基于 **Amazon Echo** 和 **Raspberry Pi** 的自动窗帘控制

Fortware Wang

September 8, 2017

目录

基于 **Amazon Echo** 和 **Raspberry Pi** 的自动窗帘控制 **3**

步骤一**:** 设计概述 **3**

步骤二**:** 整合电动窗帘轨道和 **Raspberry Pi** **4**

步骤三**:** 编写打开或关闭窗帘的 **Python** 脚本 **6**

步骤四**:** 编写 **Amazon Skill** 与 **Raspberry Pi** 进行通信 **8**

步骤五**:** 在 **Raspberry Pi** 上创建本地脚本并在路由器上实现端口转发 **8**

步骤六**:** 在 **Amazon** 开发者网站创建 **Alexa Skill** **11**

步骤七**:** 创建 **Amazon AWS Lambda** 代理 **12**

步骤八**:** 整合与集成 **14**

原文链接：https://www.wandianshenme.com/play/□□-amazon-echo-□-raspberry-pi-□□□□□□

基于 **Amazon Echo** 和 **Raspberry Pi** 的自动窗帘控制

本指南说明如何整合电动窗帘轨道，Raspberry Pi 和 Amazon Echo，以便您可以请求 Alexa 打开窗帘。

一旦完成，你应该可以说：“Alexa，请打开窗帘”和“Alexa，关闭窗帘”。

步骤一：设计概述

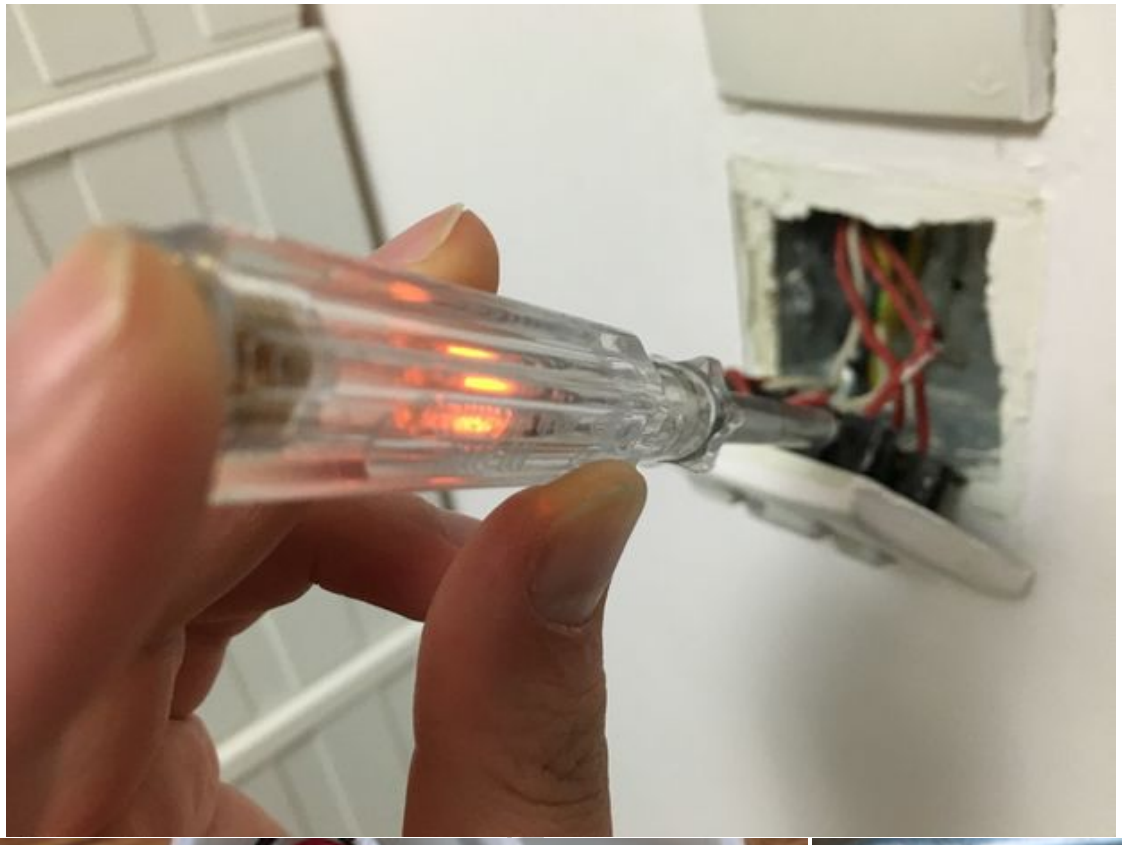
本设计由以下几个部分组成：

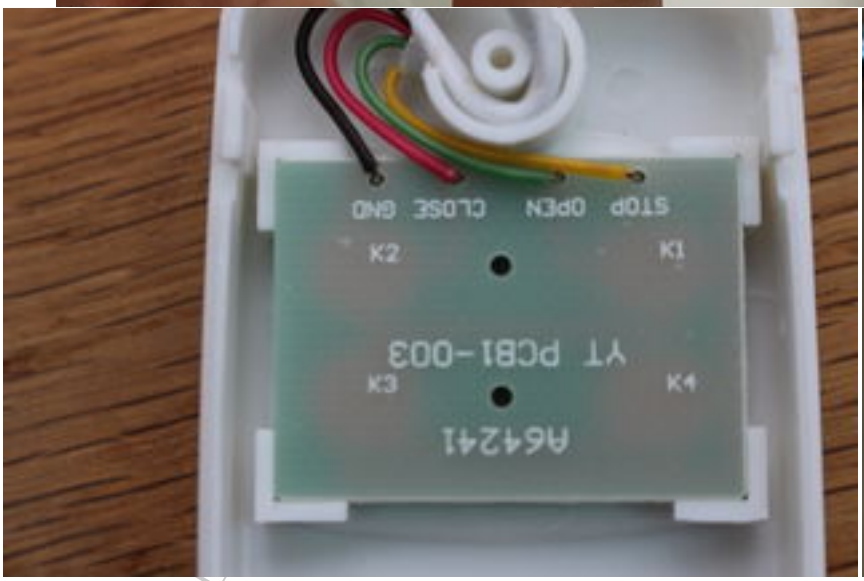
1. 将电动窗帘轨道与 Raspberry Pi 集成
2. 编写一个 Python 脚本来打开和关闭窗帘
3. 编写 Amazon skill，让您的 Echo 与 Raspberry Pi 进行通信

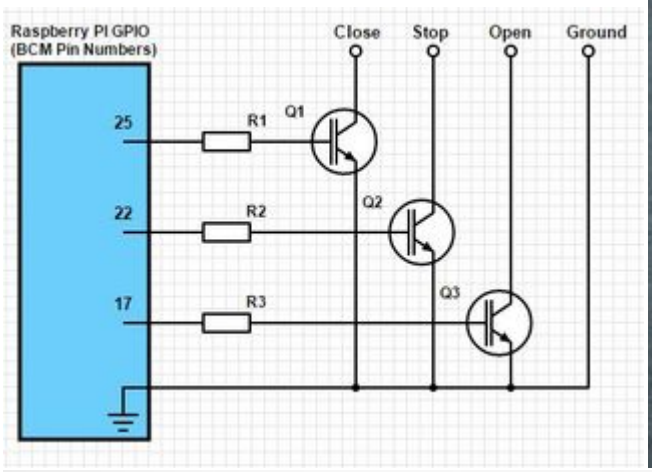
我使用以下组件：

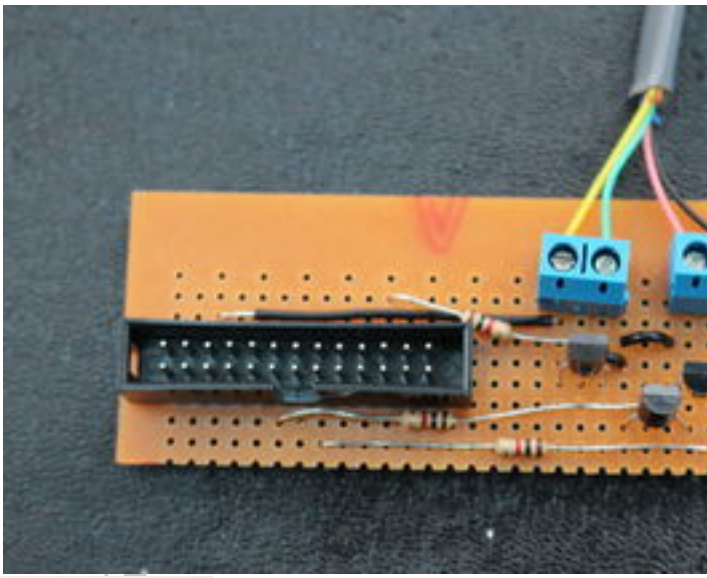
1. 一个 Amazon Echo
2. 一个电动窗帘轨道,https://www.amazon.co.uk/gp/product/B013X0BZF2
3. 连接 Raspberry Pi GPIO 的水晶头网线,http://www.ebay.co.uk/itm/190738599768?_trksid=p2057872.m2749.l2649&ssPageName=STRK%3AMEBIDX%3AIT
4. 一块 Raspberry Pi 开发板
5. 与 Raspberry Pi GPIO 整合所需的组件（我从 Proto-Pic 订购），包括 Raspberry Pi GPIO 转接口 (2x13)、带状排线、3 个型号为 BC547 的 NPN 晶体管、3 个 1K 电阻、1 块合适的单面洞洞板、适量导线

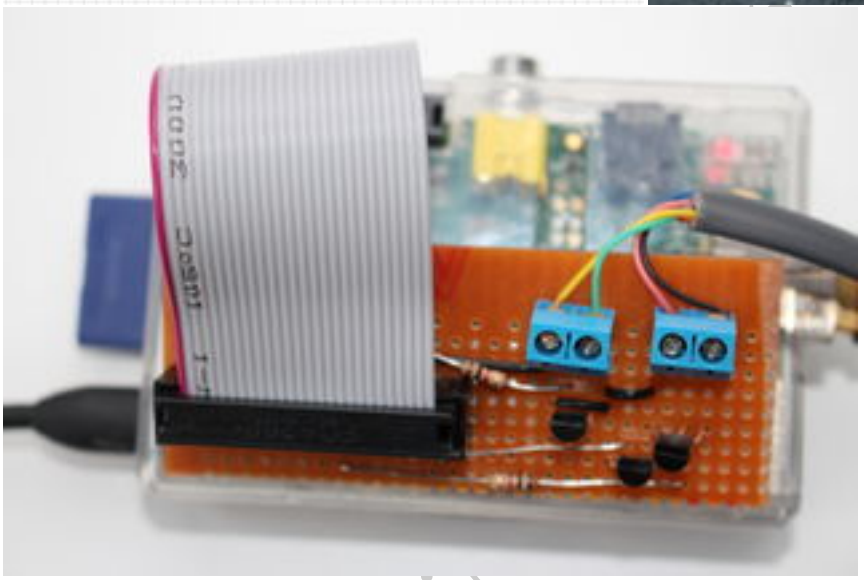
步骤二: 整合电动窗帘轨道和 **Raspberry Pi**











我购买的窗帘轨道带两个控制器 - 有线和无线。我没有碰到无线的，选择将 Raspberry Pi 与有线控制器整合。

一旦拆卸，您可以看到有线的控制器由四个开关组成（见电路图）-通过跟踪电线，您可以看到按钮将三根电线连接到地（第四根导线）。标示为 'open' 的短电线导致电机单向启动; 标记为 'close' 的短电线使电机以其他方式驱动。标有 'stop' （按停止按钮引起）的短线使电机停止。

切掉水晶头网线的一端并剥去外皮。

将原始控制器插头的背面与水晶头网线进行比较，以确定导线线序（在图中，原始控制线为白色; 水晶头网线为灰色）。使用四根线; 两个不需要。通过短路打开，关闭和停止接地线的电线，确认您有正确的电线。窗帘导轨应打开，关闭并停止。

导线和接地之间的电势是 5V - 我认为这是一个电阻上拉的的逻辑输入，接地时切换

到低电平。

与 Raspberry Pi GPIO 集成相对简单。任何基本的晶体管都可以用作开关来临时接地每根线，从而模拟开关按压。

图中给出了一个示例电路。注意电线应确保 Raspberry Pi 和窗帘共地（Raspberry PI 6 号引脚）。

我使用标准 BJT 晶体管 - NPN BC547 和 1K 电阻将晶体管的基极连接到 Raspberry Pi。

我连接到物理引脚 11,15 和 22 号，映射到 BCM 格式的 GPIO 引脚 17,22 和 25，更多关于物理和逻辑引脚之间的关系参考：https://projects.drogon.net/raspberry-pi/wiringpi/pins/

步骤三: 编写打开或关闭窗帘的 **Python** 脚本

Raspberry Pi 具有非常灵活的通用输入/输出端口 - 简称为 GPIO。有关使用 Raspberry Pi GPIO 有许多很好的指导和文档。

还有一个优秀的的图书馆，可用于 Python 和 C，这使得驱动 GPIO 非常简单 - Wiring Pi。原始页面位于:http://wiringpi.com/

在http://raspi.tv/how-to-install-wiringpi2-for-python-on-the-raspberry-pi你 可以找到一组有用的安装说明。

一旦安装完成，以下代码应使窗帘打开和关闭，每次更改之间有两秒的间隙。该代码通过将 GPIO 引脚置为高电平 0.5s 来模拟按钮，提高晶体管的基极电压，从而允许电流在集电极和发射极之间流动 - 将电线短路到地。

```
1 import RPi.GPIO as GPIO
2 import time
3 GPIO.setmode(GPIO.BCM)
4 openPin=17
5 closePin=25
6 stopPin=22
7 GPIO.setup(openPin, GPIO.OUT)
8 GPIO.setup(closePin, GPIO.OUT)
9 GPIO.setup(stopPin, GPIO.OUT)
10
11 time.sleep(2)
```

```
12
13 while (1==1) :  print "Opening\n"
14   GPIO.output(openPin, GPIO.HIGH)
15   time.sleep(0.5)
16   GPIO.output(openPin, GPIO.LOW)
17
18   time.sleep(2)
19
20   print "Stopping\n"
21   GPIO.output(openPin, GPIO.HIGH)
22   time.sleep(0.5)
23   GPIO.output(openPin, GPIO.LOW)
24
25   time.sleep(2)
26
27   print "Stopping\n"
28   GPIO.output(stopPin, GPIO.HIGH)
29   time.sleep(0.5)
30   GPIO.output(stopPin, GPIO.LOW)
31
32   time.sleep(2)
33
34   print "Closing\n"
35   GPIO.output(closePin, GPIO.HIGH)
36   time.sleep(0.5)
37   GPIO.output(closePin, GPIO.LOW)
38
39   time.sleep(2)
40
41   print "Stopping\n"
42   GPIO.output(stopPin, GPIO.HIGH)
43   time.sleep(0.5)
44   GPIO.output(stopPin, GPIO.LOW)
45
46   time.sleep(2)
```

步骤四: 编写 **Amazon Skill** 与 **Raspberry Pi** 进行通信

询问 **Alexa**

理想情况下，我们可以要求 Alexa 打开窗帘。但是我现在不知道 Echo 上提供的语音接口是否可以使用该类型的请求。所以，我们会说：'Alexa，请让窗帘打开'。然后“Alexa，请让窗帘关闭”。

有三个步骤:

1. 在 Raspberry Pi 上创建本地脚本，并在路由器上实现端口转发
2. 在亚马逊开发者网站上创建一个 Alexa skill
3. 创建 Amazon AWS Lambda 代理

值得欣慰的是，关于此方案前人已经有一些优秀的分享，因此我从以下链接收益颇丰:https://www.instructables.com/id/Control-Raspberry-Pi-GPIO-With-Amazon-Echo-and-Pyt/

该指南说明了如何在 Amazon 开发人员网站上创建一个 skill，指出 Amazon 使用 ngrok（ssl 为 http 代理）生成输出，然后使用名为 Flask-Ask 的库在 Raspberry Pi 上处理请求。ngrok 必不可少，因为开发者网站只会通过 ssl 发送请求 - Flask-ask 监听 http。

ngrok 作为一个临时代理很出色，但除非您要支付年费，否则每次重新启动您的 pi 上的 ngrok 客户端时，您需要修改指向 Amazon 脚本的 URL 。

另一种方法是在 Amazon AWS 上创建一个基本代理，然后可以将请求重定向到您的 Pi（假设您已经在防火墙上配置端口转发以允许请求）。

我使用了Matt创建的代理:https://forums.developer.amazon.com/questions/8155/how-to-use-aws-lambda-as-a-proxy-for-non-ssl-serve.html

感谢 Matt !

步骤五: 在 **Raspberry Pi** 上创建本地脚本并在路由器上实现端口转发

在 Raspberry Pi 创建以下脚本:

```
1 from flask import Flask
2
3 from flask_ask import Ask, statement, convert_errors import RPi.GPIO as
    GPIO import logging import time
```

```
4
5 GPIO.setmode(GPIO.BCM)
6 openPin=17
7
8 closePin=25
9
10 stopPin=22
11
12 GPIO.setup(openPin, GPIO.OUT)
13
14 GPIO.setup(closePin, GPIO.OUT)
15
16 GPIO.setup(stopPin, GPIO.OUT)
17
18 app = Flask(__name__) ask = Ask(app, '/')
19
20 app.config['ASK_VERIFY_REQUESTS'] = False
21
22 app.config['ASK_APPLICATION_ID'] = 'Unique amazon skill reference''
23
24 logging.getLogger("flask_ask").setLevel(logging.DEBUG)
25
26
27
28 def openCurtains():
29
30 print 'Opening curtains'
31
32 GPIO.output(openPin, GPIO.HIGH)
33
34 time.sleep(0.5)
35
36 GPIO.output(openPin, GPIO.LOW)
37
38 time.sleep(5)
39
```

```
40 GPIO.output(stopPin, GPIO.HIGH)
41 time.sleep(0.5)
42 GPIO.output(stopPin, GPIO.LOW)
43
44
45
46 def closeCurtains():
47
48 print 'Closing curtains'
49
50 GPIO.output(closePin, GPIO.HIGH)
51
52 time.sleep(0.5)
53
54 GPIO.output(closePin, GPIO.LOW)
55
56 time.sleep(5)
57
58 GPIO.output(stopPin, GPIO.HIGH)
59
60 time.sleep(0.5)
61
62 GPIO.output(stopPin, GPIO.LOW)
63
64
65
66 @ask.intent('GPIOControlIntent', mapping={'status': 'status'})
67 def gpio_control(status):
68
69 if status in ['open', 'yield']: openCurtains() return statement('Good
      morning Hart Household')
70
71 if status in ['close', 'shield']: closeCurtains() return statement('Good
      evening Hart Household')
72
73 if __name__ == '__main__': app.run(host='IP address of Pi', port=5000,
```

```
    debug=True)
```

启动 Pi 运行进程。在路由器上设置端口转发，以便发送到您的公共地址 5000 端口的请求转发到您的 Pi 的端口 5000。请注意，如果您的 IP 地址不是静态的，您可能需要订阅诸如 DYNDNS 之类的服务。

步骤六: 在 **Amazon** 开发者网站创建 **Alexa Skill**

让我们跟随 Patrick 的指引 - 从第 4 步开始：https://www.instructables.com/id/Control-Raspberry-Pi-GPIO-With-Amazon-Echo-and-Pyt/step4/AWS-Account/

Patrick 的第四步: 创建一个 Amazon 开发者帐号。

Patrick 的第五步: 创建一个新的 skill, 我使用了这个调用名字 “窗帘”，这样 Alexa 就会回应 “Alexa，问窗帘……”

请注意应用程序 ID - 您需要将其粘贴到您编写的 Curtains 脚本中，再加上要编写的 Lambda 代理。

Patrick 的第六步: 这里有一些微妙的变化，如截图所示。

意图很简单:

```
1 {
2
3 "intents": [{
4
```

```
"intent": "GPIOControlIntent",
"slots": [{
"name": "status",

"type": "GPIO_CONTROL"

}]

}]

}
```

slots 值包含两种: open 和 close

调用更简单:

```
GPIOControlIntent to {status}
```

Patrick 的第七步: 这是我们要使用 Lambda 代理而不是 Ngrok 的地方 - 看下一个阶段。

步骤七: 创建 **Amazon AWS Lambda** 代理

建立 AWS 帐户和创建 lambda 函数的最佳指导可从 Amazon 获得 - https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/developing-an-alexa-skill-as-a-lam

按照以下说明，直到您准备好将 node.js 代码输入到函数中。

然后使用来自 Matt 的代理:https://forums.developer.amazon.com/questions/8155/how-to-use-aws-lambda-as-a-proxy-for-non-ssl-serve.html。

(我在格式化代码的地方错过了一个 skill - 抱歉):

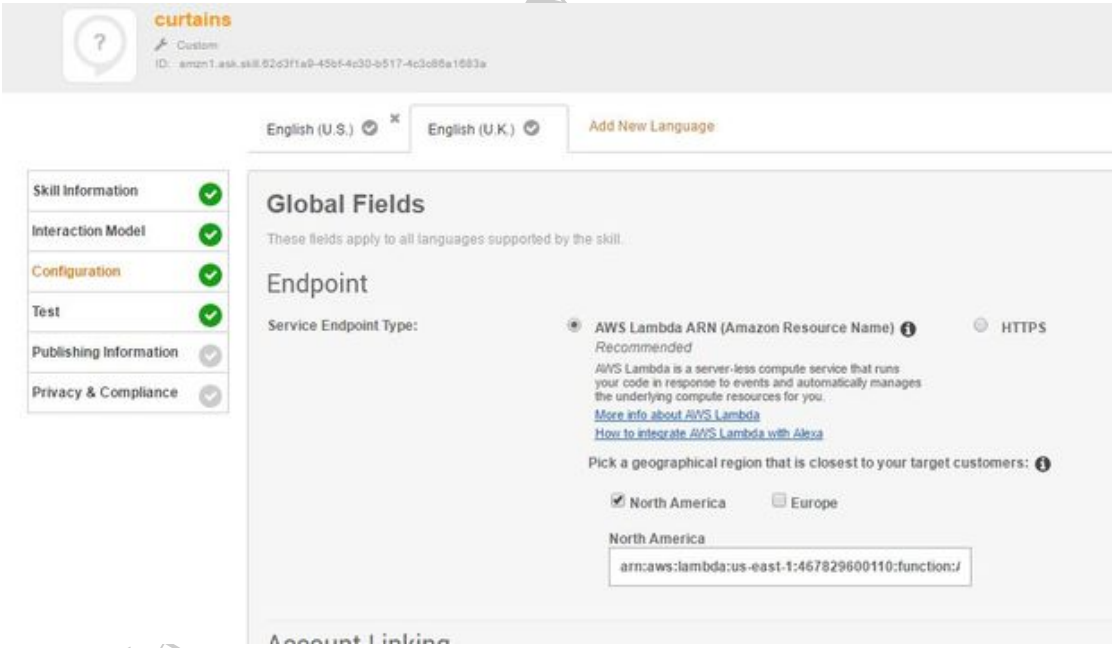
```
var http = require('http');
var URLParser = require('url');

console.log('Loading proxy function');

exports.handler = function (json, context)

```

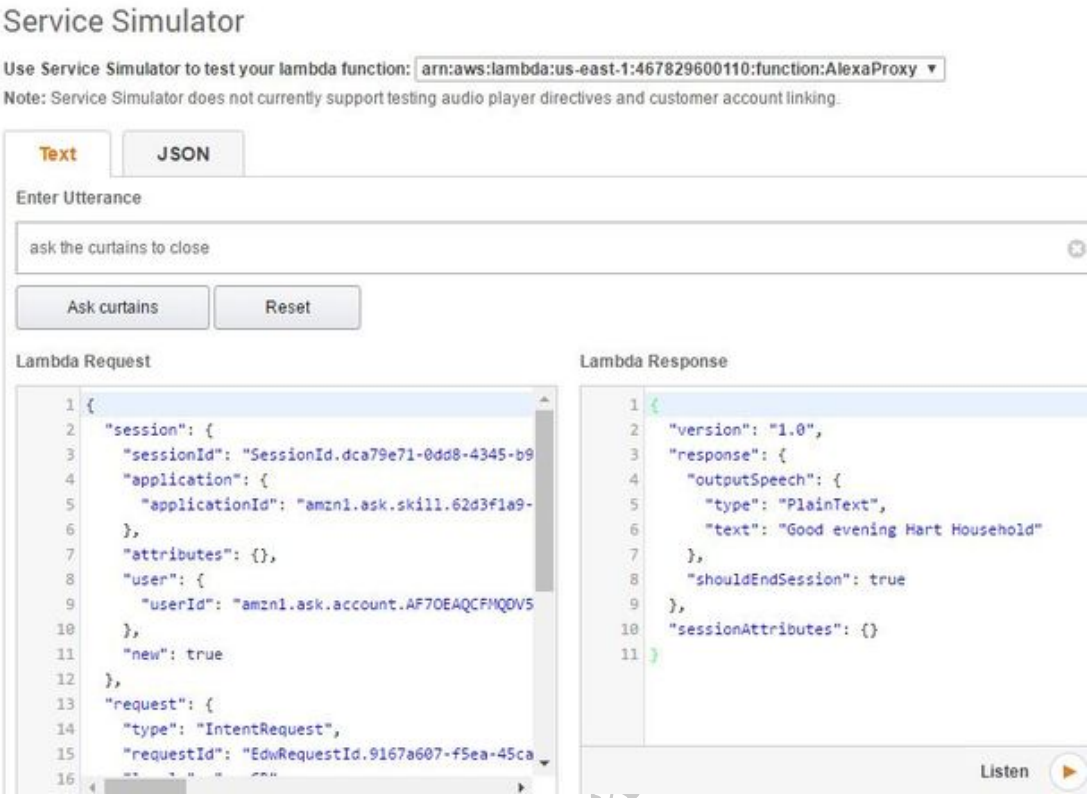
```
8 {
9 try {
10 // A list of URL's to call for each applicationId
11
12 var handlers = { 'appId':'url', 'your amazon app id':'http://your router
       address:5000/' };
13
14 // Look up the url to call based on the appId
15
16 var url = handlers[json.session.application.applicationId];
17
18 if (!url) { context.fail("No url found for application id"); }
19
20 console.log('Trying url')
21
22 console.log(url)
23
24 var parts = URLParser.parse(url);
25
26 var post_data = JSON.stringify(json);
27
28 // An object of options to indicate where to post to
29
30 var post_options = { host: parts.hostname, port: (parts.port || 80), path:
       parts.path, method: 'POST', headers: { 'Content-Type':
       'application/json', 'Content-Length': post_data.length } };
31
32 // Initiate the request to the HTTP endpoint
33
34 console.log(post_options)
35
36 var req = http.request(post_options,function(res) {
37
38 var body = "";
39
40 // Data may be chunked
```

```
41
42 res.on('data', function(chunk) { body += chunk; });
43 res.on('end', function() { // When data is done, finish the request
44
45 context.succeed(JSON.parse(body)); }); });
46
47 req.on('error', function(e) { context.fail('problem with request: ' +
       e.message); });
48
49 // Send the JSON data console.log('Sending data') req.write(post_data);
       req.end(); } catch (e) { context.fail("Exception: " + e); } };
```

我还将高级配置中的超时增加到 7 秒 - 否则，在窗帘完成 5 秒的活动之前，Lambda 代理将超时。

步骤八: 整合与集成





复制 Lambda ID - 您需要将其粘贴到 Amazon 开发中，指示 Amazon 将请求引导到您的新 Lambda 代理，将其转发到 Raspberry Pi 上 - 请参阅第 1 个屏幕截图。

从亚马逊开发者网站测试 skill - 你应该看到第二个屏幕截图的响应。

如果它不起作用，请检查 Lambda 日志的调用错误和 Raspberry Pi 脚本输出的错误。

然后打开 Echo 的开发 skill - 不要将其对外发送，否则您将失去对窗帘的控制。

坐下来享受自动化窗帘！

原文链接:https://www.instructables.com/id/Amazon-Echo-Controlled-Curtains/

原文链接：https://www.wandianshenme.com/play/□□-amazon-echo-□-raspberry-pi-□□□□□□