Firmware Version : CEC1702\_devBoard\_diceRIoT\_build\_0800

[20:39:04.932] Dec 28 2017 20:25:54

[20:39:04.932] ------------------------------------------------

[20:39:05.882] winc1500\_wifi\_cb: 2c

[20:39:05.882] M2M\_WIFI\_RESP\_CON\_STATE\_CHANGED: CONNECTED

[20:39:05.882] winc1500\_wifi\_cb: 32

[20:39:05.882] M2M\_WIFI\_REQ\_DHCP\_CONF: IP is 192.168.43.70

[20:39:05.882] WINC is connected to Akshaya successfully!

[20:39:05.882] winc1500\_wifi\_init: Done

[20:39:05.882] Initializing rando.

[20:39:05.882] DPS\_HSM: initialize\_riot\_system:

[20:39:05.933] Iothub Version: 1.1.22

[20:39:05.933] -------------------- IoTHubClient\_LL\_CreateFromDeviceAuth -------

[20:39:05.933] IOTHUB CLIENT: uri azure-iothub-mchp-ny-1.azure-devices.net

[20:39:05.933] IOTHUB CLIENT: device id dice-device-1

[20:39:05.933] DPS\_HSM: dps\_hsm\_riot\_create:

[20:39:05.933] RIOT: CDI

[20:39:05.933] ce1c95020a4902a87fde4dbf8977b68362a944eac301c4c6c25828c92bd80780

[20:39:07.039] RIOT: Device Certificate

[20:39:07.089] -----BEGIN CERTIFICATE-----

[20:39:07.089] MIIBdTCCARugAwIBAgIFDg0MCwowCgYIKoZIzj0EAwIwNDESMBAGA1UEAwwJcmlvdC1yb290MQswCQYDVQQGDAJVUzERMA8GA1UECgwITVNSX1RFU1QwHhcNMTcwMTAxMDAwMDAwWhcNMzcwMTAxMDAwMDAwWjA7MRkwFwYDVQQDDBByaW90LXNpZ25lci1jb3JlMQswCQYDVQQGDAJVUzERMA8GA1UECgwITVNSX1RFU1QwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAR2C/Fa9fotRGPpEN91dRjfT7iHPyBnpx/blWJ9CFvfA+65lPGvT2cuyVRh0XjaoDpWwhSPV/KbqYu/YrpqAu+9oxMwETAPBgNVHRMECDAGAQH/AgEBMAoGCCqGSM49BAMCA0gAMEUCIQDFhqlOoamzbIz/1NUWm2njxwjDcxM/uSOw8fc2iUkEEwIgBl2wjtTF2TXnWfN8wTj/vHRysR3lK6mit4J

jvO9C0oE=

[20:39:07.089] -----END CERTIFICATE-----

[20:39:07.390] DPS\_HSM: dps\_hsm\_riot\_get\_certificate:

[20:39:07.390] RIOT: Alias Certificate

[20:39:07.390] -----BEGIN CERTIFICATE-----

[20:39:07.490] MIICNjCCAdugAwIBAgIFCgsMDQ4wCgYIKoZIzj0EAwIwOzEZMBcGA1UEAwwQcmlvdC1zaWduZXItY29yZTELMAkGA1UEBgwCVVMxETAPBgNVBAoMCE1TUl9URVNUMB4XDTE3MDEwMTAwMDAwMFoXDTM3MDEwMTAwMDAwMFowOzEZMBcGA1UEAwwQcmlvdC1kZXZpY2UtY2VydDELMAkGA1UEBgwCVVMxETAPBgNVBAoMCE1TUl9URVNUMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEXl8k/JM5Trrd+T2hCzH8J8FzdMo+nqEBctNE0wUQWesM+QbsF1PBmEHxb/FpqegLlleJW4niMxaN8z3J7T/imKOByzCByDAWBgNVHSUBAf8EDDAKBggrBgEFBQcDAjCBrQYDVR0RAQH/BIGiMIGfoIGcBgorBgEEAYI3WQMBMIGNAgEBMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgA

EdgvxWvX6LURj6RDfdXUY30+4hz8gZ6cf25VifQhb3wPuuZTxr09nLslUYdF42qA6VsIUj1fym6mLv2K6agLvvTAtBglghkgBZQMEAgEEIGvpsYTJN8KOEi7uUSto6o4Aw90VnqToXoTLqWb0Rs1OMAoGCCqGSM49BAMCA0kAMEYCIQDFhqlOoamzbIz/1NUWm2njxwjDcxM/uSOw8fc2iUkEEwIhAMts/WpPgP8xjvYkDaI4n0x30euQUj78oTvlY4Qp61yk

[20:39:07.490] -----END CERTIFICATE-----

[20:39:07.490] DPS\_HSM: dps\_hsm\_riot\_get\_alias\_key:

[20:39:07.490] RIOT: Alias key

[20:39:07.490] -----BEGIN EC PRIVATE KEY-----

[20:39:07.490] MHcCAQEEINXd3WSS7LqDEFGdpbQi0V51w01XkpMVPUYlG9V7/ldUoAoGCCqGSM49AwEHoUQDQgAEXl8k/JM5Trrd+T2hCzH8J8FzdMo+nqEBctNE0wUQWesM+QbsF1PBmEHxb/FpqegLlleJW4niMxaN8z3J7T/imA==

[20:39:07.490] -----END EC PRIVATE KEY-----

[20:39:07.490] DPS\_HSM: dps\_hsm\_riot\_destroy:

[20:39:07.490] socketio\_create:

[20:39:07.490] TLSIO MBEDTLS: setoption: x509EccCertificate

[20:39:07.542] TLSIO MBEDTLS: setoption: x509EccAliasKey

[20:39:07.542] TLSIO MBEDTLS: setoption: TrustedCerts

[20:39:07.542] Sending temperature reading to IoTHub every 2 seconds

[20:39:07.542] socketio\_open:

[20:39:07.592] winc1500\_wifi\_cb: 20

[20:39:07.642] winc1500\_wifi\_cb: 1b

[20:39:07.842] SERVER IP is 40.76.71.185

[20:39:08.092] winc1500\_connect: socket: 0

[20:39:14.595] mbedtls\_connect: sts 0

[20:39:14.897] -> CONNECT | VER: 4 | KEEPALIVE: 25 | FLAGS: 128 | USERNAME: azure-iothub-mchp-ny-1.azure-devices.net/dice-device-1/api-version=2016-11-14&DeviceClientType=iothubclient%2f1.1.22%20(CEC1702) | CLEAN: 0

[20:39:15.897] <- CONNACK | SESSION\_PRESENT: true | RETURN\_CODE: 0x0

[20:39:17.903] IoTHubClient\_LL\_SendEventAsync accepted message [1] for transmission to IoT Hub.

[20:39:18.203] -> SUBSCRIBE | PACKET\_ID: 2 | TOPIC\_NAME: devices/dice-device-1/messages/devicebound/# | QOS: 1

[20:39:19.204] <- SUBACK | PACKET\_ID: 2 | RETURN\_CODE: 1

[20:39:21.255] IoTHubClient\_LL\_SendEventAsync accepted message [2] for transmission to IoT Hub.

[20:39:24.608] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 3 | PAYLOAD\_LEN: 83

[20:39:26.509] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 4 | PAYLOAD\_LEN: 83

[20:39:27.510] <- PUBACK | PACKET\_ID: 3

[20:39:27.510] <- PUBACK | PACKET\_ID: 4

[20:39:29.517] IoTHubClient\_LL\_SendEventAsync accepted message [3] for transmission to IoT Hub.

[20:39:29.868] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 5 | PAYLOAD\_LEN: 83

[20:39:30.869] <- PUBACK | PACKET\_ID: 5

[20:39:32.871] IoTHubClient\_LL\_SendEventAsync accepted message [4] for transmission to IoT Hub.

[20:39:33.172] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 6 | PAYLOAD\_LEN: 83

[20:39:34.175] <- PUBACK | PACKET\_ID: 6

[20:39:36.175] IoTHubClient\_LL\_SendEventAsync accepted message [5] for transmission to IoT Hub.

[20:39:36.477] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 7 | PAYLOAD\_LEN: 83

[20:39:37.477] <- PUBACK | PACKET\_ID: 7

[20:39:39.480] IoTHubClient\_LL\_SendEventAsync accepted message [6] for transmission to IoT Hub.

[20:39:39.831] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 8 | PAYLOAD\_LEN: 83

[20:39:40.832] <- PUBACK | PACKET\_ID: 8

[20:39:42.883] IoTHubClient\_LL\_SendEventAsync accepted message [7] for transmission to IoT Hub.

[20:39:43.183] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 9 | PAYLOAD\_LEN: 83

[20:39:44.184] <- PUBACK | PACKET\_ID: 9

[20:39:46.185] IoTHubClient\_LL\_SendEventAsync accepted message [8] for transmission to IoT Hub.

[20:39:46.586] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 10 | PAYLOAD\_LEN: 83

[20:39:47.588] <- PUBACK | PACKET\_ID: 10

[20:39:49.590] IoTHubClient\_LL\_SendEventAsync accepted message [9] for transmission to IoT Hub.

[20:39:49.890] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 11 | PAYLOAD\_LEN: 83

[20:39:50.890] <- PUBACK | PACKET\_ID: 11

[20:39:52.944] IoTHubClient\_LL\_SendEventAsync accepted message [10] for transmission to IoT Hub.

[20:39:53.395] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 12 | PAYLOAD\_LEN: 83

[20:39:54.396] <- PUBACK | PACKET\_ID: 12

[20:39:56.403] IoTHubClient\_LL\_SendEventAsync accepted message [11] for transmission to IoT Hub.

[20:39:56.754] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 13 | PAYLOAD\_LEN: 83

[20:39:57.755] <- PUBACK | PACKET\_ID: 13

[20:39:59.758] IoTHubClient\_LL\_SendEventAsync accepted message [12] for transmission to IoT Hub.

[20:40:00.160] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 14 | PAYLOAD\_LEN: 83

[20:40:01.161] <- PUBACK | PACKET\_ID: 14

[20:40:03.162] IoTHubClient\_LL\_SendEventAsync accepted message [13] for transmission to IoT Hub.

[20:40:03.512] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 15 | PAYLOAD\_LEN: 83

[20:40:04.514] <- PUBACK | PACKET\_ID: 15

[20:40:06.565] IoTHubClient\_LL\_SendEventAsync accepted message [14] for transmission to IoT Hub.

[20:40:06.868] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 16 | PAYLOAD\_LEN: 83

[20:40:07.871] <- PUBACK | PACKET\_ID: 16

[20:40:09.873] IoTHubClient\_LL\_SendEventAsync accepted message [15] for transmission to IoT Hub.

[20:40:10.225] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 17 | PAYLOAD\_LEN: 83

[20:40:11.226] <- PUBACK | PACKET\_ID: 17

[20:40:13.229] IoTHubClient\_LL\_SendEventAsync accepted message [16] for transmission to IoT Hub.

[20:40:13.580] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 18 | PAYLOAD\_LEN: 83

[20:40:14.584] <- PUBACK | PACKET\_ID: 18

[20:40:16.592] IoTHubClient\_LL\_SendEventAsync accepted message [17] for transmission to IoT Hub.

[20:40:16.893] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 19 | PAYLOAD\_LEN: 83

[20:40:17.943] <- PUBACK | PACKET\_ID: 19

See [Device Explorer](https://github.com/fsautomata/azure-iot-sdks/blob/master/tools/DeviceExplorer/doc/how_to_use_device_explorer.md) to learn how to observe the messages IoT Hub receives from the device.

### 3.4 Receive messages to Azure IoT Hub

See [Device Explorer](https://github.com/fsautomata/azure-iot-sdks/blob/master/tools/DeviceExplorer/doc/how_to_use_device_explorer.md) to learn how to send cloud-to-device messages from IoT hub. The received messages are displayed in the serial terminal.

Example :

[20:44:03.366] <- PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE = 0x01 | TOPIC\_NAME: devices/dice-device-1/messages/devicebound/%24.mid=22a4184a-ecbb-473b-b816-3d5fae0a63e5&%24.to=%2Fdevices%2Fdice-device-1%2Fmessages%2FdeviceBound&iothub-ack=full&led=green | PACKET\_ID: 3 | PAYLOAD\_LEN: 0

[20:44:03.366] Received Message [0]

[20:44:03.366] Message ID: 22a4184a-ecbb-473b-b816-3d5fae0a63e5

[20:44:03.366] Correlation ID: <null>

[20:44:03.377] Data: <<<>>> & Size=0

[20:44:03.377] Message Properties:

[20:44:03.377] Key: led Value: green

[20:44:05.754] IoTHubClient\_LL\_SendEventAsync accepted message [77] for transmission to IoT Hub.

[20:44:06.106] -> PUBLISH | IS\_DUP: false | RETAIN: 0 | QOS: DELIVER\_AT\_LEAST\_ONCE | TOPIC\_NAME: devices/dice-device-1/messages/events/ | PACKET\_ID: 79 | PAYLOAD\_LEN: 83

[20:44:07.107] <- PUBACK | PACKET\_ID: 79

### 3.5 Steps to use UDS data from eFuse

1. Disable (comment) the macro USE\_TEST\_UDS in dps\_hsm\_riot.c
2. Compile the code and generate spi image.
3. Program the spi\_image.bin (See step 3.2)

Note: Once you program this image to SPI; the UDS will be locked after the bootrom loads and executes this application. Hence later if you run any application through JTAG, it will always read the UDS as 0.

#### Deriving the X.509 certificate and the thumbprint

1. On powering the board, the alias certificate is displayed on the serial terminal, after the following the text:

RIOT: Alias Certificate

Example:

*-----BEGIN CERTIFICATE-----*

*[15:11:14.359] MIIBdTCCARugAwIBAgIFDg0MCwowCgYIKoZIzj0EAwIwNDESMBAGA1UEAwwJcmlvdC1yb290MQswCQYDVQQGDAJVUzERMA8GA1UECgwITVNSX1RFU1QwHhcNMTcwMTAxMDAwMDAwWhcNMzcwMTAxMDAwMDAwWjA7MRkwFwYDVQQDDBByaW90LXNpZ25lci1jb3JlMQswCQYDVQQGDAJVUzERMA8GA1UECgwITVNSX1RFU1QwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAR2C/Fa9fotRGPpEN91dRjfT7iHPyBnpx/blWJ9CFvfA+65lPGvT2cuyVRh0XjaoDpWwhSPV/KbqYu/YrpqAu+9oxMwETAPBgNVHRMECDAGAQH/AgEBMAoGCCqGSM49BAMCA0gAMEUCIQDFhqlOoamzbIz/1NUWm2njxwjDcxM/uSOw8fc2iUkEEwIgBl2wjtTF2TXnWfN8wTj/vHRysR3lK6mit4J*

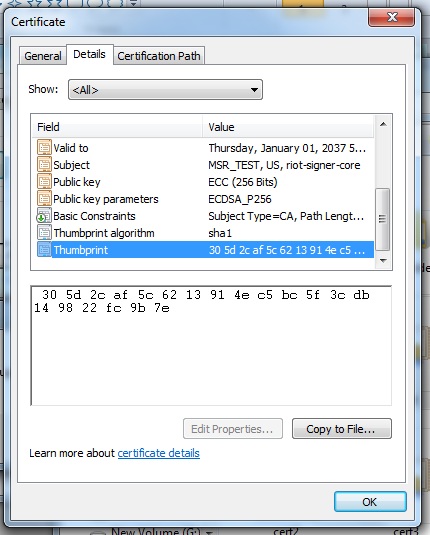
*jvO9C0oE=*

*[15:11:14.359] -----END CERTIFICATE-----*

1. Copy the text from ‘BEGIN CERTIFICATE’ to ‘END CERTIFICATE’ to a text editor file.

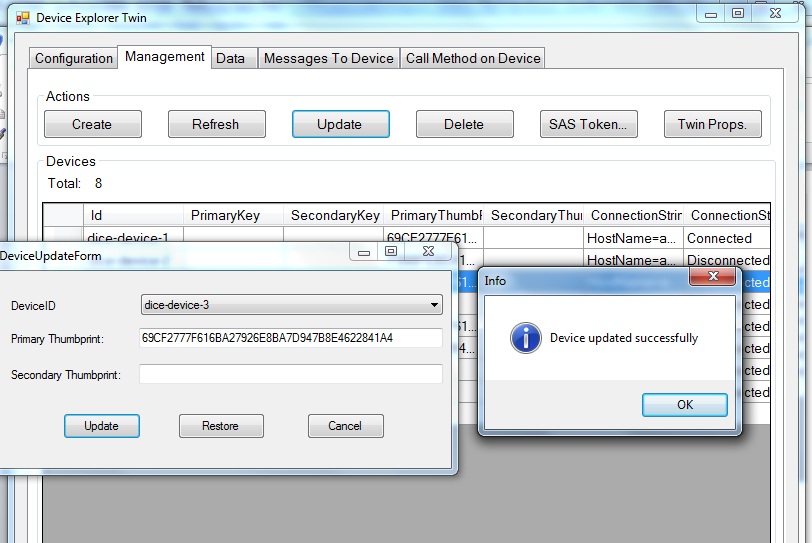
Remove the time stamps, and save with .crt extension.

1. Double click the file with .crt extension; in details tab, thumbprint bytes can be found.



1. The thumbprint should be used as primary key when registering the device in IoT hub.

Use upper case for the thumbprint bytes.



Once the thumbprint is registered for the device; on the next power-on, the device should be able to connect to the Azure IoT hub.

## Next Steps

You have now learned how to run a sample application that collects sensor data and sends it to your IoT hub. To explore how to store, analyze and visualize the data from this application in Azure using a variety of different services, please click on the following lessons:

* [Manage cloud device messaging with iothub-explorer](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-explorer-cloud-device-messaging)
* [Save IoT Hub messages to Azure data storage](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-store-data-in-azure-table-storage)
* [Use Power BI to visualize real-time sensor data from Azure IoT Hub](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-live-data-visualization-in-power-bi)
* [Use Azure Web Apps to visualize real-time sensor data from Azure IoT Hub](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-live-data-visualization-in-web-apps)
* [Weather forecast using the sensor data from your IoT hub in Azure Machine Learning](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-weather-forecast-machine-learning)
* [Remote monitoring and notifications with Logic Apps](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-monitoring-notifications-with-azure-logic-apps)

-------------------------------------------------- END OF DOCUMENT -----------------------------------------------------------