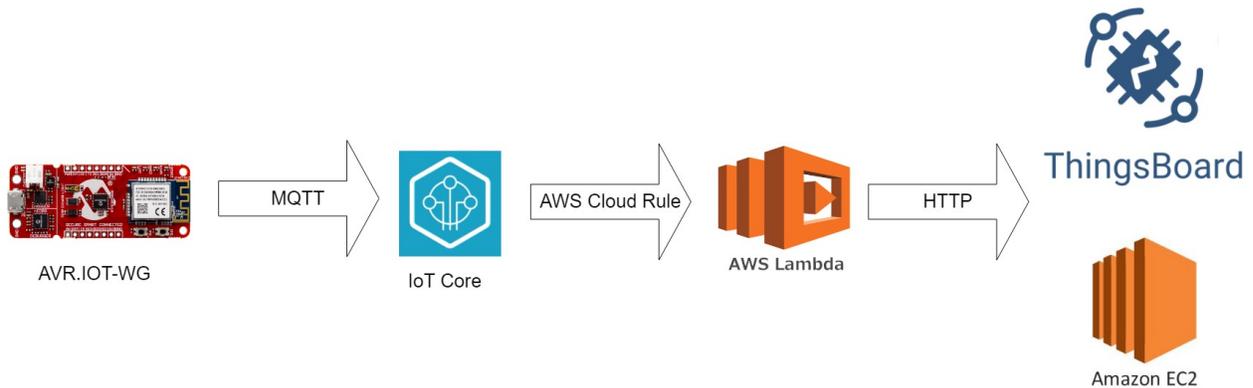

调整 AVR-IoT WG 的用途以连接到 AWS

简介

本应用笔记介绍如何将 **AVR-IoT WG** 开发板连接到 **AWS IoT Core**，如何发送光和温度数据并将其绘制为折线图。本文档介绍构建与亚马逊 Web 服务（Amazon Web Services, AWS）云进行深度集成的解决方案的详细步骤。它使用 **AWS IoT Core** 来管理设备，并与其他 AWS 服务（例如 **AWS Lambda**）集成，以便将数据发送到 **ThingsBoard**，利用 **ThingsBoard** 实现数据可视化。

此应用的高级架构如下图所示。

图 1. 应用的高级架构



GitHub 上提供代码。



目录

简介.....	1
1. AWS-IoT WG 是什么.....	3
2. 连接到 AWS IoT Core.....	4
3. 创建和注册私有 CA.....	5
4. 置备 ATWINC1510.....	6
5. TLS 连接.....	7
6. MQTT 连接.....	8
7. 将数据发送到 AWS Cloud.....	9
8. 指令.....	10
9. 创建 Lambda 以接收设备消息.....	13
10. 在 ThingsBoard 上提供可视化图表.....	15
11. 固件.....	20
12. 结论.....	21
13. 附录：在 AWS EC2 上安装 ThingsBoard.....	22
Microchip 网站.....	23
产品变更通知服务.....	23
客户支持.....	23
Microchip 器件代码保护功能.....	23
法律声明.....	23
商标.....	24
质量管理体系.....	24
全球销售及服务网点.....	25

2. 连接到 AWS IoT Core

与 AWS IoT Core 的连接在安全套接字层（secure socket layer, TLS）上使用 MQTT 协议。TLS 可以对服务器和客户端进行身份验证。

目前大部分 Web 通信中都会标配服务器身份验证要求。AVR-IoT 开发板的连接模块 ATWINC1510 可以简化这一过程，就像将正确的证书颁发机构（Certificate Authority, CA）证书上传到其存储器中一样简单。另一方面，客户端身份验证需要更多步骤，并且需要获取客户端证书。

亚马逊提供三种生成客户端证书的方式：

- 一键式证书创建：将使用 AWS IoT 的证书颁发机构生成证书、公钥和私钥；
- 使用证书签名请求（Certificate Signing Request, CSR）创建：上传 CSR，然后 AWS IoT 的 CA 将生成证书；
- 使用私有 CA：在 AWS IoT Core 中注册 CA 并用其生成证书。

使用私有 CA 的优势在于，可以在首次尝试连接到云之前延迟进行设备注册。这称为即时注册（Just in Time Registration, JITR）。当连接请求使用由先前注册的 CA 签名的客户端证书且证书尚未连接到设备时，这种注册将自动执行并触发。

本应用使用私有 CA。ATECC608A 安全元件保存密钥对，CA 将通过该密钥对生成证书，而连接元件 ATWINC1510 则保存证书本身、CA 证书和 CA 公钥。

3. 创建和注册私有 CA

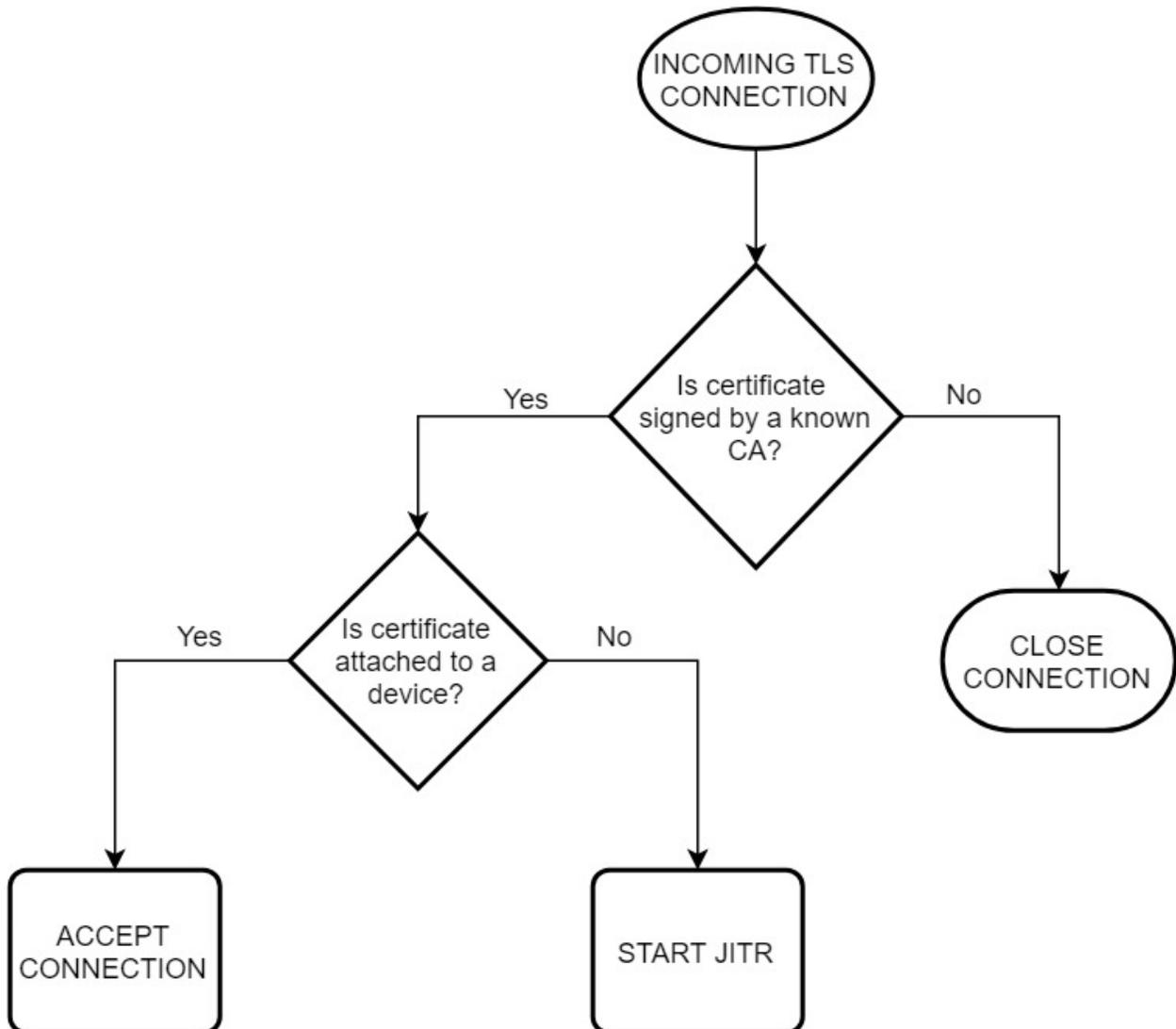
创建由一个根 CA 和一个或多个签名者 CA 组成的信任链是一种常见的做法。

根 CA 为签名者 CA 签发证书。然后，签名者 CA 可以通过签发设备证书来依次为设备授权。因此，签名者是需要 AWS 中注册的 CA。创建根/签名者和在 AWS 上注册签名者的过程由 Microchip 提供的 Python™ 脚本处理。

当设备首次使用已注册 CA 签发的证书进行连接时，会触发 JITR。这会将设备注册为 AWS IoT Core 事物。在随后的连接中，设备将是已经注册的状态，因此它可以开始发送数据，JITR 将不会被触发。

这一流程如下图所示。

图 3-1. 连接流程



注：只有提供已注册 CA 签发的证书的连接才可触发 JITR。

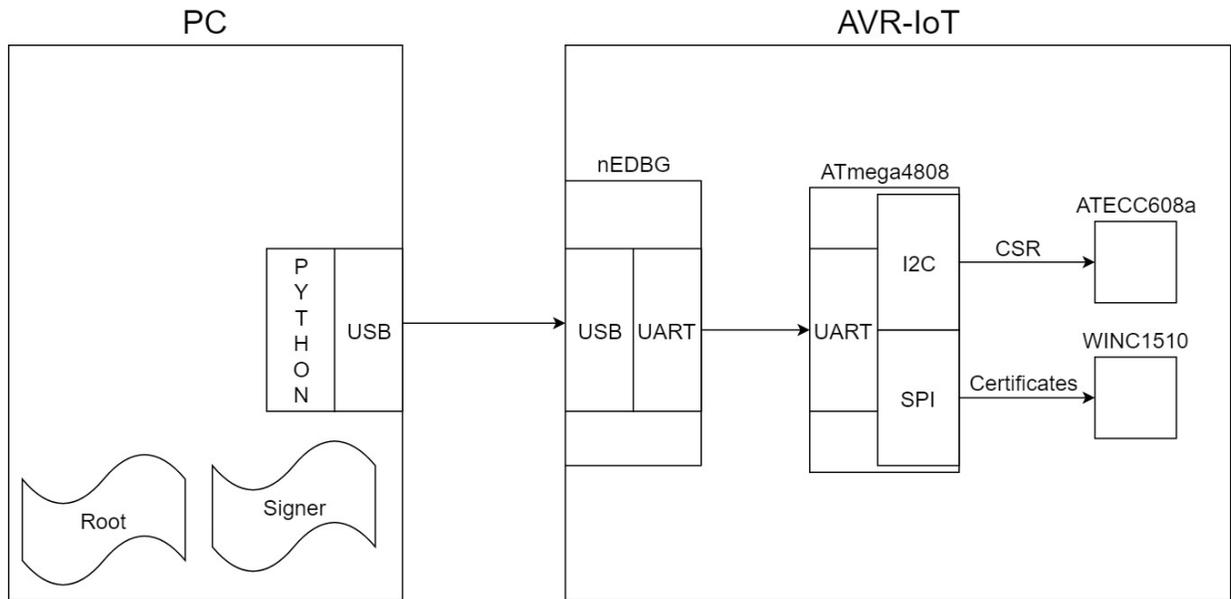
4. 置备 ATWINC1510

本节介绍 ATWINC1510 如何获取设备证书、CA 证书和 CA 公钥。设备证书基于先前生成的密钥对并且已经驻留在 ATECC608A 中。设备证书是 TLS 层用于客户端身份验证的客户端证书（这两个名称在本文档中可以互换使用）。

置备包含两个步骤：

1. 运行 Python 脚本的 PC 请求并接收 ATECC608A 中密钥对的 CSR。
2. 签名者 CA 生成证书，然后将证书连同其自己的证书和公钥一起返回到开发板，开发板将这些证书和公钥存储在 ATWINC1510 中。

图 4-1. WINC1510 置备设置高级架构



注： 签名者必须在 AWS Cloud 中注册。

GitHub 代码库中的代码包括 Python 脚本和 AVR-IoT 开发板的置备固件。置备完成后，脚本将打印设备证书的主题密钥。此密钥很重要。当 JITR 在 AWS IoT Core 中注册设备时，事物名称将设置为与其证书的主题密钥相同。

5. TLS 连接

TLS 连接提供身份验证和加密。身份验证包括两个部分：

1. 服务器身份验证；开发板对服务器进行身份验证。
2. 客户端身份验证；服务器对开发板进行身份验证。

服务器身份验证对用户来说是透明的，因为 AR-IoT 开发板上的 ATWINC1510 已预装所需的 CA 证书。

注： 本应用笔记不讨论 CA 证书。

在客户端身份验证期间，必须使用客户端私钥，但是由于它存储在 ATECC608A 芯片内部且无法提取，因此所有计算都必须在 ATECC608A 内部进行。通常，这些计算将由“连接性”元件 ATWINC1510 完成。由于无法读出私钥，因此不可能进行计算，所以 ATWINC1510 库提供了一个 API，用于将 TLS 计算任务委托给主应用完成。主应用（在 ATmega4808 上运行，“智能”组成部分）将依次调用 ATECC608A 库 API 进行计算。

在完成 TLS 连接前，必须在服务器和客户端之间协商共享密钥。此密钥用于对后续通信进行加密。

6. MQTT 连接

在 TLS 层成功连接后，此开发板将开始建立 MQTT 连接。由于 TLS 处理了身份验证和安全性问题，因此 MQTT 不必提供用户名或密码。

另一方面，JITR 附加到所有设备的安全策略会强制要求 MQTT 客户端 ID 必须与 AWS IoT Core 中的事物名称匹配。该策略还会要求设备仅发布遵循<THING_NAME>/#格式的主题。如有必要，可以更改此策略，可从 JITR 代码本身进行更改（它将应用于后续将要注册的设备），也可在设备已注册之后更改。

注：

1. 主题格式中的#是通配符，可以表示任意数量的子主题。
2. 事物名称是设备证书的主题密钥。

7. 将数据发送到 AWS Cloud

建立 MQTT 连接后，AVR-IoT 开发板将立即读取光强度和温度值，然后将其发布到<THING_NAME>/sensors MQTT 主题。

Amazon Lambda 是一项可以部署单一功能 Lambda 的服务。Lambda 根据事件执行。例如，JITR 是为了响应新设备与 IoT Core 连接而触发的 Lambda。同样，另一个 Lambda 由设备通过 MQTT 发送的消息触发。Lambda 接收设备的光强度和温度数据，然后将其转发到 ThingsBoard。数据既不保存在 AWS 服务器上，也不保存在 ThingsBoard 服务器上。

注： 在本示例中，此 Lambda 将与单个主题绑定。它只会接收来自该主题的消息，因此只会接收来自单个设备的消息。可以创建与一组主题绑定的类似 Lambda，以接收来自更多设备的消息。

ThingsBoard 是一个可以将传感器数据显示为图表、计量器和其他小工具的平台。它还具有复杂的用户和设备管理系统，但是在本应用笔记中不会广泛使用这些系统。使用了实时演示 <https://demo.thingsboard.io>。

另外，ThingsBoard 是开源的，也可以使用带有公共 IP/URL 的 **AWS EC2** 上的 Web 服务器实例在 AWS 上进行自我托管。

8. 指令

1. 按照 [Zero Touch Secure Provisioning Kit for AWS IoT \(AWS IoT 零接触安全置备工具包\)](#) 指南直到 VII. Provision the Device 段来配置 AWS 帐户。零接触工具包使用的代码在此 [代码库](#) 中。

注:

1. AWS 控制台中的某些命名可能已更改，但是新名称直观上与旧名称类似。如果运行 `aws_register_signer.py` (或其他脚本) 有任何问题，参见 [修复程序](#)。类似的修复程序可应用于其他脚本
 2. 第一步是最难复制的!
 2. 在代码库的 `ecc-provision` 文件夹中使用置备固件对 AVR-IoT WG 进行编程。
 3. 在步骤 1 中，将 `scripts/provision/manual_kit_provision.py` 脚本复制到创建根 CA 和签名者 CA 的文件夹。
 4. 使用终端，然后使用 `pip` 安装 `pyserial`: `pip install pyserial` 或 `python -m pip install pyserial`。
 5. 当 AVR-IoT 仍连接到 PC 时，在其新位置执行 `manual_kit_provision.py`。必须指定 COM 端口 (在 Windows® 上) 或 USB 文件 (在 Linux/macOS 上)。下面是一个示例命令: `python manual_kit_provision.py COM10`。输出应类似于下图，执行时间不应超过五秒。
- 注: 该脚本还显示了在 AWS 中设备 (也称为事物) 将具有的名称。保存此名称以供将来使用。

图 8-1. 置备脚本输出

```

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. Wi-Fi 网络的默认 SSID 和密码为 “MCHP.IOT” 和 “microchip”。若要进行更改，请编辑 `demo/avr.iot-aws-demo/cloud/cloud.h` 文件。
7. 在 `demo/avr.iot-aws-demo/cloud/cloud.h` 中，有一个 `#define AWS_HOST_ENDPOINT` 部分，需要进行更新。转到 **AWS IoT Core Console > Test (AWS IoT Core 控制台 > 测试)**，右上角有一个名为 **Connected to iotconsole-xxx-x (已连接到 iotconsole-xxx-x)** 的下拉列表；点击它，然后选择 **View endpoint (查看端点)**，一个侧边栏将从右侧滑入。复制端点。它的第一部分包含 “-ats” 后缀，必须将其删除。因此，例如 `a3mnn069kqq6d6-ats.iot.us-west-2.amazonaws.com` 变为 `a3mnn069kqq6d6.iot.us-west-2.amazonaws.com`。不带 “-ats” 后缀的端点是 `demo/avr.iot-aws-demo/cloud/cloud.h` 中 `#define AWS_HOST_ENDPOINT` 的值。

图 8-2. AWS IoT Core Console > Test, 名为 “Connected to iotconsole-xxx-x” 的下拉列表



8. 在 `demo/avr.iot-aws-demo/cloud/cloud.h` 中，有一个 `#define AWS_THING_ID` 部分，需要进行更新。其值必须是终端在步骤 5 中显示的事物名称。
9. 在 `/demo` 项目中用固件对 AVR-IoT WG 进行编程。
10. 第一次运行此代码时，红色 LED 会闪烁一次，因为它会启动并执行 JITR，但连接失败。该代码会自动恢复并自动重新连接。这种情况仅在第一次连接期间发生。
11. 访问 AWS Console > IoT Core > Manage > Things (AWS 控制台 > IoT Core > 管理 > 事物)， “Things (事物)” 列表中将出现新的事物，其名称为步骤 5 中显示的名称。复制事物名称。
12. 在 AWS Console > IoT Core > Test (AWS 控制台 > IoT Core > 测试) 中查看消息。在 “Subscription topic (订阅主题)” 字段中，粘贴事物名称，然后附加 `/sensors`。这将订阅一个类似于 `aa0b820832ee20f38d88c072a7c192cfe426683c/sensors` 的主题。
13. 点击 “Subscribe to topic (订阅主题)”，它将开始显示实时收到的消息。

图 8-3. 在 AWS Test Console (AWS 测试控制台) 中显示为 JSON 的实时消息

The screenshot displays the AWS Test Console interface for a subscription. On the left, under 'Subscriptions', a subscription is selected with ID '6cb6b6cd25eadd7671f21ad...'. The main area shows the 'Publish' section with a text input field containing the topic '6cb6b6cd25eadd7671f21ad0fc42d0e2c2bb84d4/sensors'. Below this, a code editor shows a JSON message:

```
1 {  
2   "message": "Hello from AWS IoT console"  
3 }
```

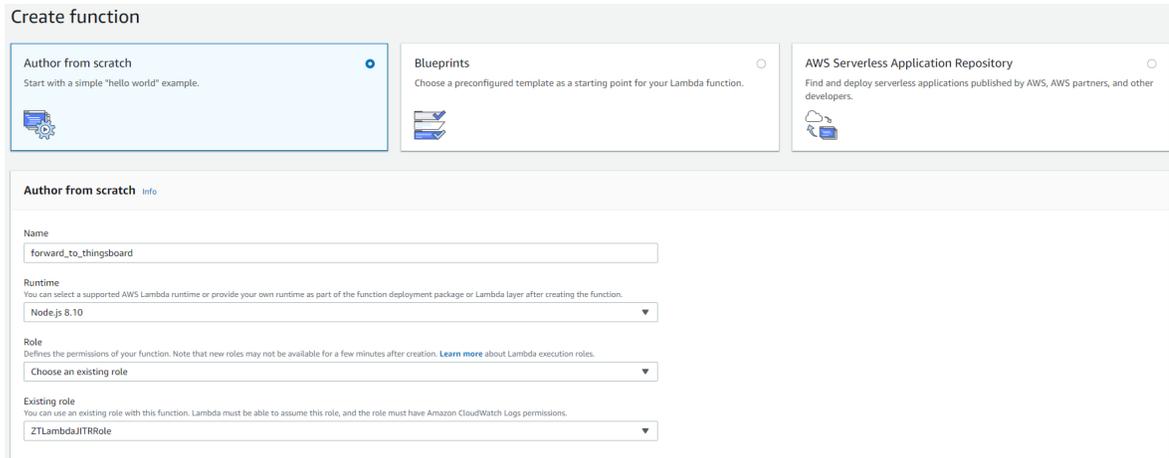
Below the publish section, three real-time messages are shown, each with a timestamp and a JSON payload:

- 6cb6b6cd25eadd7671f21ad0fc42d0e2c2bb84d... Feb 15, 2019 12:21:06 PM +0200
{
 "L": 185,
 "T": "34.50"
}
- 6cb6b6cd25eadd7671f21ad0fc42d0e2c2bb84d... Feb 15, 2019 12:21:05 PM +0200
{
 "L": 168,
 "T": "34.50"
}
- 6cb6b6cd25eadd7671f21ad0fc42d0e2c2bb84d... Feb 15, 2019 12:21:04 PM +0200
{
 "L": 198,
 "T": "34.50"
}

9. 创建 Lambda 以接收设备消息

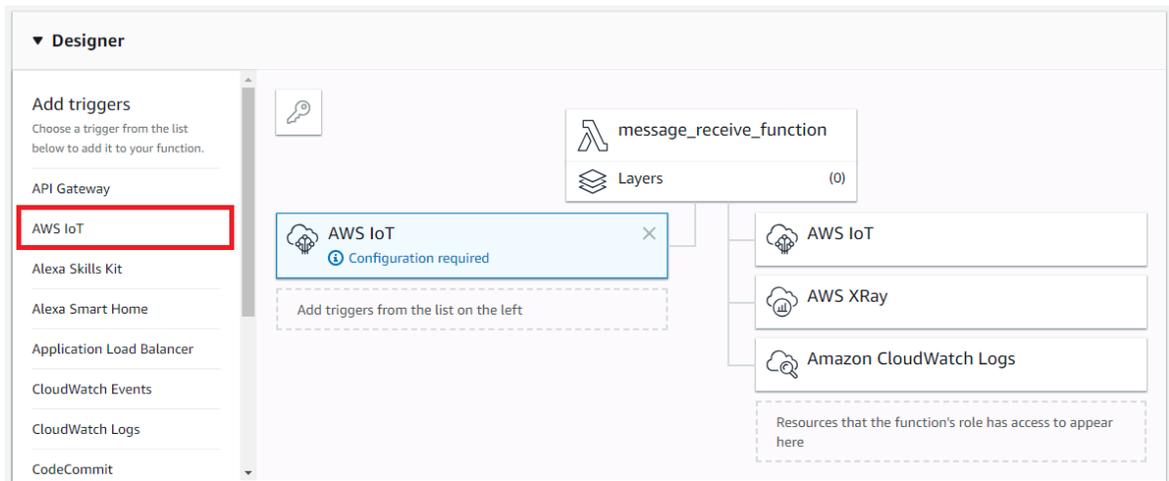
1. 在 AWS Lambda 控制台中，点击“**Create function（创建功能）**”，为其命名并使用 ZTLambdaJITRole 角色。

图 9-1. AWS Lambda，创建功能



2. 在功能屏幕的左侧，“Designer（设计工具）”下方，有一个名为“Add triggers（添加触发器）”的部分；在此部分中，点击“**AWS IoT**”。

图 9-2. AWS Lambda，添加 AWS IoT 触发器



3. 名为“**Configure triggers（配置触发器）**”的部分将在下方打开；选择“**Custom IoT rule（自定义物联网规则）**”。
4. 在规则下拉列表中，选择“**Create a new rule（创建新规则）**”。
5. 为规则命名并进行说明；对于“**Rule query statement（规则查询语句）**”，复制 `SELECT * FROM <YOUR_THING_NAME>/sensors`，然后将 `<YOUR_THING_NAME>` 替换为实际的事物名称。

图 9-3. 配置自定义规则以触发 Lambda 功能

Configure triggers

IoT type
Configure a custom IoT rule, or set up an IoT button.

Custom IoT rule
 IoT Button

Rule
Pick an existing rule, or create a new one.
Create a new rule

Rule name
Enter a name to uniquely identify your IoT rule.
aa0b8_message_forward

Rule description
Provide an optional description for your rule.
Forward messages from aa0b8 to Lambda.

Rule query statement
Create a SQL statement for this rule. For example, to set up your first dash button: `SELECT * FROM "iotbutton/*"`.
`SELECT * FROM "aa0b820832ee20f38d88c072a7c192cfe426683c/sensors"`

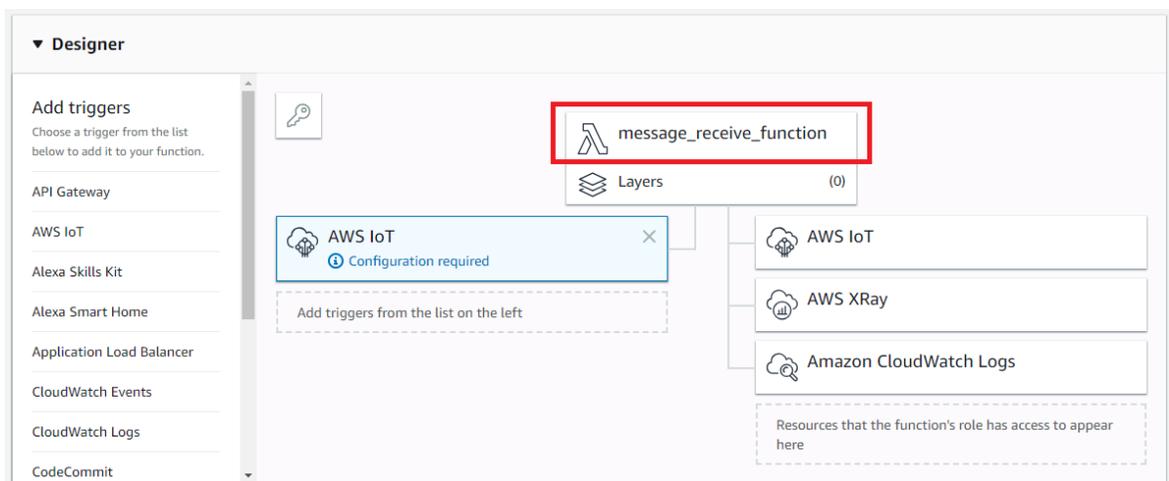
Lambda will add the necessary permissions for AWS IoT to invoke your Lambda function from this trigger. [Learn more about the Lambda permissions model.](#)

Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).

Cancel Add

- 在“Designer”部分，主要区域显示了一个树状图像：点击树中最上面的块，该块标有功能名称。

图 9-4. 配置自定义规则以触发 Lambda



- 一个名为“Function code（功能代码）”的新部分将在“Designer”部分下方打开。在“Code entry type（代码输入类型）”下拉列表中，选择“Upload a .zip file（上传.zip 文件）”。

图 9-5. 上传 Lambda 功能的代码

Function code Info

Code entry type
Upload a .zip file

Runtime
Node.js 8.10

Handler Info
index.handler

Function package
Upload

For files larger than 10 MB, consider uploading using Amazon S3.

- 在 `scripts/lambda/forward_to_thingsboard` 中上传 .zip。
- 点击页面左上角的“Save（保存）”。

10. 在 ThingsBoard 上提供可视化图表

1. 转到 <https://demo.thingsboard.io/signup> 并创建一个帐户；将通过电子邮件发送一个激活链接。
2. 登录 ThingsBoard 帐户，然后转到“Devices（设备）”页面；点击右下角的悬浮按钮（Floating Action Button, FAB）创建新设备。

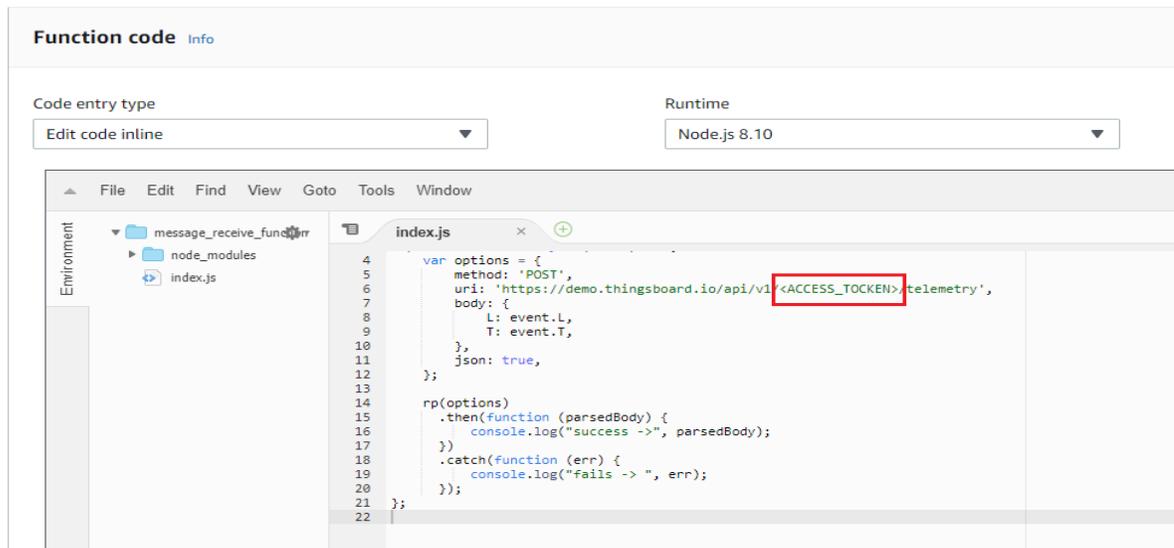
图 10-1. 添加设备

3. 点击带有新设备名称的卡，侧边栏将从屏幕右侧滑入，然后点击“Copy access token（复制访问令牌）”按钮。

图 10-2. 复制访问令牌按钮

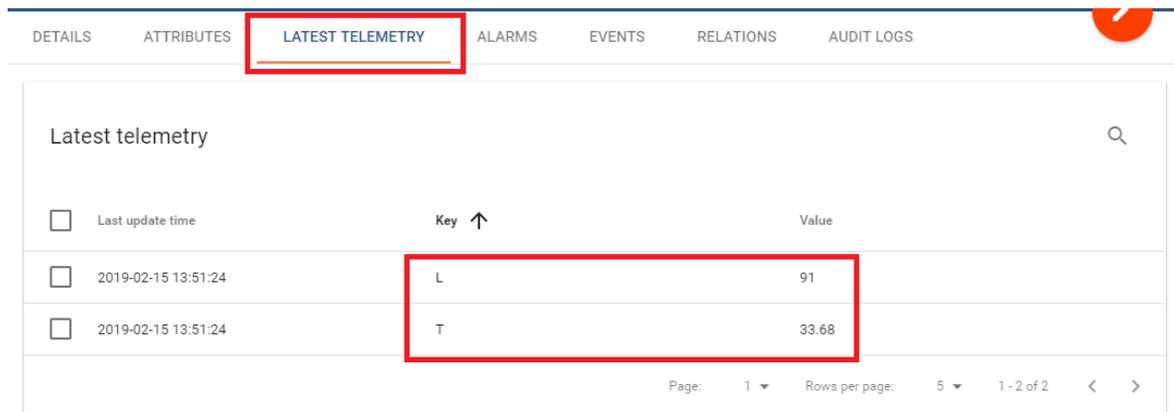
4. 转到先前创建的功能的 AWS Lambda 控制台页面。
5. 在“Function code（功能代码）”部分，确保在“Code entry type”下拉列表中选择“Edit code inline（编辑内嵌代码）”，并且编辑器中具有先前作为 .zip 上传的文件。
6. 在第 6 行的 `index.js` 中，将 `<ACCESS_TOKEN>` 替换为刚从 ThingsBoard 复制的访问令牌。

图 10-3. 在 Lambda 代码中使用设备访问令牌



7. 点击页面左上角的“Save”。
8. 连接开发板并确保其正在发送数据（黄色 LED 每秒切换一次）。
9. 返回到 Devices 页面上的 ThingsBoard，然后点击上面带有设备名称的卡。侧边栏将从右侧滑入，它包含多个标签；点击名为“Latest Telemetry（最新遥测）”的选项卡，将出现一个有两行的表。在“Key（密钥）”列的值中，“L”表示光，“T”表示温度；“Value（值）”列大约每秒会更新一次来自开发板的数据。

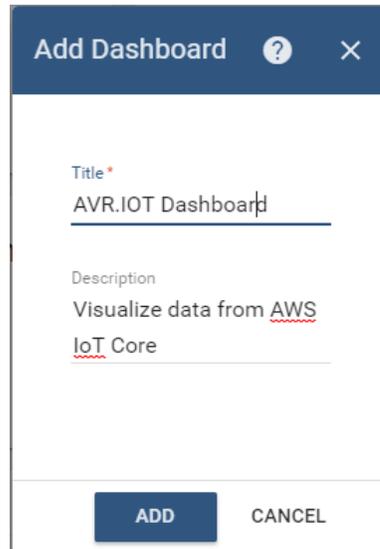
图 10-4. 设备最新遥测取值



数据一路传递到 ThingsBoard 服务器之后，就可以配置仪表板来绘制这些数据。

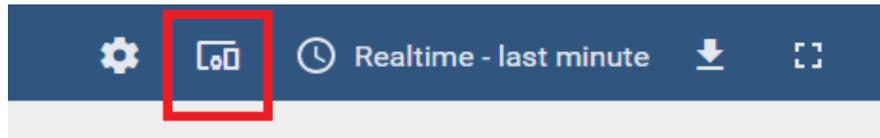
10. 在 ThingsBoard 中，转到“Dashboards（仪表板）”页面。
11. 点击右下角的 FAB 创建新的仪表板。

图 10-5. 添加仪表板



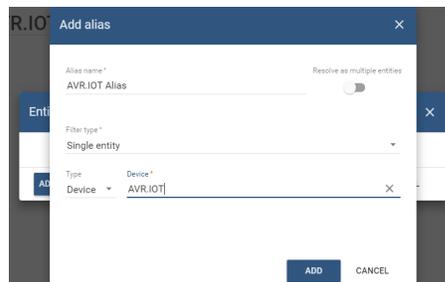
12. 点击带有新建仪表板名称的卡，然后移至 Dashboard 页面。
13. 点击右下角的 FAB 来编辑仪表板。
14. 在右上角的名称下方有一些图标；点击左侧名为“Entity aliases (实体别名)”的第二个图标，然后将打开一个模态窗口。

图 10-6. 在仪表板中编辑实体别名图标



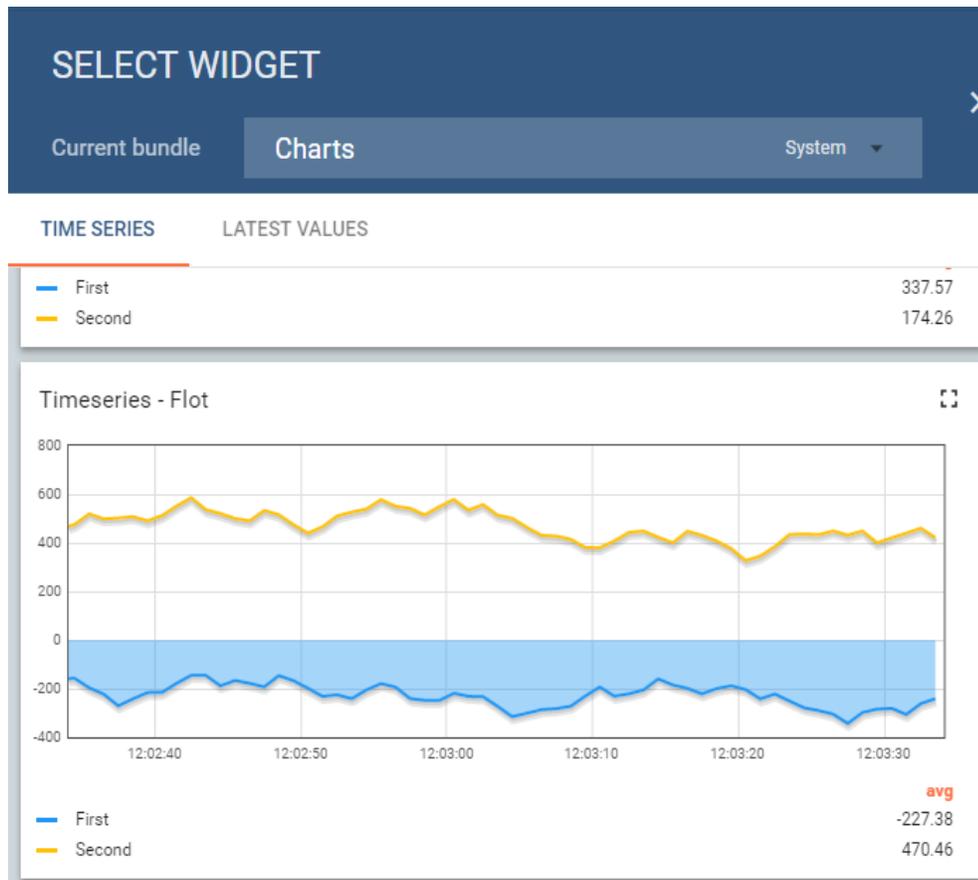
15. 点击“Add alias (添加别名)”并为其命名；在“Filter type (过滤类型)”下拉列表中，选择“Single entity (单个实体)”；对于“Type (类型)”，选择“Device (设备)”，对于“Device”，选择创建的设备，然后点击“Add (添加)”。

图 10-7. 添加别名



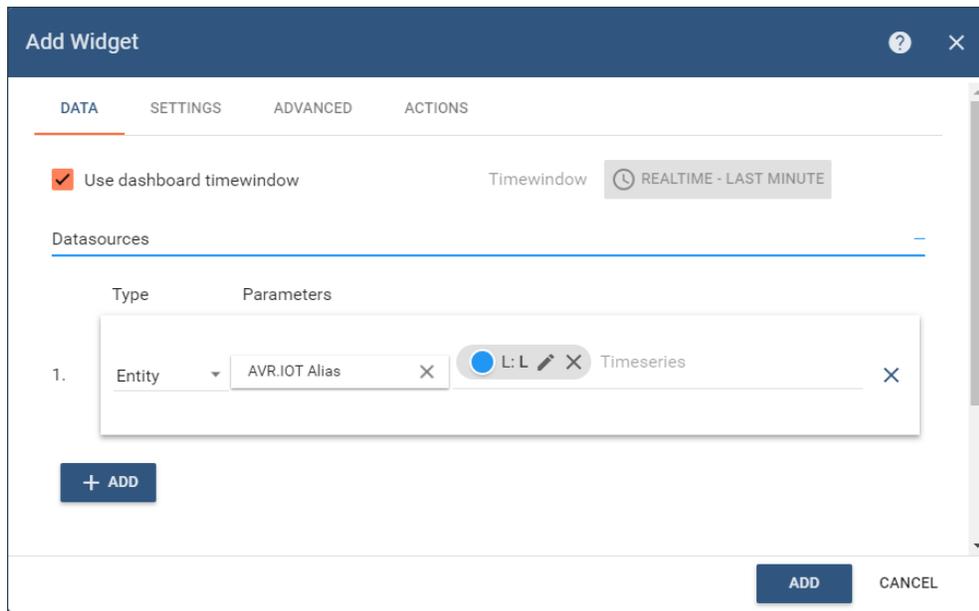
16. 点击“Save”。
17. 屏幕中心有一个名为“Add new widget (新增小工具)”的按钮；点击它，然后一个侧边栏将从右侧滑入。
18. 在“Current bundle (当前捆绑包)”下拉列表中，选择“Charts (图表)”，然后向下滚动到图表的“TimeSeries-Flot (时间序列—Flot)”类型。点击它，然后将显示一个带有配置选项的模态窗口。

图 10-8. 添加图表的时间序列—Flot 类型



19. 在“Datasets（数据集）”下方点击“Add”。
20. 类型将是“Entity（实体）”；对于“Entity alias”，选择先前创建的别名。
21. 在“TimeSeries（时间序列）”下拉列表中，将加载设备数据。要显示光数据图表，选择“L”，然后点击“Add”。

图 10-9. 配置图表数据源

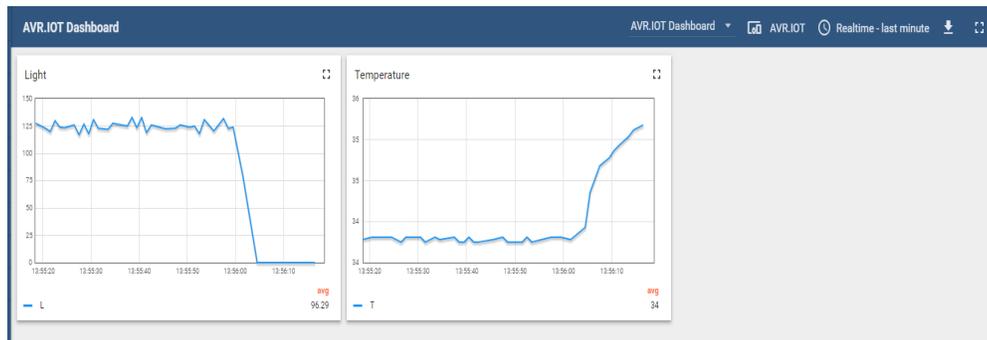


22. 类似地，添加温度图表。

23. 右下角有三个 FAB；点击带勾号的 FAB，完成仪表板的自定义。

配置现已准备就绪。用户现在可以看到图表上显示的实时数据，如下图所示。

图 10-10. AVR-IoT 收集的传感器数据在 ThingsBoard 中的显示



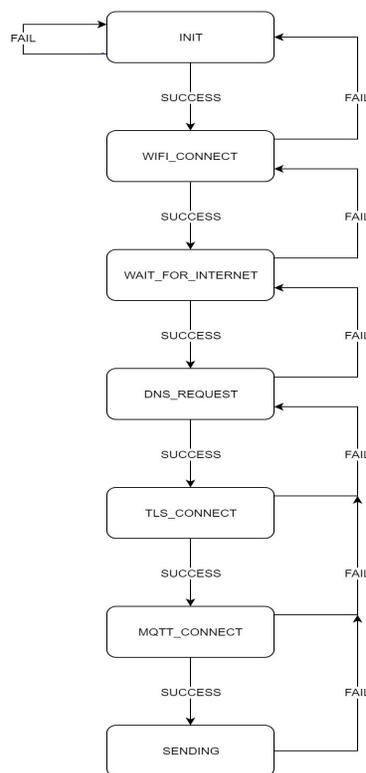
11. 固件

本节涉及 /demo 项目中的固件。下面是在云中发送数据之前固件遵循的步骤：

- 连接到 Wi-Fi
- 获取 AWS 服务器 IP
- 在 TLS 层连接到 AWS
- 在 MQTT 层连接到 AWS

这些步骤会作为状态机的状态在软件中建模。下图显示软件的状态图。

图 11-1. AVR-IoT 状态



开发板上的四个 LED 用于通知用户应用程序的状态。LED 的含义如下表所述。

表 11-1. LED 含义

蓝色 LED 亮起	已连接到 Wi-Fi
蓝色 LED 闪烁	已连接到 Wi-Fi, 但没有互联网连接
绿色 LED 亮起	已连接到 AWS
黄色 LED 闪烁	正在发送数据
红色 LED 亮起	发生错误

12. 结论

在物联网中，安全是基石，本应用笔记描述的演示设计可以实现高级别的安全性，以便成功连接到 AWS。AVR-IoT 开发板的模块化设计能够将 ATECC608A 和 ATWINC1510 的职责完全分离，而 ATmega4808 单片机则将它们联结在一起并实现应用逻辑。

AWS Cloud 提供多种服务，这些服务可以充当基于物联网设备所收集数据而实现的高级应用构建模块。本演示使用 AWS IoT Core 进行设备管理，使用 AWS Lambda 将数据转发到 ThingsBoard 进行可视化。最终解决方案也可使用其他服务进行存储、处理和其他操作。

ThingsBoard 提供易于使用的可视化工具，此外，如果使用复杂的用户管理系统，它还可以变成一个功能完整的物联网解决方案前端。

13. 附录：在 AWS EC2 上安装 ThingsBoard

由于 ThingsBoard 是一个开源平台，因此用户可轻松地将其安装在任何虚拟专用服务器（Virtual Private Server，VPS）上，例如 AWS EC2。这样，用户便可摆脱 ThingsBoard 的演示仪表板，构建一个基于 AWS Cloud 的完整系统。有关更多信息，参见“[Getting Started with Amazon EC2](#)”指南。

配置 VPS 之后，请继续阅读本 [ThingsBoard Installation](#) 指南。

注：最便捷的选项是 ThingsBoard Professional Edition（PE），它用于创建 VPS 和安装 ThingsBoard，因此已包含在 AWS Marketplace 中。有关更多信息，参见 [Installing ThingsBoard PE from AWS Marketplace](#) 指南。

Microchip 网站

Microchip 网站 (<http://www.microchip.com/>) 为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答 (FAQ)、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 <http://www.microchip.com/pcn>，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 嵌入式解决方案工程师 (ESE)
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式和地址。

也可通过以下网站获得技术支持：<http://www.microchip.com/support>

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿意与关心代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 (Digital Millennium Copyright Act)》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担

保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，否则在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、Microchip 徽标、Adaptec、AnyRate、AVR、AVR 徽标、AVR Freaks、BesTime、BitCloud、chipKIT、chipKIT 徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi 徽标、MOST、MOST 徽标、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32 徽标、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST 徽标、SuperFlash、Symmetricom、SyncServer、Tachyon、TempTrackr、TimeSource、tinyAVR、UNI/O、Vectron 及 XMEGA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、HyperLight Load、IntelliMOS、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plus 徽标、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、Vite、WinPath 和 ZL 均为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Incorporated 在美国的服务标记。

Adaptec 徽标、Frequency on Demand、Silicon Storage Technology 和 Symmcom 均为 Microchip Technology Inc. 在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc. 的子公司 Microchip Technology Germany II GmbH & Co. KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2020, Microchip Technology Incorporated 版权所有。

ISBN: 978-1-5224-6159-3

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 <http://www.microchip.com/quality>。

全球销售及服务中心

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话: 480-792-7200 传真: 480-792-7277 技术支持: http://www.microchip.com/support 网址: http://www.microchip.com	澳大利亚 - 悉尼 电话: 61-2-9868-6733 中国 - 北京 电话: 86-10-8569-7000 中国 - 成都 电话: 86-28-8665-5511 中国 - 重庆 电话: 86-23-8980-9588 中国 - 东莞 电话: 86-769-8702-9880 中国 - 广州 电话: 86-20-8755-8029 中国 - 杭州 电话: 86-571-8792-8115 中国 - 香港特别行政区 电话: 852-2943-5100 中国 - 南京 电话: 86-25-8473-2460 中国 - 青岛 电话: 86-532-8502-7355 中国 - 上海 电话: 86-21-3326-8000 中国 - 沈阳 电话: 86-24-2334-2829 中国 - 深圳 电话: 86-755-8864-2200 中国 - 苏州 电话: 86-186-6233-1526 中国 - 武汉 电话: 86-27-5980-5300 中国 - 西安 电话: 86-29-8833-7252 中国 - 厦门 电话: 86-592-2388138 中国 - 珠海 电话: 86-756-3210040	印度 - 班加罗尔 电话: 91-80-3090-4444 印度 - 新德里 电话: 91-11-4160-8631 印度 - 浦那 电话: 91-20-4121-0141 日本 - 大阪 电话: 81-6-6152-7160 日本 - 东京 电话: 81-3-6880-3770 韩国 - 大邱 电话: 82-53-744-4301 韩国 - 首尔 电话: 82-2-554-7200 马来西亚 - 吉隆坡 电话: 60-3-7651-7906 马来西亚 - 槟榔屿 电话: 60-4-227-8870 菲律宾 - 马尼拉 电话: 63-2-634-9065 新加坡 电话: 65-6334-8870 台湾地区 - 新竹 电话: 886-3-577-8366 台湾地区 - 高雄 电话: 886-7-213-7830 台湾地区 - 台北 电话: 886-2-2508-8600 泰国 - 曼谷 电话: 66-2-694-1351 越南 - 胡志明市 电话: 84-28-5448-2100	奥地利 - 韦尔斯 电话: 43-7242-2244-39 传真: 43-7242-2244-393 丹麦 - 哥本哈根 电话: 45-4485-5910 传真: 45-4485-2829 芬兰 - 埃斯波 电话: 358-9-4520-820 法国 - 巴黎 电话: 33-1-69-53-63-20 传真: 33-1-69-30-90-79 德国 - 加兴 电话: 49-8931-9700 德国 - 哈恩 电话: 49-2129-3766400 德国 - 海尔布隆 电话: 49-7131-72400 德国 - 卡尔斯鲁厄 电话: 49-721-625370 德国 - 慕尼黑 电话: 49-89-627-144-0 传真: 49-89-627-144-44 德国 - 罗森海姆 电话: 49-8031-354-560 以色列 - 若那那市 电话: 972-9-744-7705 意大利 - 米兰 电话: 39-0331-742611 传真: 39-0331-466781 意大利 - 帕多瓦 电话: 39-049-7625286 荷兰 - 德卢内市 电话: 31-416-690399 传真: 31-416-690340 挪威 - 特隆赫姆 电话: 47-72884388 波兰 - 华沙 电话: 48-22-3325737 罗马尼亚 - 布加勒斯特 电话: 40-21-407-87-50 西班牙 - 马德里 电话: 34-91-708-08-90 传真: 34-91-708-08-91 瑞典 - 哥德堡 电话: 46-31-704-60-40 瑞典 - 斯德哥尔摩 电话: 46-8-5090-4654 英国 - 沃金厄姆 电话: 44-118-921-5800 传真: 44-118-921-5820
亚特兰大 德卢斯, 佐治亚州 电话: 678-957-9614 传真: 678-957-1455 奥斯汀, 德克萨斯州 电话: 512-257-3370 波士顿 韦斯特伯鲁, 马萨诸塞州 电话: 774-760-0087 传真: 774-760-0088 芝加哥 艾塔斯卡, 伊利诺伊州 电话: 630-285-0071 传真: 630-285-0075 达拉斯 阿迪森, 德克萨斯州 电话: 972-818-7423 传真: 972-818-2924 底特律 诺维, 密歇根州 电话: 248-848-4000 休斯顿, 德克萨斯州 电话: 281-894-5983 印第安纳波利斯 诺布尔斯特维尔, 印第安纳州 电话: 317-773-8323 传真: 317-773-5453 电话: 317-536-2380 洛杉矶 米慎维荷, 加利福尼亚州 电话: 949-462-9523 传真: 949-462-9608 电话: 951-273-7800 罗利, 北卡罗来纳州 电话: 919-844-7510 纽约, 纽约州 电话: 631-435-6000 圣何塞, 加利福尼亚州 电话: 408-735-9110 电话: 408-436-4270 加拿大 - 多伦多 电话: 905-695-1980 传真: 905-695-2078			