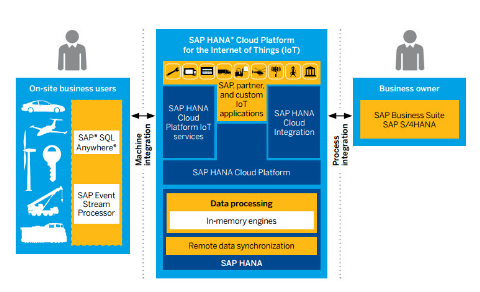
IoT1C07 – Create a Dashboard (climate module) – on Premise

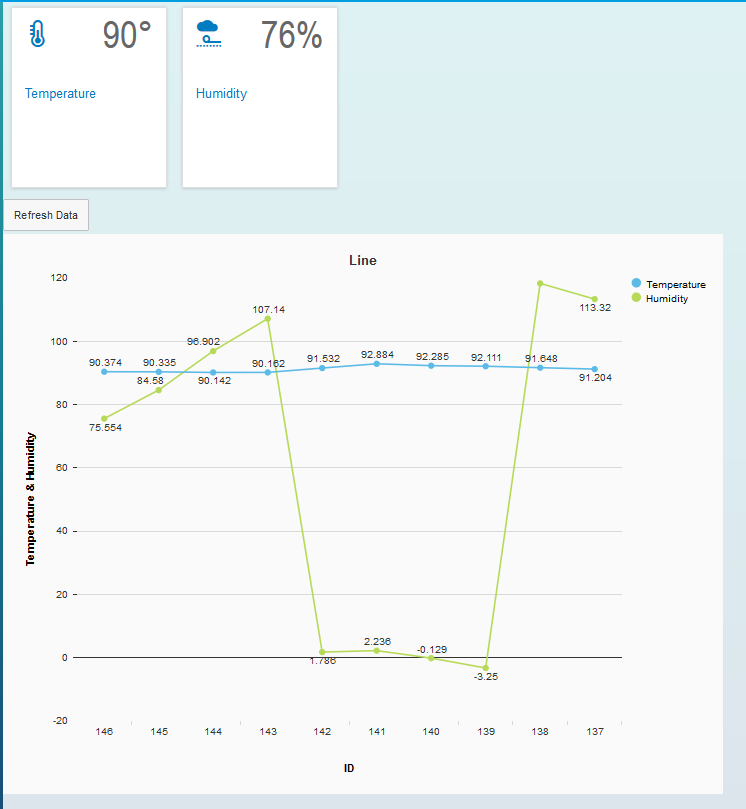
|  |  |
| --- | --- |
| **Product**  HANA Cloud Platform IoT Services  **Level**  Undergraduate/Graduate  Beginner  **Focus**  HANA Cloud Platform  **Author** Ross Hightower | MOTIVATION  In this case you will create a dashboard to show the data from your IoT climate device.  **PREREQUISITES**  IoT1Cxx - |



# 

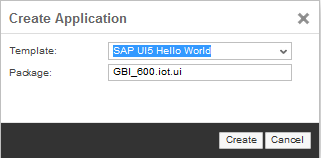
# Display the Data

In this case you will create a small UI5 application to display the data transmitted by the Tessel. The application looks like the image below. The two tiles at the top show the most recent temperature and humidity and the chart shows the last 10 values. The application refreshes every 3 seconds but the refresh can be toggled off using the button.



## Create the Application

Right-click the iot package and select **Create Application.** Select **SAP UI5 Hello World** in the Template field and enter **.ui** after the iot in the Package field



### index.html

Replace the code in the index.html with the code shown below:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta charset="utf-8">  <title>Walkthrough</title>  <script id="sap-ui-bootstrap"  type="text/javascript"  src="/sap/ui5/1/resources/sap-ui-core.js"  data-sap-ui-theme="sap\_bluecrystal"  data-sap-ui-libs="sap.m"  data-sap-ui-xx-bindingSyntax="complex"  data-sap-ui-resourceroots='{ "iot": "./" }'></script>    <script>  sap.ui.getCore().attachInit(function () {  sap.ui.xmlview({  viewName : "iot.view.App"  }).placeAt("content");  });  </script>  </head>  <body class="sapUiBody" id="content">  </body>  </html> |

Listing 1

This code loads the core UI5 libraries and initializes a view called App.view.xml located in the view package.

### App.view.xml

Right-click the ui package and create a new package called **view.** Create a file in the view package called **App.view.xml** and paste the code below into it.

|  |
| --- |
| <mvc:View  controllerName="iot.view.App"  xmlns="sap.m"  xmlns:mvc="sap.ui.core.mvc"  xmlns:viz="sap.viz.ui5.controls">  <HBox>  <StandardTile id="tempTile" icon="sap-icon://temperature"  title="Temperature"  number="{iot>/d/results/0/TEMPERATURE}" />  <StandardTile id="humTile" icon="sap-icon://weather-proofing"  title="Humidity"  number="{iot>/d/results/0/HUMIDITY}" />  </HBox>  </mvc:View> |

Listing 2

This code implements the two tiles. The number property of the tiles are bound to the TEMPERATURE and HUMIDITY properties of the first record ( /0/ ) in the iot model. We will instantiate the iot model in the controller.

### App.controller.js

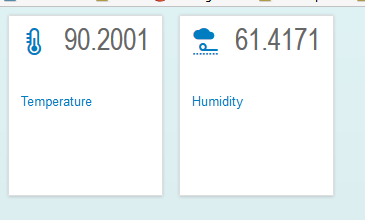
Right-click the ui package and create a new package called **controller.** Create a file in the controller package called **App.controller.js** and paste the code below into it.

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller",  "sap/m/MessageToast",  "sap/ui/model/json/JSONModel"  ], function (Controller, MessageToast, JSONModel) {  "use strict";  return Controller.extend("iot.view.App", {  onInit : function () {    this.cModel = new JSONModel();  this.cModel.loadData("**http://db1.hana2.ucc.uwm.edu:8002/gbi-student-366/iot/data/iot.xsodata**/DATA?$top=10&$format=json&$orderby=ID desc");  this.getView().setModel(this.cModel, "iot");    }  });  }); |

Listing 3

This code implements the onInit function which will execute when the view is instantiated. The code initializes a JSON model and provides the URI of your service. Replace the highlighted portion so that it matches your URI. The URI includes some parameters that limit the return the top 10 records ($top=10), formats the data as JSON ($format=json) and sorts the data by ID in descending order. The result is that only the last record is returned.

Open the index.html file and click the run icon.



Next, we will clean up the number formatting.

### App.view.xml

Change the code in the App.view.xml file to look like the code shown below. The highlighted portions show the changes.

|  |
| --- |
| <mvc:View  controllerName="iot.view.App"  xmlns="sap.m"  xmlns:mvc="sap.ui.core.mvc"  xmlns:viz="sap.viz.ui5.controls">  <HBox>  <StandardTile id="tempTile" press="doIt" icon="sap-icon://temperature"  title="Temperature"  number="{path: 'iot>/d/results/0/TEMPERATURE', formatter: '.degrees'}" />  <StandardTile id="humTile" press="doIt" icon="sap-icon://weather-proofing"  title="Humidity"  number="{path: 'iot>/d/results/0/HUMIDITY', formatter: '.humidity'}" />  </HBox></mvc:View> |

Listing 4

This code configures functions (degrees() and humidity()) which will are called before the value is displayed so that we can format the values.

### App.controller.js

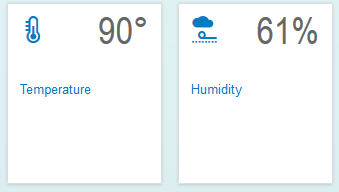
Now add the highlighted portion below to the App.controller.js file.

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller",  "sap/m/MessageToast",  "sap/ui/model/json/JSONModel"  ], function (Controller, MessageToast, JSONModel) {  "use strict";  return Controller.extend("iot.view.App", {  onInit : function () {    this.cModel = new JSONModel();  this.cModel.loadData("http://db1.hana2.ucc.uwm.edu:8002/gbi-student-366/iot/data/iot.xsodata/DATA?$top=1&$format=json&$orderby=ID desc");  this.getView().setModel(this.cModel, "iot");    } ,    **degrees: function(temp){**  **return Math.round(temp) + "\u00b0";**  **},**    **humidity: function(hum){**  **return Math.round(hum)+ "\u0025";**  **}**  });  }); |

Listing 5

IF YOU COPY ALL OF THIS CODE, REMEMBER TO UPDATE THE URL.

These functions round the values to integers and add a degrees and percent sign. Now refresh or run the application again.



## Add a Chart

Next, we’ll add a chart. The JavaScript for the chart is a bit complex but if you look at it carefully you will see what it is doing.

### App.view.xml

Make the following changes to the App.view.xml file.

|  |
| --- |
| <mvc:View  controllerName="iot.view.App"  xmlns="sap.m"  xmlns:mvc="sap.ui.core.mvc"  xmlns:viz="sap.viz.ui5.controls">  <HBox>  <StandardTile id="tempTile" press="doIt" icon="sap-icon://temperature"  title="Temperature"  number="{path: 'iot>/d/results/0/TEMPERATURE', formatter: '.degrees'}" />  <StandardTile id="humTile" press="doIt" icon="sap-icon://weather-proofing"  title="Humidity"  number="{path: 'iot>/d/results/0/HUMIDITY', formatter: '.humidity'}" />  </HBox>  <ToggleButton id="refreshButtonID" text="Refresh Data" pressed="true" press="doIt" />  <HBox>  <viz:Popover id="idPopOver"/>  <viz:VizFrame id="idVizFrameLine" height="700px" width="900px" vizType="line" uiConfig="{applicationSet:'fiori'}"> </viz:VizFrame>  </HBox>  </mvc:View> |

Listing 6

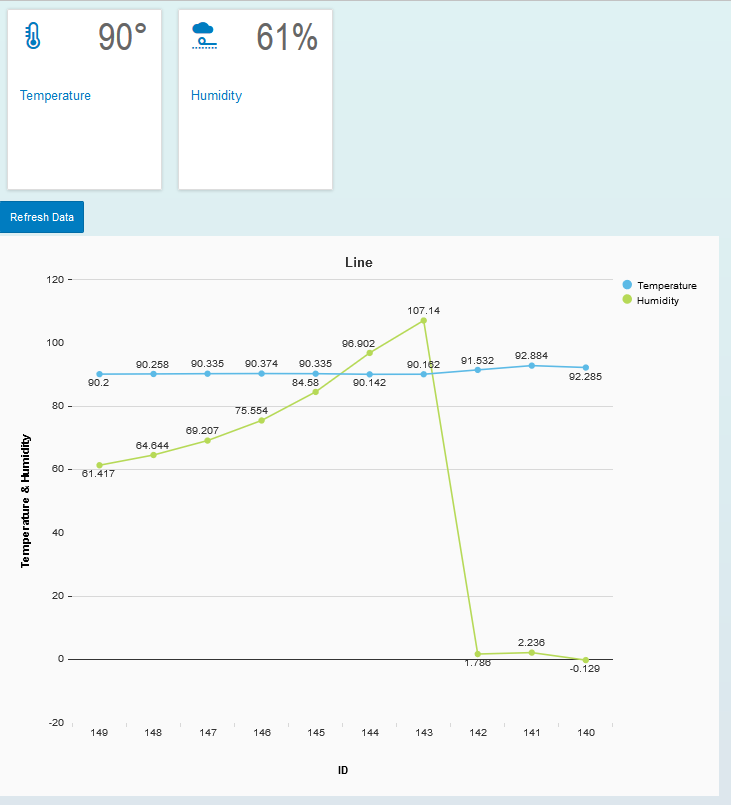
### App.controller.js

Now make the following changes to the App.controller.js file.

|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller",  "sap/m/MessageToast",  "sap/ui/model/json/JSONModel"  ], function (Controller, MessageToast, JSONModel) {  "use strict";  return Controller.extend("iot.view.App", {  onInit : function () {    this.cModel = new JSONModel();  this.cModel.loadData("http://hd3.hana.ucc.uwm.edu:8003/GBI\_600/iot/iot.xsodata/DATA?$top=10&$format=json&$orderby=ID desc");  this.getView().setModel(this.cModel, "iot");    var oVizFrame = this.getView().byId("idVizFrameLine");  var oPopOver = this.getView().byId("idPopOver");    var oDataset = new sap.viz.ui5.data.FlattenedDataset({  dimensions : [ {  name : 'ID',  value : "{ID}"  } ],  measures : [  {  name : 'Temperature',  value : '{TEMPERATURE}'  }, {  name : 'Humidity',  value : '{HUMIDITY}'  }],  data : {  path : "/d/results"  }  });    oVizFrame.setVizProperties({  plotArea : {  isFixedDataPointSize : true,  categorySize : {  desktop : {  minValue : 100  }  },  dataLabel : {visible : true},    lineStyle: {  rules: [  {  dataContext: [  {ID: "\*"}  ],  properties: {  width: 6  }  }]  }  },  legend : {  title: {visible : false}  },    title: {  visible: true,  text: 'Line'  }  });  oVizFrame.setDataset(oDataset);  oVizFrame.setModel(this.cModel);  var feedPrimaryValues = new sap.viz.ui5.controls.common.feeds.FeedItem({  'uid' : "primaryValues",  'type' : "Measure",  'values' : ["Temperature", "Humidity"]  }), feedAxisLabels = new sap.viz.ui5.controls.common.feeds.FeedItem({  'uid' : "axisLabels",  'type' : "Dimension",  'values' : ["ID"]  });  oVizFrame.addFeed(feedPrimaryValues);  oVizFrame.addFeed(feedAxisLabels);  oPopOver.connect(oVizFrame.getVizUid());    },      degrees: function(temp){  return Math.round(temp) + "\u00b0";  },    humidity: function(hum){  return Math.round(hum)+ "\u0025";  }  });  }); |

Listing 7

Refresh or run the application.



### Refresh the Data

The last change is to add the code that will cause the data to refresh automatically.

### App.controller.js

Change the code in the App.controller.js file as shown below.

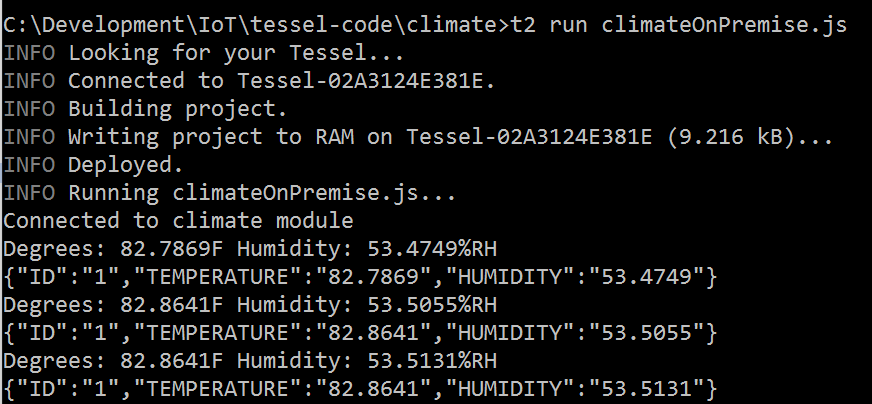
|  |
| --- |
| sap.ui.define([  "sap/ui/core/mvc/Controller",  "sap/m/MessageToast",  "sap/ui/model/json/JSONModel"  ], function (Controller, MessageToast, JSONModel) {  "use strict";  return Controller.extend("iot.view.App", {  onInit : function () {  this.refresh = true;  this.cModel = new JSONModel();  this.cModel.loadData("http://hd3.hana.ucc.uwm.edu:8003/GBI\_600/iot/iot.xsodata/DATA?$top=10&$format=json&$orderby=ID desc");  this.getView().setModel(this.cModel, "iot");    var oVizFrame = this.getView().byId("idVizFrameLine");  var oPopOver = this.getView().byId("idPopOver");    var oDataset = new sap.viz.ui5.data.FlattenedDataset({  dimensions : [ {  name : 'ID',  value : "{ID}"  } ],  measures : [  {  name : 'Temperature',  value : '{TEMPERATURE}'  }, {  name : 'Humidity',  value : '{HUMIDITY}'  }],  data : {  path : "/d/results"  }  });    oVizFrame.setVizProperties({  plotArea : {  isFixedDataPointSize : true,  categorySize : {  desktop : {  minValue : 100  }  },  dataLabel : {visible : true},    lineStyle: {  rules: [  {  dataContext: [  {ID: "\*"}  ],  properties: {  width: 6  }  }]  }  },  legend : {  title: {visible : false}  },    title: {  visible: true,  text: 'Line'  }  });  oVizFrame.setDataset(oDataset);  oVizFrame.setModel(this.cModel);  var feedPrimaryValues = new sap.viz.ui5.controls.common.feeds.FeedItem({  'uid' : "primaryValues",  'type' : "Measure",  'values' : ["Temperature", "Humidity"]  }), feedAxisLabels = new sap.viz.ui5.controls.common.feeds.FeedItem({  'uid' : "axisLabels",  'type' : "Dimension",  'values' : ["ID"]  });  oVizFrame.addFeed(feedPrimaryValues);  oVizFrame.addFeed(feedAxisLabels);  oPopOver.connect(oVizFrame.getVizUid());    var that = this;  this.refresh = true;  setInterval(function(){  if(that.refresh){  that.loadData();  }  }, 3000);  },    loadData: function(){  this.cModel.loadData("http://hd3.hana.ucc.uwm.edu:8003/GBI\_600/iot/iot.xsodata/DATA?$top=10&$format=json&$orderby=ID desc");    },    doIt: function(){  this.refresh = !this.refresh;  },    degrees: function(temp){  return Math.round(temp) + "\u00b0";  },    humidity: function(hum){  return Math.round(hum)+ "\u0025";  }  });  }); |

Listing 8

The added code will run the loadData() function every 3 seconds (3000 ms). The loadData() function simply fetches the data from the server again if the this.refresh variable is true. The doIt function toggles the this.refresh variable when the button is pressed. The variable this.refresh is initialized at the top of the onInit() function.

## Test the Climate App

If you implemented the Tessel climate module as shown in case IoT1Cxx – Setup a Tessel Device – On Premise, start the **climateOnPremise.js** program to begin loading data.



If you aren’t using a Tessel but used the node.js simulation, start deviceSim.js application.

Run the UI application and carefully cup your hands around the Tessel climate module and breathe gently on it. Watch the temperature and humidity rise.

