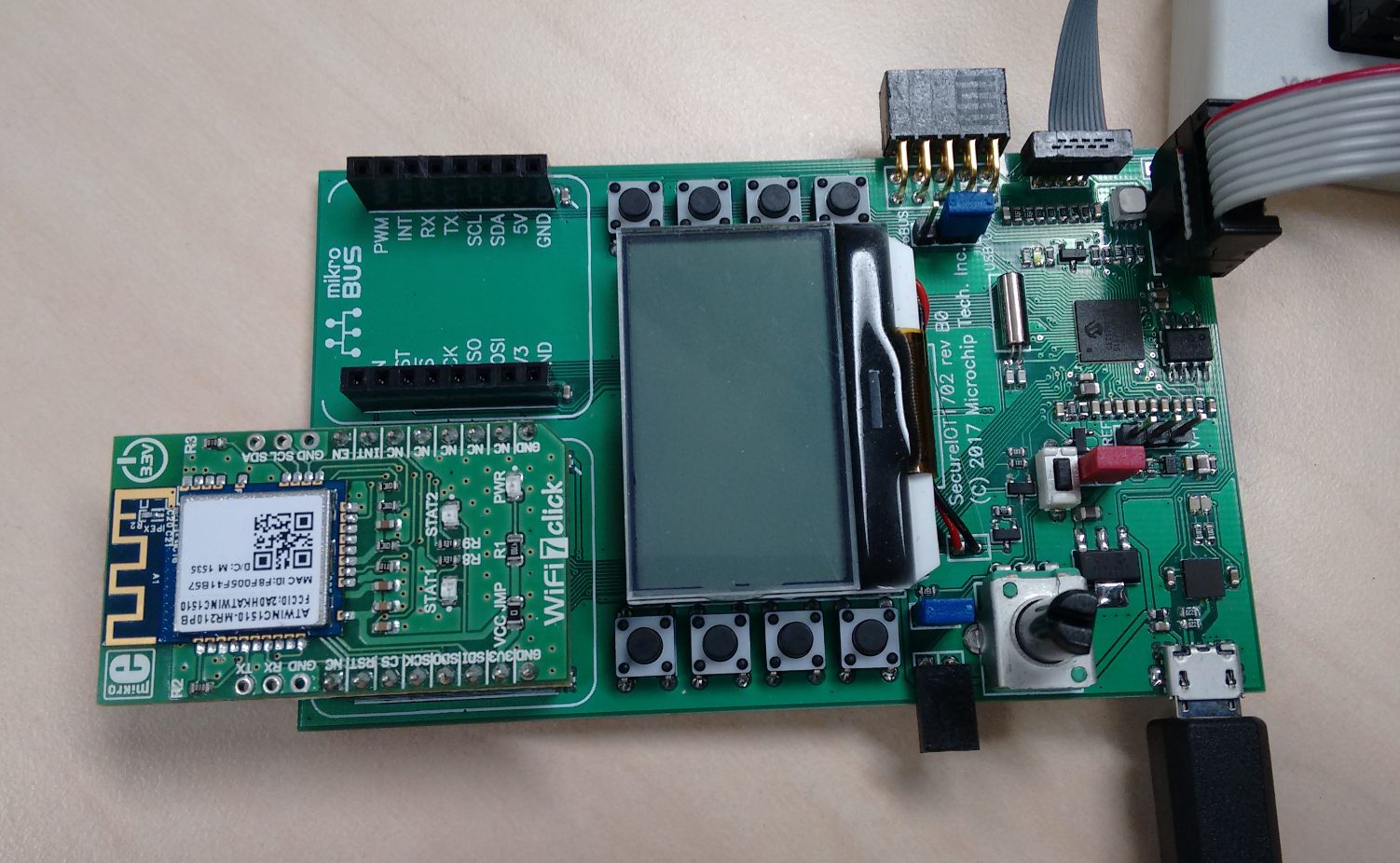
# AWS IOT on SecureIOT1702



## Hardware

***SecureIOT CEC1702 board*** – For running the AWS IOT demo application

***Winc1510 Clicker board*** – connected on mikroBus 1. This provides the Wifi connectivity

## Tools/Utilities

***Keil*** - For project build, downloading and debugging code

***DediProg SF 100*** – For loading spi image in flash

***ComXDBG.exe*** – UART host application for viewing trace messages from secureIOT1702 board

## Build firmware

The codetree **secureIOT1702\_led\_demo** contains the project and source files for compiling and building the keil project.

### Project File

The Keil project file path:

**secureIOT1702\_led\_demo\led\_demo\_project\aws\_iot\_cec1702.uvprojx**

### mbedTLS

mbedTLS is added in the project as a library

**secureIOT1702\_led\_demo\framework\aws\external\_libs\mbedTLS\libmbedtls240.lib**

### AWS IOT C SDK

AWS IOT C SDK path:

**secureIOT1702\_led\_demo\framework\aws**

The SDK is from AWS github - <https://github.com/aws/aws-iot-device-sdk-embedded-C>

The SDK is updated for CEC1702 platform.

The platform specific adapters for CEC1702:

* secureIOT1702\_led\_demo\framework\aws\platform\cec1702\common\timer.c
* secureIOT1702\_led\_demo\framework\ aws\platform\cec1702\mbedtls\ mbedtls\_net.c
* secureIOT1702\_led\_demo\framework\ aws\platform\cec1702\ mbedtls\ network\_mbedtls\_wrapper.c
* secureIOT1702\_led\_demo\framework\ aws\platform\cec1702\ mbedtls\ winc1500\_connect.c
* secureIOT1702\_led\_demo\framework\ aws\platform\cec1702\pthread\ threads\_pthread\_wrapper.c

Note: The demo application doesn’t have any RTOS or kernel, it is a simple loop; hence thread wrapper functions are empty stub functions.

### AWS Application

The AWS application is **subscribe\_publish\_app.** It uses MQTT as the transport for communicating to the AWS IOT cloud.

This file is same as one provided with AWS SDK samples; it has been updated for CEC1702 led demo.

The certificates and keys required for authentication using X.509 is hardcoded in the **subscribe\_publish\_app.c** file.

* ***Root Certificate*** : char rootCA[ ] = “ “
* ***Client Certificate*** : char clientCRT[ ] = “ “
* ***Client Private Key*** : char clientKey[ ] == “ “

Configuration for AWS connectivity is done through aws\_iot\_config.h file. Update the host, client\_id and thing name based on your AWS setup.

### WiFi Configuration

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

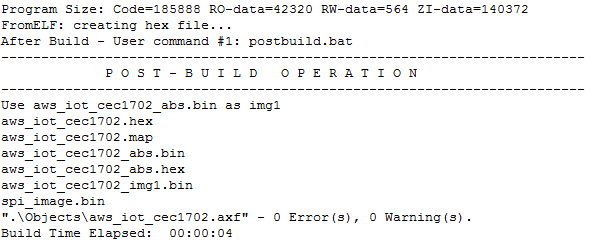


Modify the above values to match your wifi router.

### Compiler

* Keil uVision V5.20.0.0

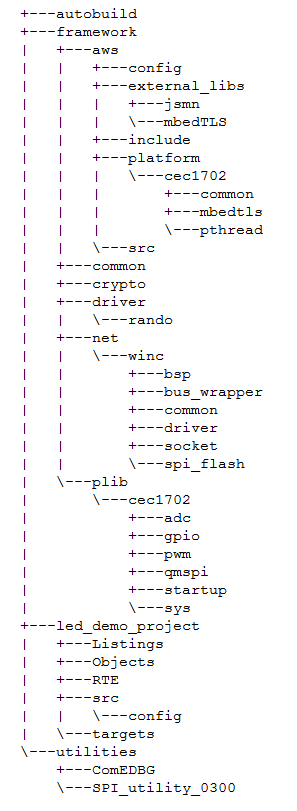
### Output



The spi\_image is created as part of the post-build process. The output files are placed in:

**secureIOT1702\_led\_demo\led\_demo\_project\targets**

### Code Tree Structure



## Build Firmware into Flash

Use ***DediProg SF 100*** tool to program the image – ***spi\_image.bin*** (created during the postbuild process) into flash on the secureIOT1702 board.

## ComXDBG

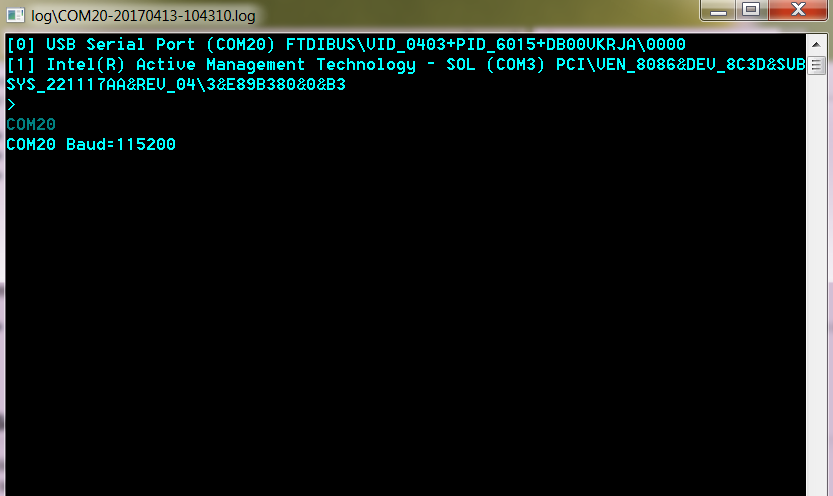
To view the UART traces:

1. connect USB cable between secureIOT1702 board and Windows host
2. After driver installation, start ComXDBG.exe

**secureIOT1702\_led\_demo\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 secureIOT1702 board

### Sample log

[11:45:11.175] ORIGINAL IMAGE

[11:45:11.175] AWS IOT Node

[11:45:11.175] Microchip CEC1702

[11:45:11.175] HW Version: SECUREIOT1702\_REV\_B0

[11:45:11.175] FW Version/Label: SECUREIOT1702\_AWS\_LED\_0100

[11:45:11.175] May 11 2017 11:43:05

[11:45:11.479] $Ts:

[11:45:12.033] winc1500\_wifi\_cb: 2c

[11:45:12.136] M2M\_WIFI\_RESP\_CON\_STATE\_CHANGED: CONNECTED

[11:45:12.136] winc1500\_wifi\_cb: 32

[11:45:12.136] M2M\_WIFI\_REQ\_DHCP\_CONF: IP is 192.168.0.102

[11:45:12.136] WINC is connected to TP-LINK\_258C successfully!

[11:45:12.136] winc1500\_wifi\_init: Done

[11:45:12.136] Initializing rando.

[11:45:12.136] AWS IoT SDK Version 2.1.1-

[11:45:12.136] Connecting...

[11:45:12.136] . Seeding the random number generator...

[11:45:12.136] . Loading the CA root certificate ...

[11:45:12.136] ok (0 skipped)

[11:45:12.136] . Loading the client cert. and key...

[11:45:12.447] ok

[11:45:12.497] . Connecting to a3hglwcan0yf7v.iot.us-west-2.amazonaws.com/8883

...winc1500\_wifi\_cb: 20

[11:45:12.547] SERVER IP is 52.88.193.255

[11:45:12.597] ok

[11:45:12.597] . Setting up the SSL/TLS structure...

[11:45:12.648] SSL state connect : 0 ok

[11:45:12.648] SSL state connect : 0 . Performing the SSL/TLS handshake...

[11:45:14.256] Verify requested for (Depth 2):

[11:45:14.256] cert. version : 3

[11:45:14.256] serial number : 18:DA:D1:9E:26:7D:E8:BB:4A:21:58:CD:CC:6B:3B:

4A

[11:45:14.256] issuer name : C=US, O=VeriSign, Inc., OU=VeriSign Trust Net

work, OU=(c) 2006 VeriSign, Inc. - For authorized use only, CN=VeriSign Class 3

Public Primary Certification Authority - G5

[11:45:14.256] subject name : C=US, O=VeriSign, Inc., OU=VeriSign Trust Net

work, OU=(c) 2006 VeriSign, Inc. - For authorized use only, CN=VeriSign Class 3

Public Primary Certification Authority - G5

[11:45:14.256] issued on : 2006-11-08 00:00:00

[11:45:14.256] expires on : 2036-07-16 23:59:59

[11:45:14.256] signed using : RSA with SHA1

[11:45:14.256] RSA key size : 2048 bits

[11:45:14.256] basic constraints : CA=true

[11:45:14.256] key usage : Key Cert Sign, CRL Sign

[11:45:14.256] This certificate has no flags

[11:45:14.256] Verify requested for (Depth 1):

[11:45:14.256] cert. version : 3

[11:45:14.256] serial number : 3F:92:87:BE:9D:1D:A4:A3:7A:9D:F6:28:2E:77:5A:

C4

[11:45:14.256] issuer name : C=US, O=VeriSign, Inc., OU=VeriSign Trust Net

work, OU=(c) 2006 VeriSign, Inc. - For authorized use only, CN=VeriSign Class 3

Public Primary Certification Authority - G5

[11:45:14.256] subject name : C=US, O=Symantec Corporation, OU=Symantec Tru

st Network, CN=Symantec Class 3 ECC 256 bit SSL CA - G2

[11:45:14.256] issued on : 2015-05-12 00:00:00

[11:45:14.256] expires on : 2025-05-11 23:59:59

[11:45:14.256] signed using : RSA with SHA-256

[11:45:14.256] EC key size : 256 bits

[11:45:14.256] basic constraints : CA=true, max\_pathlen=0

[11:45:14.256] subject alt name :

[11:45:14.256] key usage : Key Cert Sign, CRL Sign

[11:45:14.308] This certificate has no flags

[11:45:14.308] Verify requested for (Depth 0):

[11:45:14.308] cert. version : 3

[11:45:14.308] serial number : 20:88:52:7E:45:DF:8F:76:68:0D:39:EA:E2:26:12:

D2

[11:45:14.308] issuer name : C=US, O=Symantec Corporation, OU=Symantec Tru

st Network, CN=Symantec Class 3 ECC 256 bit SSL CA - G2

[11:45:14.308] subject name : C=US, ST=Washington, L=Seattle, O=Amazon.com,

Inc., CN=\*.iot.us-west-2.amazonaws.com

[11:45:14.308] issued on : 2017-03-07 00:00:00

[11:45:14.308] expires on : 2018-03-08 23:59:59

[11:45:14.308] signed using : ECDSA with SHA256

[11:45:14.308] EC key size : 256 bits

[11:45:14.308] basic constraints : CA=false

[11:45:14.308] subject alt name : iot.us-west-2.amazonaws.com, \*.iot.us-west-2.

amazonaws.com

[11:45:14.308] key usage : Digital Signature

[11:45:14.308] ext key usage : TLS Web Server Authentication, TLS Web Client

Authentication

[11:45:14.308] This certificate has no flags

[11:45:22.677] ok

[11:45:22.677] [ Protocol is TLSv1.2 ]

[11:45:22.677] [ Ciphersuite is TLS-ECDHE-ECDSA-WITH-AES-256-GCM-SHA384 ]

[11:45:22.677] [ Record expansion is 29 ]

[11:45:22.677] . Verifying peer X.509 certificate...

[11:45:22.677] ok

[11:45:22.796] autoreconnect\_set\_status...

[11:45:22.796] Subscribing...

[11:45:22.902] aws\_iot\_mqtt\_yield...

[11:45:27.926] ======================================================

[11:45:28.035] Temperature is: 25.50 Celsius

[11:45:28.035] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[11:45:28.035] aws\_iot\_mqtt\_yield...

[11:45:33.123] ======================================================

[11:45:33.276] Temperature is: 26.00 Celsius

[11:45:33.276] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[11:45:33.276] aws\_iot\_mqtt\_yield...

[11:45:38.915] ======================================================

[11:45:39.020] Temperature is: 26.50 Celsius

[11:45:39.020] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Steps to run the AWS demo

### Setup AWS Account

1. Create Amazon AWS Account

<https://aws.amazon.com/account/>

2. Install and setup Amazon Command Line Interface

<http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-set-up.html>

3. Configure AWS Command Line interface

<http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html>

### AWS Thing and Certificate Creation

1. Create your IOT Thing

**aws iot create-thing --thing-name "yourThingName"**

2. Create certificate and keys

**aws iot create-keys-and-certificate --set-as-active --certificate-pem-outfile cert.pem --public-key-outfile publicKey.pem --private-key-outfile privkey.pem**

Note the certificate-arn in output. You can also get it by listing all certificates

**aws iot list-certificates**

### Attach a Policy to the Certificate

1. Create a simple iotpolicy.json file with following as its contents:

{

"Version*"*: "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Action":["iot:\*"],

"Resource": ["\*"]

}]

}

This is a simple test policy file allowing all iot actions on all resources

2. Create the AWS policy using the iotpolicy.json file

**aws iot create-policy --policy-name "PubSubToAnyTopic" --policy-document file://iotpolicy.json**

3. Attach the policy to the certificate

Use your **certificate-arn** in the command

**aws iot attach-principal-policy --principal "certificate-arn" --policy-name "PubSubToAnyTopic"**

### Build AWS IOT SecureIOT1702 project

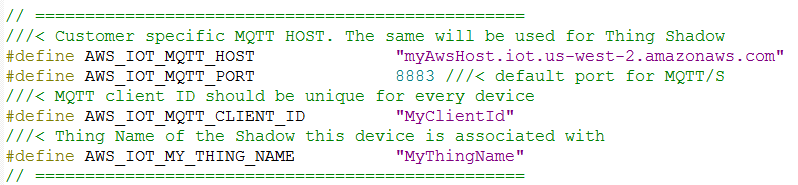
1. Open the **secureIOT1702\_led\_demo** Keil project
2. Update subscribe\_publish\_app.c with your certificates and keys:
   * ***Root Certificate*** : char rootCA[ ] = “ “
   * ***Client Certificate*** : char clientCRT[ ] = “ “
   * ***Client Private Key*** : char clientKey[ ] == “ “

*Example for Root Certificate*:



1. Update ***aws\_iot\_config.h*** to match your AWS host, client and thing

Example:



1. Update CONN\_SSID, CONN\_PSK\_PWD in winc1500\_connect.c to match your wifi router.
2. Build the Keil project
3. Program the firmware binary (***targets\ spi\_image.bin***) into flash using DediProg SF 100

### AWS Led Demo

1. After the firmware is programmed, turn on the secureIOT1702 board
2. Once securely connected, the AWS demo will send emulated temperature values to the AWS IoT cloud.

Basically it is posting emulated temperature values to the MQTT topic ***sdk/test/temperature***.

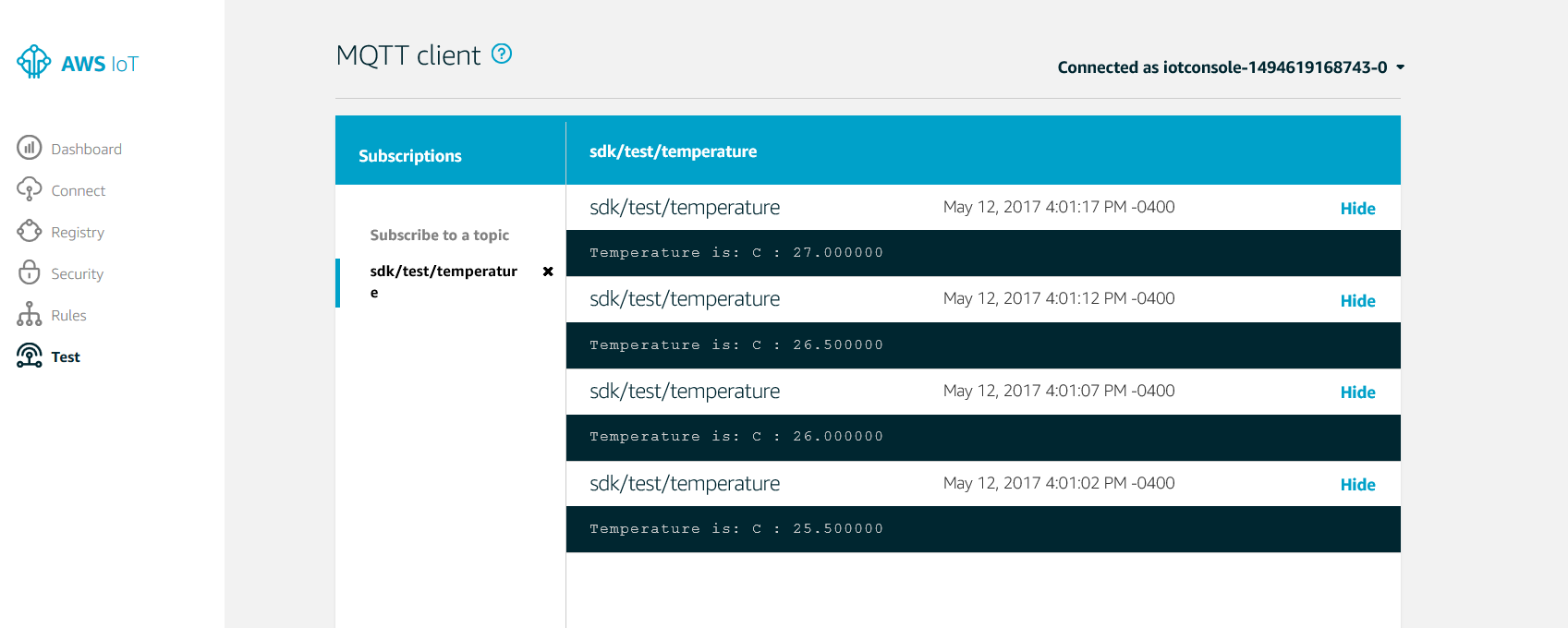
You should be able to see these updating values in AWS IoT Dashboard.

1. Open <https://aws.amazon.com/iot/>
2. Sign In to IoT Console
3. Go to All Services -> Internet of Things and click on AWS IOT
4. On the AWS IoT Console click on “Test” on left hand side pane
5. Here we need to subscribe to the following topic:

sdk/test/temperature

The subscribed topic will be added on the left hand side under subscriptions

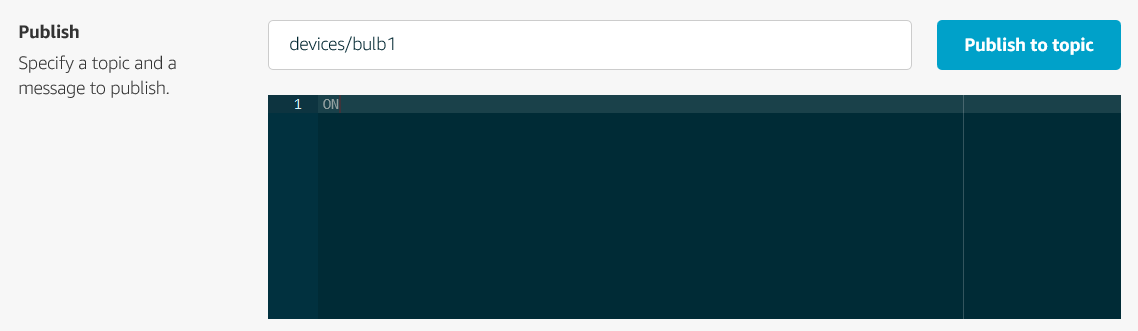
1. Click on the subscribed temperature topic to see the temperature reading updates every 5 seconds.



1. To turn on/off the led on the secureIOT1702 board, we need to post to the following topic

***devices/bulb1***

In the same “Test” tab in the AWS IoT Console, you can publish to topics. Simply publish ON or OFF to the topic devices/bulb1



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