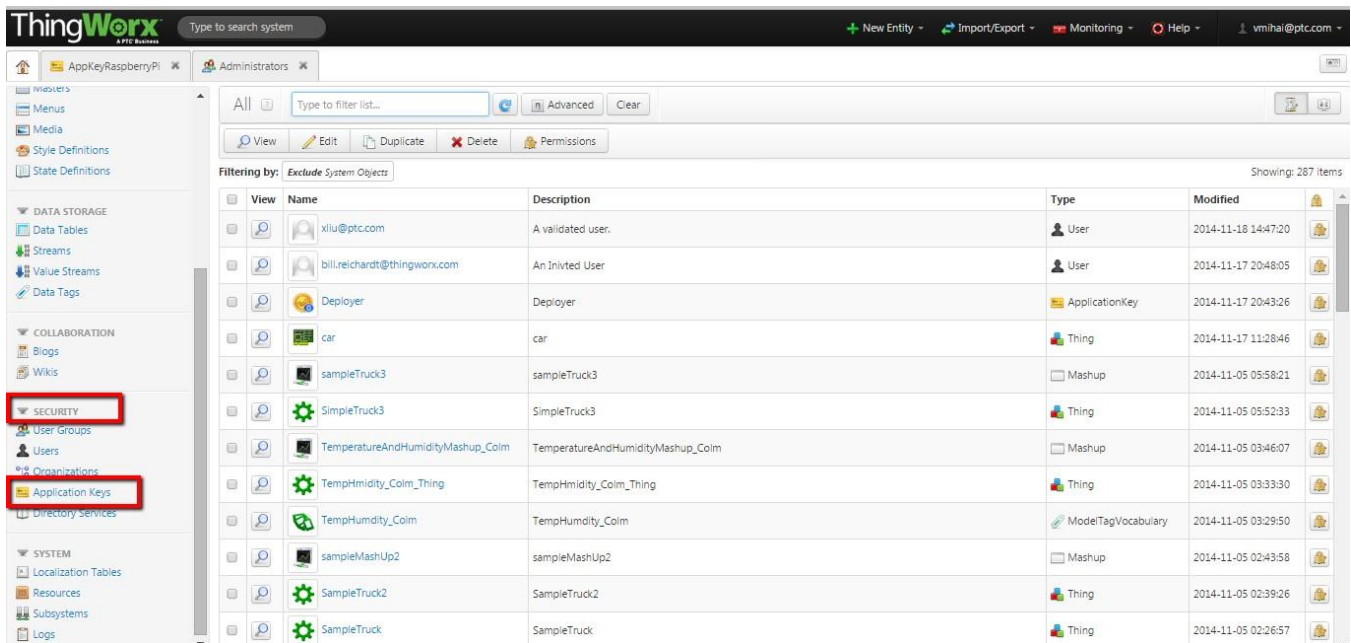
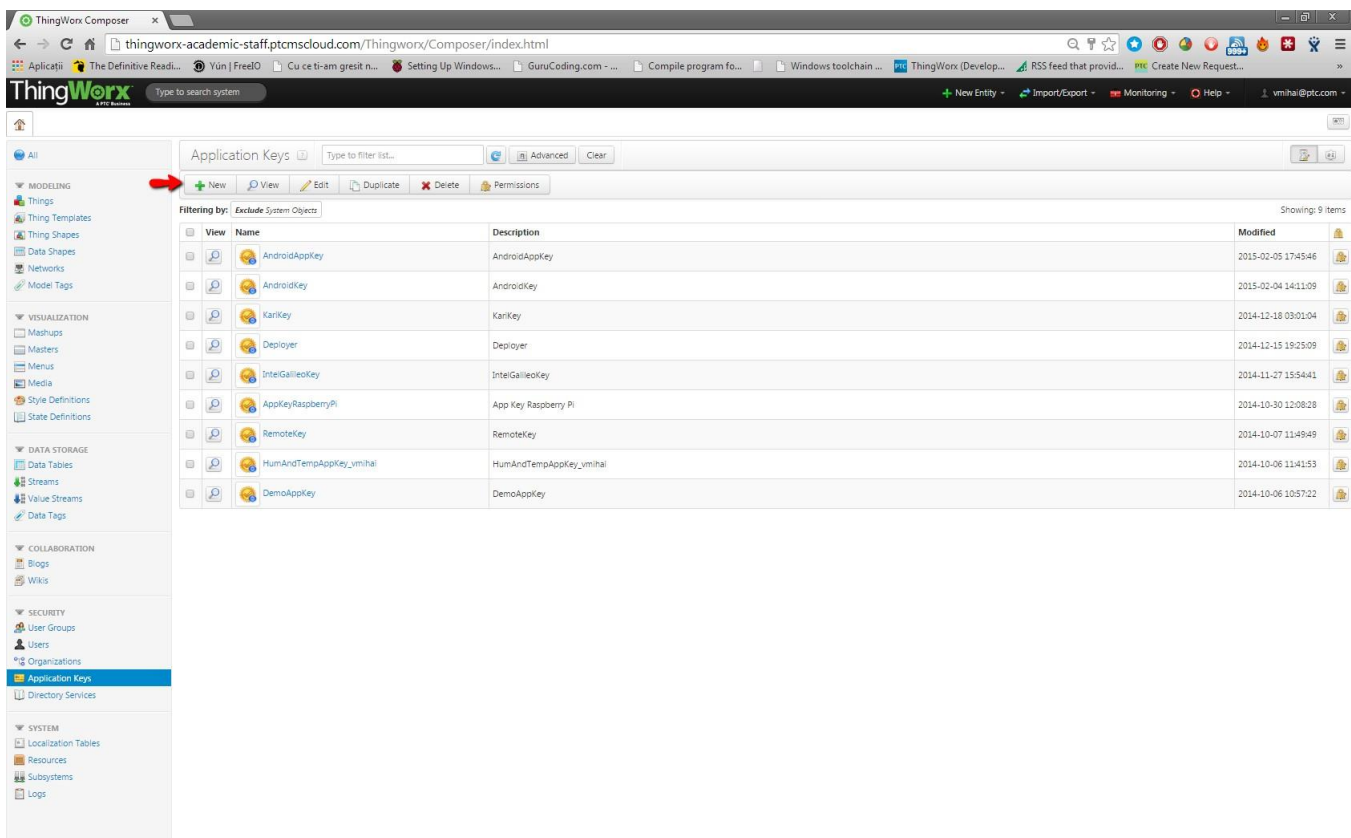


## Lab ThingWorx REST Interface Experiment

- Use the same hardware previously used for Cayenne
1. Meet ThingWorx (PowerPoint slides)
  2. Install Arduino IDE (if you have not done before).  
<https://www.arduino.cc/en/Guide/Windows>
  3. Install ESP8266 Library from the following link (if you have not done before).  
<https://github.com/esp8266/Arduino>
  4. Download this code and modify (instructions are given later).  
REST\_ESP8266\_ThingWorx\_JSON\_LED\_control.ino from the following repository:  
[https://github.com/indystar1/ESP8266-code-various/REST\\_ESP8266\\_ThingWorx](https://github.com/indystar1/ESP8266-code-various/REST_ESP8266_ThingWorx)
  5. Follow tutorial here  
<https://www.udemy.com/thingworx-fundamentals/>
  6. Create an account on <https://www.thingworxacademic.com/>
  7. Use the following composer for experiment (use same user ID and password as above)  
<https://academic.cloud.thingworx.com/Thingworx/Composer/>
  8. The following steps are very similar to the tutorial appearing here (but see below for some property names for our project that are different)  
<https://www.thingworx.com/ecosystem/academic-program/iot-projects/weather-app-arduino-uno/>
  9. Create an App key
    - a. The App key is a ThingWorx created object that is used in the authentication into the ThingWorx Server process. Application keys are used whenever one would like to interact with a ThingWorx Application outside of the actual ThingWorx Web Application. Thus an app key replaces the login credentials input that is normally required for each communication with ThingWorx.
    - b. To get this value go to your ThingWorx Composer home tab and locate Application Key in the Security section.



10. Click it and click the +New Button:



11. Enter a name, for example: ArduinoUnoKey (or ESP8266Key) and Reference field :

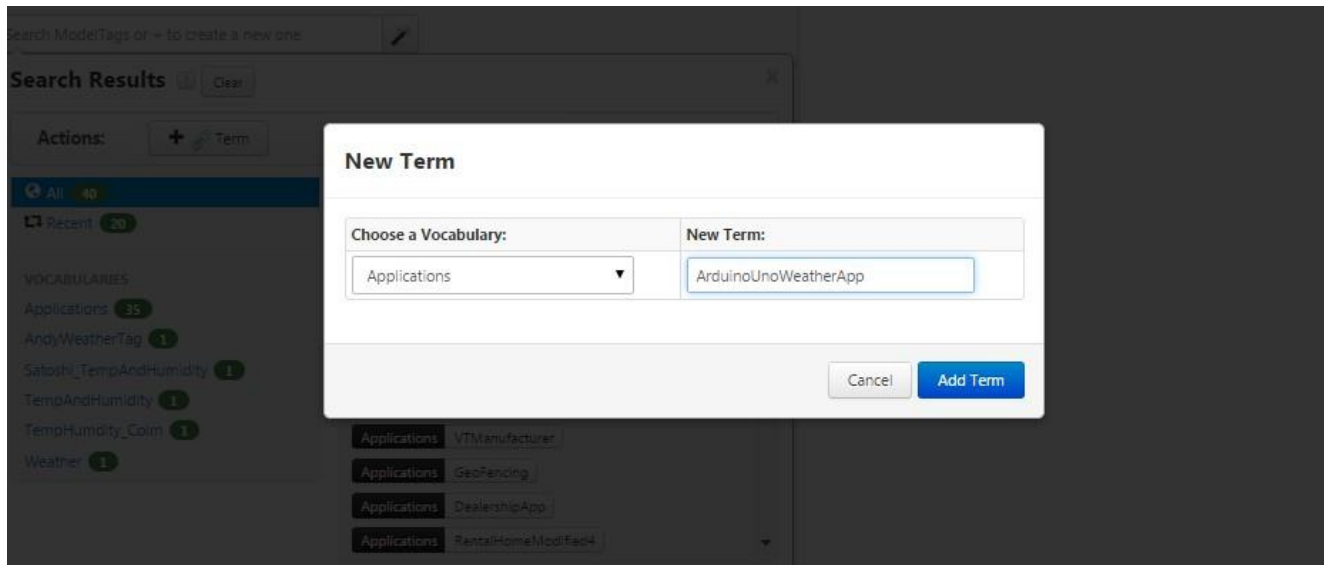
The screenshot shows the 'New Application Key' form in ThingWorx. The 'Name' field is filled with 'ArduinoUnoKey' and the 'User Name Reference' field is filled with 'ymihai@ptc.com'. Red arrows point to these fields. The form also includes fields for 'Description', 'Tags', 'IP Whitelist', 'Client Name', and 'keyId'. On the right, there are fields for 'Home Mashup', 'Avatar', and 'Last Modified Date'.

12. You can also tag it. This is useful if you plan on exporting for example all the objects you made for a project. If you tag all your Things, ThingTemplates, ThingShapes etc. with the same tag you can then filter the export based on your tag and have an xml file with all the information. To read more about tags go to Wiki page of [ThingWorx Community](#), section 03.13 Tags.

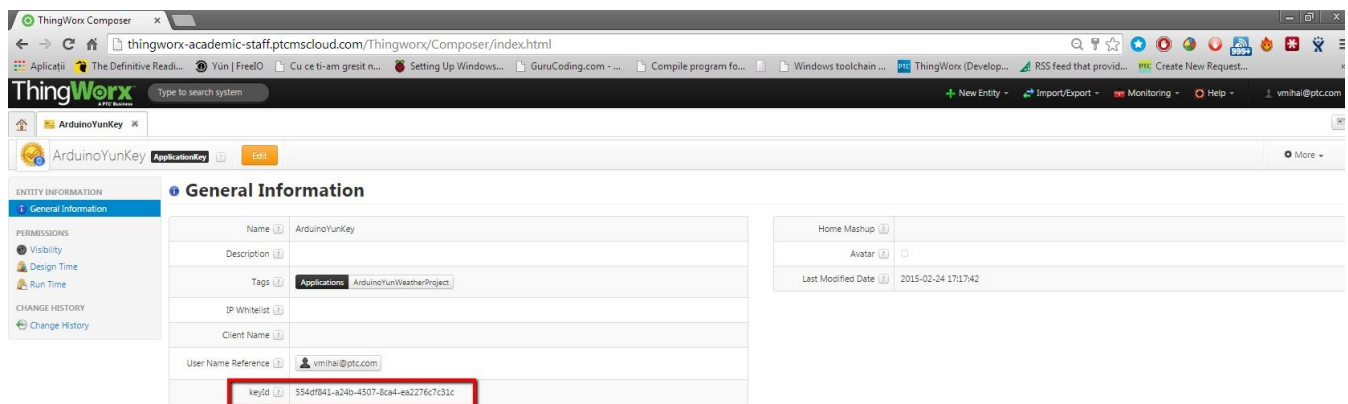
13. To create a tag, click on the wand in the tag field and select + Term

The screenshot shows the 'New Application Key' form with the 'Tags' field open. The 'Actions' dropdown shows '+ Term' selected. The 'Search Results' panel shows a list of vocabularies and applications. The 'VOCABULARIES' list includes 'Applications', 'AndyWeatherTag', 'Satosh\_TempAndHumidity', 'TempAndHumidity', 'TempHumidity\_Coim', and 'Weather'. The 'Applications' list includes 'ArduinoUno', 'ArduinoUnoWeatherProject', 'ACMEVendingCarProject', 'AcmeTractor', 'AcmeVending', 'AcmeFoods', 'VTManufacturer', 'GeoPencing', 'DealershipApp', and 'RentalHomeModFeedk'.

14. Choose the Vocabulary in which you want to create it and give the term a name. For example ArduinoUnoWeatherProject.

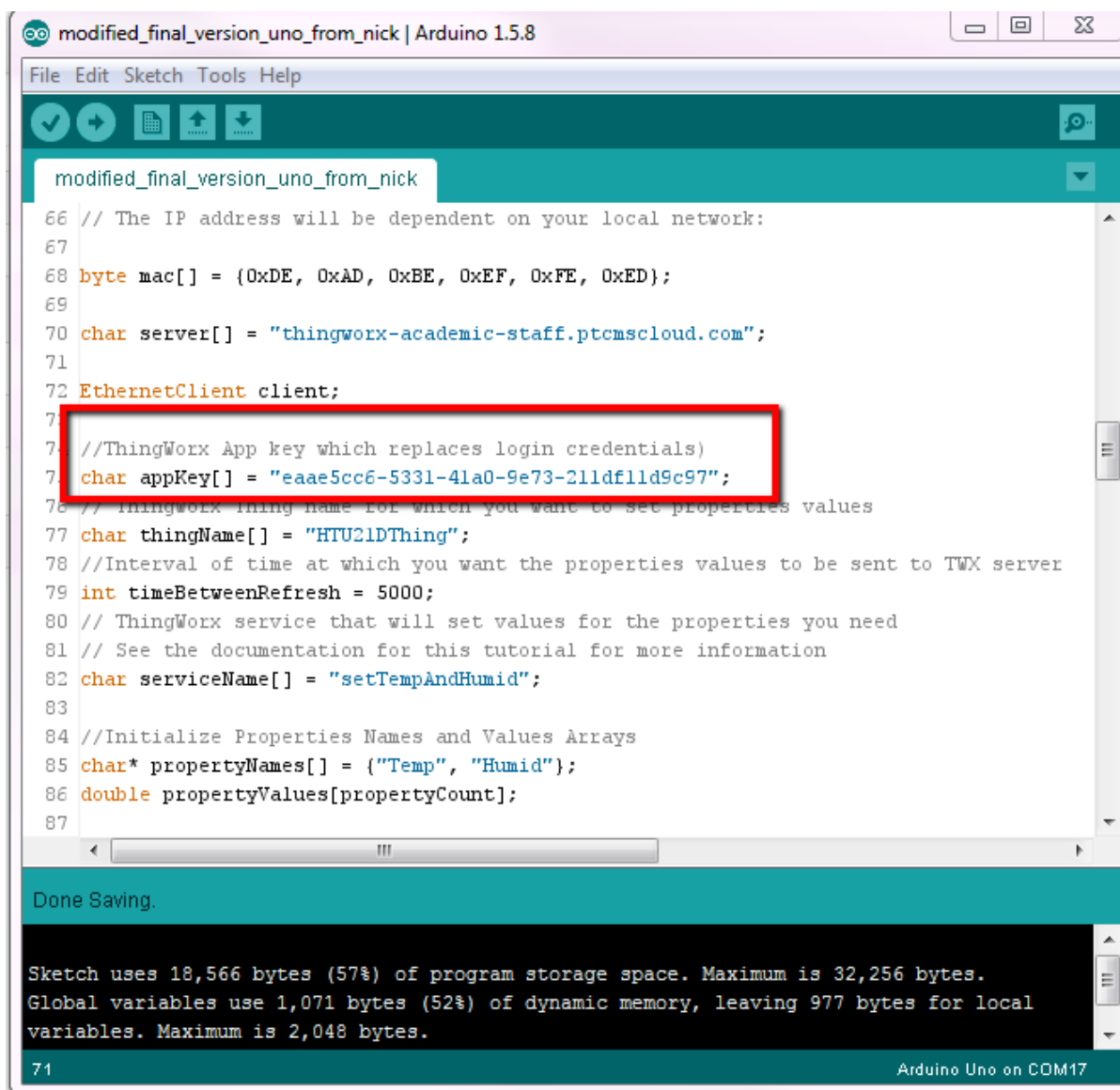


15. Next, click Add Term. Save the App key and copy the keyId



16. And paste it as value for the appKey in the your\_Arduino\_source\_code.ino file.

**Note:** When creating the Application key, make sure that in the User Name Reference field you select the same user you use to log into ThingWorx to follow this tutorial. Otherwise the data sent from the ThingWorx java SDK program won't be received by ThingWorx.



```
modified_final_version_uno_from_nick | Arduino 1.5.8
File Edit Sketch Tools Help

modified_final_version_uno_from_nick

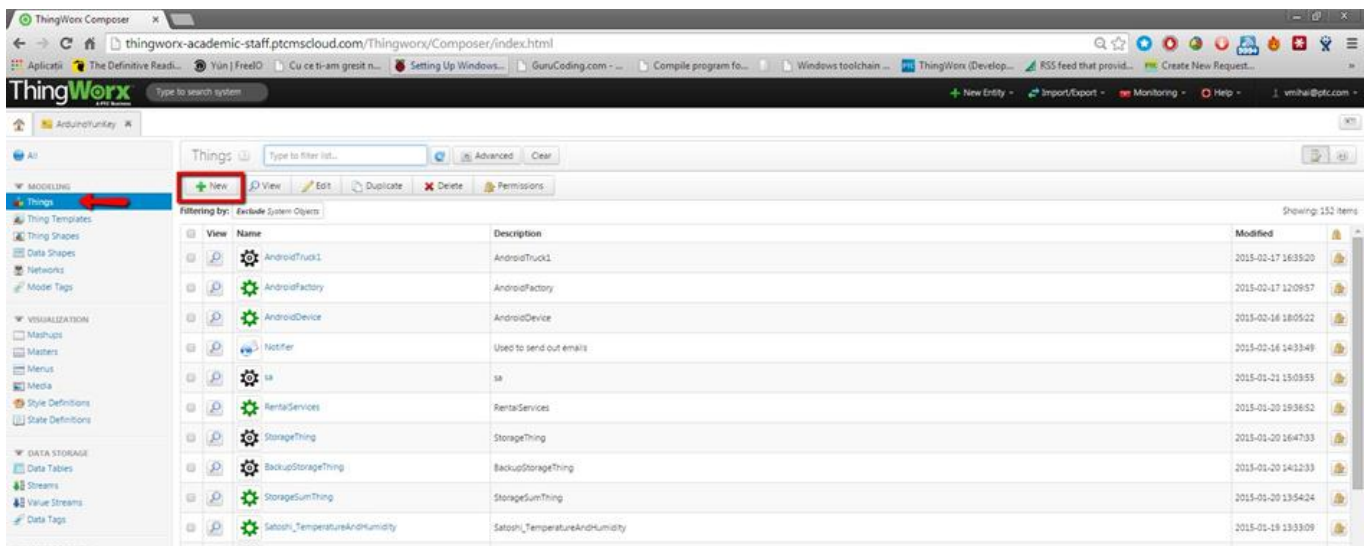
66 // The IP address will be dependent on your local network:
67
68 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
69
70 char server[] = "thingworx-academic-staff.ptcmscloud.com";
71
72 EthernetClient client;
73
74 //ThingWorx App key which replaces login credentials)
75 char appKey[] = "eaae5cc6-5331-41a0-9e73-211df11d9c97";
76 // ThingWorx thing name for which you want to set properties values
77 char thingName[] = "HTU21DThing";
78 //Interval of time at which you want the properties values to be sent to TWX server
79 int timeBetweenRefresh = 5000;
80 // ThingWorx service that will set values for the properties you need
81 // See the documentation for this tutorial for more information
82 char serviceName[] = "setTempAndHumid";
83
84 //Initialize Properties Names and Values Arrays
85 char* propertyNames[] = {"Temp", "Humid"};
86 double propertyValues[propertyCount];
87

Done Saving.

Sketch uses 18,566 bytes (57%) of program storage space. Maximum is 32,256 bytes.
Global variables use 1,071 bytes (52%) of dynamic memory, leaving 977 bytes for local
variables. Maximum is 2,048 bytes.

71 Arduino Uno on COM17
```

17. To create this Thing on your ThingWorx Platform, Go to Things and click New.

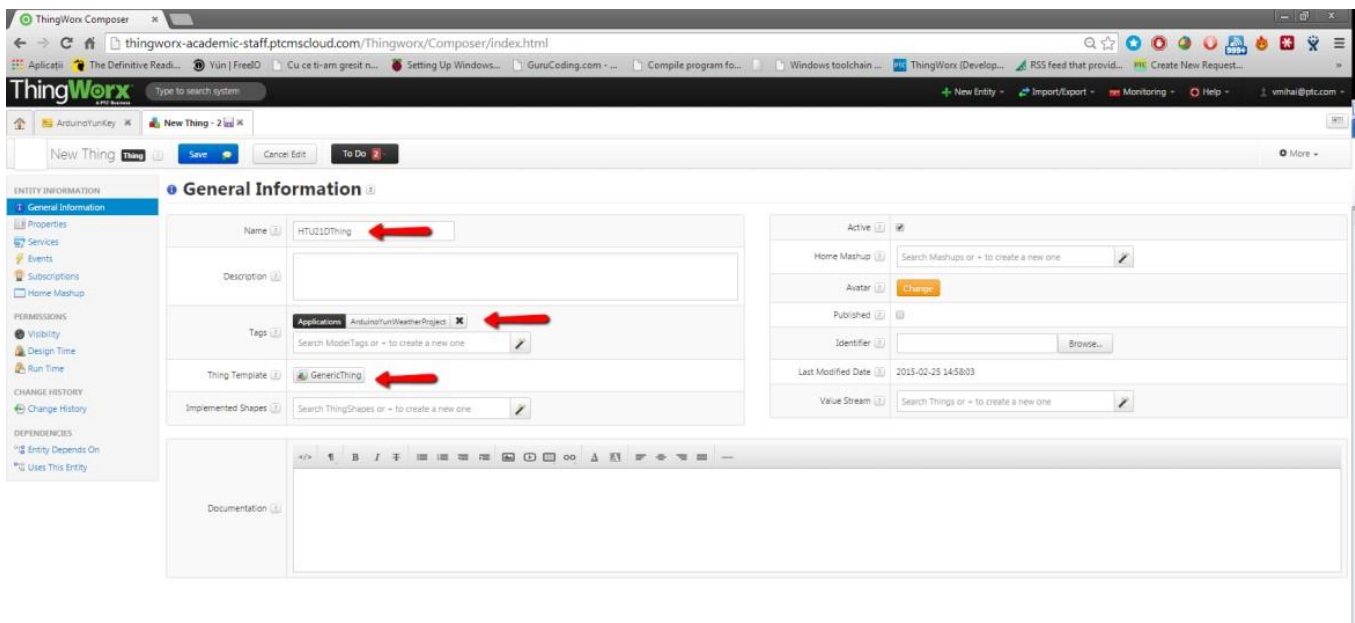


18. Name the Thing “ESP8266\_REST\_yours” or whatever you want.

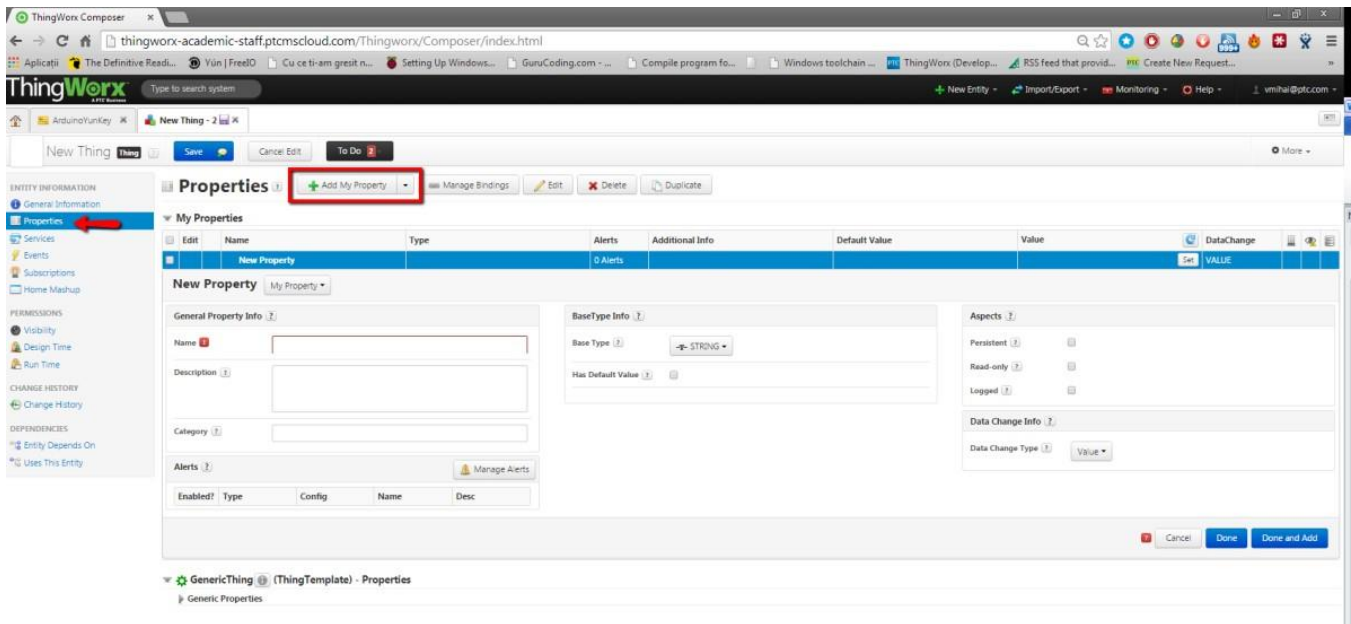
**Note:** Make sure you use this exact name in your Arduino sketch, otherwise you will need to change also the REST call in your Arduino sketch to reflect the new name of the Thing you want to set property values for.

19. Tag it with your ArduinoUnoWeatherProject tag

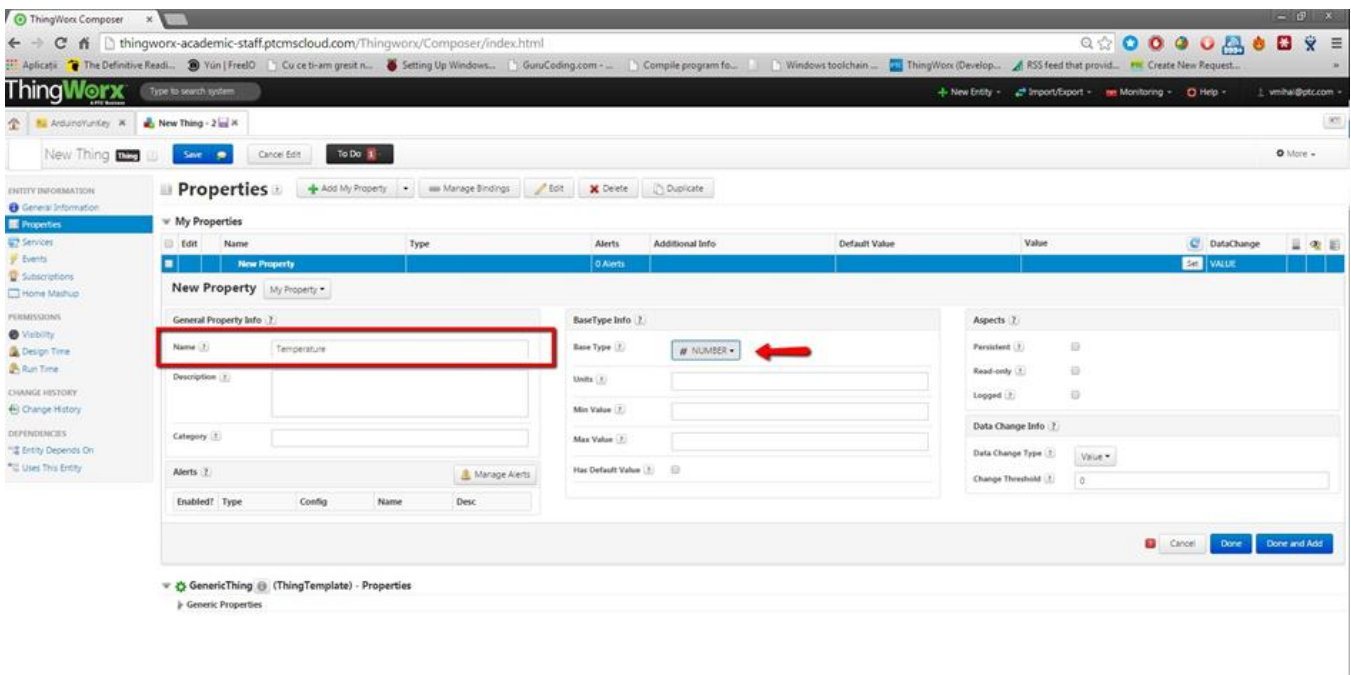
20. Select GenericThing as a Template.



21. Next click Save and go to the Properties Section of your Thing. Here make sure your Thing is in edit mode and click Add My Property.



22. Create five Properties: Temperature (Number), Humidity (Number), AnalogValue (Number), BoolStatus (Number), and MyControl (Boolean). Note: Property names are case-sensitive in REST calls.
- Start with the Temperature property. Type in the Name field and select Number as Base Type.



23. Next, click [Done and Add] to add also the Humidity property. Add the rest of properties. Note that BoolStatus and MyControl need to be set Boolean as Base Type. Click Done. At the end you should have something like this.



ESP8266\_REST\_indystar
Thing
Save
Cancel Edit
To Do

ENTITY INFORMATION

General Information

Properties

Services

Events

Subscriptions

Home Mashup

PERMISSIONS

Visibility

Design Time

Run Time

CHANGE HISTORY

Change History

Properties

My Properties

	Edit	Name	Type	Alerts
		# Temperature		0 Alerts
		# Humidity		0 Alerts
		# AnalogValue		0 Alerts
		BoolStatus		0 Alerts
		IndyCtrl		0 Alerts

GenericThing (ThingTemplate) - Properties

Generic Properties

24. Move to the Services section.

25. Click Add My Service and name the new Service setTempAndHumid.

HTU21DThing

Save

Cancel Edit

To Do

ENTITY INFORMATION

General Information

Properties

Services

Events

Subscriptions

Home Mashup

PERMISSIONS

Visibility

Design Time

Run Time

CHANGE HISTORY

Change History

Services

My Services

Edit	Service Name	Test	Service Type	Inputs	Output
	setTempAndHumid		Local (JavaScript)	Temp Humid	

setTempAndHumid

Service Info

Inputs/Outputs

Snippets

Me

Entities

Name

setTempAndHumid

Description

Category

Async

☐

Script

Search script

Find and replace

```
1
```

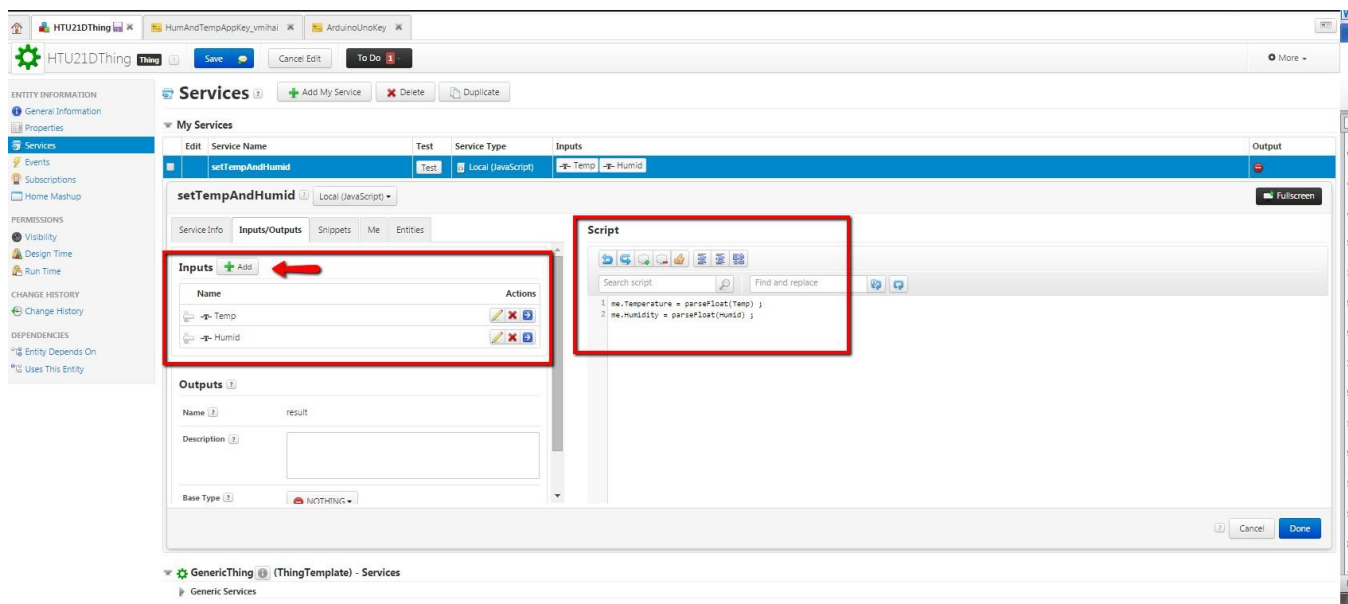
GenericThing (ThingTemplate) - Services

Generic Services

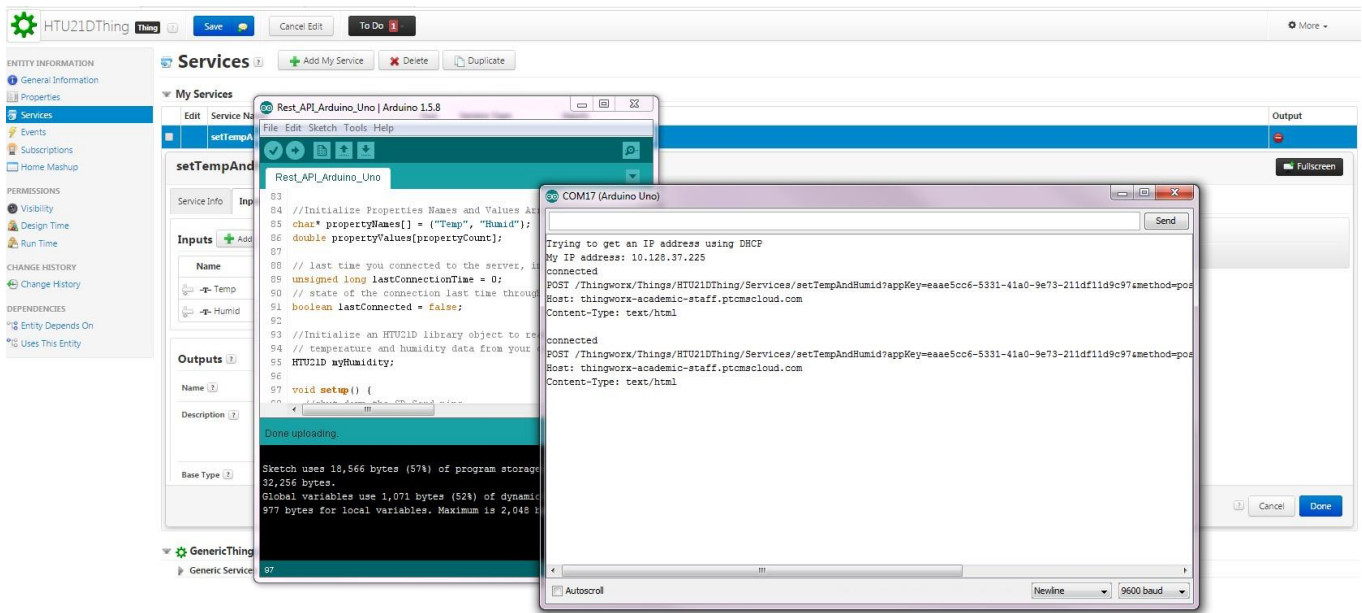


26. Click the Inputs/Outputs tab and add four String inputs. Name them: "Temp", "Humid", "Bool", "Analog" (Make sure you use the exact names and Base types used in your Arduino sketch because this is taken into account when building the HTTP Post Request in your Arduino sketch.)
27. In the script area paste in the following code that sets your Thing properties values to the values of the input parameters of your setTempAndHumid service.

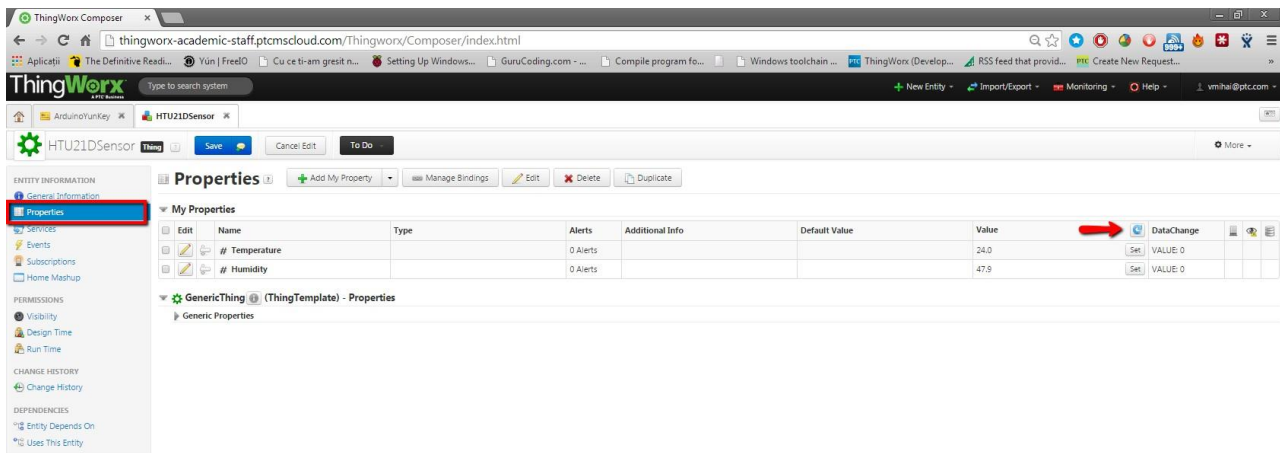
```
var isTrueSet = (Bool.toLowerCase() === 'true');  
me.Temperature = parseFloat(Temp) ;  
me.Humidity = parseFloat(Humid) ;  
me.BoolStatus = isTrueSet ;  
me.AnalogValue = parseFloat(Analog) ;
```



28. Compile and make sure no errors exist.
29. Now, you can go ahead and upload the sketch to the board.
- Connect power to the hardware. Make sure you have the right port selected under Tools→Serial Port and Arduino Uno select under Tools→Board. Then click upload. After seeing the Transfer completed message, go to Tools →Serial Monitor to see the temperature and humidity values are displayed and sent to ThingWorx.

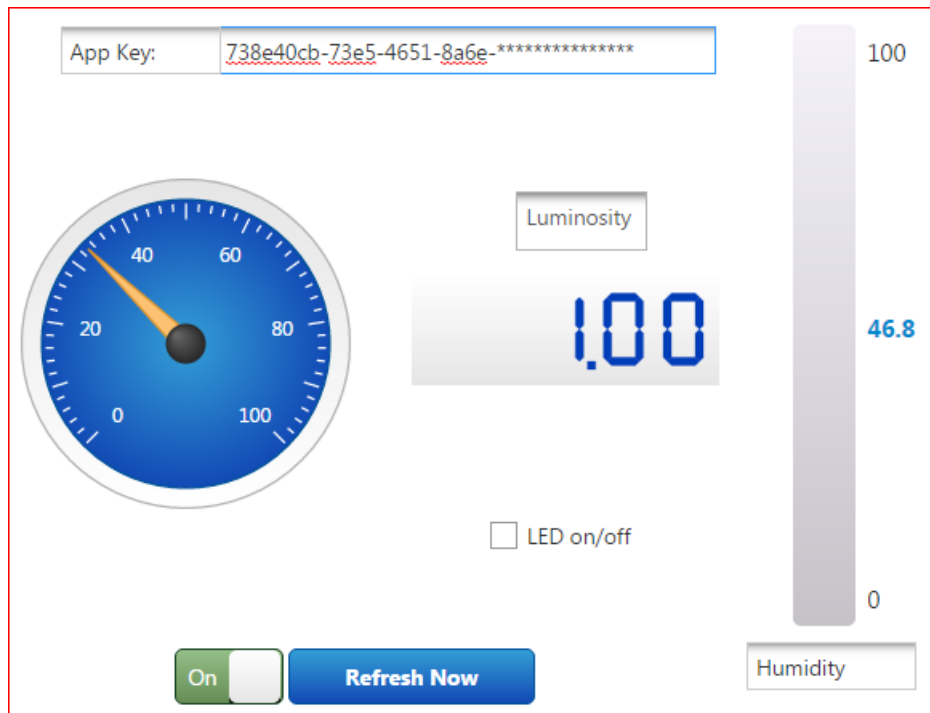


If you go back to your Thing properties section, you should be able to see the values being constantly updated when clicking on the update button.



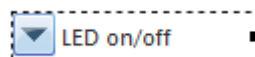
Note: Programs running on the Arduino Uno cannot be stopped. This sketch will run continuously until you replace it with a different one. Note that the sketch is stored on the board's memory and will start running even after cutting the power supply and then connecting it again.

30. Next, create a Mashup like this (to visualize your data):

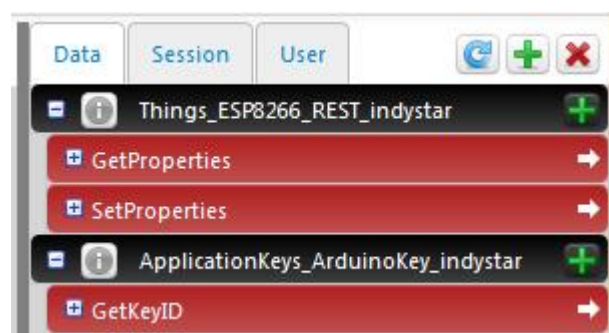


31. To do that, follow *How to Create a Mashup for Arduino Uno: Part One – Instructions.pdf* files <https://www.thingworx.com/ecosystem/academic-program/iot-projects/weather-app-arduino-uno/> but replace every reference of HTU21DThing to Your\_Thing\_name, and also some property names such as Analog.

32. For controlling an LED, need to add a Checkbox widget. Drag and drop Checkbox widget and name it (at the display name) something like “LED on/off.”

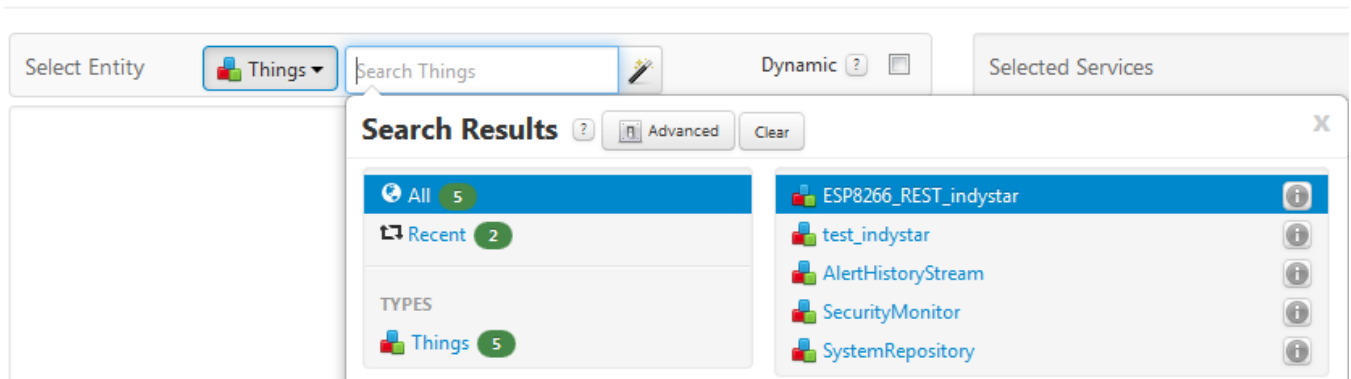


33. Add setProperties Services by clicking + button next to Things\_ESP8266\_REST bar below.

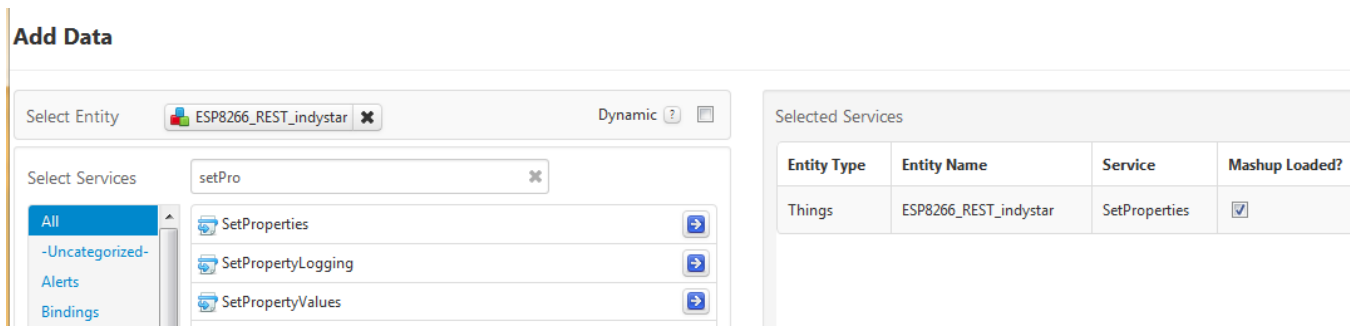


34. Then click [All], select Things, and then select your thing.

## Add Data



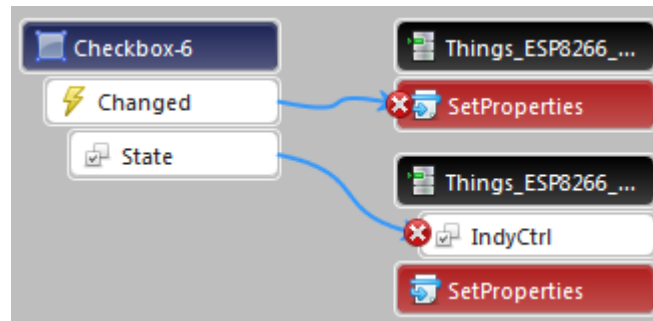
35. Then type setPro at the Filer window, and click [->] to add to Selected Services. Don't forget to check "Mashup Loaded?" checkbox. Click Done and go back.



36. Bind the State attribute (with ➔ mark) of the checkbox widget (it appears when you hover your mouse on the checkbox) to the MyControl on the setProperties Service that you brought in at the previous step.

37. Trigger the setProperties Service by the Changed event of the checkbox widget. That is, bind the Changed attribute (with ➔ mark) of the checkbox widget to the MyControl on the setProperties Service.

38. Now when you click, “LED on/off” checkbox widget, you should see the following on the Connections window.



39. Save frequently while creating mashup.

40. Once you successfully create your mashup, launch the mashup by clicking [View Mashup] button. Your browser’s pop-up blocker may not allow mashup pop-up. In that case, enable pop-ups for this site.

41. See if data change. Try to cover photoresistor to see if Luminosity changes.

42. Now check or un-check “LED on/off” checkbox widget to see if LED2 on the board is turned on or off.

43. Testing the Service: you could test your Service by clicking [Test] button.

The screenshot shows a 'Services' management interface. At the top, there's a header with the title 'Services' and three buttons: '+ Add My Service', 'X Delete', and 'Duplicate'. Below this is a section titled 'My Services' which contains a table. The table has columns for 'Edit', 'Service Name', 'Test', and 'Service Type'. There is one service listed: 'setTempAndHumid'. The 'Test' column for this service has a 'Test' button. The 'Service Type' column shows 'Local (JavaScript)'. Below the 'My Services' section, there's another section titled 'GenericThing (ThingTemplate) - Services' which contains a sub-section 'Generic Services'.

Edit	Service Name	Test	Service Type
	setTempAndHumid	Test	Local (JavaScript)

44. It will show the following Test Service Window, by which you can test your service.

ThingWorx

APTIC Business

Type to search system

setTempAndHumid - Test Service

Please be careful. Only execute services and queries where you understand the consequences. Note that this is testing the currently saved script - any changes you have made since the last save will be lost.

Inputs:

Temp

Bool

☐

Humid

Analog

45.