

□

AWS IoT and Adafruit WICED Feather

Created by Kevin Townsend



Last updated on 2016-07-26 02:46:52 AM UTC

Guide Contents

| | |
|---|----|
| Guide Contents | 2 |
| Introduction | 3 |
| AWS IoT Setup | 5 |
| Enable AWS Services for your Amazon Account | 5 |
| Login to AWS IoT Admin | 6 |
| Create a Thing | 7 |
| Connect a Device | 9 |
| Store the Thing Connection Details | 10 |
| Arduino Sketch Setup | 11 |
| Convert Certificate with pycert | 11 |
| Add the Private Key to the Sketch | 11 |
| Add the AWS IoT MQTT Details | 12 |
| Add AP Connection Details | 13 |

Introduction



[AWS IoT \(http://adafru.it/pex\)](http://adafru.it/pex), part of Amazon's ongoing attempt to corner all aspects of the 'cloud', is essentially a secure MQTT broker with a management console and some back-end logic behind it to make it easier and more flexible to work with than a vanilla broker like [Mosquitto \(http://adafru.it/pey\)](http://adafru.it/pey).

One of the key aspects that makes AWS IoT stand out compared to other MQTT solutions is the security layer. In addition to validating the remote MQTT server with a certificate (similar to how HTTPS works), the individual sensor nodes are *also* verified via a second device-side certificate. What this means is that there is a very high level of certainty that the device pushing the data out to the MQTT broker or subscribing to data from it is exclusively the device you think it is.

Unfortunately, this this also makes using AWS IoT more of a technical challenge, and there aren't a lot of embedded platforms out there that can jump through all of the technical hoops required to work with AWS IoT. Thankfully, with the [WICED Feather \(http://adafru.it/pez\)](#), we've got you covered!

This guide will walk you through configuring a new 'thing' with AWS IoT, and run you through the steps required to start securely communicating with Amazon's servers using a WICED Feather, including getting the two required certificates into a form the WICED Feather can use.

New to AWS? Don't worry about it, you should have everything you need here to get up and running in under an hour!



AWS IoT Setup

This page will guide you through the process of setting up a new 'thing' on AWS, and creating the required certificates to enable the WICED Feather to talk to the AWS IoT MQTT servers.

Enable AWS Services for your Amazon Account

Before you can do anything, you will need to make sure that your Amazon account has billing enabled (even if you are using a free AWS plan!) and that your account is associated with a valid credit card.

You will also need to verify your identity, which you can do by clicking on the following link: <https://aws-portal.amazon.com/gp/aws/developer/registration/index.html> (<http://adafru.it/peA>)

This will call the phone you associated with your account, and you will supply a PIN number, and you will then be able to select the AWS support plan that you wish to use:

Support Plan

AWS Support offers a selection of plans to meet your needs. All plans provide 24x7 access to customer service, AWS documentation, whitepapers, and support forums. For access to technical support and additional resources to help you plan, deploy, and optimize your AWS environment, we recommend selecting a support plan that best aligns with your AWS usage.

Please Select One

- Basic**
Description: Customer Service for account and billing questions and access to the AWS Community Forums.
Price: Included
- Developer**
Use case: Experimenting with AWS
Description: One primary contact may ask technical questions through Support Center and get a response within 12–24 hours during local business hours.
Price: \$49/month

If you don't have a valid credit card associated with your account and if you haven't verified your account, you will get the following error whenever you try to use AWS IoT via the AWS Console:



Access to the resource AWSIoT is denied.

There may be a delay of up to 24 hours for the account activation to take effect. You can test the account status by clicking a related link like <https://console.aws.amazon.com/lambda/home> (<http://adafru.it/peB>) which will show you the following screen if your account activation is still in progress:

Your service sign-up is almost complete!

Thanks for signing up with Amazon Web Services. Your services may take up to 24 hours to fully activate. If you're unable to access AWS services after that time, here are a few things you can do to expedite the process:

1. Make sure you provided all necessary information during signup. [Complete your AWS registration.](#)
2. Check your email to see if you have received any requests for additional information. If you have, please respond to those emails with the information requested.
3. Verify your [credit card information](#) is correct. Also, check your credit card activity to see if there's a \$1 authorization (this is not a charge). You may need to contact your card issuer to approve the authorization.

If the problem persists, please contact Support:

If your billing information is up to date, but the error message persists when trying to create a 'thing' (later in this tutorial) you can try resubscribing to AWS services with the following link: <https://portal.aws.amazon.com/billing/signup?type=resubscribe#/> (<http://adafru.it/peC>)

Login to AWS IoT Admin

The first thing to do once you have a verified account is to login to the [AWS Console](#) (<http://adafru.it/peD>). If you don't already have an account you can create one, or login with your existing details

From the AWS Console home page, find the **AWS IoT** icon shown below and click on it:



Click the **Get Started** button on the AWS IoT homepage:

AWS IoT

AWS IoT is a managed cloud platform that lets connected devices -- cars, light bulbs, sensor grids and more -- easily and securely interact with cloud applications and other devices.

Get started

Start interactive tutorial

[Getting started documentation](#)

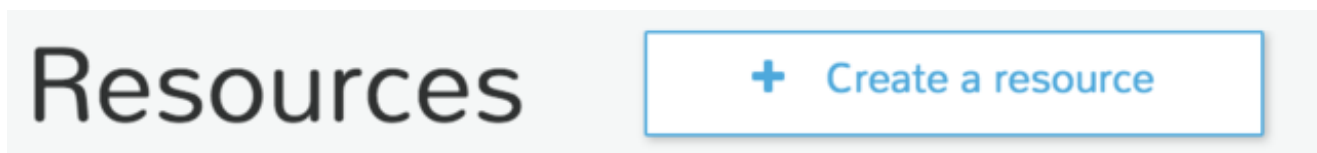
Create a Thing

Before you can work with AWS IoT you will need to 'Create a Thing', which is done via the wizard shown below:

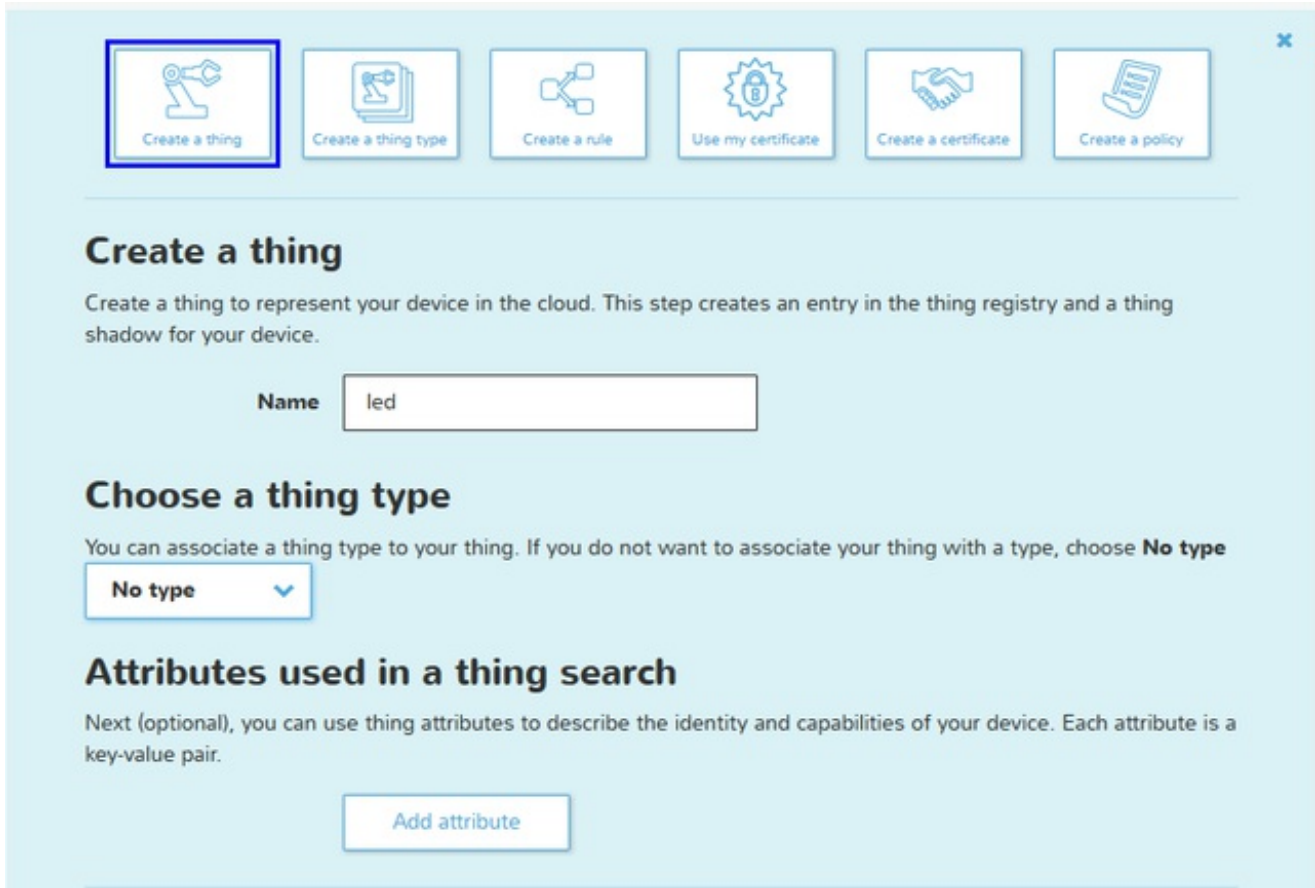


The screenshot shows the 'Create a thing' wizard in the AWS IoT console. At the top, there are five navigation buttons: 'Create a thing' (highlighted with a blue border), 'Create a rule', 'Use my certificate', 'Create a certificate', and 'Create a policy'. Below the navigation bar, the main heading is 'Create a thing' with a sub-heading: 'Create a thing to represent your device in the cloud. This step creates an entry in Device Registry and also a Device Shadow for your device.' There is a text input field for 'Name'. Below that, the 'Attributes' section is shown with a sub-heading: 'Next (optional), you can use thing attributes to describe the identity and capabilities of your device. Each attribute is a key-value pair.' There is an 'Add attribute' button. At the bottom right, there is a 'Create' button.

If you didn't use the **Get Started** button and don't see the wizard above, you can find it by clicking on the **Resources** link in the top-right hand corner, and then clicking the **Create a resource** button shown below:

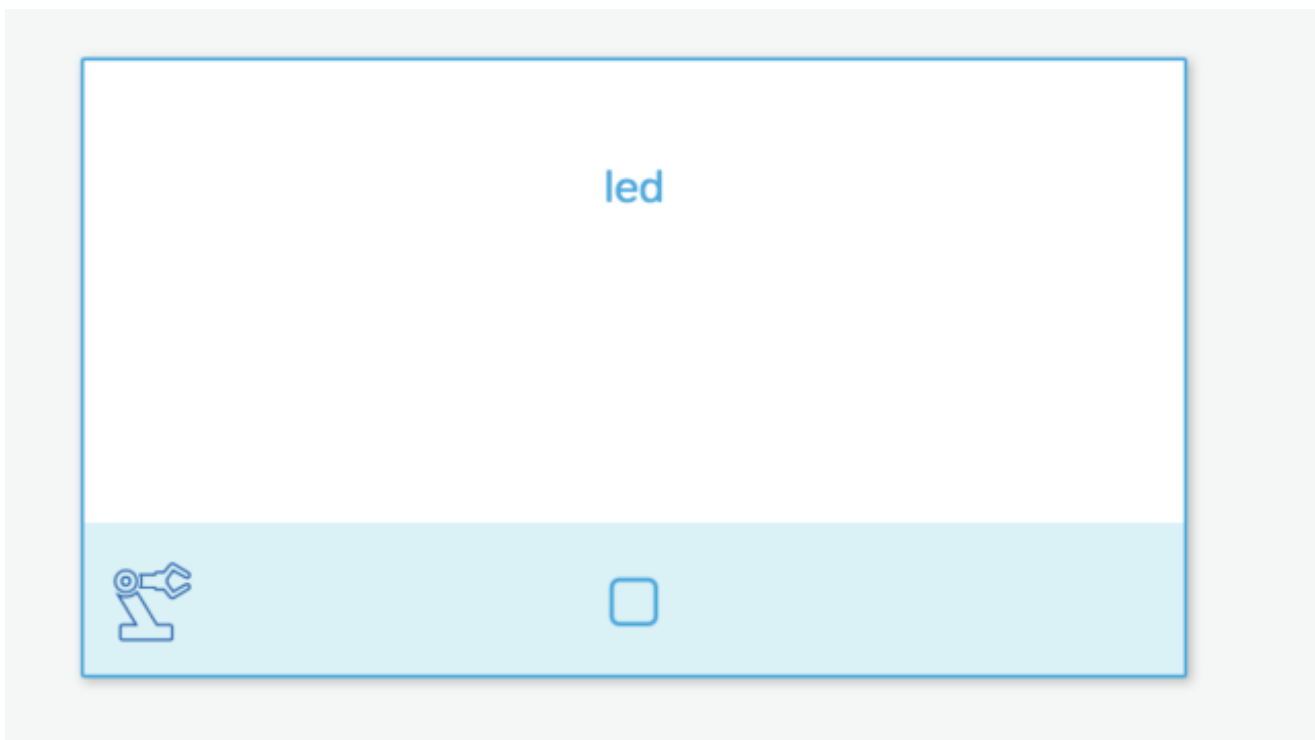


Name the thing **led** (no attributes or type) and click the **create** button:



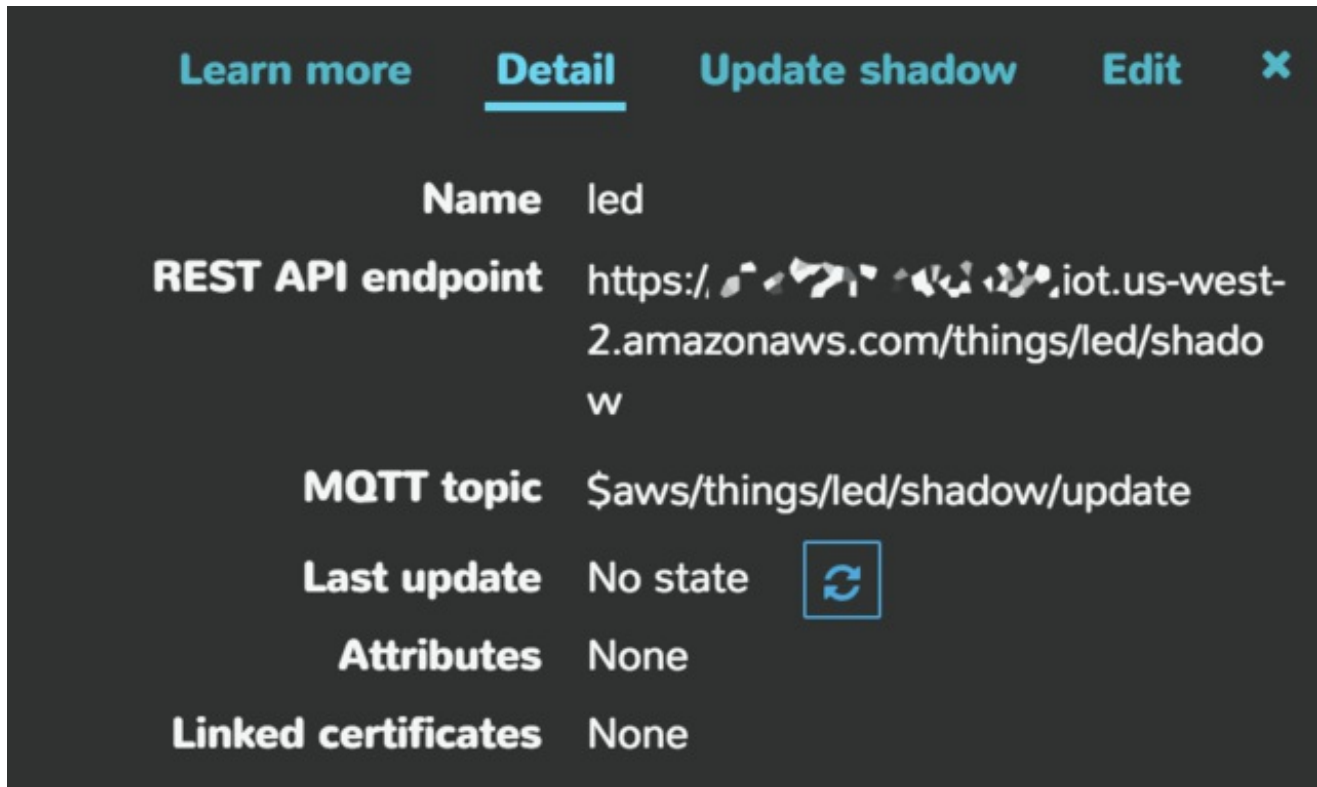
The screenshot shows the 'Create a thing' wizard in the AWS IoT console. At the top, there is a navigation bar with six icons: 'Create a thing' (highlighted with a blue border), 'Create a thing type', 'Create a rule', 'Use my certificate', 'Create a certificate', and 'Create a policy'. Below the navigation bar, the main heading is 'Create a thing'. The instructions state: 'Create a thing to represent your device in the cloud. This step creates an entry in the thing registry and a thing shadow for your device.' There is a 'Name' input field containing the text 'led'. Below that is the 'Choose a thing type' section, which says: 'You can associate a thing type to your thing. If you do not want to associate your thing with a type, choose **No type**'. A dropdown menu is set to 'No type'. The next section is 'Attributes used in a thing search', with instructions: 'Next (optional), you can use thing attributes to describe the identity and capabilities of your device. Each attribute is a key-value pair.' There is an 'Add attribute' button at the bottom of this section.

You should see a small box with your new 'thing' in the bottom left-hand side of the screen:





Click the **View Thing** button to see the details for your new 'led' thing on the right-hand

side of the admin console:

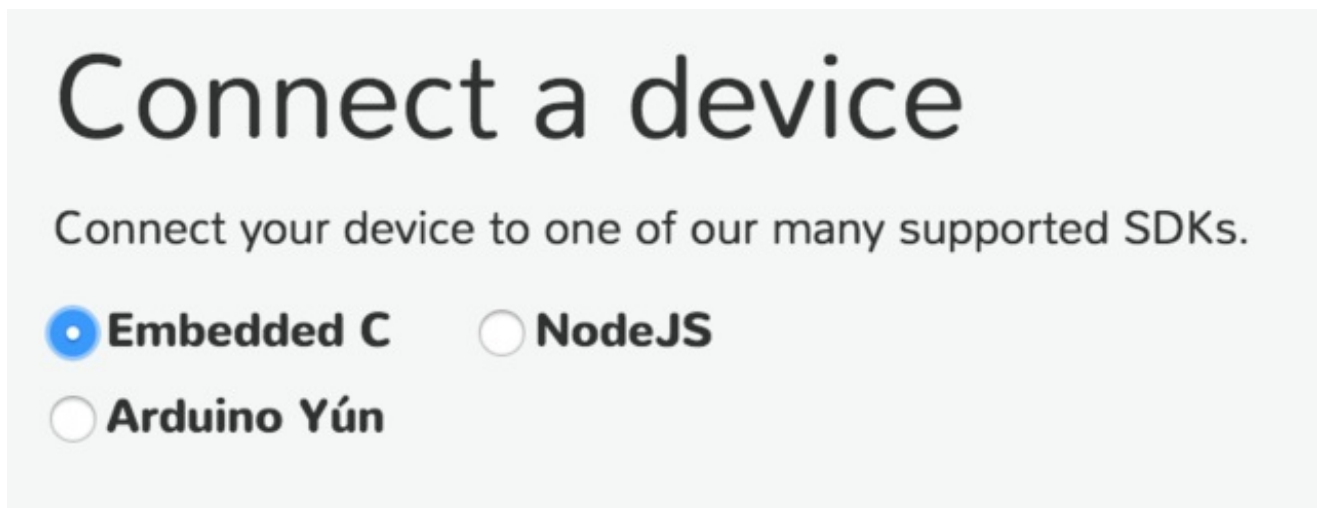


The screenshot shows the 'Detail' page for an IoT shadow in the AWS IoT console. At the top, there are navigation links: 'Learn more', 'Detail' (underlined), 'Update shadow', 'Edit', and a close button 'x'. Below these are several key-value pairs:

| | |
|----------------------------|--|
| Name | led |
| REST API endpoint | https://  ,iot.us-west-2.amazonaws.com/things/led/shadow |
| MQTT topic | \$aws/things/led/shadow/update |
| Last update | No state  |
| Attributes | None |
| Linked certificates | None |

Connect a Device

With the thing opened up on the side panel, click the **Connect a Device** button in the bottom right-hand corner, and select the **Embedded C** option:



The screenshot shows a dialog box titled 'Connect a device'. Below the title is the text 'Connect your device to one of our many supported SDKs.' There are three radio button options: 'Embedded C' (which is selected), 'NodeJS', and 'Arduino Yún'.

Then click the **Generate certificate and policy** button that appears on the right-hand side:

First, you will need to create and download security credentials for your device. The following steps will help you to create and download security credentials (a certificate for authentication, and a policy that defines what the device using this certificate is allowed to do).

You can generate a certificate with 1-click. When you generate a certificate, we will also generate a default security policy named led-Policy. You can modify this security policy at any time through the 'Resources' panel of this console.

Generate certificate and policy

Download the **Private Key** and **Certificate** to a folder, since you will need to place these files in the same location as your WICED Arduino sketch later on:

Please download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys will not be retrievable after closing this form.

- [Download public key](#)
- [Download private key](#)
- [Download certificate](#)

Confirm & start connecting

Click the **Confirm & start connecting** button to continue.

Store the Thing Connection Details

At this point, you should see some C pre-processor macros that contains the information necessary for the WICED Feather to connect to the AWS IoT servers. Copy these values, and store them somewhere in a text file since they will need to be added to the AWS Arduino Sketch we'll edit in a moment:

Note: You only need the first four entries in the code snippet below

```
// Get from console
// =====
#define AWS_IOT_MQTT_HOST      "a21n9iuswv1kz1.1702.iot.us-west-2.amazonaws.com"
#define AWS_IOT_MQTT_PORT      8883
#define AWS_IOT_MQTT_CLIENT_ID "led"
#define AWS_IOT_MY_THING_NAME  "led"
#define AWS_IOT_ROOT_CA_FILENAME "root-CA.crt"
#define AWS_IOT_CERTIFICATE_FILENAME "a21n9iuswv1kz1-certificate.pem.crt"
#define AWS_IOT_PRIVATE_KEY_FILENAME "a21n9iuswv1kz1-private.pem.key"
// =====
```



Arduino Sketch Setup

AWS IoT uses MQTT, which is already well supported by the WICED Feather, but it has some additional requirements around security with the need for a certificate on the sensor node side (the WICED Feather) to validate that the device AWS IoT is talking to is indeed the right device.

To address some of the specificities of talking to AWS, the WICED Feather includes a sample AWS sketch in the **MQTT/aws** folder in examples. You should have this available on your system, but for convenience sake you can also browse the latest code online in github using the link below:

[Show AWS Example Sketch on Github](http://adafru.it/peE)

<http://adafru.it/peE>

Convert Certificate with pycert

To validate the WICED Feather, we need to convert the certificates downloaded in the previous page of this tutorial into a format the WICED SDK understands. You can do this with the [pycert tool](http://adafru.it/peF) (<http://adafru.it/peF>) we provide as part of the WICED Feather BSP.

Place the ***certificate.pem.crt** and ***private.pem.key** files in the sketch folder, and run the pycert tools as follows (you may need to adjust the '..' path to points to tools/pycert in the WICED Feather BSP):

```
python ../../tools/pycert/pycert.py convert -c local_cert -l LOCAL_CERT_LEN *****-certificate.pem.crt
```

This should create a **certificates.h** file in the sketch folder containing the certificate data in 'local_cert':

```
Loaded certificate *****-certificate.pem.crt  
Wrote certificates.h
```

You can now reference certificates.h in your sketch (it is already included in the code), which takes care of one of the two certificates we need to deal with for AWS.

Add the Private Key to the Sketch

Now we need to add the private key to our demo sketch. The file downloaded is already in

a binary format, so it doesn't need to be run through the `pycert` conversion tool, but you do need to copy the contents of the certificate into the sketch and perform a bit of manual labour to make it work.

Copy the contents of the `*private.pem.key` file to the browser, and paste them into your sketch in the `aws_private_key` array that is already defined:

```
const char aws_private_key[] =  
"-----BEGIN RSA PRIVATE KEY-----\n"  
"Your Key line 1 with\n"  
"Quote each of your key's lines like this example\n"  
"-----END RSA PRIVATE KEY-----";
```

You will need to place quotes before and after each line, and also add an `\n` newline character after each line! You'll end up with something like this:

```
const char aws_private_key[] =  
"-----BEGIN RSA PRIVATE KEY-----\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n"  
"-----END RSA PRIVATE KEY-----";
```

Add the AWS IoT MQTT Details

Finally you need to add the MQTT Server details for the AWS server, which will mean

replacing these four values with the data generated in the previous page of this tutorial:

```
// =====  
#define AWS_IOT_MQTT_HOST      "*****.iot.us-west-2.amazonaws.com"  
#define AWS_IOT_MQTT_PORT      8883  
#define AWS_IOT_MQTT_CLIENT_ID "led"  
#define AWS_IOT_MY_THING_NAME  "led"  
// =====
```

Add AP Connection Details

At this point you simply need to add your connection details, and you should be able to flash the sketch to your WICED Feather and open the Serial Monitor to see the connection status:

Note: Make sure you are using the latest version of FeatherLib and the Arduino WICED BSP before running this code. You will need at least FeatherLib 0.5.5 for it to run.

AWS IOT Example

```
Bootloader : 1.0.0  
WICED SDK  : 3.5.2  
FeatherLib : 0.5.5  
Arduino API : 0.5.6
```

Please wait while connecting to: 'CLARAWIFI' ... Connected!

```
SSID      : CLARAWIFI (-78 dBm)  
Encryption : WPA2_AES  
MAC Address : FF:FF:FF:FF:FF:FF  
Local IP   : 10.0.1.10  
Gateway    : 10.0.1.1  
Subnet Mask : 255.255.255.0
```

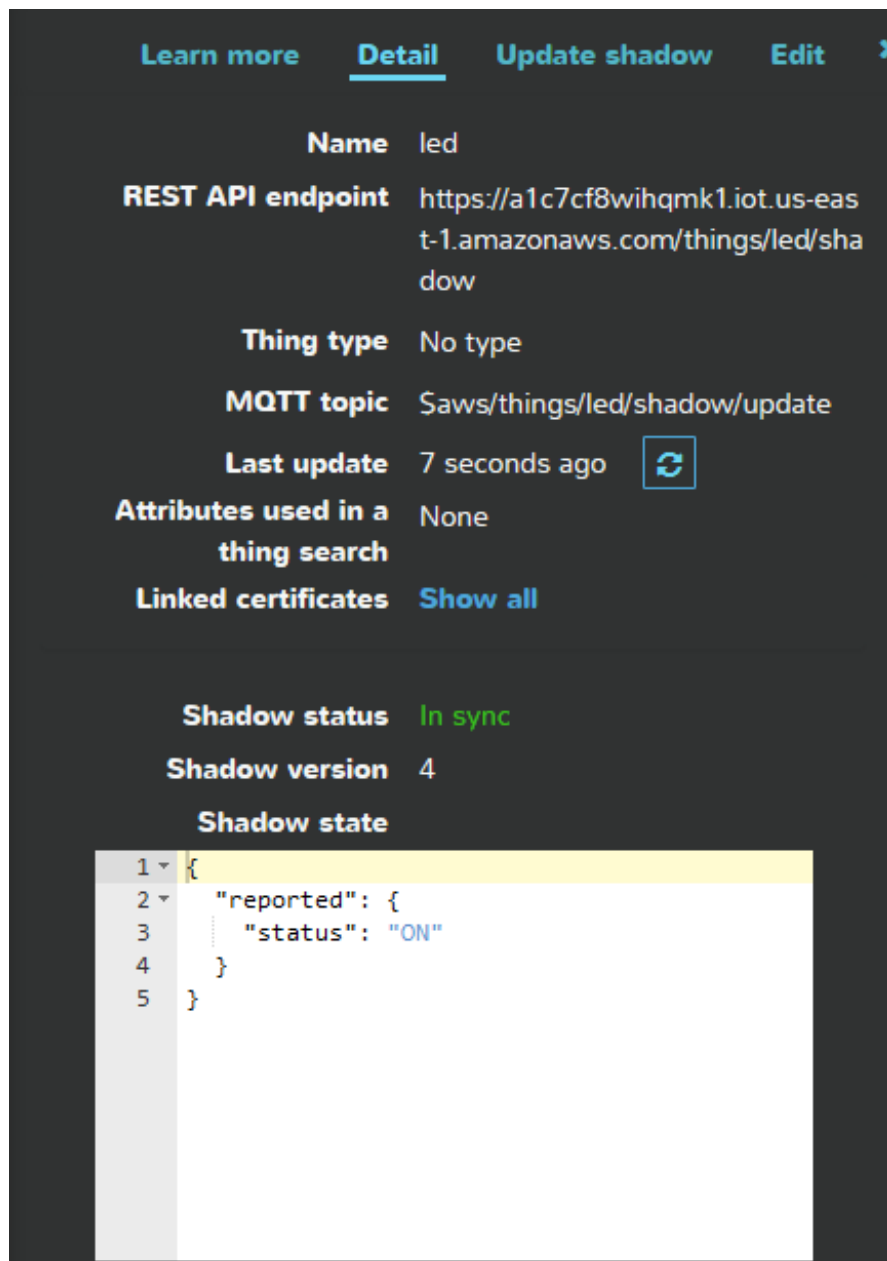
```
Connecting to *****.iot.us-west-2.amazonaws.com port 8883 ... OK  
Subscribing to $aws/things/led/shadow/update ... OK  
Enter '0' or '1' to update feed:
```

Entering either 1 or 0 into the Serial Monitor input should turn the LED on the WICED Feather on or off via AWS IoT. There will be a few seconds lag, of course, while the data makes a round trip and before the MQTT event is capture by the WICED Feather:

```
Enter '0' or '1' to update feed: 1  
Enter '0' or '1' to update feed:
```

```
Enter '0' or '1' to update feed: 0  
Enter '0' or '1' to update feed:
```

Enter '0' or '1' to update feed:



The screenshot shows the AWS IoT console interface for a device shadow. At the top, there are navigation tabs: 'Learn more', 'Detail' (which is underlined), 'Update shadow', and 'Edit'. Below the tabs, the shadow details are listed:

- Name:** led
- REST API endpoint:** https://a1c7cf8wihqmk1.iot.us-east-1.amazonaws.com/things/led/shadow
- Thing type:** No type
- MQTT topic:** Saws/things/led/shadow/update
- Last update:** 7 seconds ago (with a refresh icon)
- Attributes used in a thing search:** None
- Linked certificates:** Show all
- Shadow status:** In sync
- Shadow version:** 4

Below these details is a section titled 'Shadow state' which contains a JSON representation of the shadow's state:

```
1 {
2   "reported": {
3     "status": "ON"
4   }
5 }
```

That's it! At this point, you can start looking at the example sketch we used above to see how you can modify it to fit your needs, but you've done all the heavy lifting to get AWS IoT working, and securely connecting your WICED Feather to Amazon's servers!

Doing something neat with AWS IoT? Make sure to let us know in the forums!