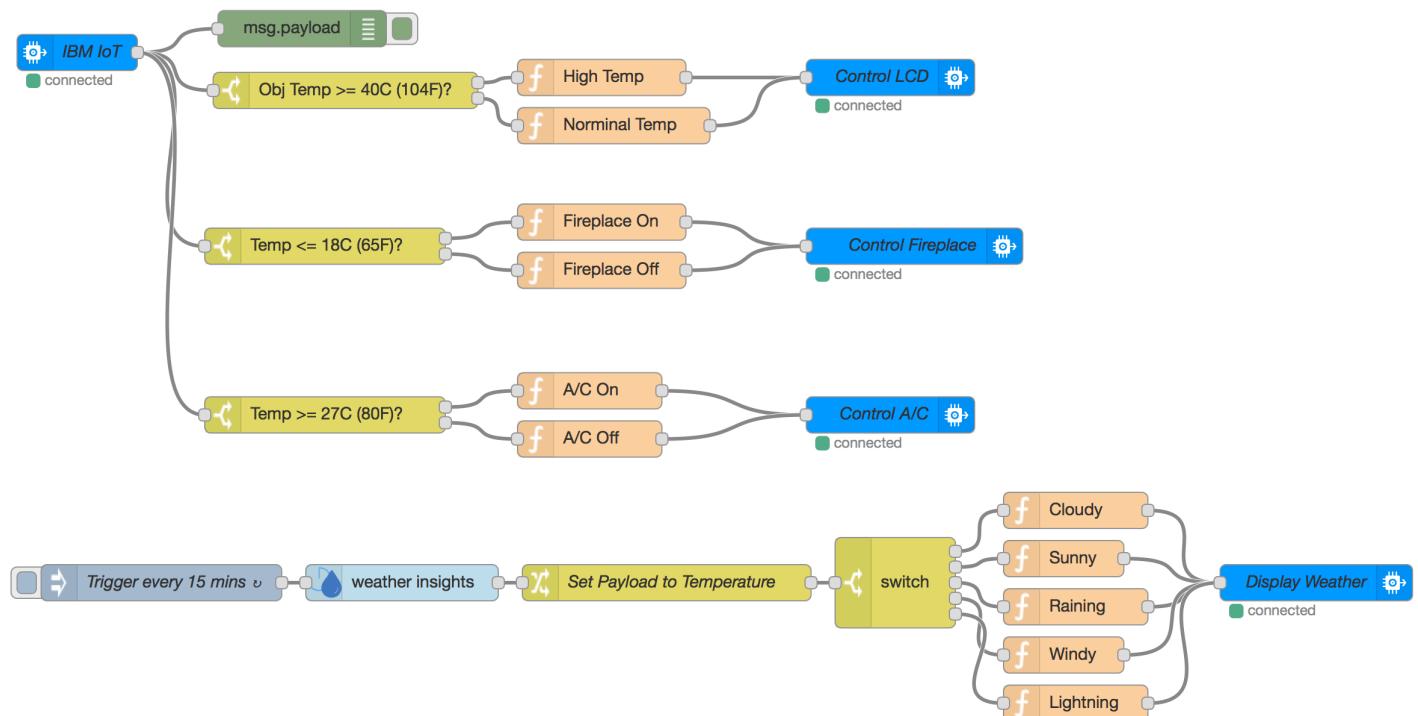
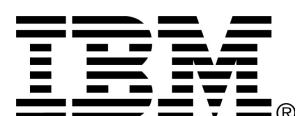


Watson Internet of Things Platform – Home Simulator

Author: JeanCarl Bisson | jbisson@us.ibm.com | [@dothewww](https://twitter.com/dothewww)



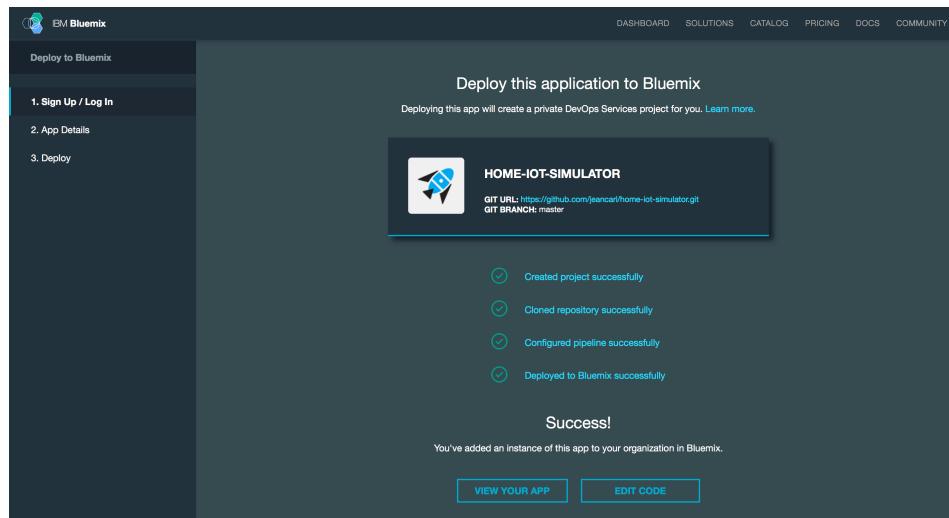
A digital copy of this lab and completed flows can be found at:
<http://ibm.biz/home-iot>



Setting up an IoT application in IBM Bluemix

In this lab, we will deploy two applications in IBM Bluemix. The first will be a simulator which contains environment sensors, including temperature, humidity, and object temperature, and several devices that will be commanded in different scenarios. With access to real hardware and sensors, such as a Raspberry Pi or the Intel Edison with Seeed Grove sensors, for example, this lab can be expanded and used in the real world environment.

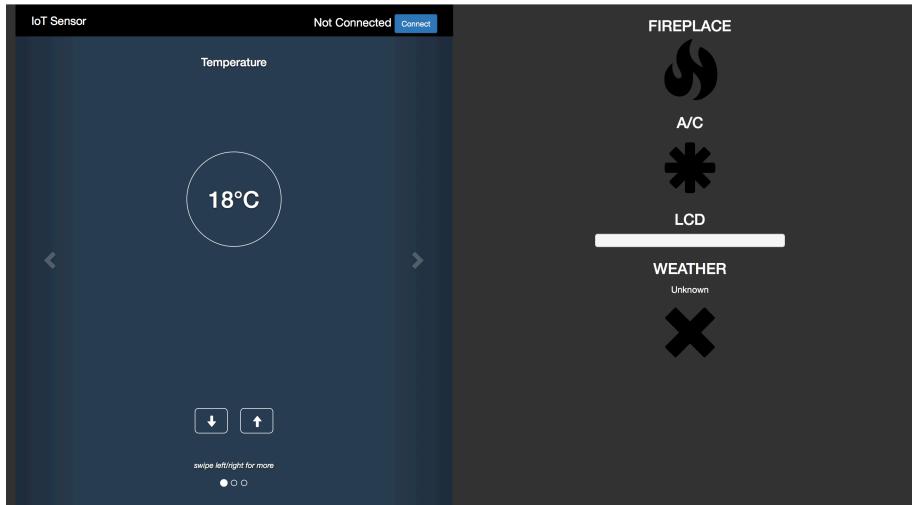
1. Deploy the simulator by visiting the following URL:
ibm.biz/home-iot-simulator
2. Select an application name, a Bluemix region, organization, and space to deploy the simulator application. Once deployed, click on the **View Your App** button.



3. This application contains several simulated things that can be commanded via the Watson IoT Platform. There are three input sensors: temperature, humidity, and object temperature. You can increase and decrease the values by clicking on the up and down arrows. Later on, we'll simulate an unsafe temperature by increasing these values above a threshold.

On the right side are several “things” we’ll control, depending on the conditions:

- a fireplace that can be turned on and off. The color of the fire can be set.
- an air conditioner that can be turned on and off.
- an LCD screen that can display messages. The text color and background color can also be set. A real world device might be a digital display on a refrigerator, a security system, or a thermostat, for example.
- a Weather indicator. This thing has a weather icon and a temperature representing the current weather condition at a location.

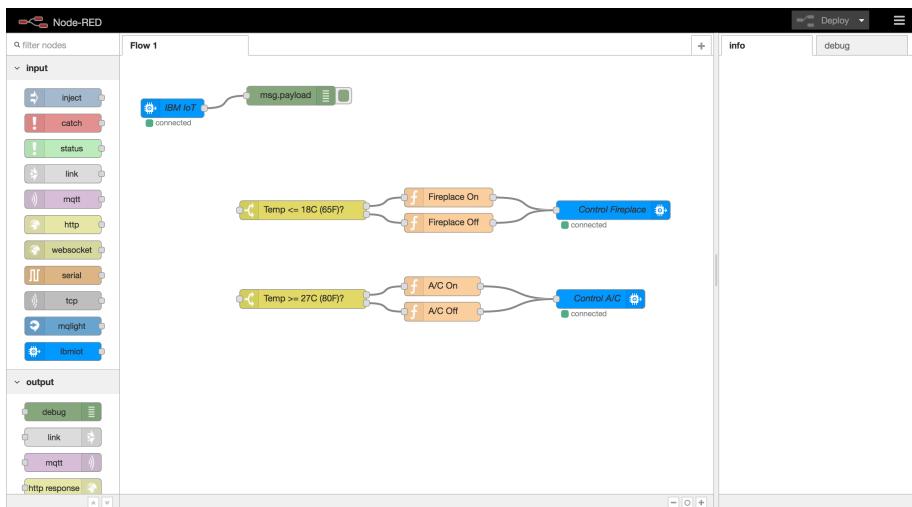


As you proceed through this lab, imagine other things that could be connected to your application and how these things can interact with each other.

Some things might have little to no processing power to keep costs down. We will create a Node-RED application in IBM Bluemix to control these devices. Node-RED is a visual interface of drag and drop nodes, written in the Node.js runtime. Node-RED makes it easy to connect together logic that takes various actions based on events.

4. Deploy the Node-RED application by visiting the following URL:
ibm.biz/home-iot-application
5. When the application is finished, click on **View Your App** at the bottom of the page. Click on the red **Go to your Node-RED flow editor**.

On the left side is a list of nodes, each with a specified behavior. Some take inputs (grey knob of the left), some have outputs (grey knob on the right) and some have multiple grey knobs (called processing nodes). Connect the grey knob from one node to the next, in a left-to-right fashion to create what are called flows. On the right side are the info tab, which displays documentation on how to use a node when selected, and the debug tab which displays messages passed into debug nodes.



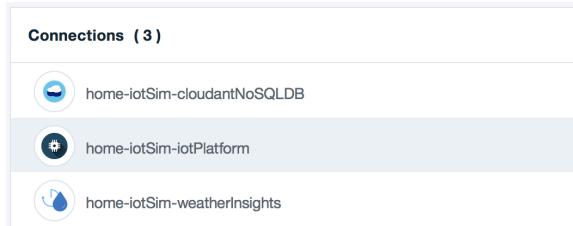
TIP!

Keep the simulator and Node-RED editor open in separate browser tabs. You will be switching between the Node-RED editor and the simulator throughout this lab.

Create a Device in Watson IBM IoT Platform

The Watson IoT Platform authenticates and manages things that connect to the Cloud. You can connect nearly anything that has Internet Connectivity and can communicate to the MQTT broker. In this section, we will create a device type and five devices that the simulator will use to connect to the Watson IoT Platform.

1. We'll start by accessing the Watson IoT Platform dashboard. In the IBM Bluemix dashboard, select your Node-RED application (the second application you deployed).
2. Under the **Connections** section, click on the **Internet of Things Platform (home-iotSim-iotPlatform)** service.

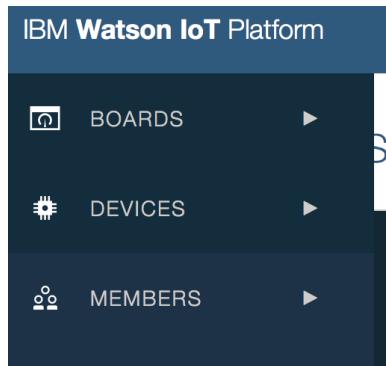


3. In the left column, click on **Launch Dashboard**.

The dashboard has a header with 'Manage' (underlined), 'Plan', and 'Connections'. The main content area has three columns:

- Connect your devices**: Includes a 'Twinkly' icon, a description about using recipes to add devices, and a note about sample connection recipes. A 'Launch dashboard' button is at the bottom.
- Analyze your data**: Includes a 'Chart' icon, a description about triggers and alerts, and a note about finding tutorials. A 'Find out more' button is at the bottom.
- Learn how to extend your app**: Includes an 'A' icon with a 'Not Available' note, a description about extending the app using other Bluemix services, and a note about available services. It shows icons for Twilio (Third Party), Cloudant NoSQL DB (IBM), Dash DB (IBM), Geospatial Analytics (IBM), Time Series Database (IBM), and IBM Analytics for Hadoop (IBM). A 'Find out how' button is at the bottom.

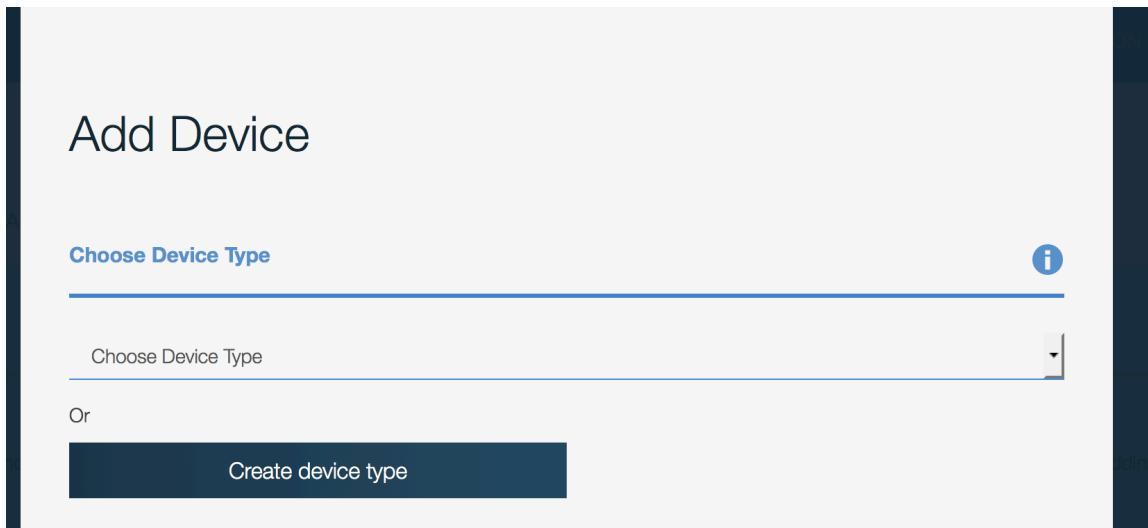
4. The Watson IoT Platform dashboard is where you can work with any registered devices that your organization has. Let's create a device type. Click on **Devices** in the left toolbar.



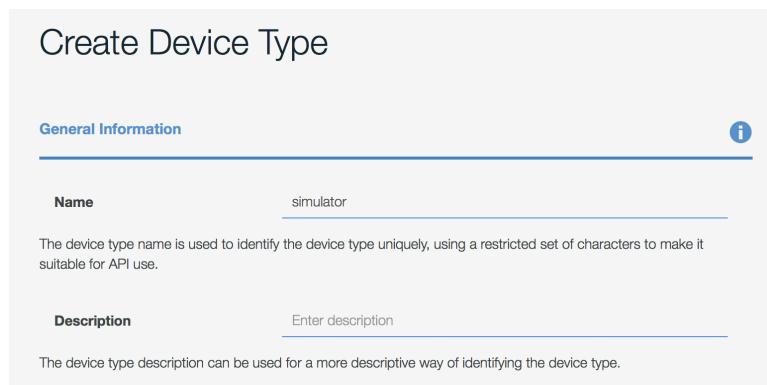
5. Click on the **Add Device** button on the right side.



6. Click on **Create device type**.



7. Select **Create device type** again.
8. A device type is a basically a label for a group of devices of the same type. Enter a device type name of `simulator`. Click on **Next** three times and **Create**.



9. You will return back to the list of device types. Your new device type should be listed in the drop down. Click on **Next** to create a specific device of the device type simulator.

Add Device

Choose Device Type

simulator

Or

Create device type

10. Enter a Device ID environment. This will create a device of type simulator, identified by the ID environment. Click **Next** twice until you get to the **Security** section.

Add Device

Device Info

Device ID

SVCC

+ Additional fields

11. Each device has a security token that the device must present to authenticate with the Watson IoT Platform. If you leave this blank, a randomized token will be generated. For simplicity, choose a token you can remember. Click **Next** and **Add** on the following screens.

Add Device

Security

You have two options:

Auto-generated authentication token

Allow the service to generate an authentication token for you. The token will be 18 characters long and will contain a mix of alphanumeric characters and symbols. The token will be returned to you at the end of the registration process.

Self-provided authentication token

Provide your own authentication token for this device. The token must be between 8 and 36 characters long, and should contain a mix of lower and upper case letters, numbers, and symbols (hyphen, underscore, and period are permitted). The token should be free of repetition, dictionary words, user names, and other predefined sequences.

Provide a token (optional)

qwerty123

Authentication tokens are encrypted before we store them.
We are not able to recover lost authentication tokens. Ensure you make a note of the authentication token after clicking Add.

12. The summary of credentials is displayed on the last screen. Copy these values as the Authentication Token cannot be retrieved again.

The screenshot shows the 'Device environment' page. At the top, it says 'Device' and has a 'Refresh' button. Below that is a section titled 'Your Device Credentials' with an information icon. A note states: 'You have registered your device to the organization. To get it connected, you need to add these credentials to your device. Once you've added these, you should see the messages sent from your device in the 'Sensor Information' section on this page.' Below this is a table showing device details:

Organization ID	d84zsm
Device Type	simulator
Device ID	environment
Authentication Method	token
Authentication Token	qwerty123

13. Repeat steps #9 to #12 to create devices with IDs of:

- fireplace
- airconditioner
- lcd
- weather

The device list in the Watson IoT Platform dashboard should look like the one shown below:

The screenshot shows the 'Devices' page in the Watson IoT Platform. The top navigation bar includes links for QUICKSTART, SERVICE STATUS, DOCUMENTATION, and BLOG, along with a user profile for jblisson@us.ibm.com. The main content area is titled 'Devices' and shows a table of device details. The columns are: Device ID, Device Type, Class ID, Date Added, and Location. There are also icons for refresh, add device, delete, search, and filter. The table lists five devices:

Device ID	Device Type	Class ID	Date Added	Location
environment	simulator	Device	Sep 30, 2016 4:43:33 PM	
fireplace	simulator	Device	Sep 30, 2016 4:44:18 PM	
airconditioner	simulator	Device	Sep 30, 2016 4:44:40 PM	
lcd	simulator	Device	Sep 30, 2016 4:44:59 PM	
weather	simulator	Device	Sep 30, 2016 4:45:15 PM	

TIP!

Keep the Watson IoT Platform dashboard open. In the next section you will look at the sensor data coming in.

Connect Simulator

Now that we have the devices registered, we'll connect the simulator to use these devices.

1. Return to the simulator application and click on the **Connect** button above the Environment sensors.



2. In the Watson IoT Platform dashboard, click on the device with the ID `environment` in the list of devices.

Devices

[Browse](#) | [Diagnose](#) | [Action](#) | [Device Types](#) | [Manage Schemas](#)

<input type="checkbox"/>	Device ID	Device Type	Class ID	Date Added
Results 1-5 of 5				
<input type="checkbox"/>	⚠ environment	simulator	Device	Sep 30, 2016 4:43:33 PM

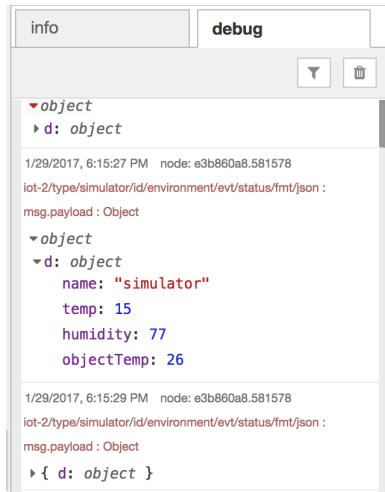
3. You can see the events that are being emitted by the simulator under the **Recent Events** section.

Recent Events		
Event	Format	Time Received
status	json	Sep 30, 2016 4:48:45 PM
status	json	Sep 30, 2016 4:48:46 PM
status	json	Sep 30, 2016 4:48:47 PM
status	json	Sep 30, 2016 4:48:48 PM
status	json	Sep 30, 2016 4:48:49 PM
status	json	Sep 30, 2016 4:48:50 PM
status	json	Sep 30, 2016 4:48:51 PM
status	json	Sep 30, 2016 4:48:52 PM

4. You can also see the sensor values that are being emitted by the simulator under the **Sensor Information** section.

Sensor Information			
Event	Datapoint	Value	Time Received
status	d.name	simulator	Sep 30, 2016 4:49:16 PM
status	d.temp	16	Sep 30, 2016 4:49:16 PM
status	d.humidity	77	Sep 30, 2016 4:49:16 PM
status	d.objectTemp	24	Sep 30, 2016 4:49:16 PM

5. Switch to the Node-RED application. The  node has been configured to subscribe to device events from this device. Select the **Debug** tab on the right side of the editor. When the simulator sends a device event to the Watson IoT Platform, the Node-RED application receives these events. The debug node displays the event in the Debug tab. We'll use these values in the next section.



```
info debug
object
d: object
1/29/2017, 6:15:27 PM node: e3b860a8.581578
iot-2/type/simulator/id/environment/evt/status/fmt/json :
msg.payload : Object
object
d: object
name: "simulator"
temp: 15
humidity: 77
objectTemp: 26
1/29/2017, 6:15:29 PM node: e3b860a8.581578
iot-2/type/simulator/id/environment/evt/status/fmt/json :
msg.payload : Object
{ d: object }
```

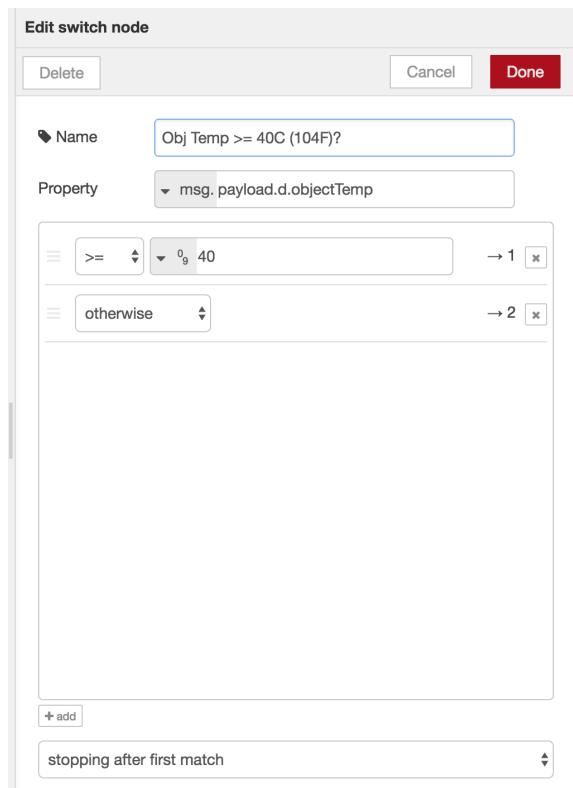
TIP!

Keep the simulator and Node-RED editor open in separate browser tabs. You will be switching between the Node-RED editor and the simulator throughout this lab.

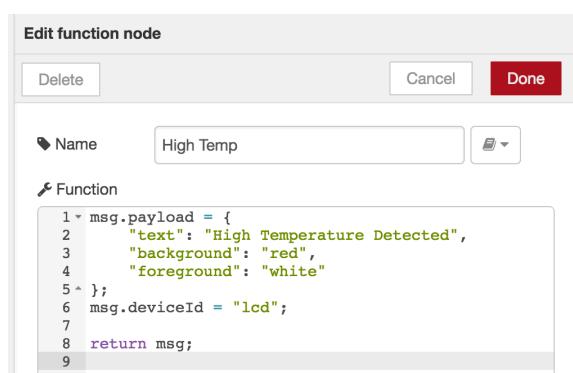
Display a Warning On LCD Screen

In this section, we'll use the simulated LCD screen. An LCD screen can be used to show a notification to someone nearby the device. For example, imagine this LCD screen is placed in the front display of a connected refrigerator, warning of a high temperature which could spoil food.

1. Add a  node as shown below.



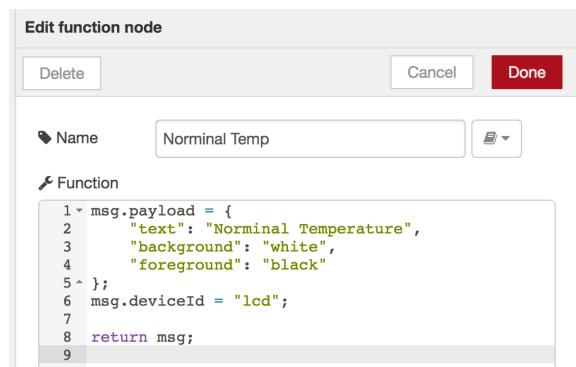
2. For the case when the temperature is too high, add a  node that constructs a message for the LCD screen we will command.



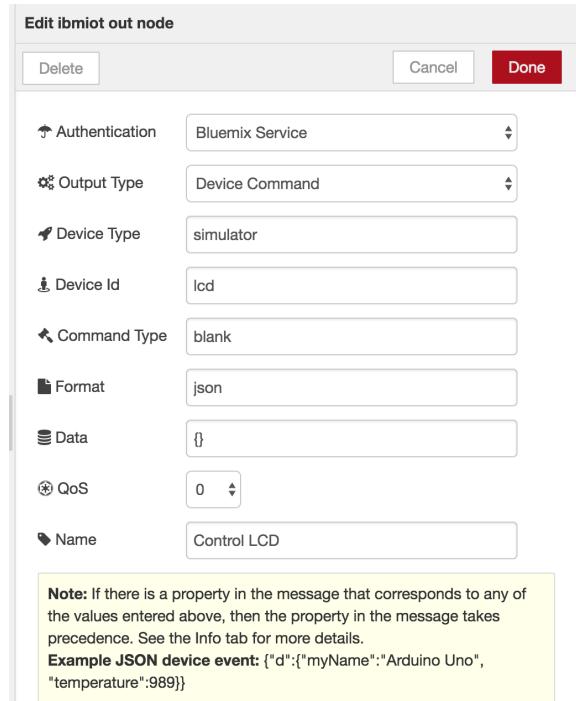
The LCD in the simulator responds to an IoT command in the following format:

```
{  
    "text": "High Temperature Detected",  
    "background": "red",  
    "foreground": "white"  
}  
  
text  
    the text can be a multiline message (lines can be separated by a new-line character, \n).  
background  
    the color of the background (a hexadecimal value like #000000, or an HTML named color)  
foreground  
    the color of the text (a hexadecimal value like #000000, or an HTML named color)
```

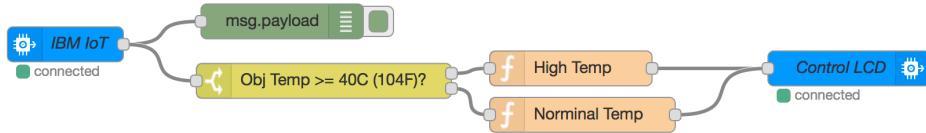
3. For the case when the temperature is normal, add a  function node that constructs a nominal message for the LCD screen we will command.



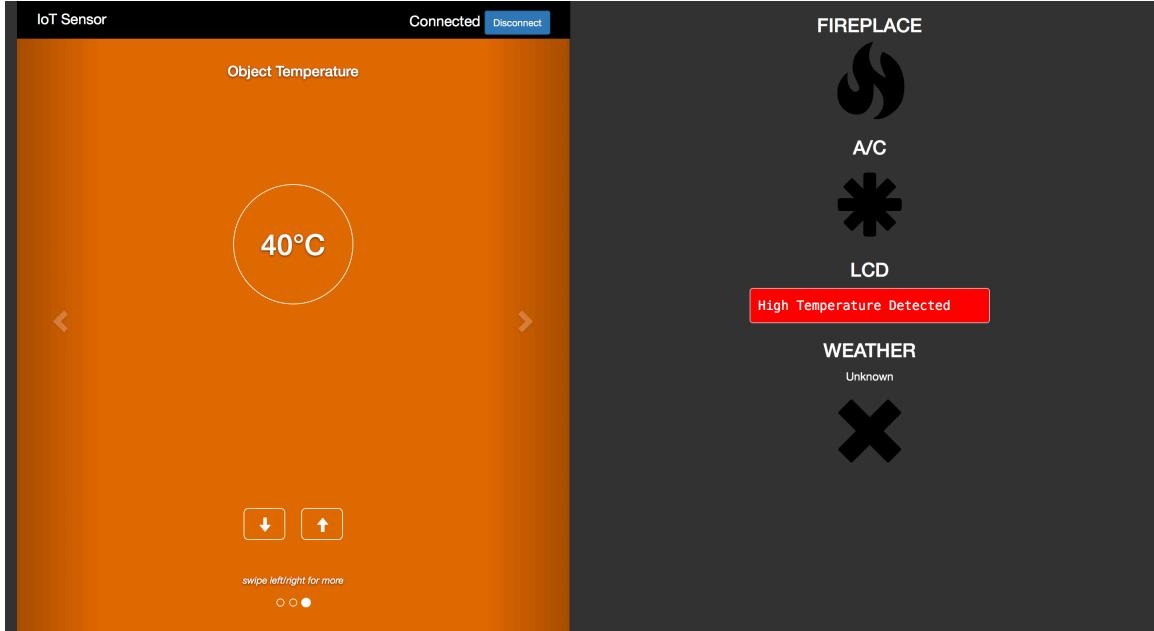
4. Send this command via a  node to the device ID lcd as shown below.



5. Connect the nodes together as shown below.



6. Click on Deploy in the top-right corner to save and deploy your changes.
7. Return to the simulator and increase the object temperature (the third environment sensor) to a value greater than or equal to 40°C. The LCD screen changes and displays a warning.

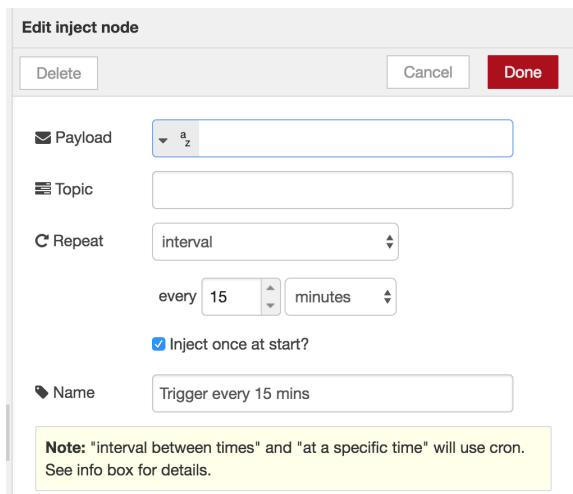


Weather Company Data for IBM Bluemix

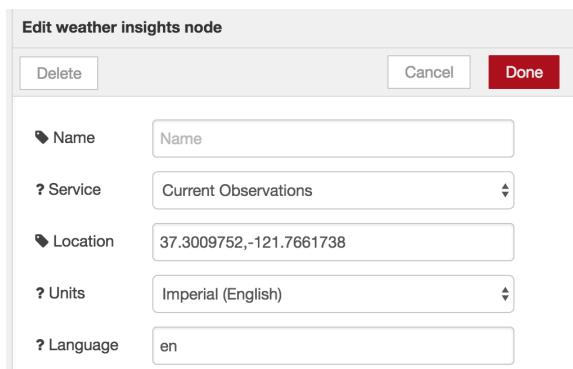
The Weather Company Data for IBM Bluemix API provides access to a variety of weather data for over two million locations worldwide. In this section, we'll retrieve the weather observation for a specified location and command the simulator to display the weather condition.

Using weather could be useful when deciding if our application should turn on an air conditioner or if it would be more efficient to open a window or an air vent.

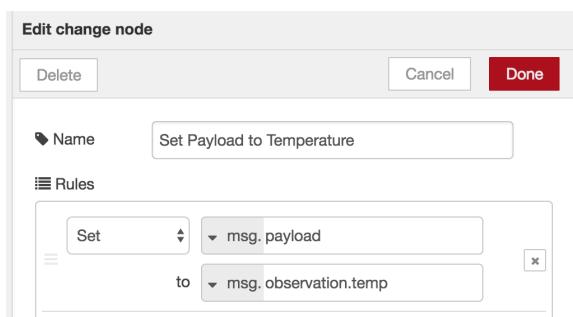
1. Add a  node as shown below. This node will trigger this flow every fifteen minutes.

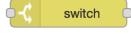


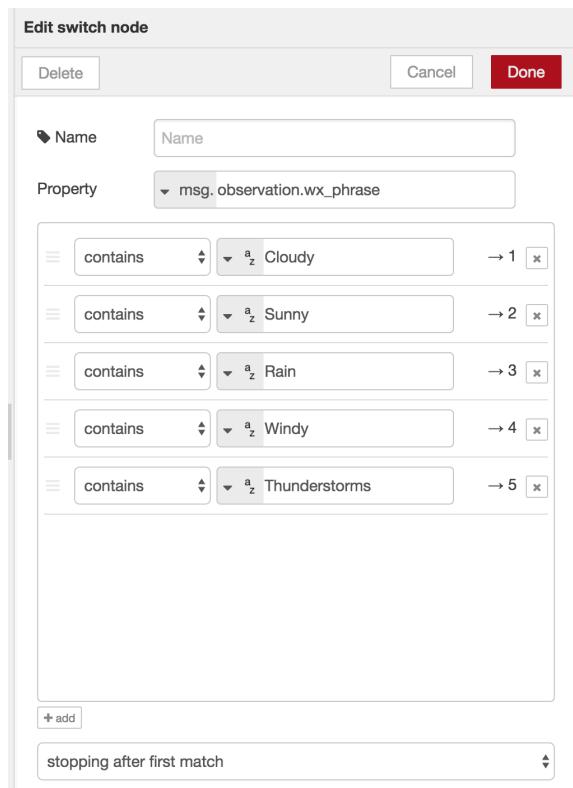
2. Add a  node as shown below.



3. Add a  node to set the msg.payload to the temperature value from the weather observation.



4. Add a  node to test the weather phrase for cloudy, sunny, raining, windy, or lightning.



The screenshot shows the 'Edit switch node' dialog. At the top, there are 'Delete', 'Cancel', and 'Done' buttons. Below that, a 'Name' field is set to 'Name'. Under the 'Property' dropdown, 'msg.observation.wx_phrase' is selected. The main area contains five 'contains' conditions, each with an 'a_z' sort option and a numbered output arrow (1 through 5). The conditions are: 'Cloudy' (arrow 1), 'Sunny' (arrow 2), 'Rain' (arrow 3), 'Windy' (arrow 4), and 'Thunderstorms' (arrow 5). A '+ add' button is at the bottom left, and a 'stopping after first match' dropdown is at the bottom right.

5. The Weather indicator in the simulator responds to an IoT command in the following format:

```
{
  "text": "Cloudy "+msg.payload+"*F",
  "condition": "cloudy",
  "color": "#C0C0C0"
}

text
  the weather condition to be displayed above weather icon.
condition
  one of the following conditions that are associated with weather icons in the simulator:
    cloudy, sunny, raining, windy, lightning
color
  the color of the weather icon (a hexadecimal value like #000000, or an HTML named color)
```

6. Add five  nodes to set values for the command we'll send to the simulator.



The screenshot shows the 'Edit function node' dialog for 'Cloudy'. It has 'Delete', 'Cancel', and 'Done' buttons. The 'Name' field is 'Cloudy'. Under the 'Function' tab, a code editor shows the following JavaScript:

```
1 msg.payload = {
2   "text": msg.observation.wx_phrase+" "+msg.payload+"*F",
3   "condition": "cloudy",
4   "color": "#C0C0C0"
5 };
6
7 return msg;
8 |
```



The screenshot shows the 'Edit function node' dialog for 'Sunny'. It has 'Delete', 'Cancel', and 'Done' buttons. The 'Name' field is 'Sunny'. Under the 'Function' tab, a code editor shows the following JavaScript:

```
1 msg.payload = {
2   "text": msg.observation.wx_phrase+" "+msg.payload+"*F",
3   "condition": "sunny",
4   "color": "yellow"
5 };
6
7 return msg;
8 |
```

Edit function node

Delete Cancel Done

Name: Raining

Function:

```

1 - msg.payload = {
2   "text": msg.observation.wx_phrase+" "+msg.payload+"*F",
3   "condition": "raining",
4   "color": "blue"
5 };
6
7 return msg;
8

```

Edit function node

Delete Cancel Done

Name: Windy

Function:

```

1 - msg.payload = {
2   "text": msg.observation.wx_phrase+" "+msg.payload+"*F",
3   "condition": "windy",
4   "color": "lightblue"
5 };
6
7 return msg;
8

```

Edit function node

Delete Cancel Done

Name: Lightning

Function:

```

1 - msg.payload = {
2   "text": msg.observation.wx_phrase+" "+msg.payload+"*F",
3   "condition": "lightning",
4   "color": "yellow"
5 };
6
7 return msg;
8

```

7. Add a  node as shown below. This node will command the weather device to display the current observation.

Edit ibmiot out node

Delete Cancel Done

Authentication: Bluemix Service

Output Type: Device Command

Device Type: simulator

Device Id: weather

Command Type: update

Format: json

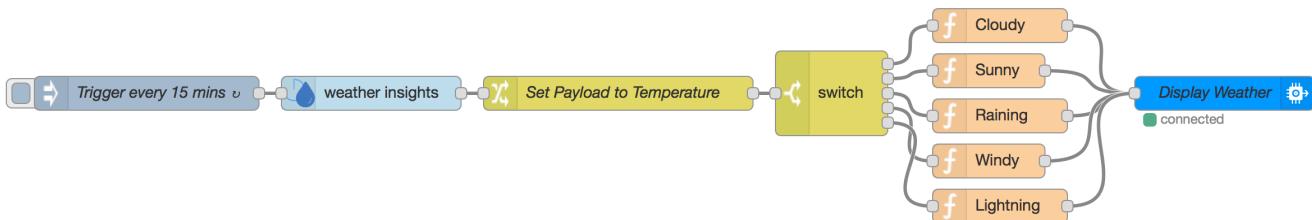
Data: {}

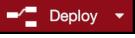
QoS: 0

Name: Display Weather

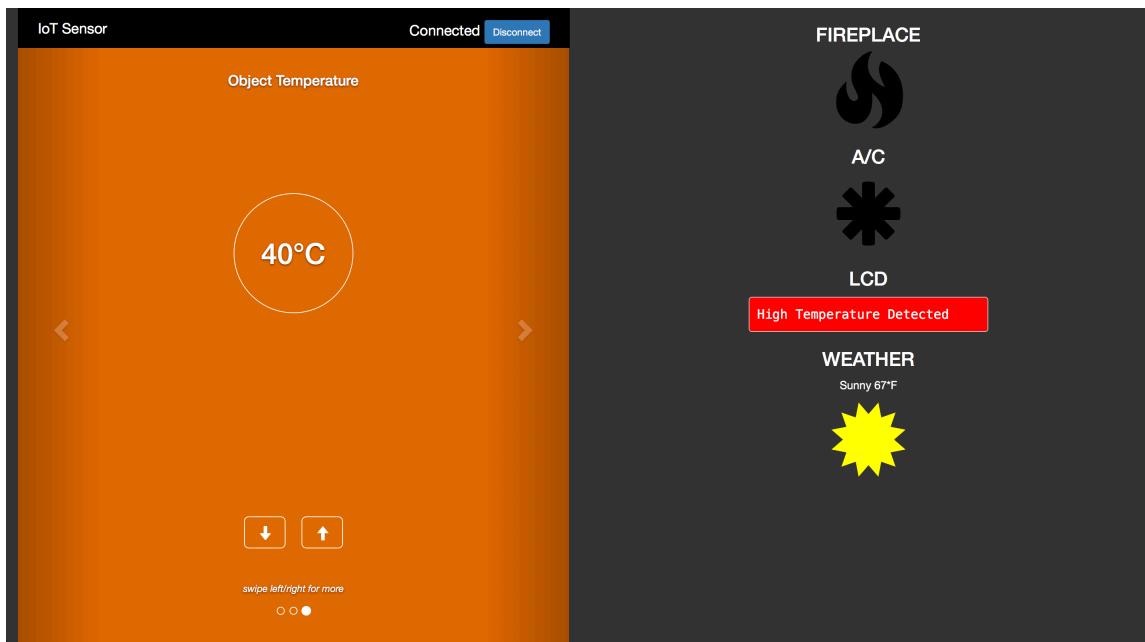
Note: If there is a property in the message that corresponds to any of the values entered above, then the property in the message takes precedence. See the Info tab for more details.
Example JSON device event: {"d":{"myName":"Arduino Uno", "temperature":989}}

8. Connect the nodes together as shown below.



9. Click on  in the top-right corner to save and deploy your changes.

10. Every 15 minutes, the inject node will trigger the flow to get the updated weather observation and send an IoT command to the simulator to update the weather indicator.



Controlling Fireplace / Air Conditioner

With an ecosystem of connected Internet of Things, it becomes possible to have things interact with each other automatically and make decisions on what state the things should be in. In this section, we will use the temperature value from the environment sensors to control the fireplace when it is colder, and control the air conditioner when it is hotter.

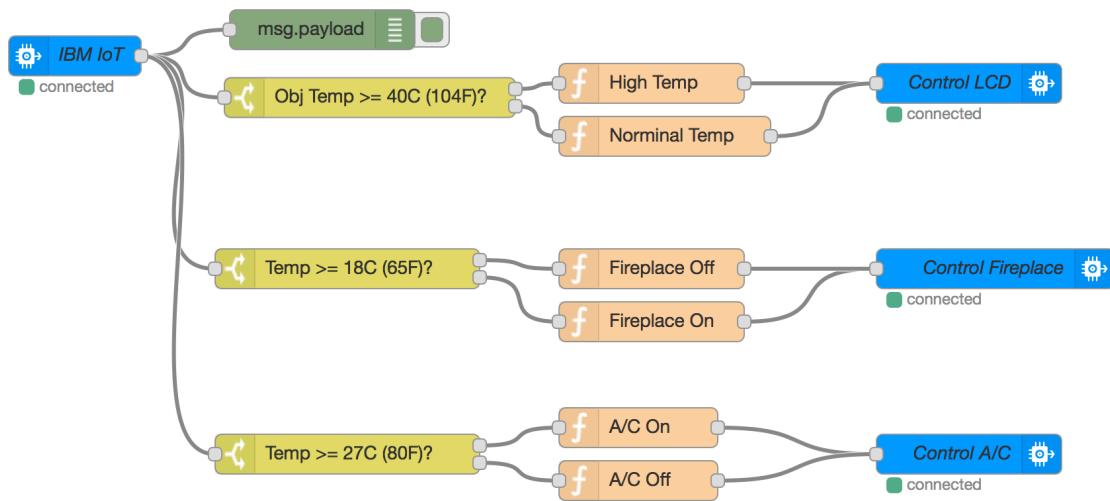
The fireplace in the simulator responds to an IoT command in the following format:

```
Command: turnOn or turnOff  
Payload:  
{  
    "color": "#FFFFFF"  
}  
  
color  
the color (a hexadecimal value like #000000, or an HTML named color like yellow) of the flame (yes, it's a fancy fireplace), if the command is turnOn. Ignored if command is turnOff.
```

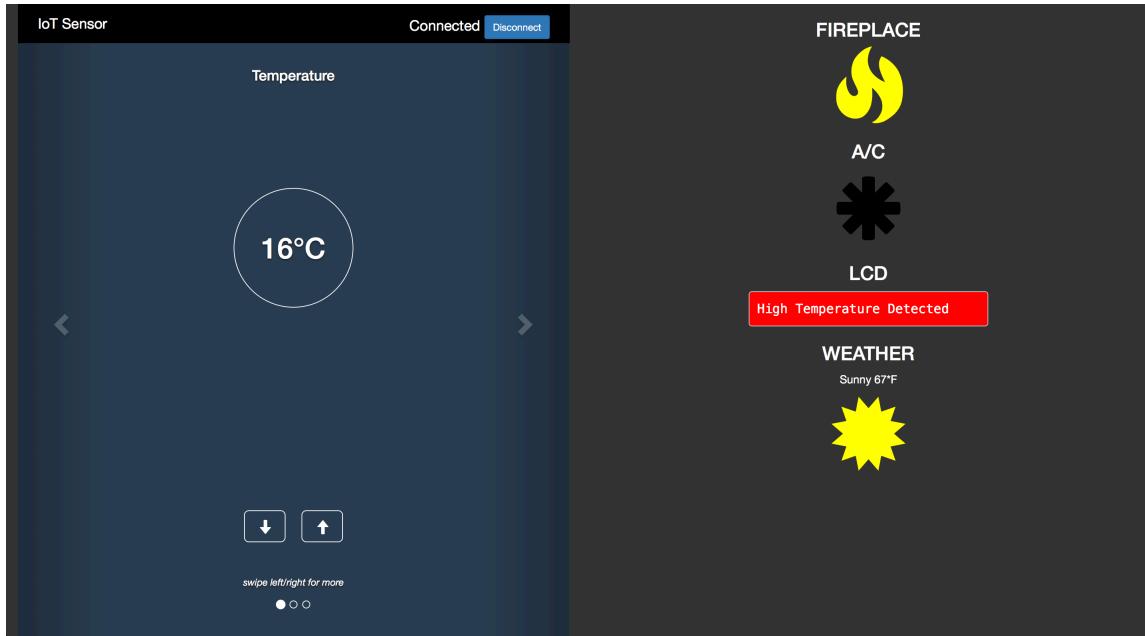
The air conditioner in the simulator responds to an IoT command in the following format:

```
Command: turnOn or turnOff  
Payload: (empty object)  
{  
}
```

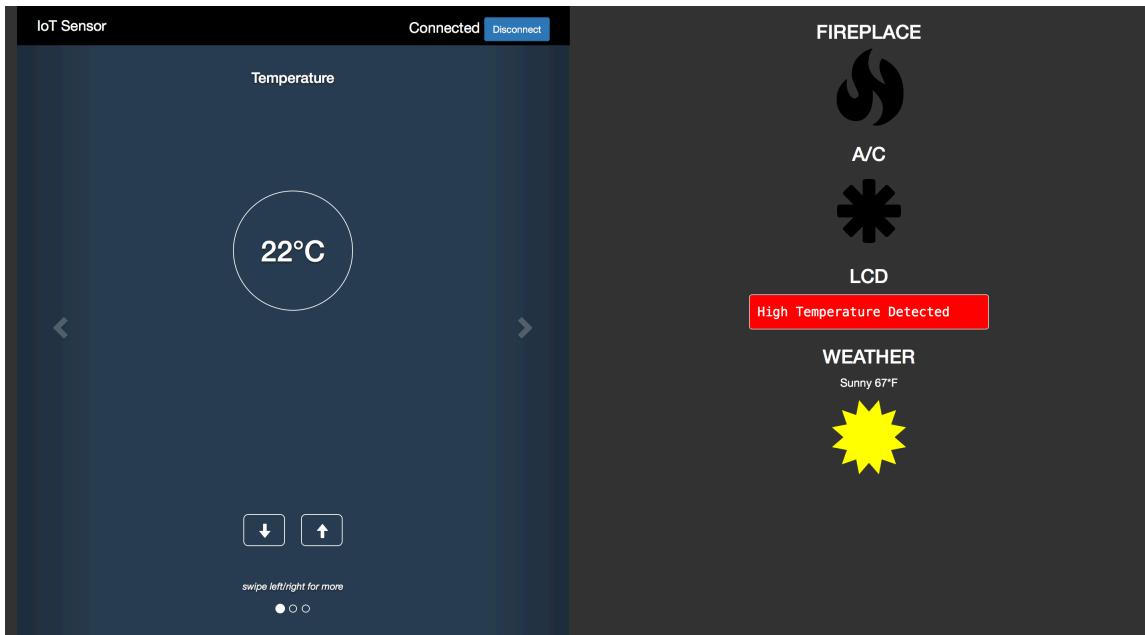
1. The two flows are already in the Node-RED editor. Connect the nodes together as shown below.



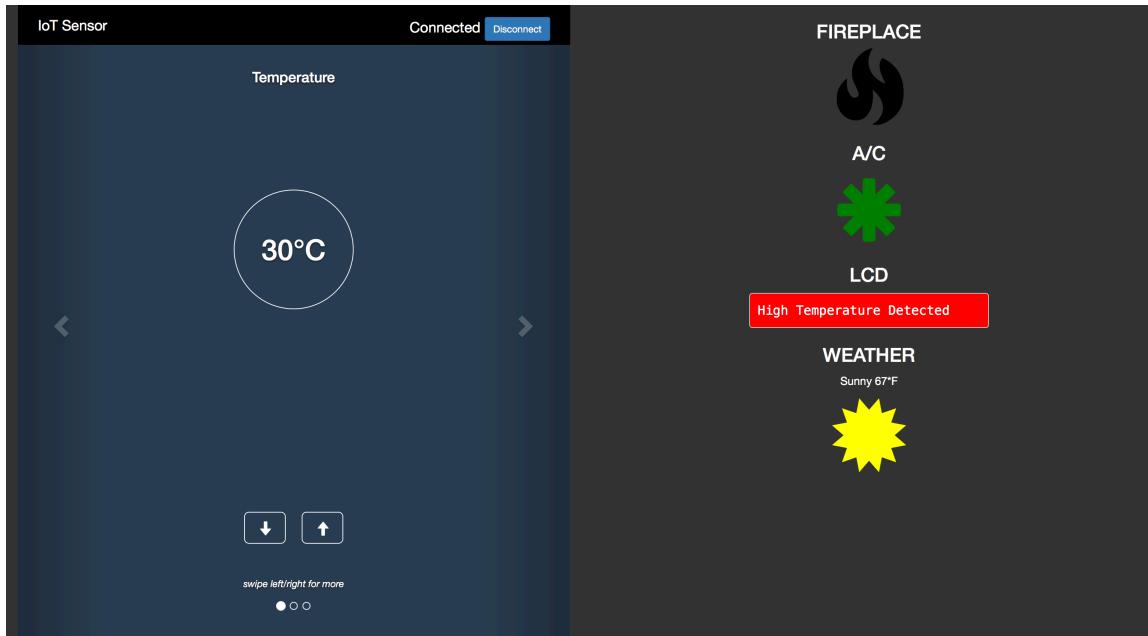
2. Click on  Deploy in the top-right corner to save and deploy your changes.
3. Return to the simulator and decrease the temperature to 16°C. Brrr! The fireplace turns on.



4. Increase the temperature to 22°C. The fireplace turns off.



5. Increase the temperature to 30°C. The air conditioner turns on.



Additional Resources

Using a simulator can help reduce costs in developing IoT solutions and help to refine the concept. Try commanding the things based on new factors. There is also a humidity environment sensor in the simulator. What can you control with high or low humidity? The Weather Company Data API makes available a variety of other weather conditions, such as wind speed, and the sunrise and sunset times. Turn on the fireplace at sunset.

If you're interested in connecting real sensors, disconnect the simulator and connect real sensors to the Watson IoT Platform. The Node-RED application can be used and modified without the simulator. For more information, please refer to the lab **Tracking Temperature with Intel Edison and Grove Sensors**, available at ibm.biz/lab-temperature-sensor-with-intel-edison and ibm.com/iot.

Twilio and Twitter

You can also use Twilio and Twitter to send notifications. Node-RED has nodes to interact with these services. Please refer to the lab **Tracking Temperature with Intel Edison and Grove Sensors**, available at ibm.biz/lab-temperature-sensor-with-intel-edison.

DeveloperWorks Recipes

Interested in more IoT. Check out IBM DeveloperWorks recipes that show tutorials on using Internet of Things in the real world. Please visit developer.ibm.com/recipes for more information.

Watson

Watson isn't just IoT. Analyze unstructured data from other sources using the suite of Cognitive APIs. Please visit ibm.com/watson and github.com/watson-developer-cloud for more information.

Node-RED

Node-RED is an open-source community project. Check out the other nodes and flows the community has developed and join the community to help make Node-RED even better. Please visit nodered.org for more information.