AN3089

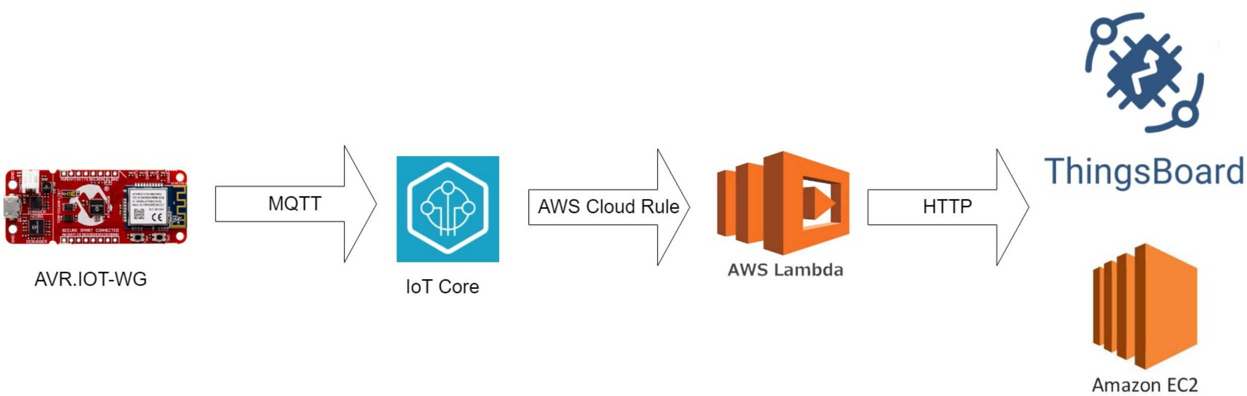
Repurpose AVR-IoT WG to Connect to AWS

Introduction

This application note describes how to connect the **AVR-IoT WG** Development Board to the **AWS IoT Core**, send light and temperature data, and plot them as line charts. This document shows the detailed steps of building a solution deeply integrated with the Amazon Web Services (AWS) Cloud. It uses AWS IoT Core for managing devices and integrates with other AWS services, such as AWS Lambda, to send data into ThingsBoard, which is used to visualize them.

The high-level architecture of this application is shown in the figure below.

Figure 1. High-Level Architecture of the Application



The code is available on GitHub.

Table of Contents

1. What is AWS-IoT WG

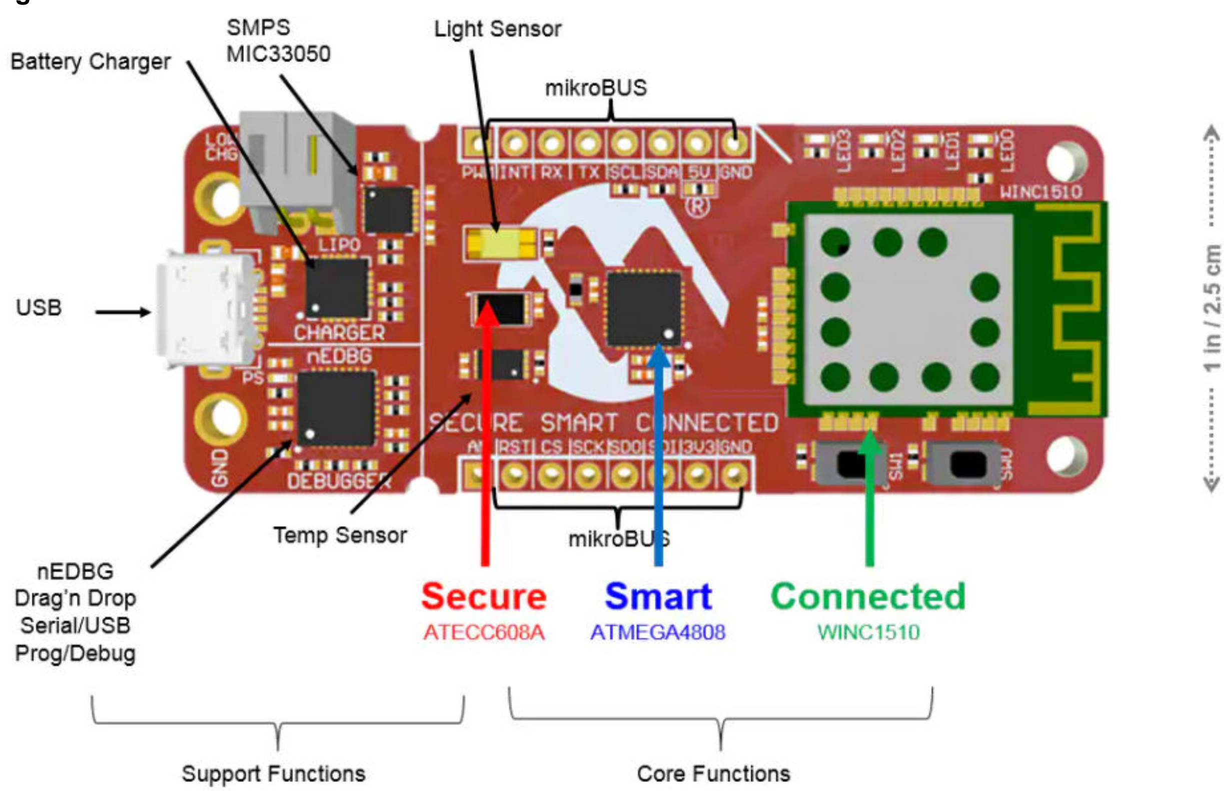
The AVR-IoT WG Development Board is a small and easily expandable demonstration and development platform for IoT solutions, based on the AVR™ microcontroller architecture using Wi-Fi® technology. It was designed to demonstrate that the design of a typical IoT application can be simplified by partitioning the problem into three blocks:

- Smart - represented by the ATmega4808 microcontroller
- Secure - represented by the ATECC608A secure element
- Connected - represented by the ATWINC1510 Wi-Fi controller module

The AVR-IoT WG Development Board features two sensors: a light sensor and high-accuracy temperature sensor, MCP9808.

The AVR-IoT WG Development Board comes preprogrammed and configured for demonstrating connectivity to the Google Cloud IoT Core. This document will demonstrate how to provision the Connected building block of the application to connect to the AWS IoT Core, while keeping the possibility to revert back to the original Google IoT Core demo.

Figure 1-1. AVR-IoT WG Blocks and Features



2. Connect to AWS IoT Core

The connection with the AWS IoT Core uses MQTT over a secure socket layer (TLS). TLS can authenticate both the server and the client.

Server authentication is a standard in most web communications today. The Connected Block of the AVR-IoT board, the ATWINC1510, makes implementing this as simple as uploading the correct Certificate Authority (CA) certificates into its memory. Client authentication, on the other hand, requires more steps and the client certificate needs to be obtained.

Amazon offers three flavors of generating the client certificate:

- One-click certificate creation: this will generate a certificate, a public and private key using AWS IoT's certificate authority;
- Create with Certificate Signing Request (CSR): upload a CSR and AWS IoT’s CA will generate a certificate;
- Use a private CA: register the CA within AWS IoT Core and use it to generate certificates.

Using a private CA has the advantage that device registration can be delayed until the first connection attempt to the cloud. This is called Just in Time Registration (JITR). The registration is automated and triggered when a connection request uses a client certificate signed by the previously registered CA and the certificate is not already attached to a device.

This application uses a private CA. The ATECC608A secure element holds the key pair by which the CA will generate the certificate, while the connectivity element, the ATWINC1510, holds the certificate itself, the CA certificate, and the CA public key.

3. Create and Register a Private CA

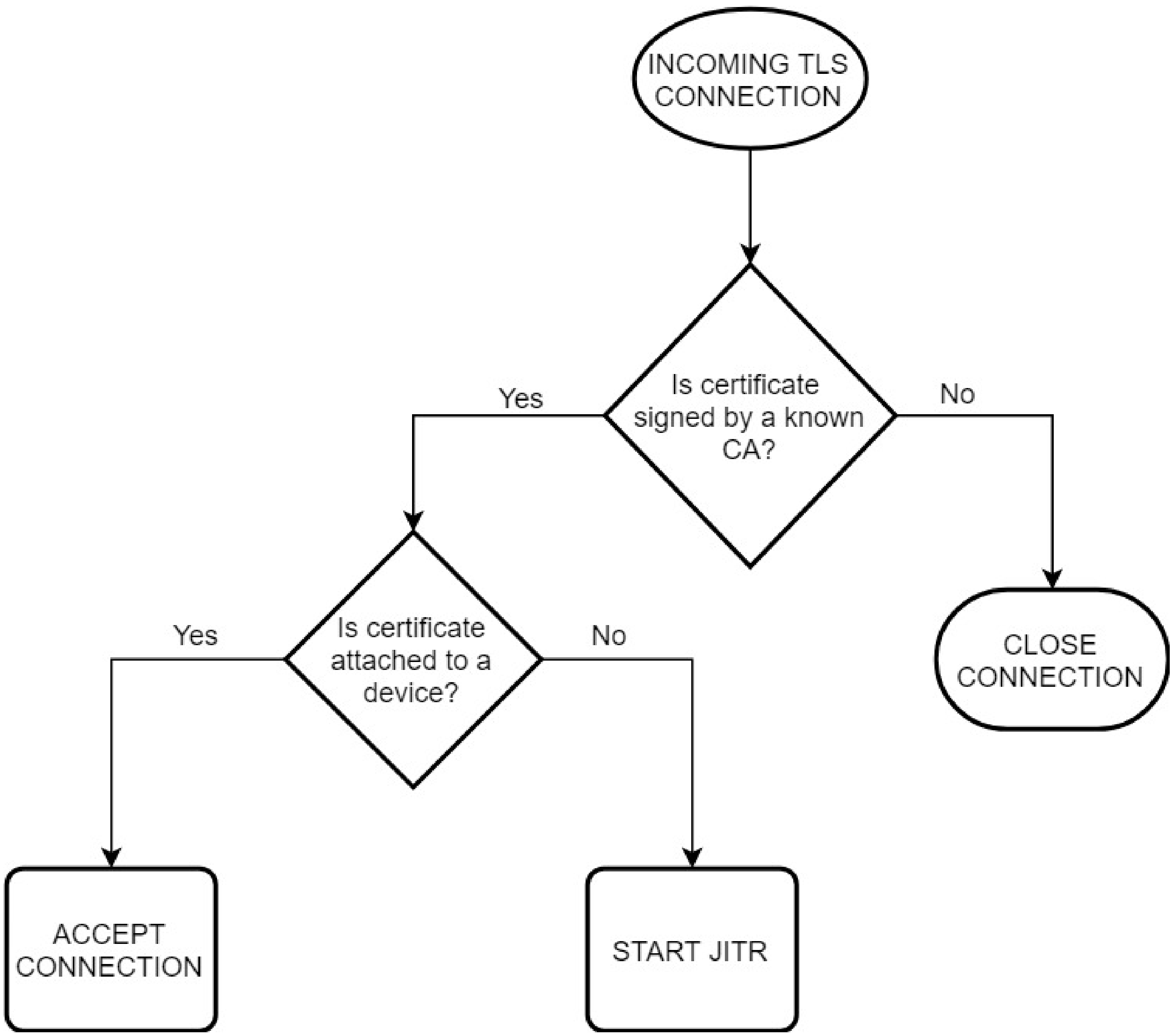
It is a common practice to create a chain of trust that consists of a root CA and one or more signer CAs.

The root CA issues certificates for signer CAs. The signer CA is then able to authorize devices in turn by issuing their device certificate. Therefore, the signer is the CA that needs to be registered into the AWS. Creating root/signer and registering the signer with the AWS are handled by the Python™ scripts provided by Microchip.

When a device first connects using a certificate issued by a registered CA, the JITR is triggered. This registers the device as an AWS IoT Core Thing. In subsequent connections the device will have already been registered, so it can start sending data and the JITR will not get triggered.

This flow is illustrated in the image below.

Figure 3-1. Connection Flow



Note: Only connections that present a certificate issued by a registered CA may trigger the JITR.

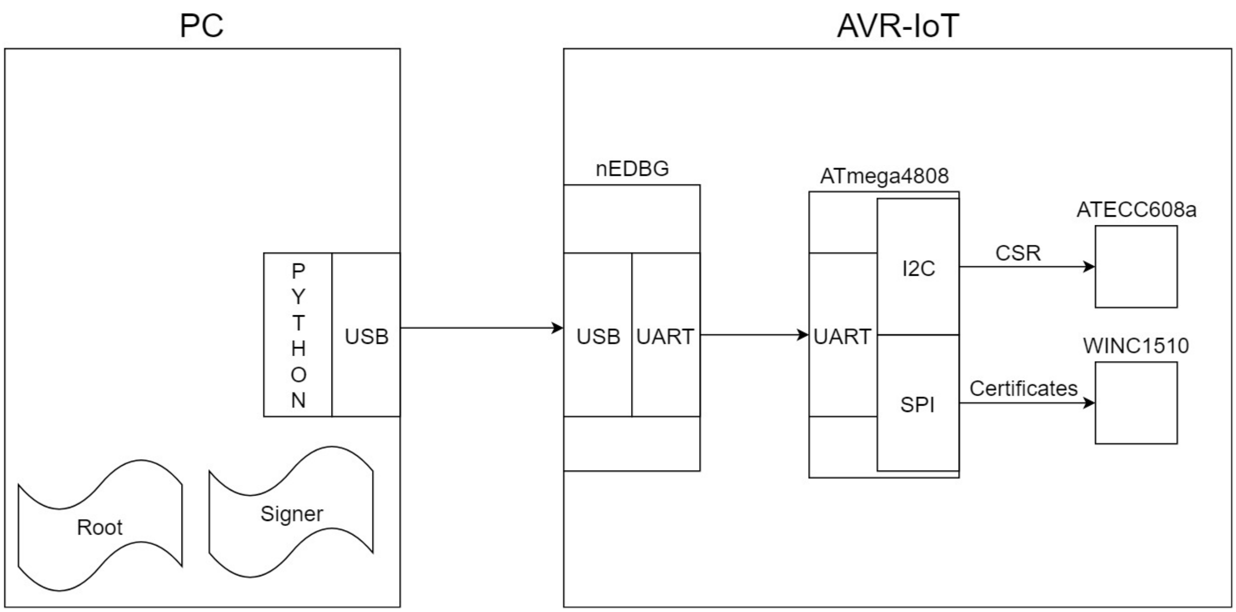
4. Provision the ATWINC1510

This section describes how the ATWINC1510 obtains the device certificate, CA certificate, and CA public key. The device certificate is based on a key pair that has previously been generated and already resides in the ATECC608A. The device certificate is the client certificate used by the TLS layer for the client authentication (the two names are used interchangeably in this document).

The provisioning is a two-step process:

1. The PC running the Python script requests and receives a CSR for the key pair in the ATECC608A.
2. The signer CA generates the certificate and returns it alongside with its own certificate and public key to the board, that will store them in the ATWINC1510.

Figure 4-1. WINC1510 Provisioning Set-Up High-Level Architecture



Note: The signer has to be registered in the AWS Cloud.

The code in the GitHub repository includes the Python script and the provisioning firmware for the AVR-IoT board. When provisioning is complete, the script will print the subject key of the device certificate; this key is important. When the JITR registers the device in the AWS IoT Core, the Thing name is set to be the same as the subject key of its certificate.

5. TLS Connection

The TLS connection provides both authentication and encryption. Authentication consists of two parts::

1. Server authentication; the board authenticates the server.
2. Client authentication; the server authenticates the board.

Server authentication happens transparently to the user, because the ATWINC1510 on the AR-IoT board comes preloaded with the required CA certificate.

Note: That CA certificate is not discussed in this application note.

During client authentication the client private key must be used, but since this is stored inside the ATECC608A chip and cannot be extracted, all calculations must be done inside the ATECC608A. Normally, these calculations would have been done by the ‘connectivity’ element, the ATWINC1510. Since this is not possible, because the private key cannot be read out, the ATWINC1510 library offers an API to delegate the TLS calculations to the main application. The main application (running on the ATmega4808, the ‘Smart’ element) will in turn call the ATECC608A library API to perform the calculations.

Before the TLS connection is complete, a shared secret key must be negotiated between the server and the client. This key is used to encrypt further communication.

6. MQTT Connection

After successfully connected on the TLS level, the board starts establishing the MQTT connection. Since the TLS handled authentication and security, MQTT does not have to provide a user name or password.

On the other hand, the security policy that the JITR attaches to all devices enforces that the MQTT Client ID must match the Thing name in the AWS IoT Core. The policy also restricts a device to only publish to topics that follow the format `<THING NAME>/#`. If necessary, this policy can be changed, whether from the JITR code itself (and it will apply to devices that will register subsequently), or after the device has already been registered.

Note:

1. The `#` in the topic format is a wild card that can represent any number of subtopics.
2. The Thing name is the subject key of the device certificate.

7. Sending Data to AWS Cloud

Once the MQTT connection is established, the AVR-IoT board will read and publish the light intensity and temperature values to the `<THING NAME>/sensors` MQTT Topic.

Amazon Lambda is a service that makes it possible to deploy single functions, called Lambdas. Lambdas execute in response to events. For example, the JITR is a Lambda triggered in response to the connection of a new device to the IoT Core. Similarly, another Lambda is triggered by the messages the devices send via MQTT. That Lambda receives the light intensity and temperature data from devices and forwards them to ThingsBoard. The data are not saved on the AWS server, nor on the ThingsBoard server.

Note: This Lambda will be, in this example, bound to a single topic. It will only receive messages from that topic, therefore from a single device. Similar Lambdas can be created bound to a set of topics, to receive messages from more devices.

ThingsBoard is a platform that can display sensor data as charts, gauges, and other widgets. It also has a complex user and devices management system, but these will not be extensively used in this application note. The live demo https://demo.thingsboard.io is used.

Alternatively, ThingsBoard is open source and can also be self-hosted on AWS using a web server instance on **AWS EC2**, with a public IP/URL.

8. Instructions

1. Follow the Zero Touch Secure Provisioning Kit for AWS IoT guide until section `VII. Provision the Device` to configure the AWS account. The code used by the Zero Touch kit is in this repository.

   Note:
   1. Some naming in the AWS console might have changed, but the new names are intuitively similar to the old ones. If there is any issue running `aws_register_signer.py` (or other scripts), see this fix. A similar fix can be applied to other scripts
   2. This first step is the most difficult to replicate!
2. Program the AVR-IoT WG with the provisioning firmware in the folder `ecc-provision` of the code repository.
3. Copy the `scripts/provision/manual_kit_provision.py` script to the folder where the root CA and signer CA were created, in step 1.
4. Using a terminal, install `pyserial` using pip: `pip install pyserial` or `python -m pip instal pyserial`.
5. While AVR-IoT is still connected to the PC execute `manual_kit_provision.py` in its new location. COM port (on Windows™) or the USB file (on Linux/macOS) must be specified. Here is an example command: `python manual_kit_provision.py COM10`. The output should look similar to the image below and the execution time should not take longer than five seconds.

   Note: This script also shows the name the device (also called the Thing) will have in the AWS. Save this name for future use.

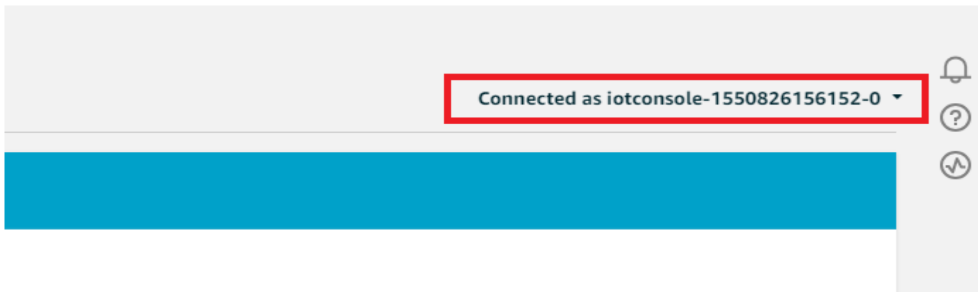
   Figure 8-1. Provisioning Script Output

```
Loading root CA certificate
    Loading from root-ca.crt

Loading signer CA key
    Loading from signer-ca.key

Loading signer CA certificate
    Loading from signer-ca.crt

Requesting device CSR
Received device CSR
    Saving to device.csr

Generating device certificate from CSR
    Saving to device.crt

Provisioning device with AWS IoT credentials

Save Singer CA Pub Key
    Saved successfully

Save Signer Certificate Message
    Saved successfully

Save Device Certificate
    Saved successfully

Done provisioning thing 8b873dd89e85493baea39c149af20d7b615028af
```

6. The default SSID and password for Wi-Fi network are ‘MCHP.IOT’ and ‘microchip’. To change them, edit the `demo/avr.iot-aws-demo/cloud/cloud.h` file.

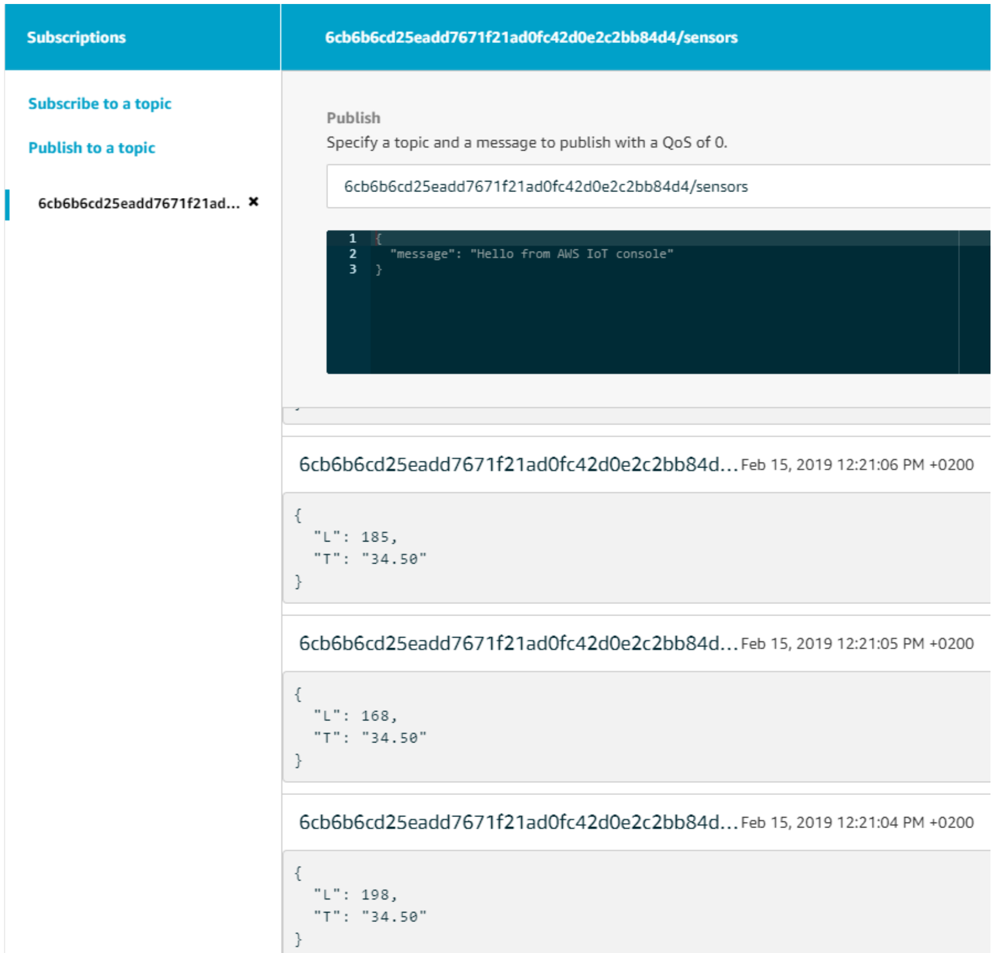
7. In `demo/avr.iot-aws-demo/cloud/cloud.h` there is a `#define AWS_HOST_ENDPOINT`, which needs to be updated. Go to the **AWS IoT Core Console > Test**, and in the upper right corner, there is a dropdown titled **Connected to iotconsole-xxx-x**; click it and select **View endpoint** and a sidebar will slide in from the right. Copy the endpoint. Its first part contains a '-ats' suffix that must be removed. Therefore, for example `a3mnn069kqq6d6-ats.iot.us-west-2.amazonaws.com` becomes `a3mnn069kqq6d6.iot.us-west-2.amazonaws.com`. The endpoint without the '-ats' suffix is the value of the `#define AWS_HOST_ENDPOINT` in `demo/avr.iot-aws-demo/cloud/cloud.h`.

Figure 8-2. AWS IoT Core Console > Test, the Dropdown Titled 'Connected to iotconsole-xxx-x'



8. In `demo/avr.iot-aws-demo/cloud/cloud.h` there is a `#define AWS_THING_ID`, which needs to be updated. Its value must be the Thing name the terminal displayed in step 5.
9. Program the AVR-IoT WG with the firmware in the /`demo` project.
10. The first time this code runs, the red LED will flash once, as it starts and does the JITR but the connection fails. The code auto-recovers and reconnects automatically. This only happens during the first connection.
11. Visit `AWS Console > IoT Core > Manage > Things` and there will be a new Thing in the Things list, and its name is the one shown at step 5. Copy the Thing name.
12. View messages in `AWS Console > IoT Core > Test`. In the 'Subscription topic' field, paste the Thing name and append '/sensors'. This will subscribe to a topic that looks like `aa0b820832ee20f38d88c072a7c192cfe426683c/sensors`.
13. Click '**Subscribe to topic**' and it will start showing messages received in real time.

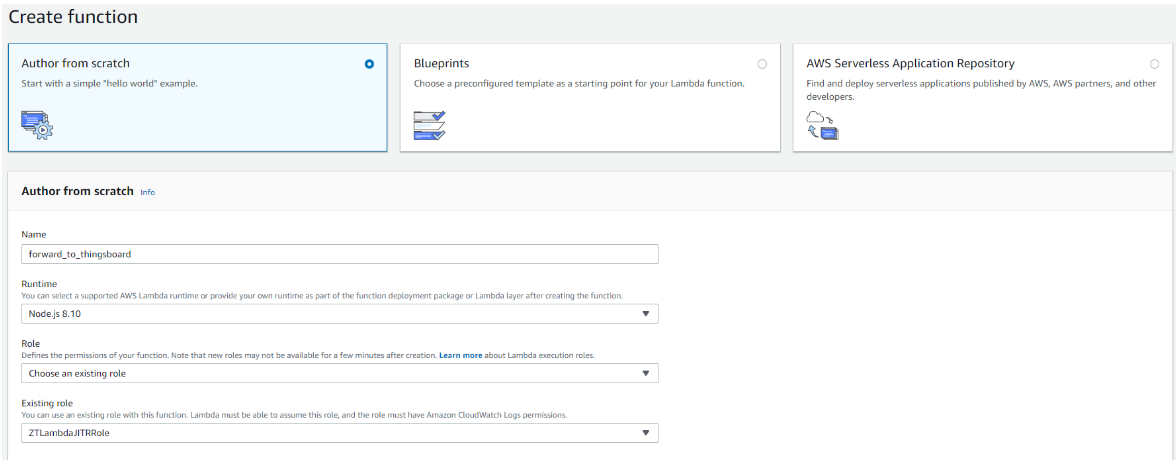
Figure 8-3. Real-Time Messages Displayed in AWS Test Console as JSON



9. Create the Lambda to Receive Device Messages

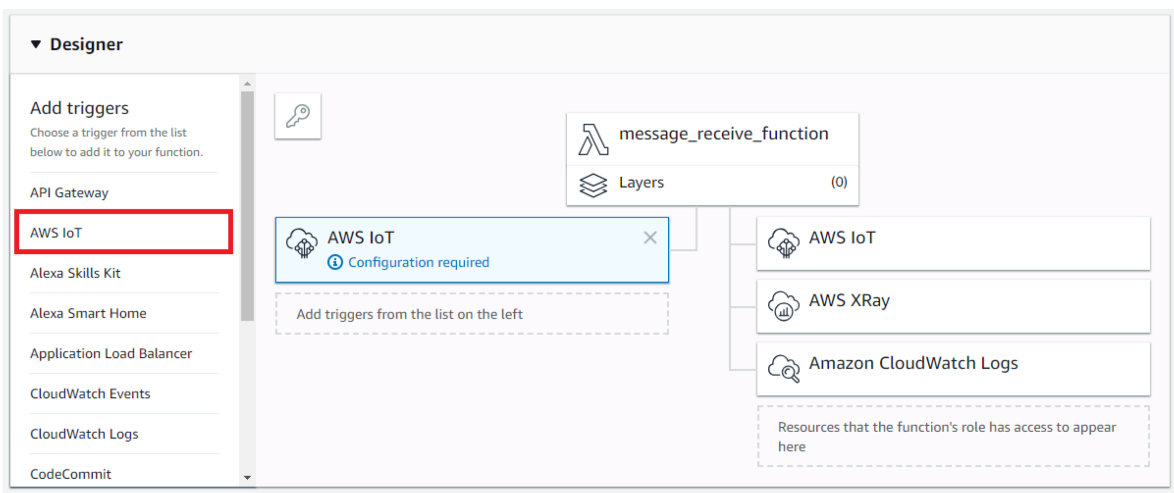
1. In the AWS Lambda console, click ‘**Create function**’, give it a name and use the role ZTLambdaJITRRole.

Figure 9-1. AWS Lambda, Create Function



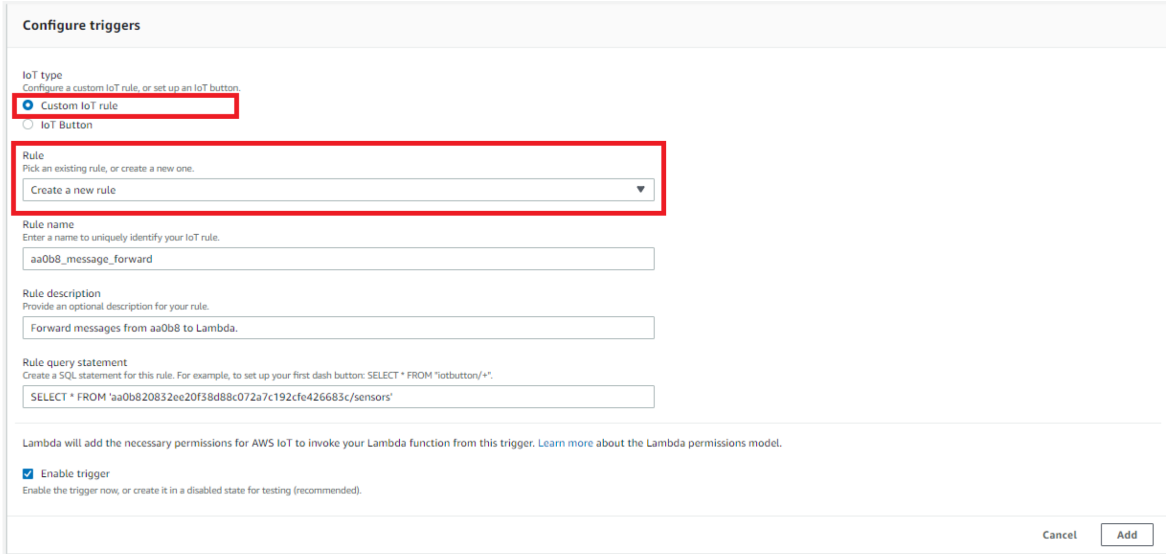
2. In the function screen, on the left side, under ‘Designer’, there is a section called ‘Add triggers’; in this section, click ‘**AWS IoT**’.

Figure 9-2. AWS Lambda, Add AWS IoT Trigger



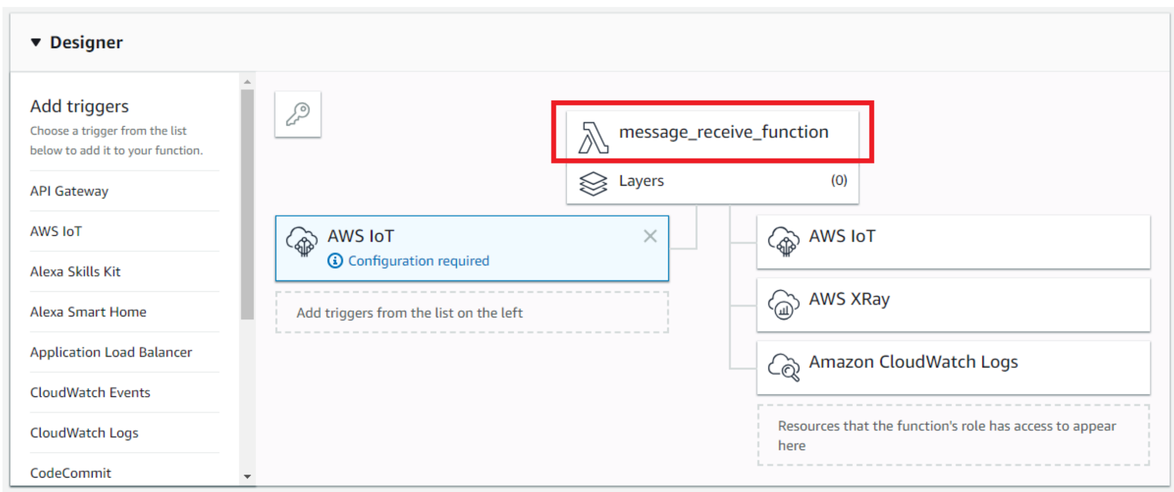
3. A section called ‘Configure triggers’ will open below; select ‘**Custom IoT rule**’.
4. In the rule dropdown, select ‘**Create a new rule**’.
5. Give the rule a name and a description; for ‘Rule query statement’ copy `SELECT * FROM <YOUR_THING_NAME>/sensors` and replace `<YOUR_THING_NAME>` with actual Thing name.

Figure 9-3. Configure a Custom Rule to Trigger Lambda Function



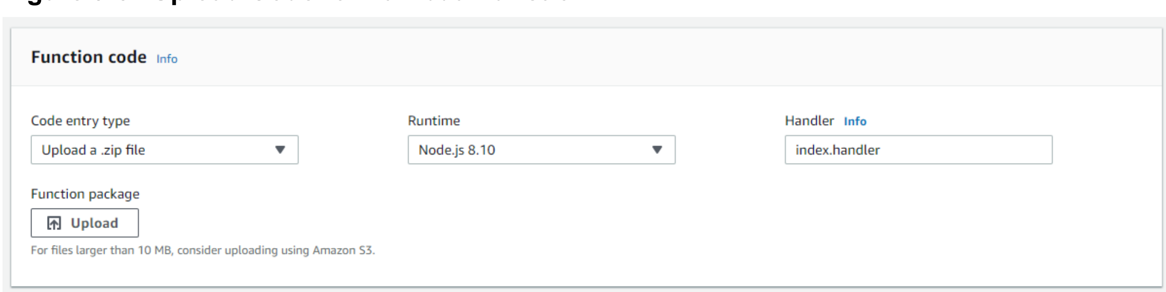
6. In the 'Designer' section, the main area shows a tree-like image; click the uppermost block in the tree, the one labeled with the function name.

Figure 9-4. Configure a Custom Rule to Trigger Lambda



7. A new section called 'Function code' will open below the 'Designer' section. In the 'Code entry type' dropdown, select '**Upload a .zip file**'.

Figure 9-5. Upload Code for Lambda Function



8. Upload the `.zip` in `scripts/lambda/forward_to_thingsboard`.
9. Click '**Save**' in the upper left corner of the page.

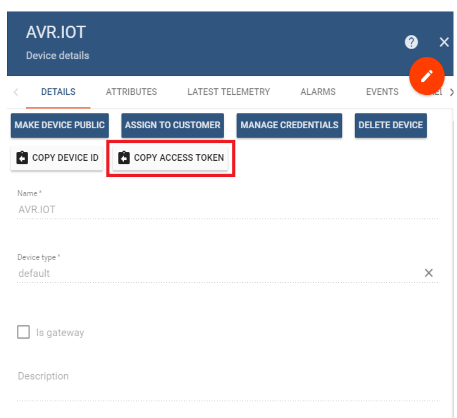
10. Visualize Chart on ThingsBoard

1. Go to https://demo.thingsboard.io/signup and create an account; an activation link will be sent via email.
2. Login into the ThingsBoard account and go to Devices page; click the Floating Action Button (FAB) in the lower right corner to create a new device.

Figure 10-1. Add Device

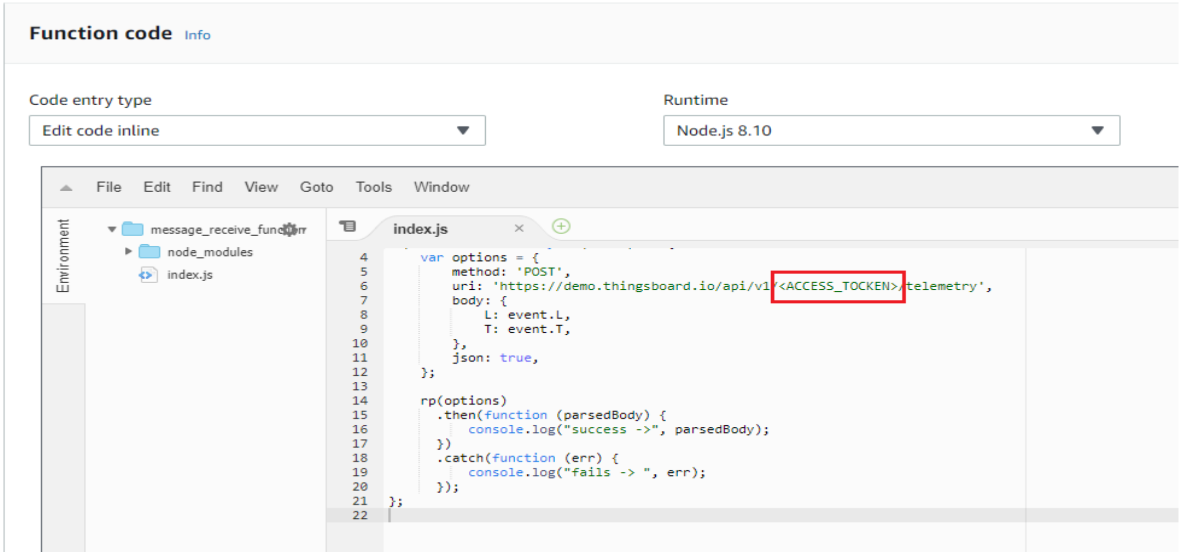
3. Click the card with the name of the newly created device and a sidebar will slide in from the right of the screen, there click the '**Copy access token**' button.

Figure 10-2. Copy Access Token Button



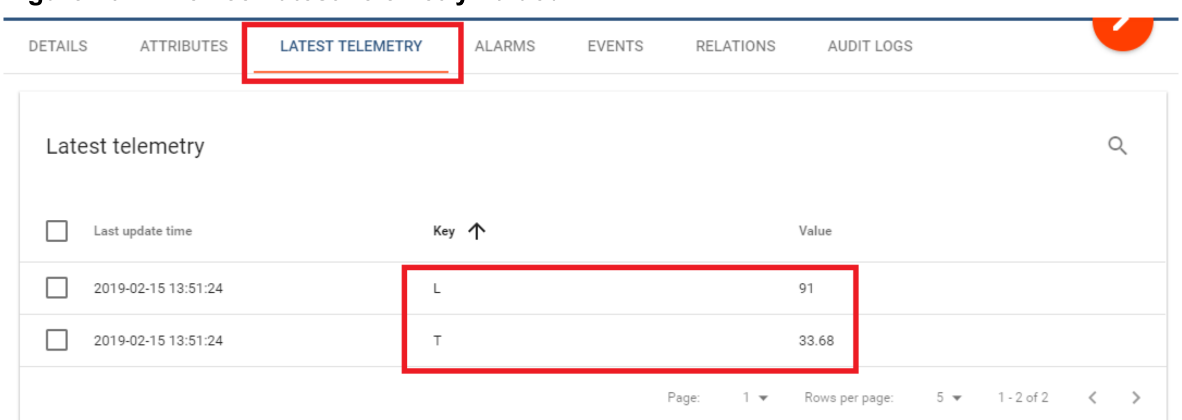
4. Go into AWS Lambda Console page of the function previously created.
5. In the 'Function code' section, make sure 'Edit code inline' is selected in the 'Code entry type' dropdown and in the editor, there will be the files previously uploaded as `.zip`.
6. In `index.js` on line 6 replace `<ACCESS_TOKEN>` with the access token just copied from ThingsBoard.

Figure 10-3. Use Device Access Token in the Lambda Code



7. Click ‘**Save**’ in the upper left corner of the page.
8. Connect the board and make sure it is sending data (the yellow LED toggles once a second).
9. Go back to ThingsBoard, on the Devices page, and click the card with the device name on it. A sidebar will slide in from the right containing several tabs; click the tab titled ‘**Latest Telemetry**’ and a table with two rows will appear. The Key column values are ‘L’ for light and ‘T’ for temperature; the Value column will update about once every second with data received from the board.

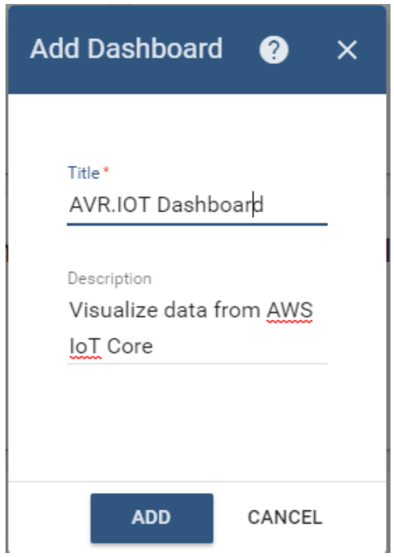
Figure 10-4. Device Latest Telemetry Valued



Now that the data made it all the way to the ThingsBoard server, it is time to configure a dashboard to plot them.

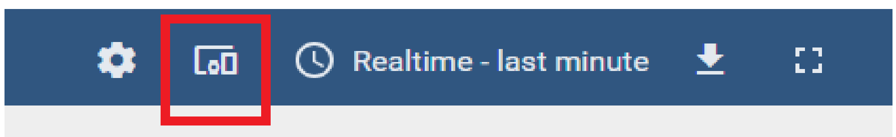
10. In ThingsBoard, go to the Dashboards page.
11. Click the FAB in the lower right corner to create a new Dashboard.

Figure 10-5. Add Dashboard



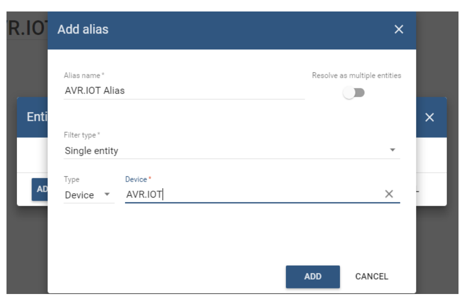
12. Click the card with the name of the newly created dashboard and move to the Dashboard page.
13. Click the FAB in the lower right corner to edit the Dashboard.
14. In the upper right corner, below the name, there are some icons; click the second one to the left, called ‘**Entity aliases**’, and a modal window will open.

Figure 10-6. Edit Entity Aliases Icon in Dashboard



15. Click ‘**Add alias**’ and give it a name; in the ‘Filter type’ drop-down, select ‘Single entity’; for ‘Type’ select ‘Device’, and, for ‘**Device**’ select the device created, then click ‘**Add**’.

Figure 10-7. Add Alias

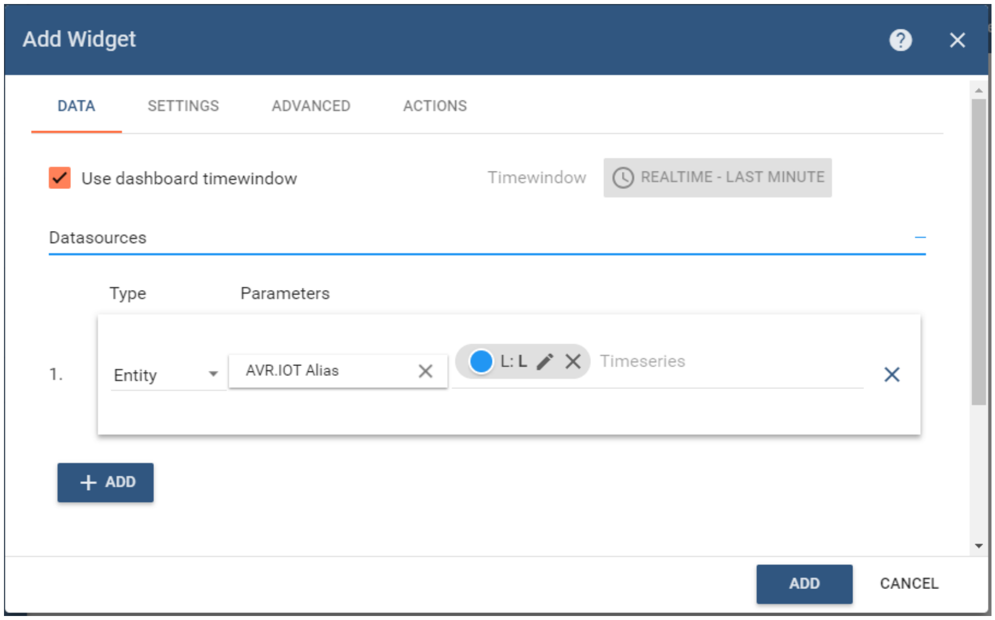


16. Click ‘**Save**’.
17. In the center of the screen, there is a button called ‘**Add new widget**’; click it and a sidebar will slide in from the right.
18. In the ‘Current bundle’ drop-down select ‘**Charts**’ and scroll down to the ‘**TimeSeries - Flot**’ type of chart. Click it and a modal window with configurations options will show.

Figure 10-8. Add TimeSeries – Flot Type of Chart

19. Under 'Datasets' click '**Add**'.
20. The type will be 'Entity'; for 'Entity alias' select the alias previously created.
21. In the 'TimeSeries' drop-down, the device data will load. To show a light chart, select '**L**' and click '**Add**'.

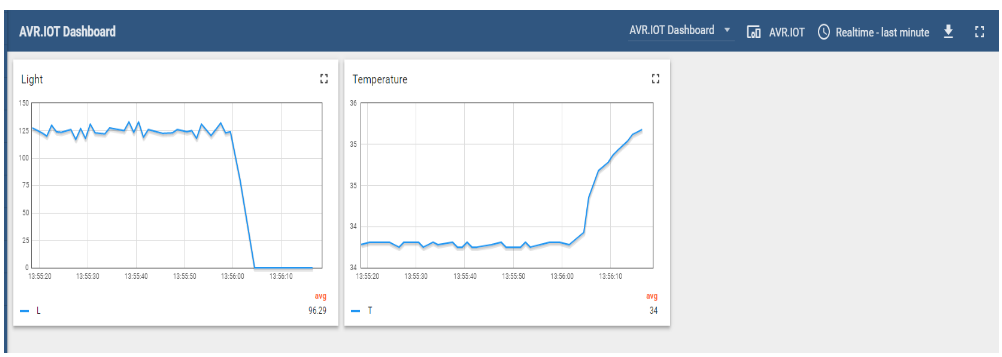
Figure 10-9. Configure the Chart DataSource



22. Similarly, add a chart for temperature.
23. In the lower right corner, there are three FABs; click the one with a tick to complete dashboard customization.

The configuration is now ready. The user can now see the live data displayed on charts, just like in the image below.

Figure 10-10. AVR-IoT Collected Sensor Data Displayed in ThingsBoard



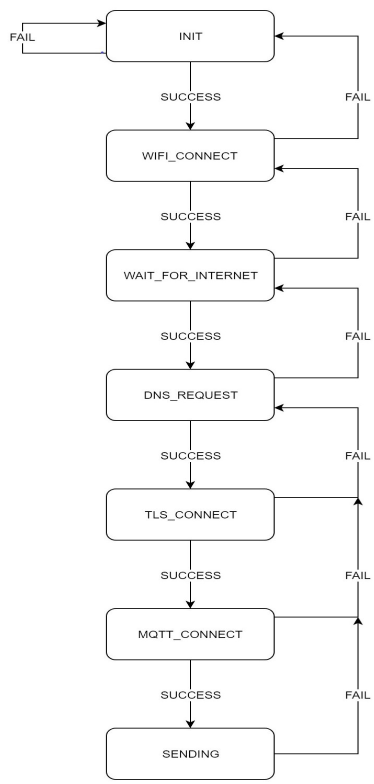
11. Firmware

This section refers to the firmware in the `/demo` project. These are the steps that the firmware follows, before sending data in the cloud:

- Connect to Wi-Fi
- Obtain the AWS server IP
- Connect to the AWS on the TLS level
- Connect to the AWS on the MQTT level

These steps are modeled in the software as states of a State Machine. The following picture illustrates the state diagram of the software.

Figure 11-1. AVR-IoT States



The four LEDs on the board notify the user about the status of the application. The meaning of the LEDs is described in the table below.

Table 11-1. LEDs Meaning

Blue LED on	Connected to Wi-Fi
Blue LED blinking	Connected to Wi-Fi but with no Internet connection
Green LED on	Connected to the AWS
Yellow LED blinking	Sending data
Red LED on	Error occurred

12. Conclusion

In IoT, security is cornerstone and the demo described in this application note achieves high levels of security in order to successfully connect to AWS. The modular design of the AVR-IoT board allows for a neat separation of responsivities between the ATECC608A and the ATWINC1510, while the ATmega4808 microcontroller ties them together and implements the application logic.

The AWS cloud has multiple services that can act as building blocks for a high-level application on top of the data collected by IoT devices. This demo used AWS IoT core for managing devices and AWS Lambda to forward data to ThingsBoard, for visualization. A final solution may also use other services for storage, processing and others.

ThingsBoard provides easy to use visualization tools and moreover, if the complex user management system is used, it can turn into a feature full front end for an IoT solution.

13. Appendix: Install ThingsBoard on AWS EC2

Since ThingsBoard is an open source platform, the user can easily install it on any Virtual Private Server (VPS), such as the AWS EC2. This way, the user can move away from the demo dashboard of ThingsBoard and built a complete system based on the AWS Cloud. Refer to the 'Getting Started with Amazon EC2' guide for more information.

After VPS configured, continue with this ThingsBoard Installation guide.

Note: The most convenient option to use is ThingsBoard Professional Edition (PE), that is already in the AWS Marketplace, because it creates the VPS and installs ThingsBoard. See the guide Installing ThingsBoard PE from AWS Marketplace for more information.

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-4598-2

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820