**Node-RED: 讲座3-基础节点与流**

在本讲座中，您将看到一些更常用的节点，并且基于以前的讲座中学到的一些内容。 您将从一系列基于流行的MQTT协议的示例开始，显示如何将一组基本但非常有用的消息处理节点连接在一起。 然后将简要介绍使用TCP，UDP和Websockets等协议将消息传入和传出流的其他方法。

在本讲座结束时，您将更好地了解Node-Red流中使用的一些基本节点。 您还将看到构建复杂的处理流程，采取现实世界的事件然后处理它们并使用常用的Internet协议来生成传达流程外的响应是多么容易。

我们进行这些讲座所使用的Node-RED云托管叫做FRED。 在FRED注册一个免费帐户。 早期讲座中的示例将与Node-RED的其他安装一起使用，如果不使用FRED，则稍后讲课将使用您需要自己安装的节点。

## 示例3.1通过MQTT消息接收JSON

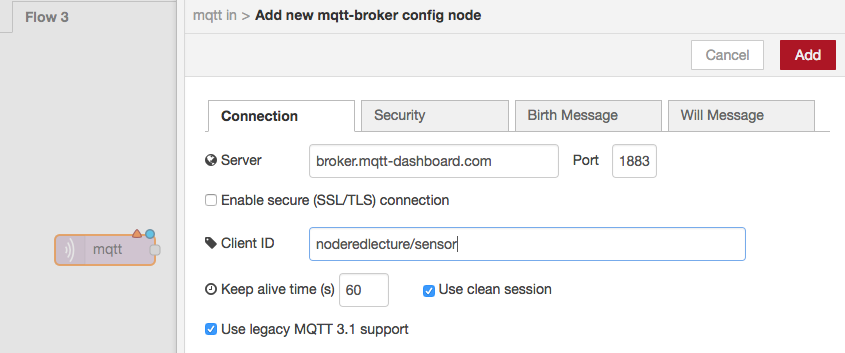
以下一系列示例均构建在mqtt节点上，该节点提供了一种方便的方式来从MQTT代理获取输入。对于不熟悉MQTT的用户，这其实是一个发布/订阅系统（通常缩写为pub / sub系统）的示例，它可以让传感器向所有订阅该传感器的客户端发送信息。 MQTT使用主题模型，允许发布者（例如传感器）创建主题并将数据发布到主题。同样，其他人可以订阅主题，并将收到发布到主题的数据的异步通知。

Pub / Sub系统是连接松散耦合的分布式系统的一种良好方式，它们很好地映射到设备或事物生成要共享的事件的典型IoT模式。 除了异步以外，MQTT协议也是轻量级的，并不像HTTP那么高; 对于资源有限的设备来说，这是一个重要的优点。 MQTT最初是在20世纪90年代后期开发的，现在已被用于各种IoT设置。 MQTT在2014年成为OASIS标准，是许多IoT工具箱的标准组成部分。 MQTT实际上代表消息队列遥测传输。

要使用mqtt节点，您需要访问代理。这些是一些免费的MQTT服务器，例如http://test.mosquitto.org/，或者在本讲座中使用的服务器www.hivemq.com。 使用代理地址和主题，您可以配置mqtt输入节点以订阅该主题，导致在该主题上发布新数据时生成新消息。 该消息将包含已发布数据的信息，包括msg.payload中的数据本身和msg.topic中的MQTT代理主题。

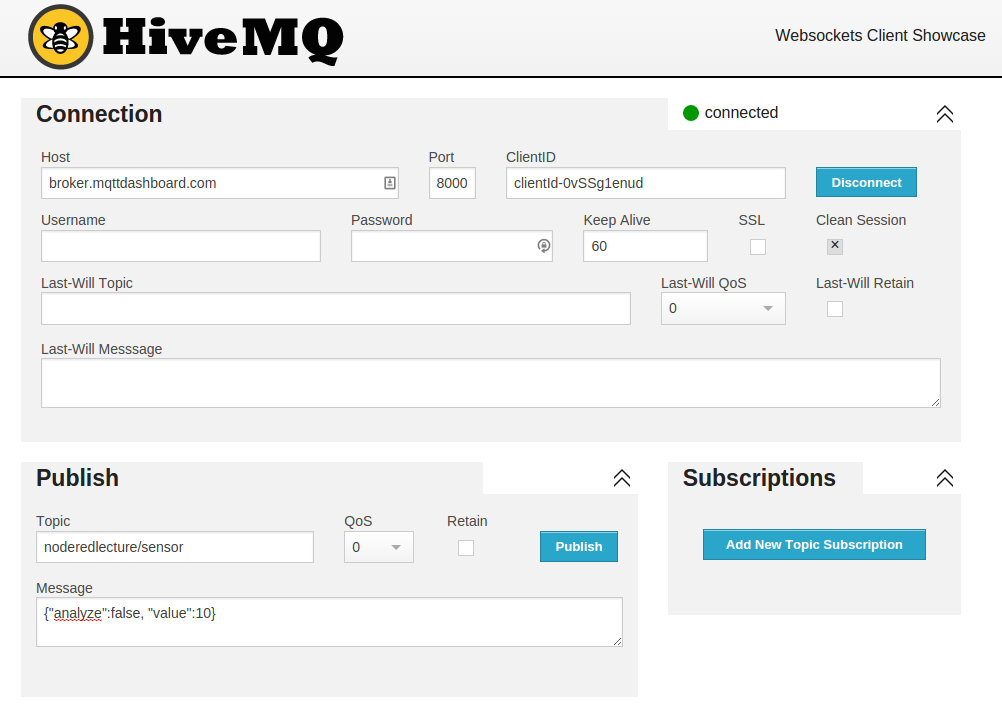
为了让您开始使用mqtt节点，您将使用免费的mqqt代理hivemq，这可以通过（http://www.hivemq.com/showcase/public-mqtt-broker/）获得。 当然，您可以使用任何MQTT代理，如果您已经安装了一个，您也可以使用您自己的代理。

首先，拖放一个mqtt输入节点并为代理配置它。 不要忘记将主题配置为独特的东西，在这个例子中，我们使用noderedlecture / sensor，但是您应该使用自己独特的主题，例如<您的名字> / sensor



图片3.1配置具有代理地址和主题的mqtt节点

有很多方法可以将mqtt消息发送到hivemq。 您可以使用他们的websockets客户端展示（http://www.hivemq.com/demos/websocket-client/），mqtt仪表板（http://www.mqtt-dashboard.com/dashboard）或您自己的库。 您将在此示例中使用其Websocket客户端，导航到该页面并连接到代理。 您将发布JSON编码的字符串到您配置Topic从而检测的mqtt节点和json节点两个节点的作用

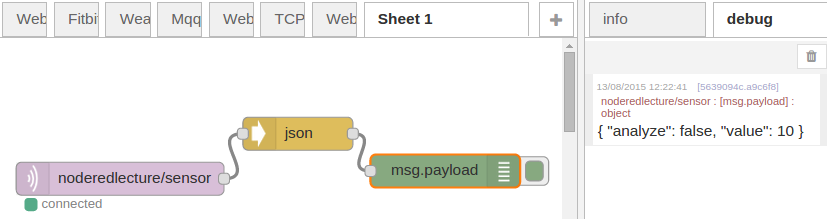


图片3.2 使用HiveMQ客户端界面发送MQTT信息

由于您正在发送一个JSON字符串，所以您需要解析mqtt节点收到MQTT消息时生成的消息。 为此，您需要拖放一个json节点并将其连接到mqtt节点的输出。

Node-RED的json节点是一种方便的功能，因为它解析了传入的消息，并尝试将其转换（从JSON转换）。 所以如果你发送一个JSON字符串，它会将它转换成一个JavaScript对象，反之亦然。

如果将通常的调试节点连接到json节点并部署，则使用HiveMQ仪表板发送JSON字符串{“analyze”：false，“value”：10}，如图3.2所示。 您将看到它在调试选项卡中打印（图3.3）。



图片3.3接收和解析作为JSON字符串发送的MQTT消息

如果仔细观察输出，可以看到msg.payload包含一个对象，它本身有两个字段，分析和赋值，每个都有自己的值。 正如您在讲座2中看到的，您可以通过msg.payload.analyze和msg.payload.value访问这些字段。 我们来看看可以做到这一点的节点。

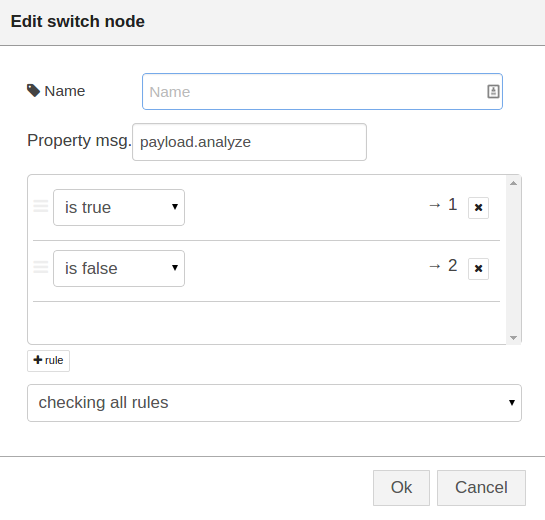
您可以在以下网址找到此流程的Node RED信息 ：

[https://raw.githubusercontent.com/SenseTecnic/nrguideflows/master/lesson3/3-1\_mqqtmessages.json](https://www.google.com/url?q=https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson3/3-1_mqqtmessages.json&sa=D&usg=AFQjCNFyKaJHmOh8VGbmZCJ7dRREnuWl6g)

## 示例3.2使用switch节点来处理JSON对象

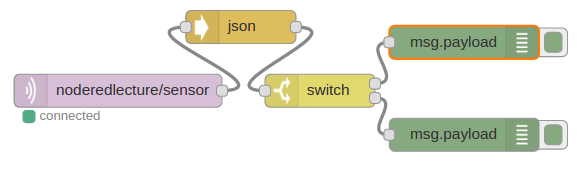
拥有JSON对象的一个很好的功能就是可以轻松地对其属性执行操作。 这时一个有用的节点是switch节点。 它的作用是根据传入的消息属性来“切换”或发送消息。 例如，您可以检查msg.payload.analyze属性，并根据其值（true / false）决定将消息发送到其中一个switch节点的输出。

拖动开关节点并双击它。 配置它来评估属性“msg.payload.analyze”。 如果为true，则将消息发送到第一个输出; 如果为false，则将其发送到第二个输出，如图3.4所示。



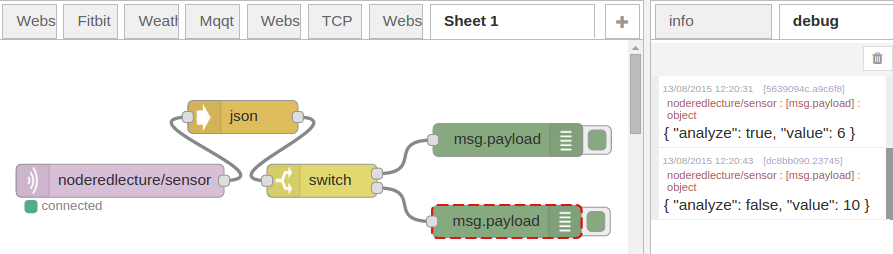
图片3.4 配置switch节点来根据数据属性传输数据

现在你可以连接两个debug节点（如图3.5）——当你为一个节点设置许多输出时，它们从上到下进行编号，因此输出1在上面，输出2在底部在图3.5中。



图片3.5 将switch 节点连接到两个debug节点

如果现在返回HiveMQ输入页面并发送MQTT消息{“analyze”：true，“value”：6}，您将看到第一个（顶部）输出被激活，传入的消息被发送了，或者“switched“ 至输出1.如果发送原始消息{”analyze“：false，”value“：10}，switch节点将激活输出2，原始调试节点将触发。 将指针悬停在调试消息上将显示哪个调试节点正在打印出消息，如图3.6所示。



图片3.6确认switch节点的发送

您可以看到，这为您提供了一个内置的Node-RED节点，可让您快速确定传入消息的内容，并根据输入将消息定向到流的不同部分。

您可以在以下网址找到此流程的Node RED信息：

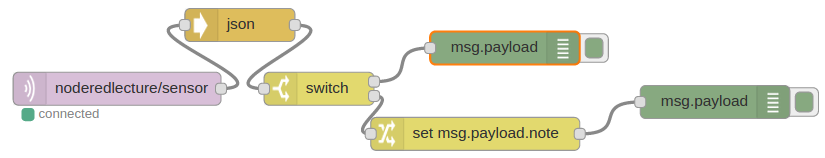
<https://raw.githubusercontent.com/SenseTecnic/nrguideflows/master/lesson3/3-2_switchnode.json>

## 示例3.3使用change节点更改或操作消息内容

另外一个非常有用的节点是change节点，能够让您修改消息内容或者为消息添加新的属性，您能够使用这个节点来影响消息中的属性，方法是更改现有属性，删除或添加新属性。

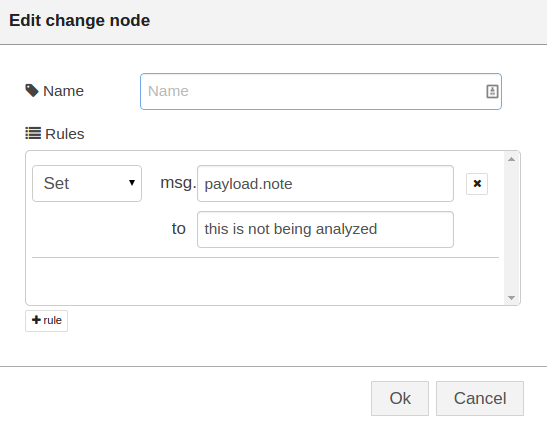
在此示例中，您将继续使用MQTT主题，并了解如何根据传入的MQTT消息成功地“切换”消息流，您可以添加新的消息属性msg.payload.note。

首先，我们拖放一个change节点并将其连接到switch机节点的第二个输出（图3.7）。 这是msg.payload.analyze设置为false时触发的输出。



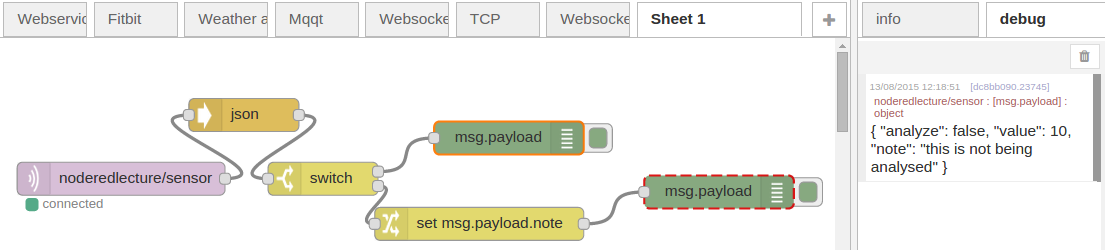
图片3.7添加一个change节点并设置新的信息属性

现在配置它，将属性msg.payload.note设置为“this is not being analyzed”，如图3.8所示。



图片3.8添加一个change节点并设置新的信息属性

当您收到switch节点在第二个输出上发送的消息时，它将被修改为包含一个“note”元素，其中的字符串“this is not being analyzed”。 如果通过从HiveMQ发送MQTT消息来部署和测试流程，您将看到如图3.9所示的输出。



图片3.9 交换和改变信息之后的结果

您可以在以下网址找到此流程的Node RED信息：

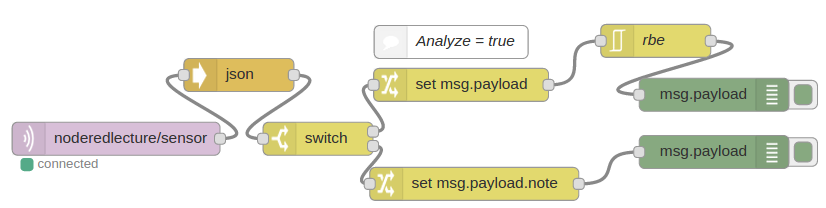
[https://raw.githubusercontent.com/SenseTecnic/nrguideflows/master/lesson3/3-3\_changenode.json](https://www.google.com/url?q=https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson3/3-3_changenode.json&sa=D&usg=AFQjCNGoD2oT5d801TLJtXMJfvhymXgl1Q)

## 示例3.4使用rbe（report by exception）节点

在此示例中，您将继续使用消息分析主题，并将节点添加到当您要分析流程时所需的部分。 您可以使用（如果原数据被更改了）rbe（report by exception）节点。 您可以将其设置为检查消息有效负载，并阻塞，直到消息更改（rbe模式）或消息更改指定量（死区模式）为止。 在rbe模式下，它适用于数字和字符串。 在死区模式下，它仅适用于数字，并将配置的死区范围增加或减小，以便传入值在触发之前会在一定范围内波动。

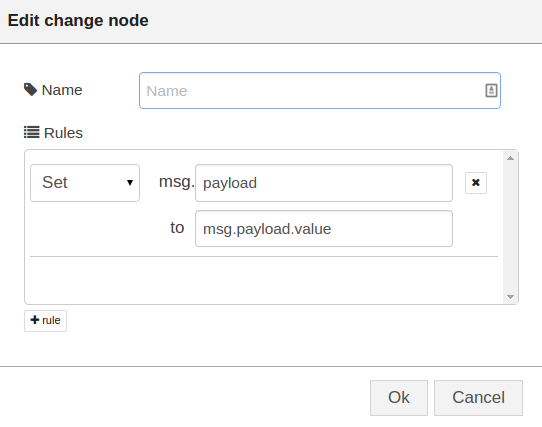
您将从添加另一个连接至switch节点的输出1的change节点开始。然后，您将连接一个rbe节点到交换机节点，如图3.10所示。

我们连接一个change节点和一个这样的rbe节点。 要提醒我们这个输出处理将其添加标志“analyze”，添加一个comment节点并写“Analyze = true”。 编写复杂流时，注释很有用。



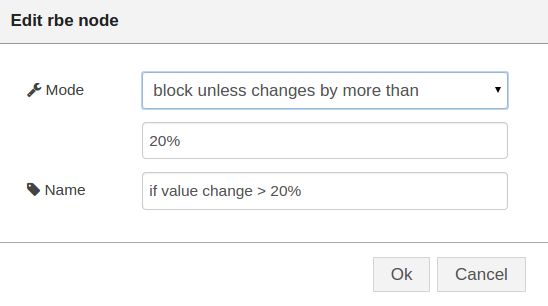
图片3.10 添加一个rbe节点来检测输入数据是否被改变超过20%

编辑change节点将msg.payload设置为msg.payload.value。 这将该节点的输出设置为接收到的输入中的msg.payload.value元素中的值（图3.11）



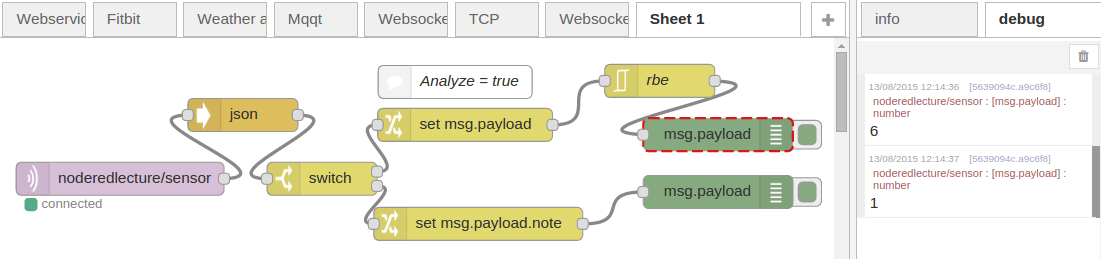
图片3.11 使用change节点来设置接收信息

由于您要确定此值是否已经更改了20％或更多，您需要双击rbe节点并将其配置为阻止，除非该值更改超过20％。



图片3.12设置rbe节点来检测接收信息的值

要测试流程，请部署此流程，然后返回HiveMQ页面并发送一系列消息。 首先，您需要将分析值设置为true，以便交换节点通过输出1上的消息发送。如果使用原始消息值6，则将无法通过rbe节点。 如果然后发送值为10的第二个消息，则rbe节点将评估6到10之间的差异，显而易见这是大于20％的，则rbe节点发送消息到最终调试节点，该调试节点将在调试窗格上打印，如图3.13所示。



图片3.13使用rbe节点确认10比6大20％以上。

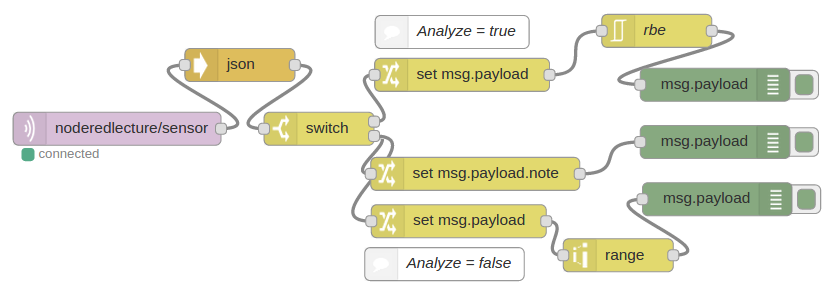
您可以在以下网址找到此流程的Node RED信息：

[https://raw.githubusercontent.com/SenseTecnic/nrguideflows/master/lesson3/3-4\_rbenode.json](https://www.google.com/url?q=https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson3/3-4_rbenode.json&sa=D&usg=AFQjCNEhJze_vy9tf7LWjCqhHQyz8qtMfQ)

## 示例3.5使用range节点缩放输入

当处理来自传感器和其他设备的真实数据输入时，通常需要扩展输入数据的能力。Node RED提供range节点以支持此功能，并允许您缩放（线性）输入值。 假设您想在不进行任何分析时将您的值（原始在0-10范围内）缩放到范围（0-255）。 这意味着当switch节点将Analyze属性评估为false时，我们正在处理流信息的下半部分。

要做到这一点，选择您上面配置的change节点（设置msg.payload）并复制ctrl + c，然后ctrl + v。 附加一个范围节点，如图3.14所示。

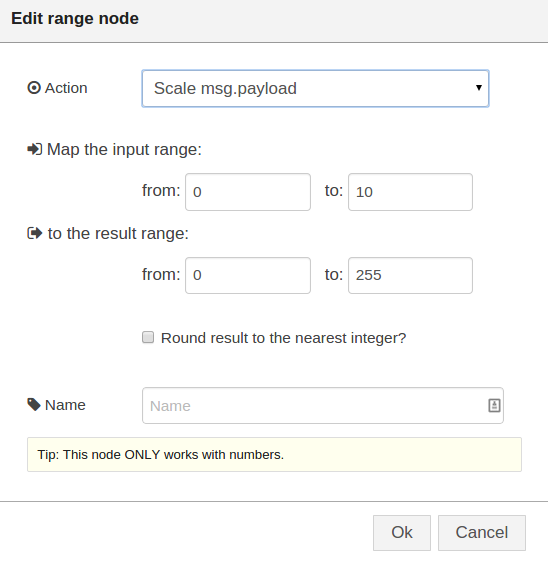


图片3.14使用range node配置输入数据

双击它，并将其配置为将输入从0-10映射到0-255，如图3.15所示。

range节点包含可设置范围的三个选项。默认值将根据给定的映射进行缩放，但如果使用相同的映射，结果将很快超出预设范围。缩放且限制到目标范围意味着结果将不会超出指定的范围。第三个选项，缩放和包装在目标范围内意味着结果将在范围内基本上是一个“模式”的回绕。

然后返回HiveMQ测试页面，并将{“analyze”：false，“value”：10}作为新的MQTT消息发送到同一主题。



图片3.15在scale节点中为输入输出设置缩放范围

如果返回到Node-RED窗口，您将看到与流程的下半部分关联的debug节点已经触发，显示出当您将其发布到MQTT中设置为10的msg.payload.value属性已按比例放大为255，如图3.16所示。

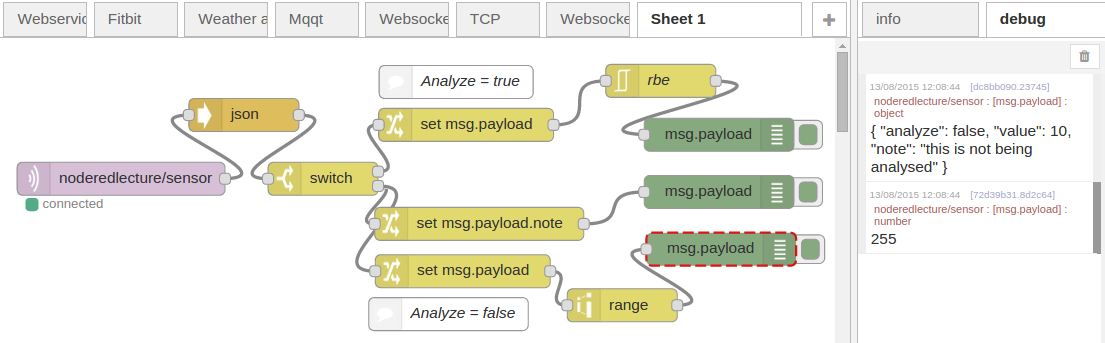


图3.16关闭分析时的最终缩放输出

您可以在以下网址找到此流程的Node RED信息:

[https://raw.githubusercontent.com/SenseTecnic/nrguideflows/master/lesson3/3-6\_mqqtout.json](https://www.google.com/url?q=https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson3/3-6_mqqtout.json&sa=D&usg=AFQjCNEYMv4nvL-NSdogbu6EloAtocR6Pw)

## 示例3.7在 Node-RED上使用 Websockets

Websockets节点是Node-RED中的另一个具有通信功能的内置节点。 Websockets提供双工TCP连接，旨在允许Web浏览器和服务器可用于增强传统HTTP交互的“反向通道”，从而在客户端不需要新的拉取请求的情况下允许服务器更新网页。

Websocket节点有两种输入和输出，允许您监听传入数据（输入）或在Websocket上发送（输出）。 输出节点版本旨在检查输出有效载荷是否起始于节点中的Websocket，在这种情况下，它将响应原始发送方。 否则它将广播有效载荷到所有连接的Websockets。

此外，输入和输出websocket节点可以被配置为服务器或客户端 - 在服务器模式下，他们可以监听一个URL，而在客户端模式下，它们可以连接到指定的IP地址。

要查看websocket节点的工作原理，您将使用在公共站点上运行的公共websockets echo服务器（https://www.websocket.org/echo.html）。

将一个inject，websocket in，websocket out和一个debug节点拖到工作区上并连接它们，如图3.18所示。

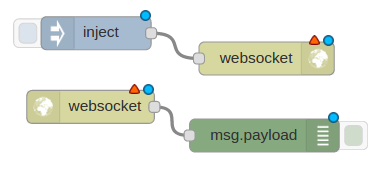


图3.18在Node-RED流中使用websockets进行通信

配置inject节点发送“Hello There”的字符串有效负载（图3.19）

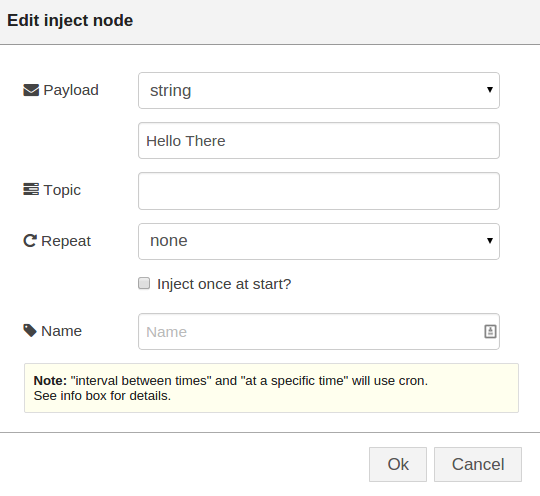


图3.19配置在websocket上发送的注入节点

配置websocket节点以连接到wss：//echo.websocket.org，如图3.20所示。

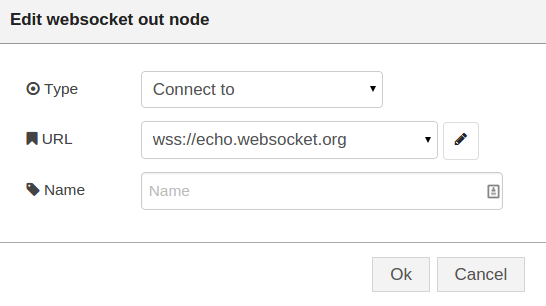


图3.20配置websocket发送到公共echo服务器。 对于out节点也一样

部署，当您点击inject节点时，您将看到如图3.21所示打印的消息

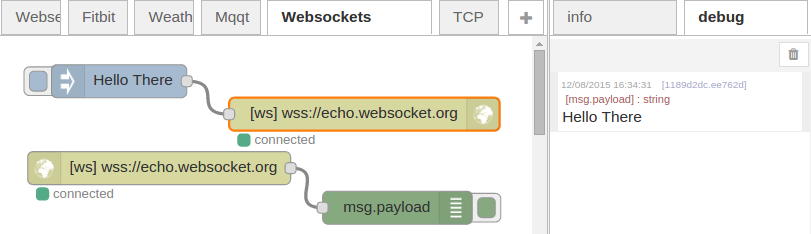


图3.21从监听处的传入数据的websocket输出

您可以在以下网址找到此流程的Node RED信息:

[https://raw.githubusercontent.com/SenseTecnic/nrguideflows/master/lesson3/3-7\_websockets.json](https://www.google.com/url?q=https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson3/3-7_websockets.json&sa=D&usg=AFQjCNFYpoisXXloDV4_9dNKGLv5XLdWog)

## 示例3.8发送TCP REQUEST。

此示例显示如何使用tcp节点发送TCP REQUEST。 在这种情况下，您将按照（http://tools.ietf.org/html/rfc2616#section-5.1.2）中的规范进行HTTP请求。

此示例显示了tcp节点的使用。 它可以用类似的方式配置udp或http节点。

一开始，我们连接一个inject，function，tcp request和debug节点，如图3.22所示。

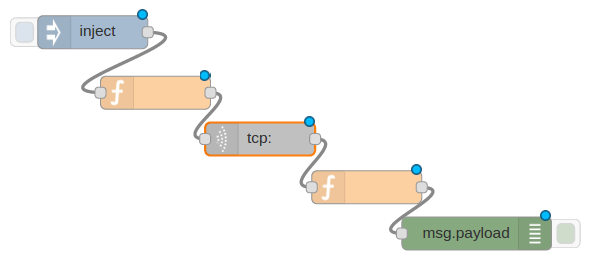


图3.22构建TCP REQUEST并在tcp输出节点上发送

编辑第一个功能节点以添加一个功能，将字符串“GET / HTTP / 1.1 \ r \ n \ r \ nHost：www.google.com”设置为有效载荷，如图3.23所示。

该字符串是标准的HTTP请求，表示它是GET请求，协议是HTTP 1.1，主机是www.google.com。 \ r \ n \ r \ n是HTTP协议中需要的两个返回/换行符号。

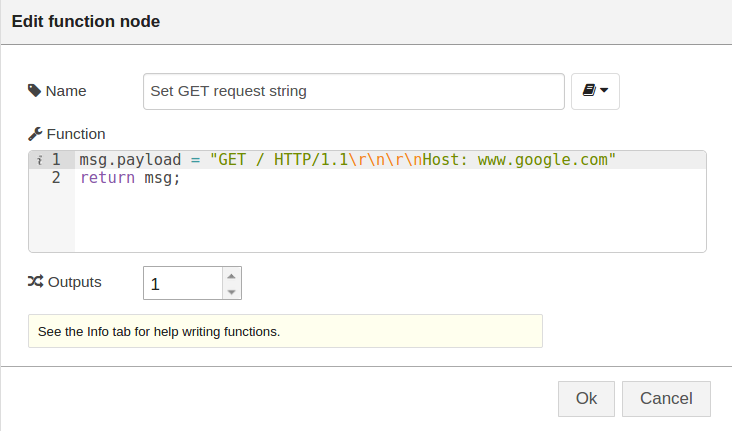


图3.23在function节点中构建TCP REQUEST

配置tcp request节点连接到端口80上的www.google.com服务器。配置它在1秒（1000毫秒）后关闭连接，如图3.24所示。

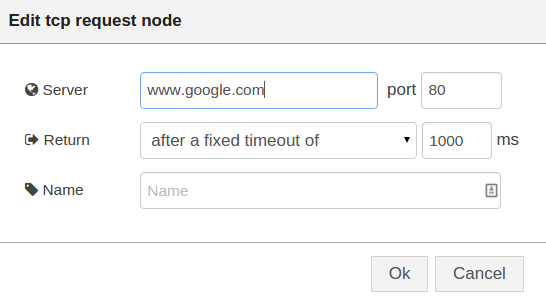


图3.24配置TCP REQUEST的终点

tcp request节点响应是一个缓冲区，需要解析。 配置第二个功能节点来解析tcp request节点响应，如图3.25所示

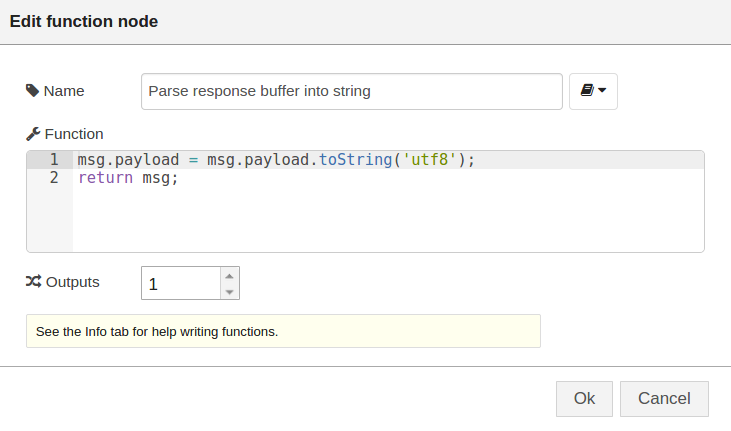


图3.25。 将响应缓冲区解析为字符串的函数节点

如果您部署流程并单击inject，您将向Google发出请求，并获得TCP响应。 调试节点将以如图3.26所示的字符串形式打印响应。

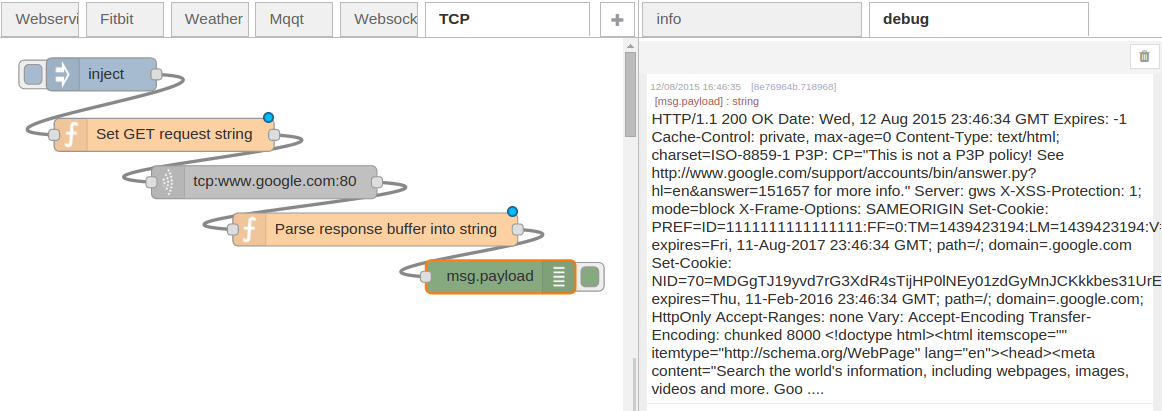


图3.26打印通过TCP连接发送的形式良好的HTTP请求的响应。

有些人可能会想知道为什么需要使用一个function节点构建我们通过TCP发送的HTTP请求。 为什么不用inject节点输入字符串？ 原因是inject节点“escapes”其使用的字符串，导致插入的返回/换行被删除。 这反过来会让接收服务器（Google）在等待丢失的返回/换行符时将其不返回响应。 因此，您可以在function节点中构建字符串。即使对经验丰富的Node-RED程序员来说这也算是“我靠这也许”之一，所以总是阅读节点的信息窗格，以确保您了解任何限制和禁止。

您可以在以下网址找到此流程的Node RED信息:

[https://raw.githubusercontent.com/SenseTecnic/nrguideflows/master/lesson3/3-8\_tcp.json](https://www.google.com/url?q=https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson3/3-8_tcp.json&sa=D&usg=AFQjCNFeOCz4I89JaGM8O7hiQeWHeyTE1g)

## 总结

在本讲座中，您已经看到一系列使用Node-RED中可用的处理和通信节点的小例子。 正如你所看到的，将现实世界输入的基本流连接在一起，进行一些处理，如简单的数据分析和返回结果是非常简单快捷的。

在这些示例中，您只是做了很少或没有编码，但仍然能够构建相当复杂的程序 ——这就是Node-RED的强大功能。

下一个讲座是对简易Node-RED中可用的基本节点的集合整理以及FRED服务提供的扩展库的快速汇总。 您可以阅读讲座来了解默认功能，您也可以将讲座作为参考，并使用它来查找本讲座系列中使用的每个节点的示例。