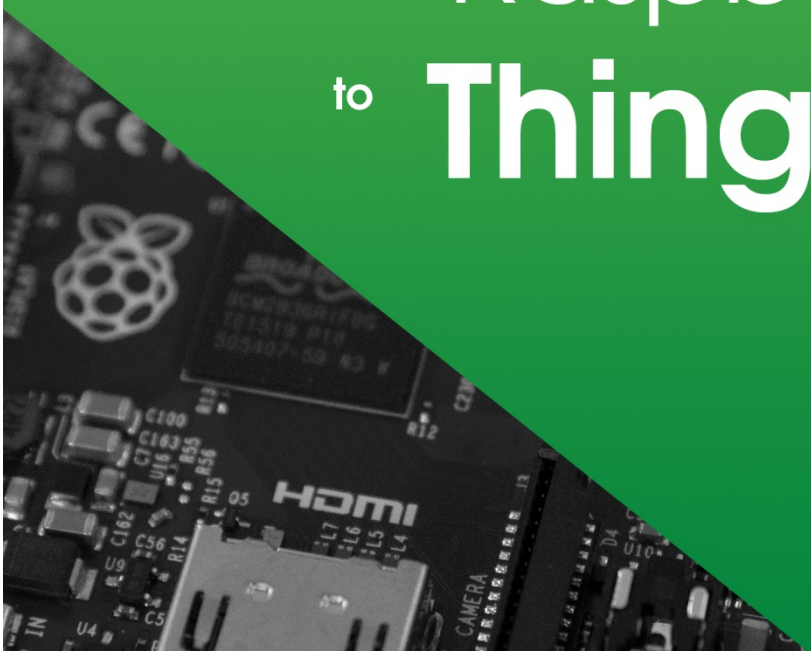
Connect Raspberry Pi to ThingWorx



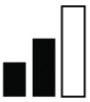
ptc

Project Introduction

Overview

In this project you will create a simple mashup in ThingWorx, PTC's industry leading IoT platform. The project introduces basic skills in modeling an IoT solution and how to visualize data in an application using the ThingWorx mashup tool.

8hrs



Intermediate

Computer Science, Development, IT

Created by PTC Academic Program

English Language

Audience

This project is intended for anyone who is interested in an introductory experience with ThingWorx. To complete this project successfully, you must have basic coding skills.

Project Milestones

1. Setup the project (2HR)
2. Model ThingWorx architecture (1HR)
3. Connect hardware with ThingWorx Composer (3HR)
4. Build a Monitoring Mashup (2HR)

Pre-requisites and Requirements

- No prior ThingWorx training is required
- Basic computer programming skills are required
- Access to ThingWorx Composer (Need access? Learn about your options on page 2)
- WinSCP or similar SFTP client
- ThingWorx Raspberry Pi files folder (you can download it here) or go to http://apps.ptc.com/schools/references/TWX_Academic_RaspberryPi.zip to download files

Bill of Materials

- Raspberry Pi Model B+ with its respective 5V power source
- Female to female jump wires (3)
- AM2302 temperature and humidity sensor
- Wi-Fi USB adapter for Raspberry Pi
- USB Keyboard
- USB Mouse
- HDMI monitor
- A computer with internet connection

How can I get access to the ThingWorx Composer?

This project is built to be used in any ThingWorx Composer version. According to your role you may have different options to access ThingWorx. Please read below:

For the individual developer

If you are using this project to get you started with IoT and ThingWorx on your own you have the following options:

1. Hosted 30-day instance: Get a free temporary ThingWorx hosted instance through the Developer Portal.
2. Downloadable 120-day instance: Get free temporary access to a local instance of ThingWorx.

For information on both options please visit the ThingWorx Developer Dashboard.

For the classroom

If you would like to use this project in the classroom, you can obtain access to our platform using the ThingWorx Academic Edition. For more information the ThingWorx Academic edition, contacts us at university@ptc.com.

For commercial customers

If you are using this project to learn how to use the ThingWorx composer in a commercial environment you can build this project using your commercial instance. Interested on ThingWorx for your company? Visit https://www.thingworx.com for more information.

Are you using ThingWorx Academic Edition?

ThingWorx Academic Edition has special features that allow optimization in the classroom. Therefore, you will experience a few changes when building this project. Please read below:

- **Entity Naming:** All entities created by a user of the system will have their username automatically filled in after the entity name. This feature is added to avoid conflict when using similar names on different entities by multiple users. You will notice that once you start typing the name of your entity an underscore followed by your username will appear (i.e. entityname_username).
  **Affects every entity created in this project.*

- **Value Streams:** If you are using the ThingWorx Academic Edition you have been provided with a personal value stream (i.e. VS_username) for data storage and will be restricted from creating other forms of data storage. However, you can use this Value Stream in multiple entities as needed. Note that ALL data stored is wiped every three days.
  **Affects the Create Value Stream segment under the Model section of this project.*

- **Configuration of Java project:** Entity naming must be consistent throughout the project in order to guarantee effective communication between ThingWorx and the Raspberry Pi. Therefore, it is necessary to modify the Main class according to the entity naming modification in the ThingWorx Academic Edition.
  **Affect the Configure your Java Project on the Computer segment under the Connect section of this project.*

Set Up

IN THIS SECTION YOU WILL:

- Install and update Raspbian on the Raspberry Pi.
- Connect the Raspberry Pi to the internet.
- Install the Python library to allow the Raspberry Pi to read temperature and humidity.
- Wire the Raspberry Pi with the Am2302 sensor to obtain temperature and humidity reading from the sensor.
- Download Java project from the ThingWorx website and pin them down in a known location.
- Install Maven and Java in the computer.

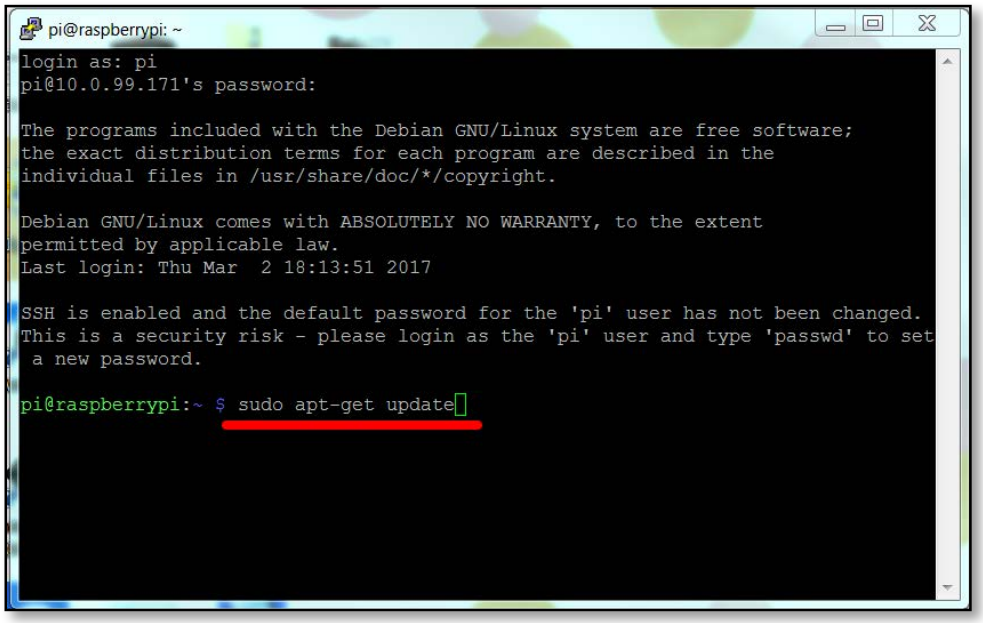
Raspberry Pi Setup

Before using the Raspberry Pi, it is necessary to install the Raspbian image to the SD card. Raspbian is basically a flavor of the Linux operating system which has been developed for the Raspberry Pi. In order to obtain Raspbian you can download NOOBS or Raspbian from the Raspberry Pi website and write the image onto the micro SD card. This tutorial will be done using a Windows Operating System. Follow this tutorial if you need to set up your Raspberry Pi. For proper functionality, make sure that you do the following:

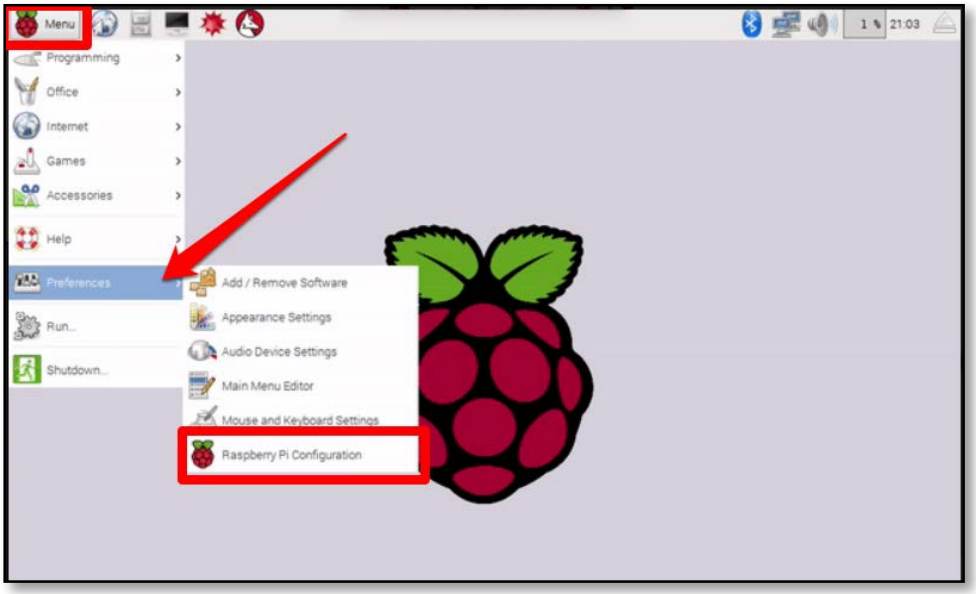
1. Type and run the following commands on the Terminal window in the Raspberry Pi:

   [sudo apt-get update]: A complete software update for the Raspberry Pi

   [sudo apt-get install oracle-java7-jdk]: Java JDK is required by ThingWorx to run the JVM and other components to enable the Raspberry Pi to run applets and applications written in Java code.



2. Enable SSH communication.
    a. Go to the Raspbian user interface
    b. Click **Menu**
    c. Go to **Preferences**
    d. Click **Raspberry Pi Configuration**
    e. Select the **Interfaces** tab
    f. Click **Enabled** on SSH

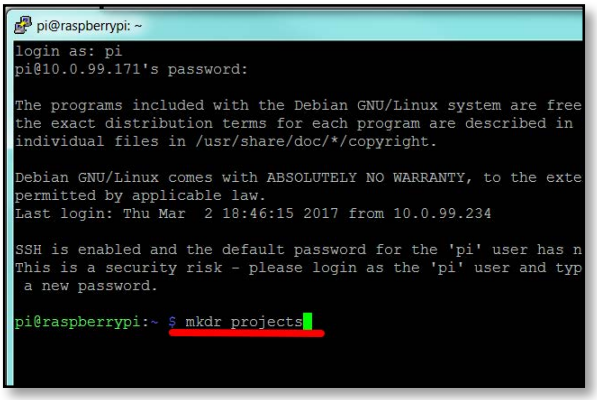


Wiring and Project Setup

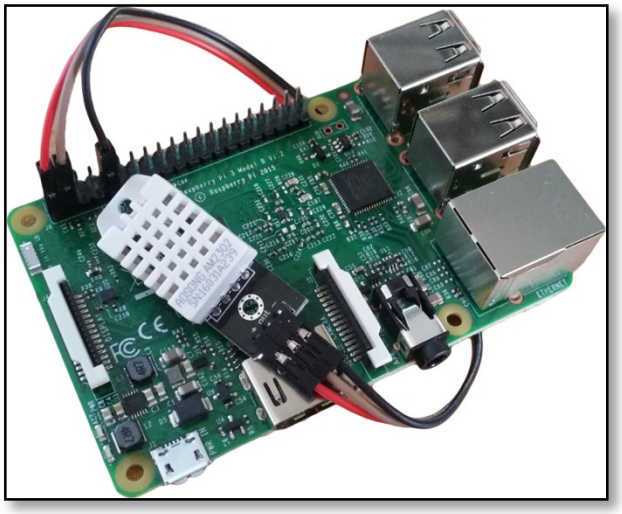
In this section you will perform the wiring for the AM2302 sensor to the Raspberry Pi and install the necessary software and Python libraries that will allow you to read data from the sensor.

On the Raspberry Pi Terminal

3. On the home directory, create a directory called **projects** to hold the project on the Raspberry Pi home directory (command: *mkdr*, i.e. ***mkdr projects***).

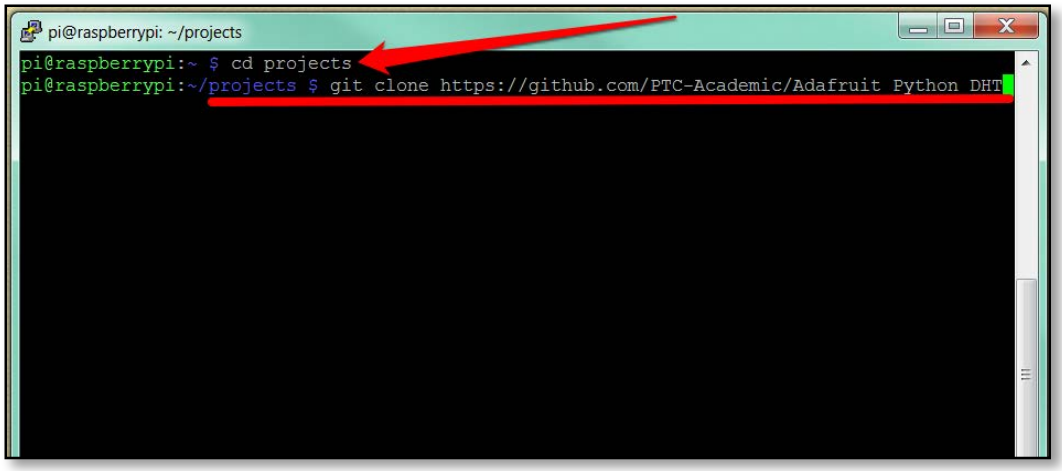


4. Navigate to the new directory you just created (command: *cd,* i.e. ***cd projects***).
5. Perform wiring and install the python library to allow the Raspberry Pi to read data from the AM2302 temperature and humidity sensor.
    a. Wire your Raspberry Pi using the pictures below as a guide (we use GPIO 4 for this project).



    b. Run the following command to obtain the Python project and to read temperature and humidity from the Am2302 sensor:

    git clone https://github.com/PTC-Academic/Adafruit_Python_DHT



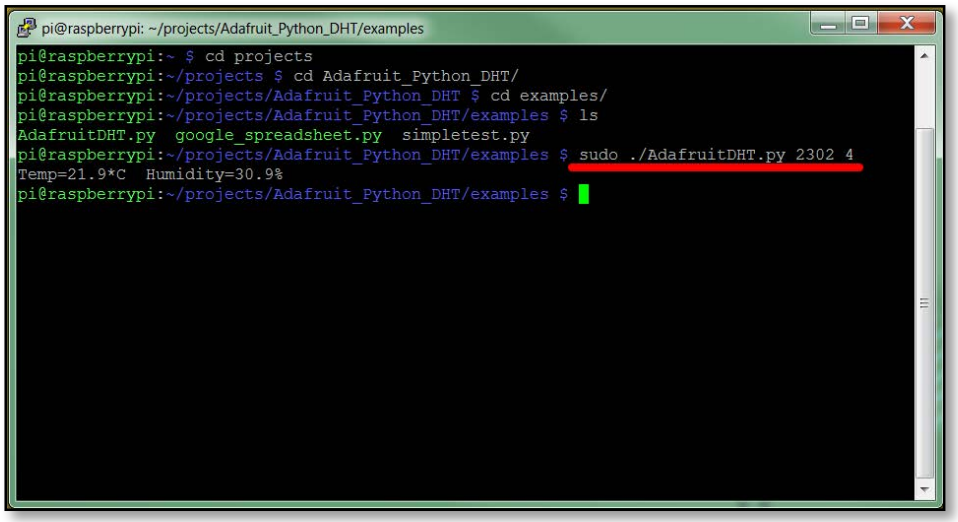
c. If you have not yet done so, install the build-essential, python-dev and python-open ssl libraries. To do so, run the following command lines:

sudo apt-get install build-essential python-dev python-openssl
sudo python setup.py install

d. On the "Adafruit_DHT" folder, open the "examples" folder and run the following command to get temperature and humidity readings

sudo ./AdafruitDHT.py 2302 4

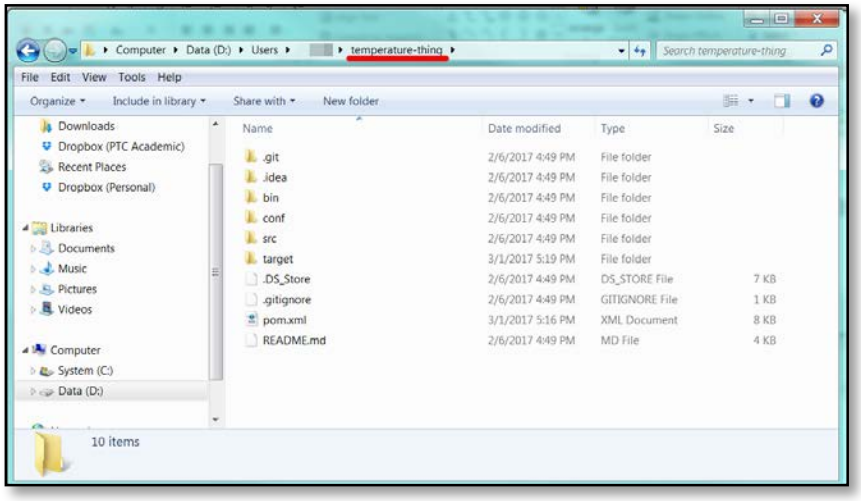
where, *2302* is the type of sensor and *4* is the GPIO pin.



NOTE: *If you choose to connect your AM2302 sensor to a different data pin you need to update the TempAndHumidityThing.java file located in the file downloaded from the ThingWorx website (If not altered, command lines 115 and 135).*

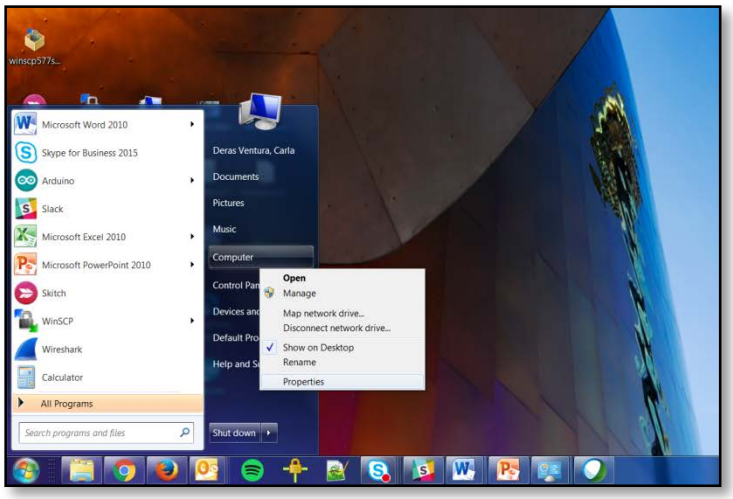
On the Computer

6. Go to the ThingWorx folder for this project (TWX_Academic_RaspberryPi.zip) and extract the folder (temperature-thing) with the Java project that will serve to push data from the AM2302 sensor into ThingWorx to your home directory.

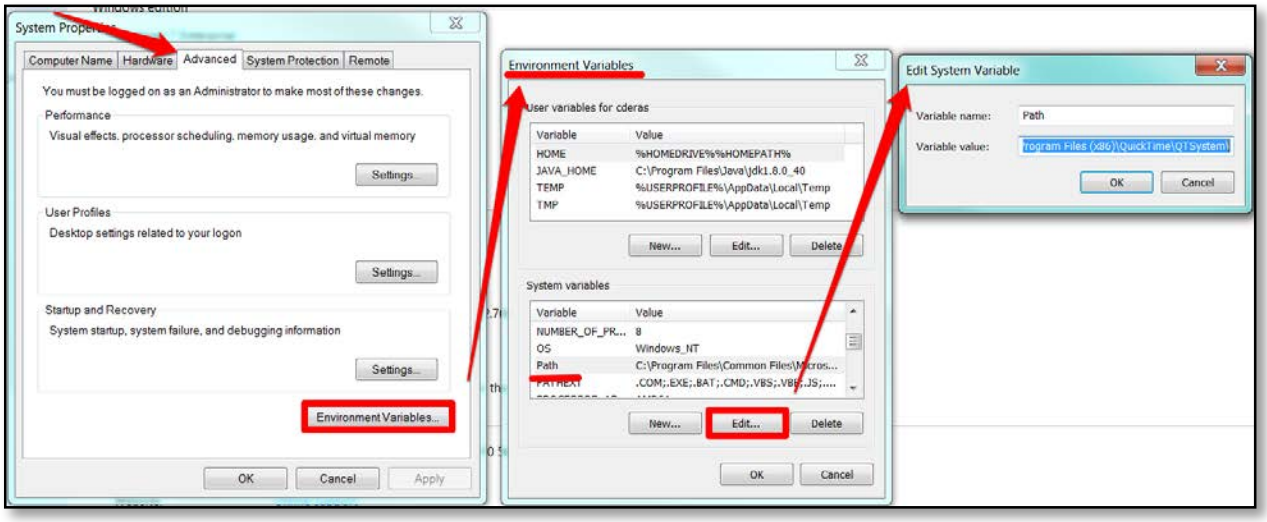


__NOTE:__ To view your home directory, open __cmd.exe__ and type "echo %USERPROFILE%", press Enter and then type "dir" and press Enter.

7. Download and install the latest version of Java JDK. This is needed for the JVM and other components.
8. Download and install Apache Maven. It is needed to manage the building process of the project. It also handles all the downloading, installing, and version updating of project dependencies.
9. Add Maven to the PATH environment variable so it can be used from the command line.
    a. Go to start, then find and click **Computer**.
    b. Right click **Computer** and select **Properties**.



c. Select **Advanced System Settings** on the right side of the window.
d. On the **Advanced** tab, click **Environment Variables…**
e. Scroll down under **System Variables**, find and select **Path**.
f. Click **Edit**

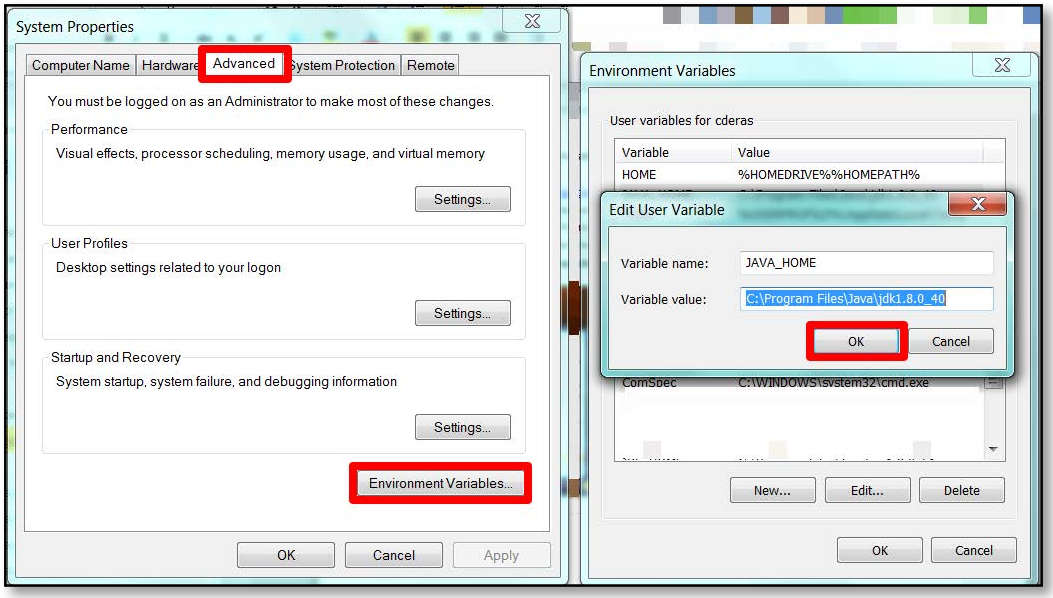


g. Select all the text and copy it
h. Paste the selected text into a text editor such as Notepad or Word for easier visibility
i. Go to the bin folder in the Maven folder and copy the location address
j. Add the bin folder from Maven address to the text (notice the address has to be between semicolons):
C:\Program Files (x86)\apache-maven-3.2.1\bin

C:\Program Files\Common Files\Microsoft Shared\Windows Live;C:\Program Files (x86)\Common Files\Microsoft Shared\Windows Live;C:\ProgramData\Oracle\Java\javapath;C:\Program Files\RSA SecurIDTokenCommon;%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\;%windir%\cde\tools;C:\apache-maven-3.3.3\bin;C:\Program Files (x86)\WebEx\Productivity Tools;C:\Program Files (x86)\WebEx\PTools020000000;C:\Program Files (x86)\Windows Live\Shared;C:\Program Files (x86)\Skype\Phone\;C:\Program Files (x86)\GtkSharp\2.12\bin;C:\Program Files\ServiceNow;C:\Program Files (x86)\QuickTime\QTSystem\

10. Select and copy the text you edited using Notepad or Word and paste it again in the **Variable Value** field in the **Edit System Variable** window.
11. Click **OK** on the **Edit System Variable** window to save it.
12. Click **OK** on the **Environmental Variables** window.
13. Click **OK** on the **System Properties** window.
14. Add Java to User Variables to utilize Java from the command line as well.

a. Go to start, then find and click **Computer**.
b. Right click **Computer** and select **Properties**.
c. Select **Advanced System Settings**.
d. On the **Advanced** tab, click **Environment Variables…**
e. Under **User variables for…** select **New**
f. Type **JAVA_HOME** in the **Variable name:** field
g. Type the address of the Java JDK in the **Variable value:** field. (C:\Program Files\Java\jdk1.7.0_60)



h. Click **OK** on the **New User Variable** window.
i. Click **OK** on the **Environmental Variables** window.
j. Click **OK** on the **System Properties** window.

This concludes the Set Up section of this Project

Model

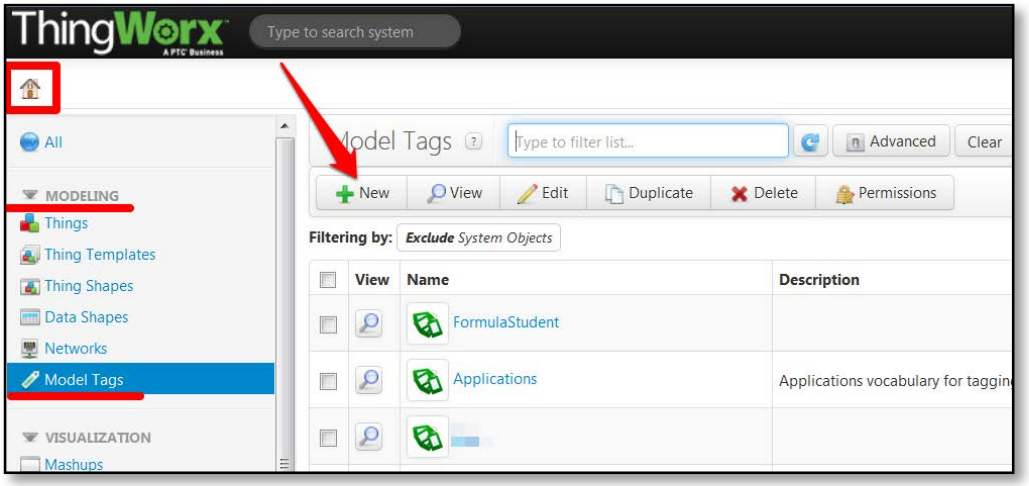
IN THIS SECTION YOU WILL:

- Create Model Tag that will help categorize and group entities in this project.
- Create a Thing Shape to allocate properties to the Am2302 Thing.
- Create a Thing Template to provide connectivity capabilities to the Am2302 Thing.
- Create a Thing that will represent the physical assets of the Am2302 sensor.
- Create an Application Key to allow communication between ThingWorx and the Raspberry Pi.
- Create a Value Stream to store data obtained from the Am2302 sensor.
- Create a Media Entity and Style Definition to add a background picture to the final user interface.

In this part, we will model the project on ThingWorx creating the necessary entities to define different properties and house the data that will be collected from the sensor.

Create a Model Tag

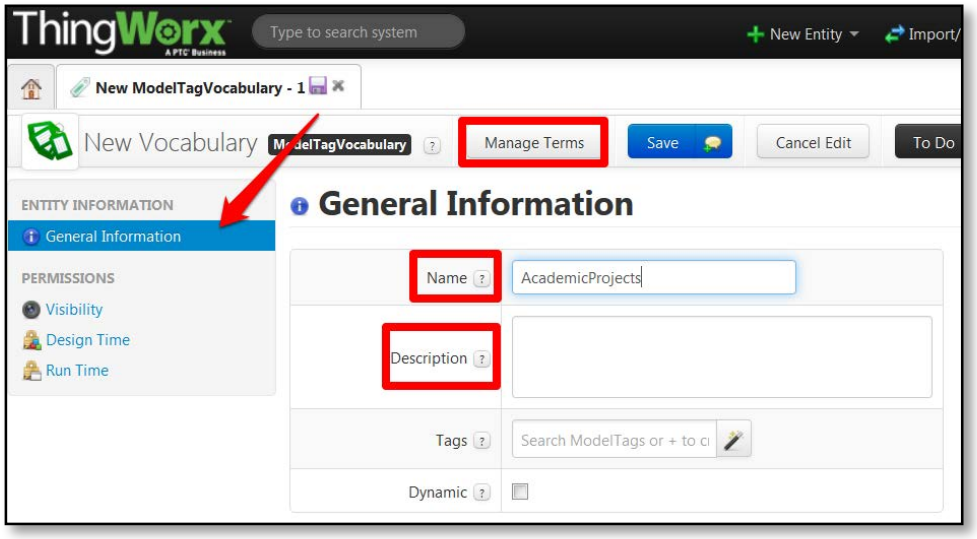
1. Open your ThingWorx Composer
2. On the **Home** tab, **Modeling** section, select **Model Tag** and click **New**.



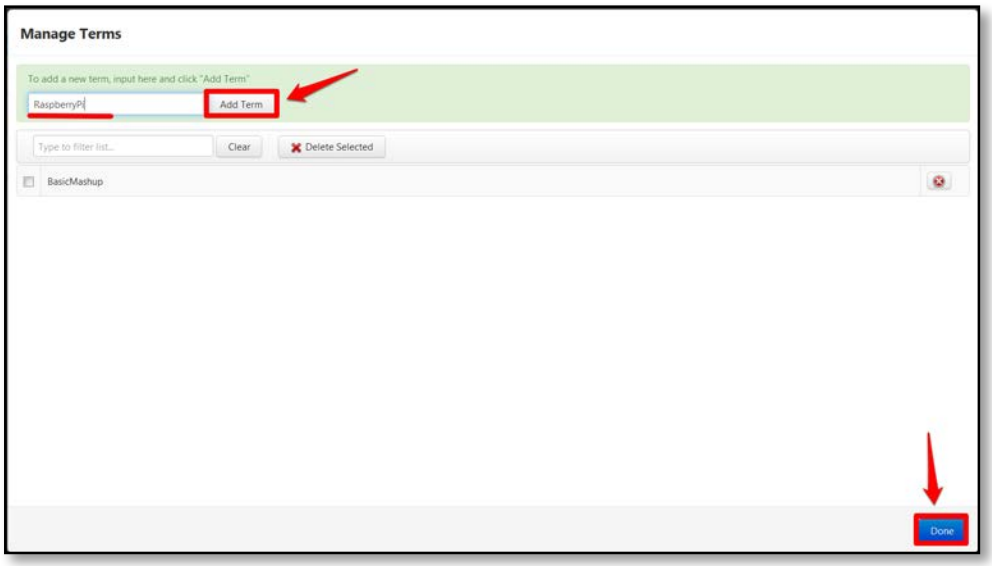
3. Enter the Name (AcademicProjects) and Description for the new model tag in the **General Information** section and save it.

Note: *If you already worked on the "Build a Basic ThingWorx Mashup" project, you can just add the Model Tag term following steps 3 to 5.*

4. Click the **Edit** button to get back to editing mode. Select the **Manage Terms** option.



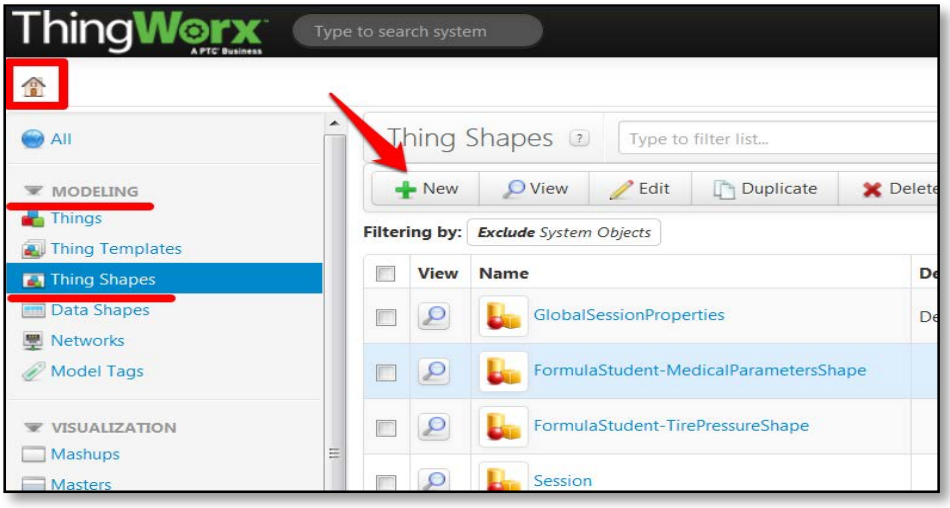
5. Add a new term by typing it in the box (RaspberryPi) and click the **Add Term** button. Then click **Done**.



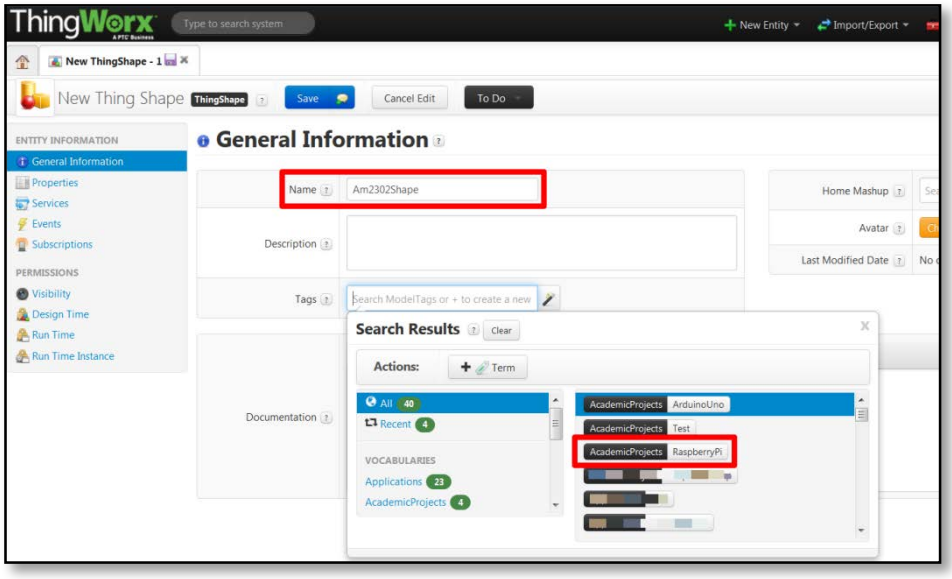
6. Click **Save** again to save the model tag with the new term.

Create a Thing Shape

7. On the **Home** tab, **Modeling** section, select **Thing Shapes** and click **New**.

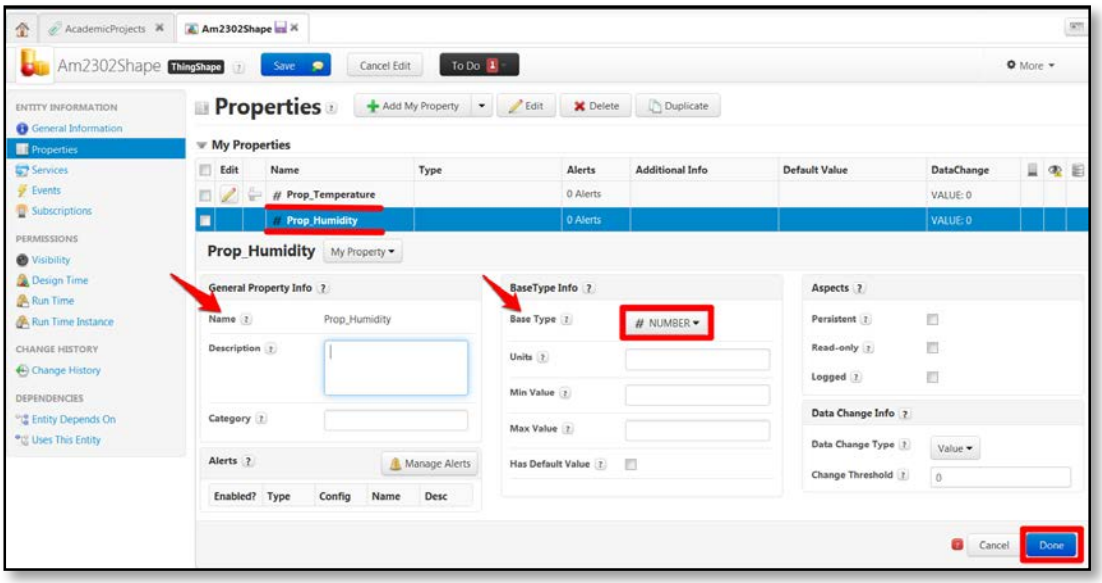


8. Enter the name (Am2302Shape) in the **Name** section.
9. Tag the entity by typing the name in the **Model Tag** section (AcademicProjects: RaspberryPi) and selecting the tag from the Search Results dialog box.



10. Select **Properties** from the **Entity Information** section.
11. Click **Add My Property**.
12. Complete the property fields as follows:
    a. **Name:** Prop_Temperature
    b. **Base Type:** Number
    c. Select the **Logged** checkbox

13. Click **Done and Add**.
14. Complete the property fields as follows:
    a. **Name:** Prop_Humidity
    b. **Base Type:** Number
    c. Select the **Logged** checkbox
15. Click **Done**.



16. Click **Save** on the top of the screen.

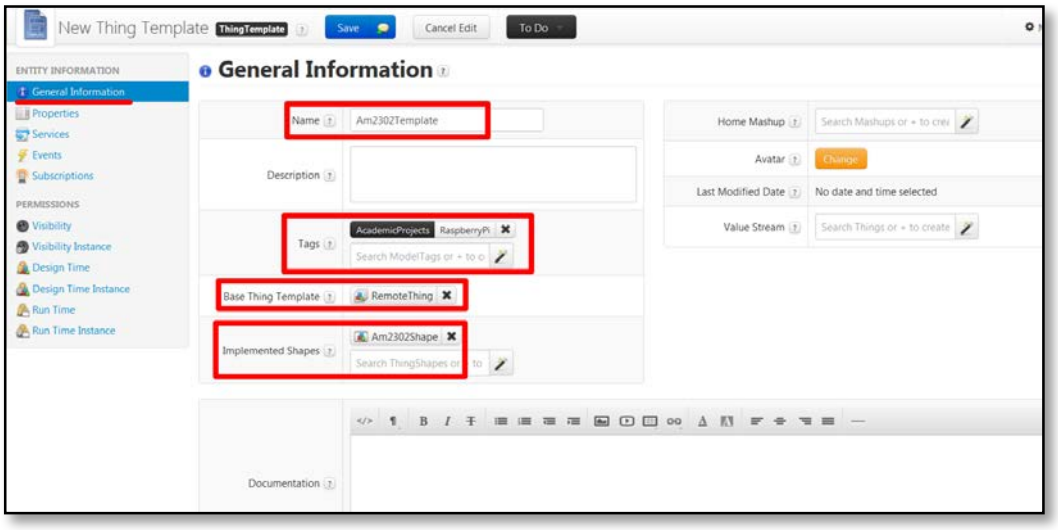
NOTE: *The Logged box specifies if the property value should be automatically logged to a* ***Value Stream*** *whenever the data changes.*

Create a Thing Template

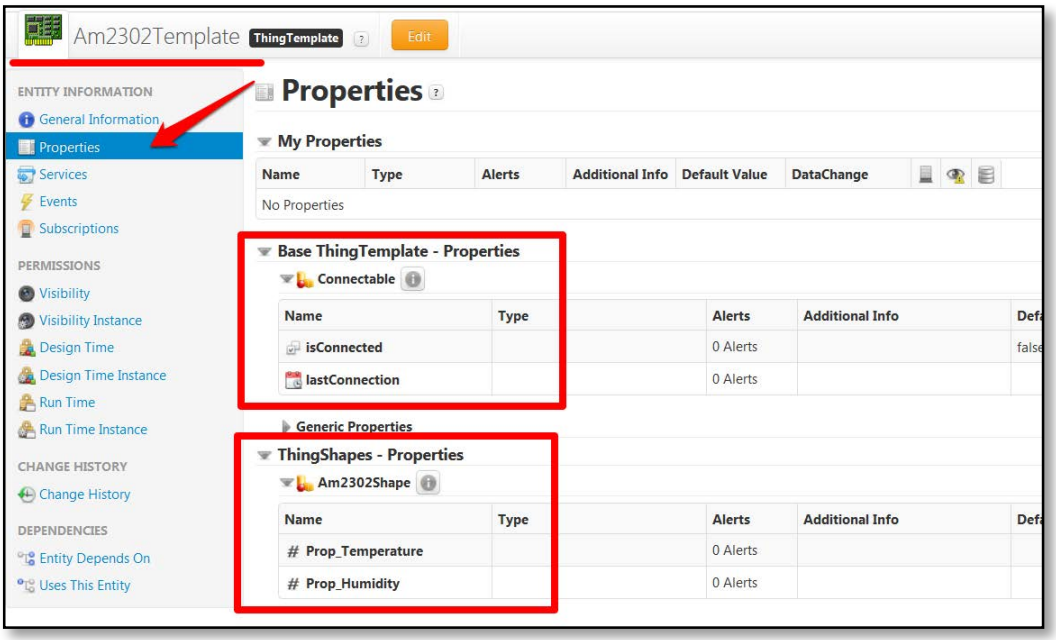
17. On the **Home** tab, **Modeling** section, select **Thing Templates** and click **New**.
18. Enter the name (Am2302Template) in the **Name** section.
19. Tag the entity by typing the name in the **Model Tag** section (AcademicProjects: RaspberryPi) and selecting the tag from the Search Results dialog box.
20. On the **Base Thing Template** field allocate a **Remote Thing Template** by typing *RemoteThing* and selecting the template from the dialog box.
21. On the **Implemented Shapes** field allocate the already created **Am2302Shape** by typing *Am2302Shape* and selecting the shape from the dialog box.

NOTE: *Using a* ***RemoteThing*** *Template gives a Thing Template the necessary properties to send or receive data remotely. Adding an* ***Implemented Shape*** *makes a Thing Template inherit all the properties on that* ***Thing Shape****.*

22. Click **Save** on the top of the screen.



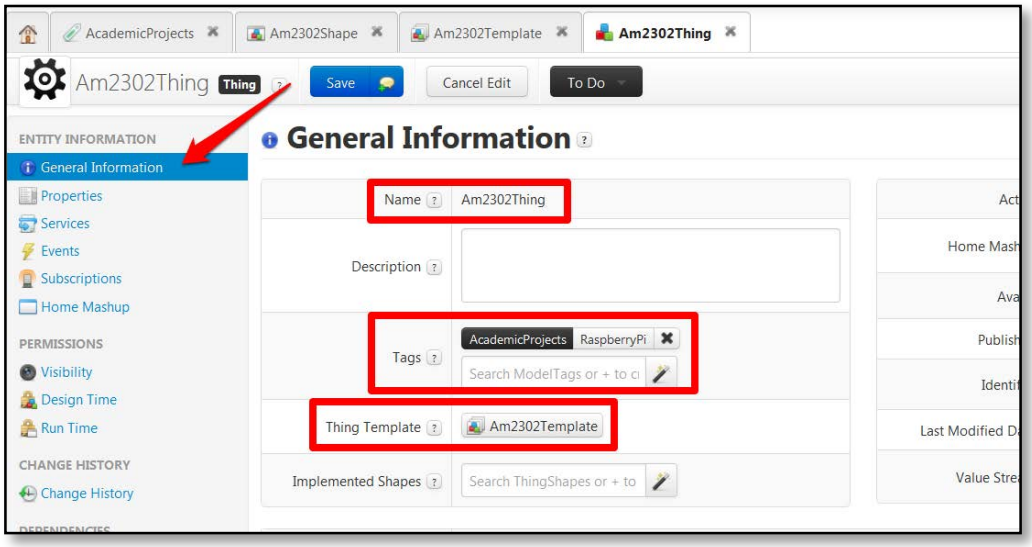
23. Click **Properties** on the left side of the screen and notice all the properties inherited from the **Base Thing Template** and the **Implemented Shape**.



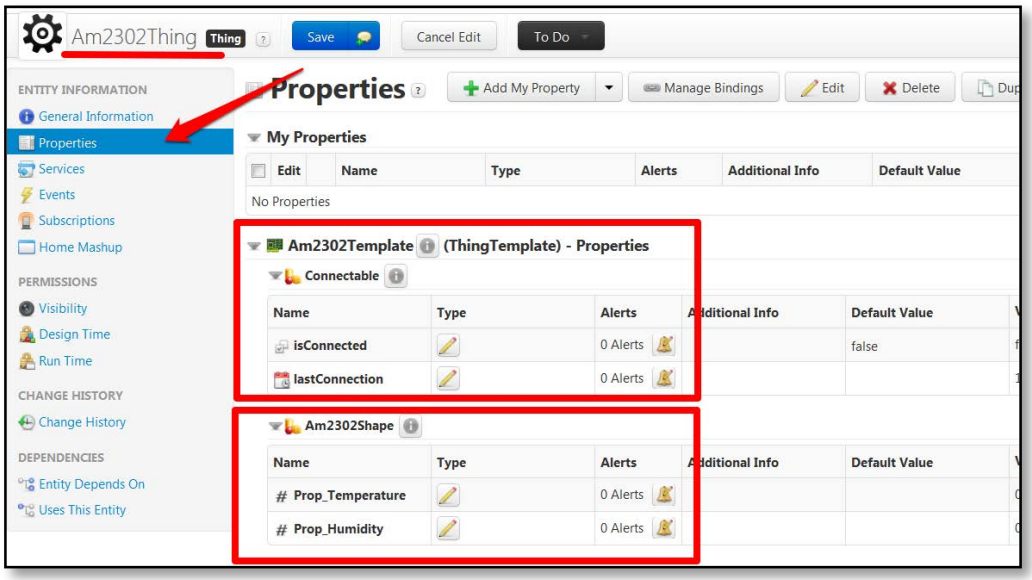
Create a Thing

It is necessary to create a **Thing** to later bind the data that will be received from the Am2302 sensor.

24. On the **Home** tab, **Modeling** section, select **Things** and click **New**.
25. Enter the name (Am2302Thing) in the **Name** section.
26. Tag the entity by typing the name in the **Model Tag** section (AcademicProjects: RaspberryPi) and selecting the tag from the Search Results dialog box.
27. On the **Thing Template** field allocate the already created **Am2302Template** by typing "Am2302Template" and selecting the template from the dialog box.
28. Click **Save** on the top of the screen.



29. Click **Properties** on the left side of the screen and notice all the properties inherited from the **Am2302Template**.



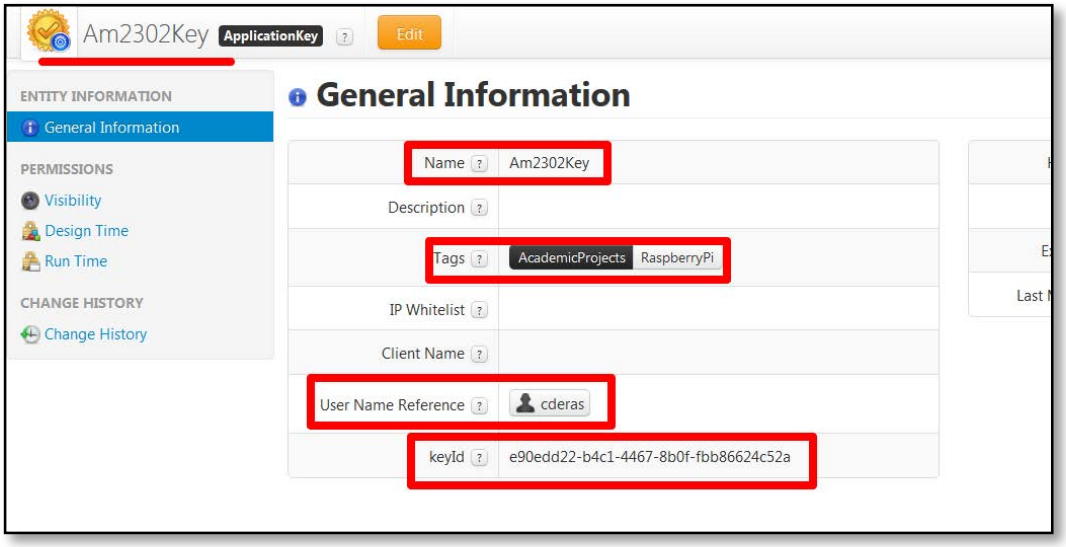
Create an Application Key

In order for ThingWorx to later identify the calling program and the user that are connecting to the website (ThingWorx Composer) it is necessary to generate an Application Key for the Raspberry Pi to access the Composer.

NOTE: *Application Keys are security tokens that can be used to login to ThingWorx instead of using standard credentials.*

30. On the **Home** tab, **Security** section, select **Application Keys** and click **New**.
31. Enter the name (Am2302Key) in the **Name** section.
32. Tag the entity by typing the name in the **Model Tag** section (AcademicProjects: RaspberryPi) and selecting the tag from the Search Results dialog box.
33. Enter your ThingWorx username by typing it in the **User Name Reference** field and selecting it from the dialog box.
34. Click **Save** on the top of the screen.

NOTE: *A new field **keyId** will appear under **User Name Reference** field with multiple characters, which is your Application Key.*



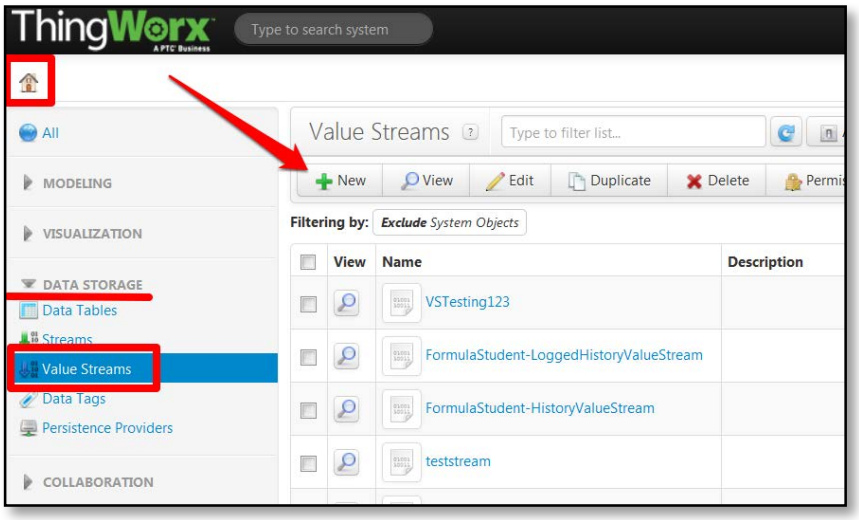
NOTE: *Make sure you use the same user to log into ThingWorx and to generate the Application Key in this tutorial. Otherwise, the data sent from the ThingWorx Java SDK program will not be received by ThingWorx. The remote thing will appear online and connected but no data will be received and no message or error will appear informing you what is causing this situation.*

Create a Value Stream

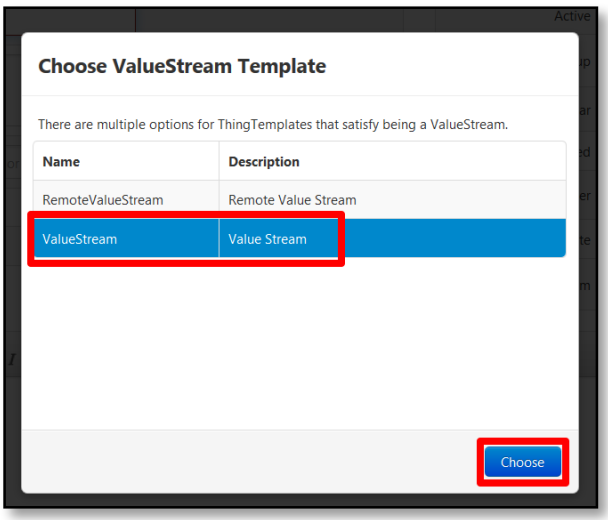
In this step you will create a Value Stream to store the Temperature and Humidity values obtained by the sensor. The Value Stream will allow you to have history data, which can later be displayed in a Mashup using a Label Chart widget.

***NOTE:** If you are using ThingWorx Academic Edition, skip steps 35 to 40.*

35. On the **Home** tab, **Data Storage** section, select **Value Streams** and click **New**.

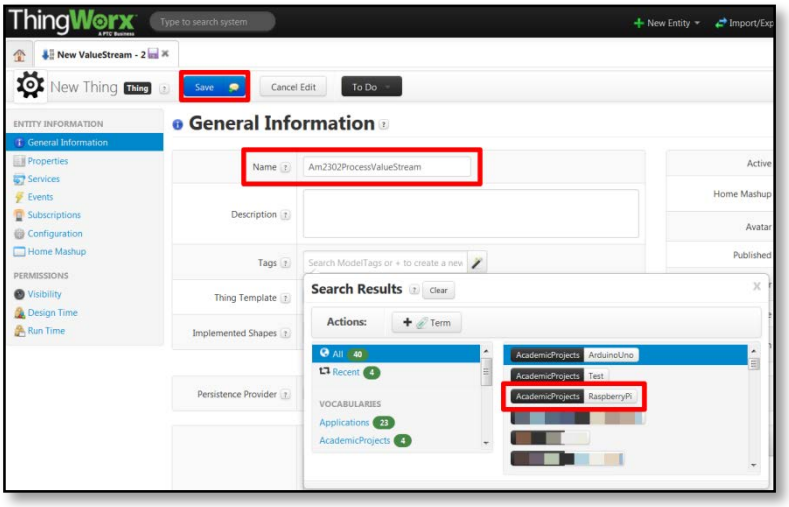


36. Select the **ValueStream** option.
37. Click **Choose**.

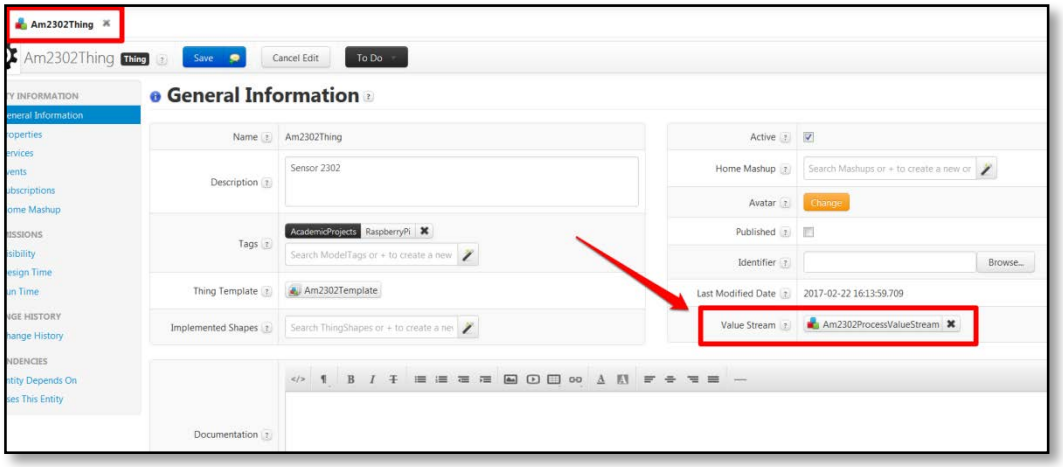


38. Enter the name (Am2302ProcessValueStream) in the **Name** section.
39. Tag the entity by typing the name in the **Model Tag** section (AcademicProjects: RaspberryPi) and selecting the tag from the Search Results dialog box.

40. Click **Save**.



41. Go to your **Am2302Thing** and open it in **Edit** mode.
42. Start typing *Am2302ProcessValueStream* (*VM_username* if you are using ThingWorx Academic Edition) in the Value Stream field and selecting the value stream you just created from the Search Results dialog box.



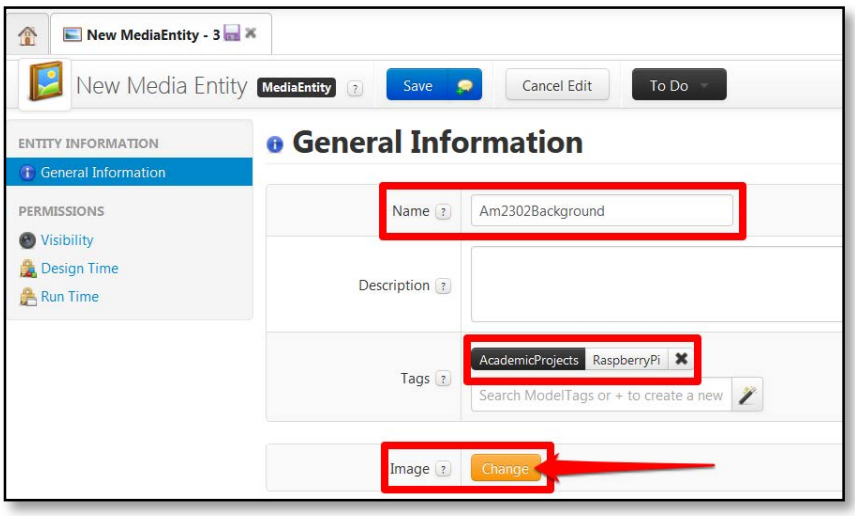
43. Click **Save**.

Create a Media Entity and Style Definitions

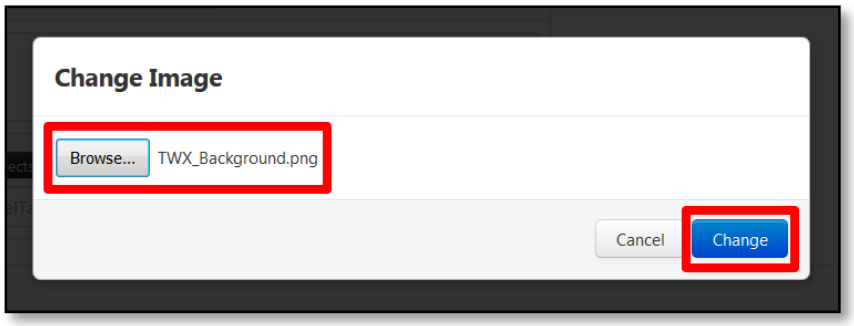
In this section we will create a Style Definition and add a Media Entity to it to customize the look of the Mashup where the data will be displayed.

44. On the **Home** tab, **Visualization** section, select **Media** and click **New**.
45. Enter the name (Am2302Background) in the **Name** section.

46. Tag the entity by typing the name in the **Model Tag** section (AcademicProjects: RaspberryPi) and selecting the tag from the Search Results dialog box.
47. On the **Image** field, click **Change**.

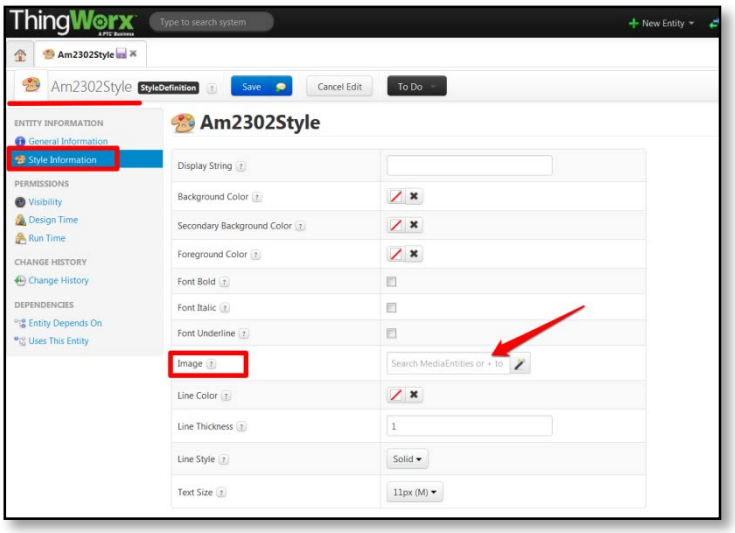


48. Click **Browse…** and select the image provided in the project files folder ("TWX_Background.png") to use as the background of your Mashup.
49. Click **Change**. You will be able to see your image in the **Image** field.

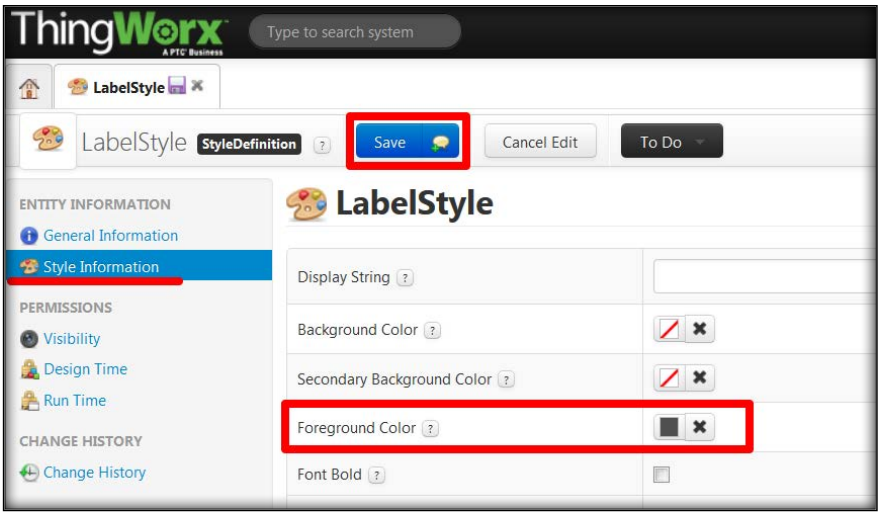


50. Click **Save**.
51. On the **Home** tab, **Visualization** section, select **Style Definition** and click **New**.
52. Enter the name (Am2302Style) in the **Name** section.
53. Tag the entity by typing the name in the **Model Tag** section (AcademicProjects: RaspberryPi) and selecting the tag from the Search Results dialog box.
54. On the left side of the screen click **Style Information**.
55. On the **Image** field, start typing *Am2302Background* and select the entity from the Search Result dialog box.

56. Click **Save**.



57. Back on the **Home** tab, **Visualization** section, select **Style Definition** and click **New**.
58. Enter the name (LabelStyle) in the **Name** section.
59. Tag the entity by typing the name in the **Model Tag** section (AcademicProjects: RaspberryPi) and selecting the tag from the Search Results dialog box.
60. On the left side of the screen click **Style Information**.
61. On the Foreground Color field click the color square next to the **x**.
62. On the pop up window, select black or any color you would like to use for your Mashup text.
63. Click **Select**.
64. Click **Save**.



This concludes the Model section of this Project

Connect

IN THIS SECTION YOU WILL:

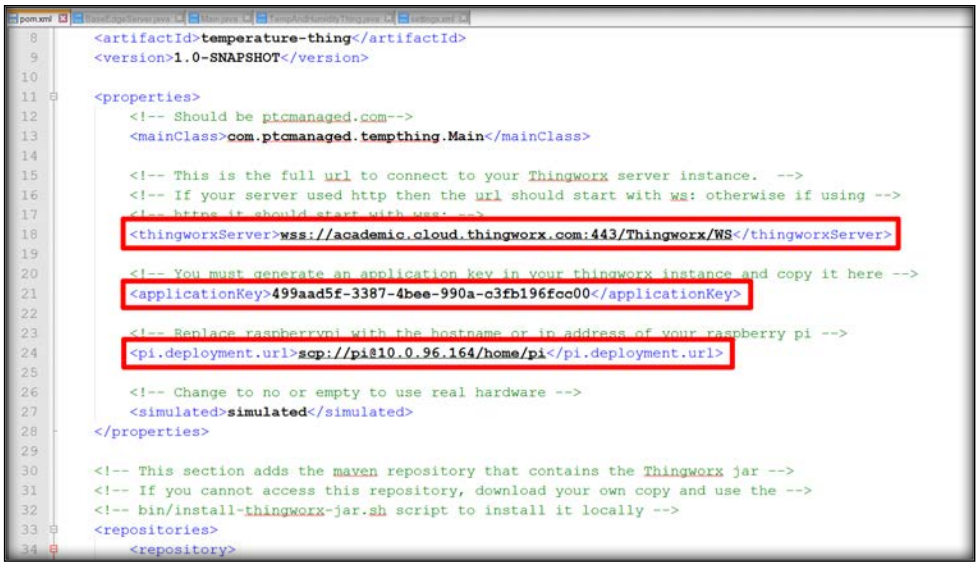
- Incorporate connectivity information to the Java project to provide dual communication capabilities between the Raspberry Pi and ThingWorx.
- Create a .jar file using Maven.
- Move the .jar file to the Raspberry Pi to execute and connect it to ThingWorx.
- Bind ThingWorx properties to temperature and humidity sensor readings.

In this section we will connect the Raspberry Pi with ThingWorx in order to transfer, receive and display data. To accomplish a successful connection we will need to configure not only ThingWorx but the Raspberry Pi and the Java SDK project, which is the main facilitator in this project when connecting to ThingWorx.

Configure your Java Project on the Computer

These steps will allow you to run an example of the java project on the computer and deliver it to the Raspberry Pi for testing.

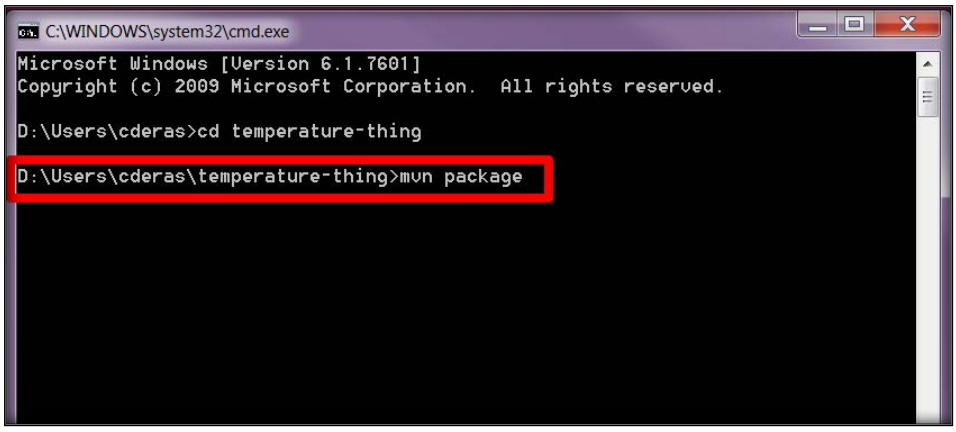
1. Open the "temperature-thing" folder saved in your home directory during the "Set Up" part of this project.
2. Open the "pom.xml" file using a source code editor like Notepad++.
3. Look for the <properties> section of the script in the "pom.xml" file and edit as follows:
    a. Update the URL where you want the JAR file to be copied to with the respective IP address (command line 18).
    b. Update the ThingWorx Application Key created on part A of the Connect section (command line 21).
    c. Update the Raspberry Pi IP address (command line 24).



4. Save the changes and close the “pom.xml” file.

NOTE: *If you are working in ThingWorx Academic Edition, you need to update your “Main.java” file [temperature-thing\src\main\java\com\ptcmanaged\tempthing\Main.java]. This file contains the information of the Thing created on ThingWorx. To assure successful connection between the Raspberry Pi and ThingWorx the name of the Thing has to be the same on ThingWorx and the “Main.java” file.*

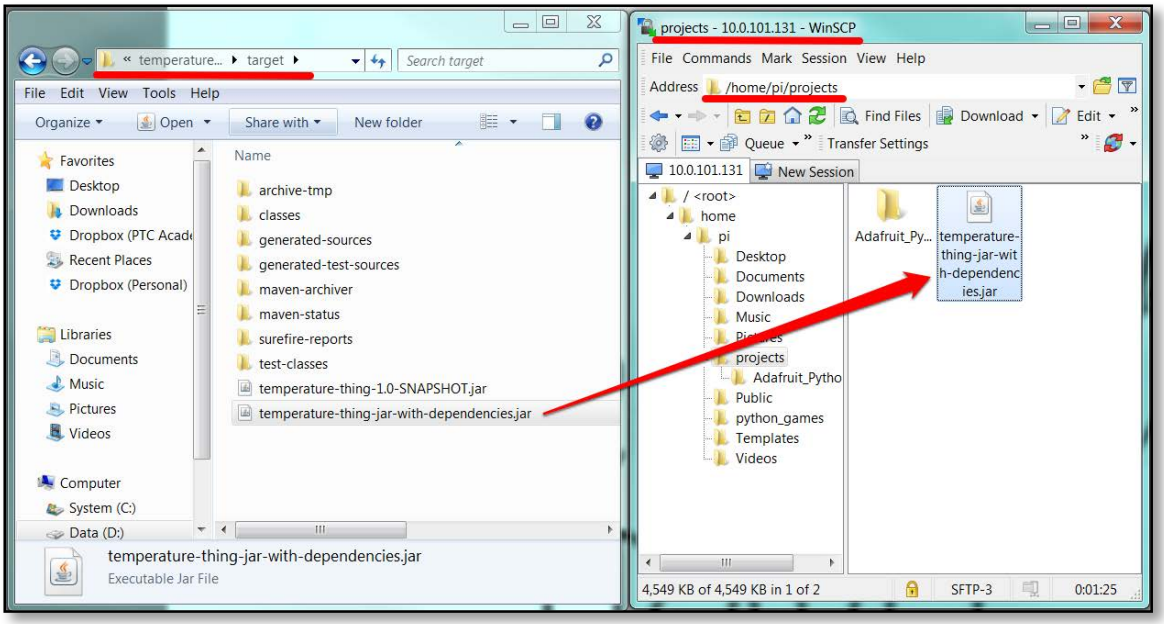
5. Open the Command Prompt on your computer and go to the directory where the ThingWorx files were extracted (It should be your home directory). The command line is *cd*.
6. Once in the directory, run the command **[*mvn package*]** to create the release JAR file that enables Java to communicate with ThingWorx.



7. Run the command **[*mvn clean install*]** to later move the .jar files to a specific location in the Raspberry Pi.

***NOTE:** Once the JAR package is built, you can deliver it to the Pi for testing.*

8. Next, turn on the Raspberry Pi and manually, using WinSCP or similar terminal emulator, transfer the created JAR file by copying from your PC from the [temperature-thing\target] folder and paste it to the "projects" folder in the Raspberry Pi.

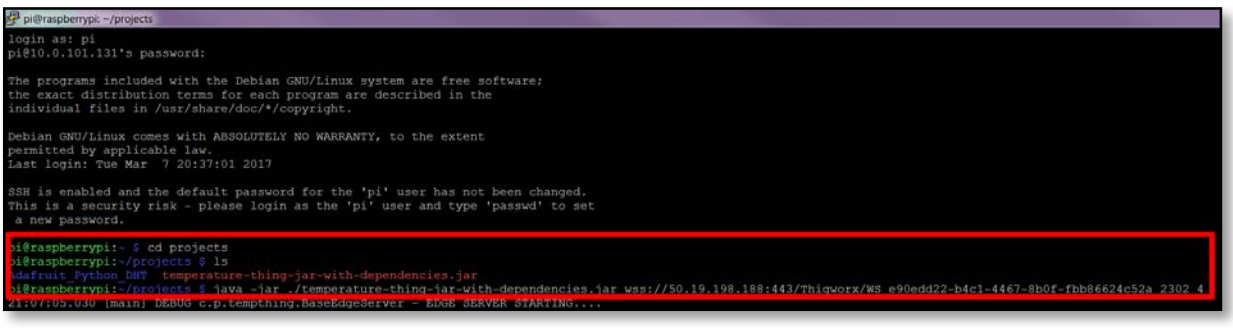


Configure the Raspberry Pi

The following steps will start the java agent from the target directory.

9. Open the Terminal window on your Raspberry Pi
10. Navigate to the "projects" folder
11. Run the following command line:

    java –jar ./<jar_project_name.jar> wss://<ThingWorx Server URL or ThingWorx Server IP Address>:443/Thingworx/WS <Application Key> <Optional Simulated or sensor and GPIO number>

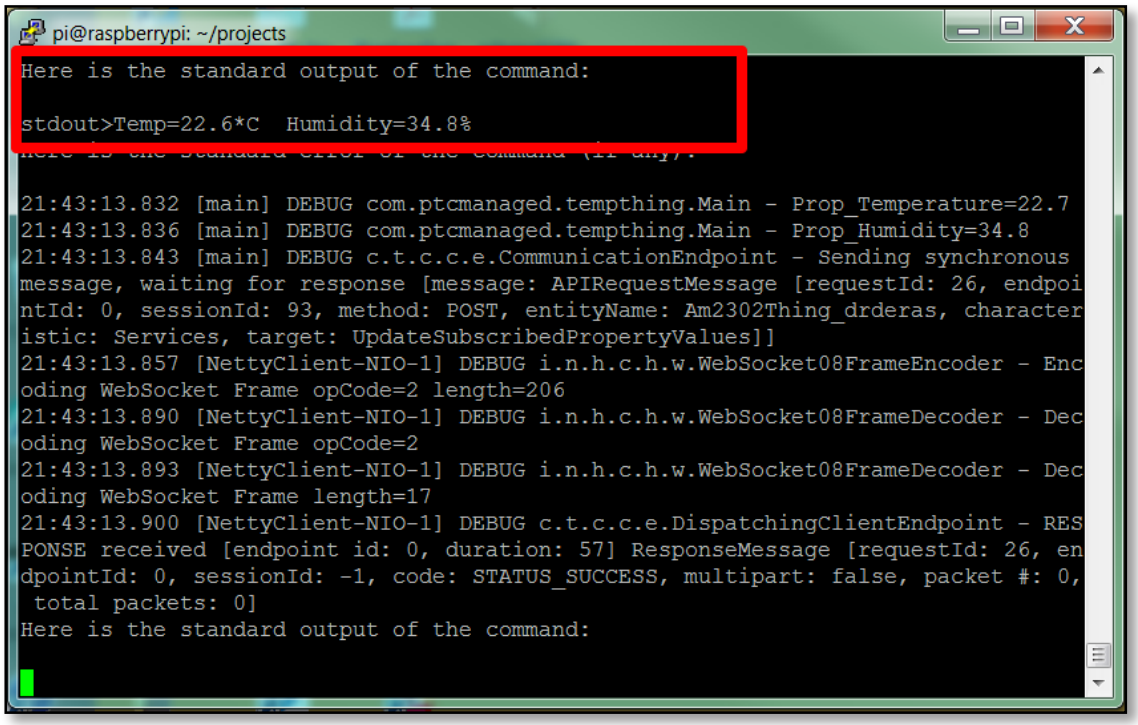


NOTE: *When the Optional Simulated parameter is provided then no hardware will be required to run this example. It will use randomly generated temperature and humidity values.*

NOTE: *For better connectivity, it is recommendable to use the IP address instead of the URL for the ThingWorx server. Also, if the server uses http then the URL/IP should start with [ws:], otherwise if using https the URL/IP should start with [wss:]*

Here are two command examples:

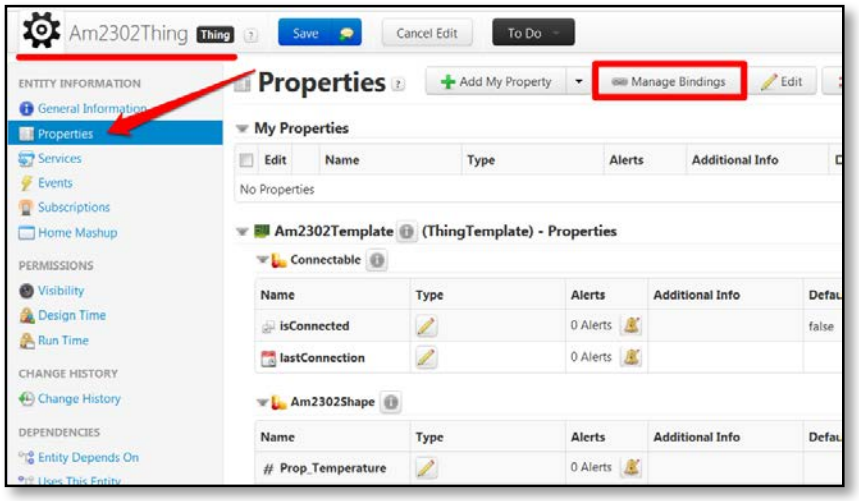
- java -jar ./temperature-thing-jar-with-dependencies.jar wss://maker01.ptcmanaged.com:443/Thingworx/WS 7c73215e-ff40-47ee-9c18-d29a0c5310e0 simulated
- java -jar ./temperature-thing-jar-with-dependencies.jar wss://54.173.81.52:443/Thingworx/WS 7c73215e-ff40-47ee-9c18-d29a0c5310e0 2302 4



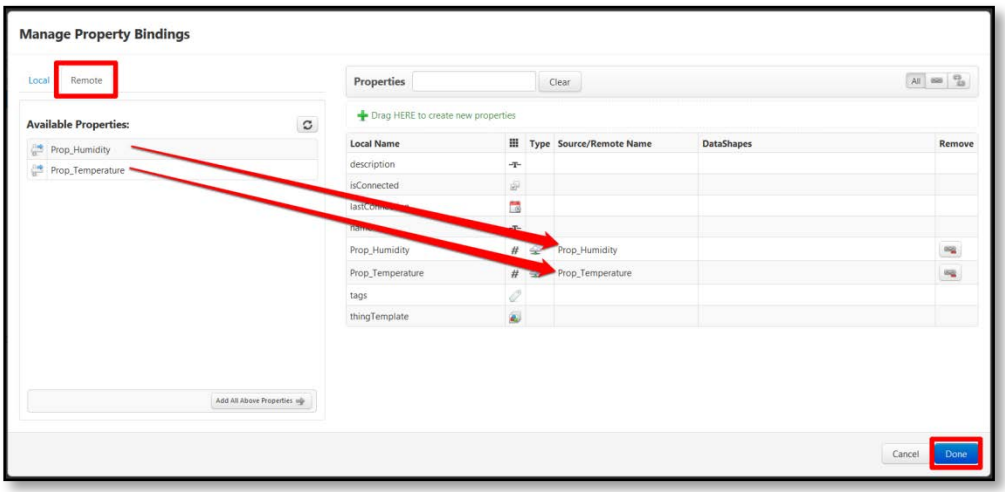
Configure ThingWorx

In this section, you will create the necessary bindings to send data to ThingWorx. With the Raspberry Pi connected and sending data go to your ThingWorx server.

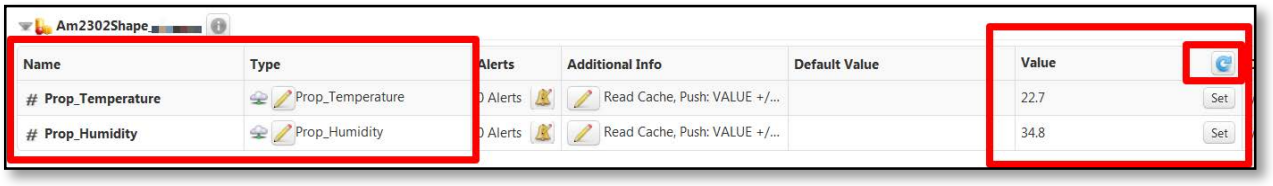
12. In the composer, open your **Am2302Thing** on "Edit" mode.
13. On the left side of the screen, go to **Properties**.
14. At the top of the page, there will be a series of buttons, click **Manage Bindings**.



15. Click the **Remote** tab on the top left side of the pop-up window.
16. Drag the **Prop_Temperature** button under Available Properties and drop it next to the **Prop_Temperature** field on the table to the right under the **Source/Remote Name** field.
17. Drag the **Prop_Temperature** button under Available Properties and drop it next to the **Prop_Temperature** field on the table to the right under the **Source/Remote Name** field.
18. Click **Done** on the bottom of the screen



19. Click the **Refresh Properties** button on the bottom part of the screen next the **Value** field on the **Am2302Shape** property table and see the values update



20. Click **Save**

NOTE: *In order to receive data from the sensor connected to the Raspberry Pi, it is necessary to use the same name given to the thing and properties created in the Model section in the temperature-thing source code.*

This concludes the Connect section of this Project

Build

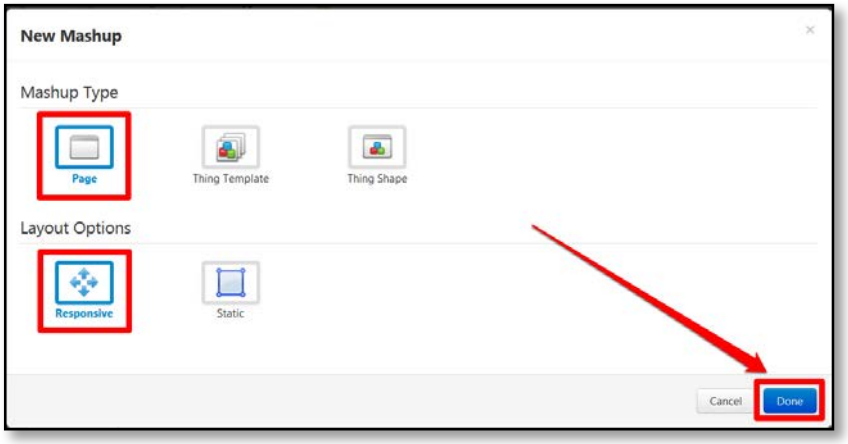
IN THIS SECTION YOU WILL:

- Create a Mashup to display the data obtained from the Am2302 sensor.
- Add widgets to the Mashup.
- Bind widgets and services to make the Mashup functional.

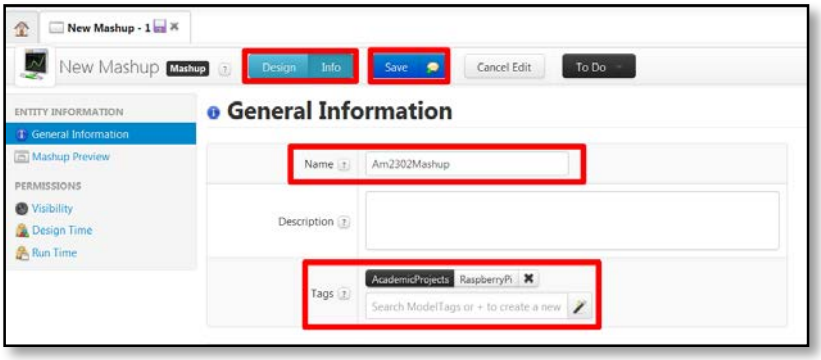
In this final section of the project, we will build the user interface called a Mashup using the ThingWorx Composer.

Create a Mashup

1. On the **Home** tab, **Visualization** section, select **Mashups** and click **New**.
2. Leave predetermined settings (Mashup Type: Page, Layout Options: Responsive) and click **Done**.
3. Click the **Info** button on the top of the screen.
4. Enter the name (Am2302Mashup) in the **Name** section.

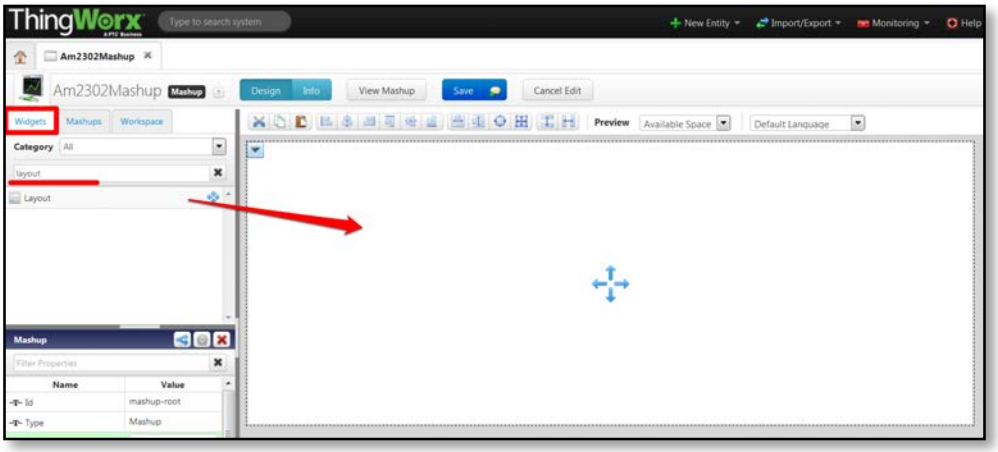


5. Tag the entity by typing the name in the **Model Tag** section (AcademicProjects: RaspberryPi) and selecting the tag from the dialog box.
6. Click **Save** on the top of the screen.
7. Click **Edit** on top of the screen, then click **Design**.



Add Widgets and Customize your Mashup

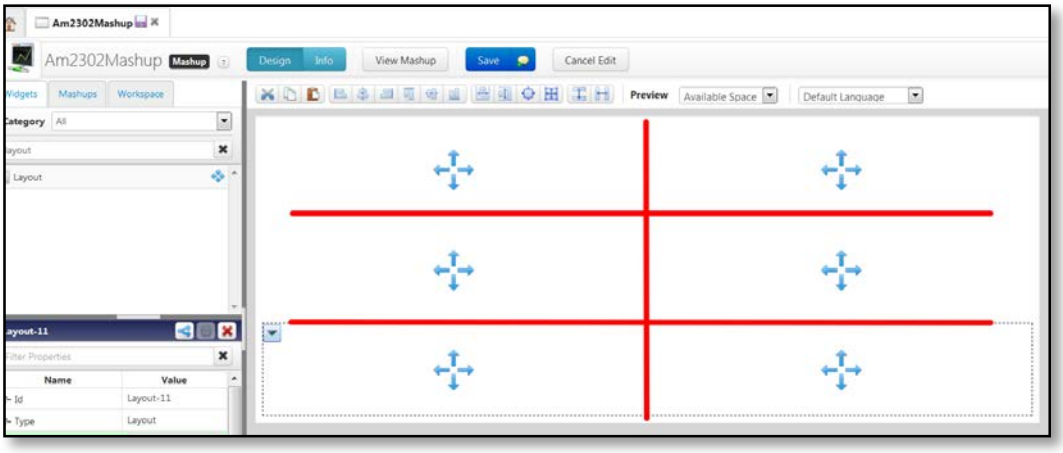
8. On the **Widgets** tab (left side of the screen), filter and/or find the **Layout** widget.
9. Drag the **Layout** widget from the **Widgets** tab and drop it in the white working area in the center of the screen.



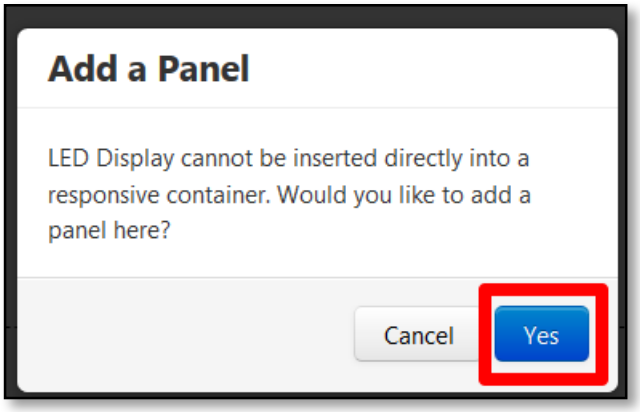
10. On the pop up window, configure as follows:
    a. Vertical Layout
    b. 3 Rows
11. Click **Done**.

12. Back on the **Widgets** tab, filter and/or find the **Layout** widget.
13. Drag the **Layout** widget from the **Widgets** tab and drop it on the top section of white working area.
14. On the pop up window, configure as follows:
    a. Horizontal Layout
    b. 2 Columns
15. Click **Done**.

16. Repeat the same process for the middle and bottom rows, resulting in a 3 rows by 2 columns section.

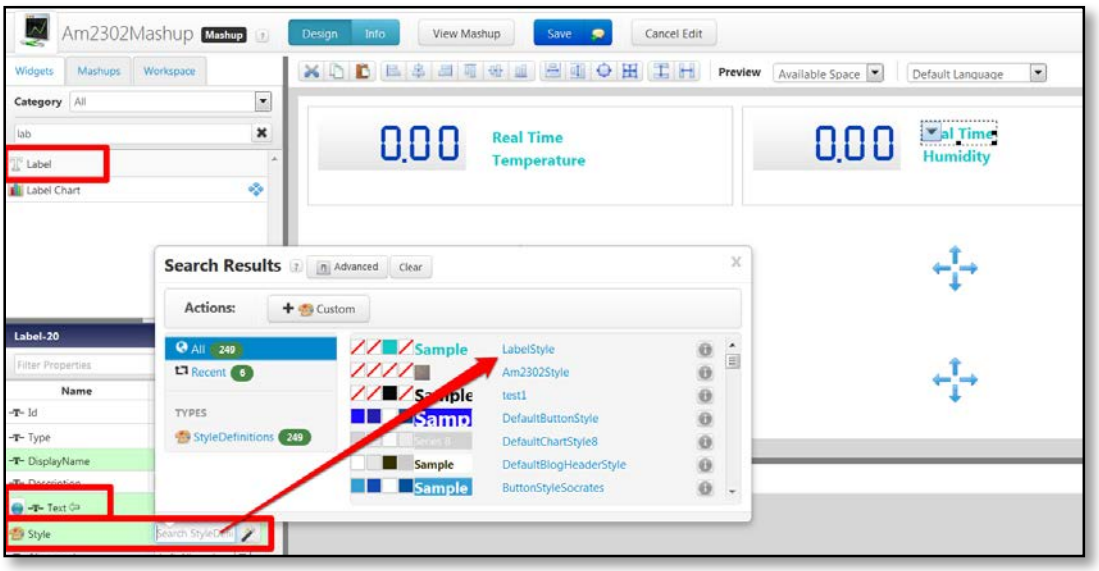


17. On the **Widgets** tab, filter and/or find the **LED Display** widget.
18. Drag the **LED Display** widget from the **Widgets** tab and drop it on the top left cell.
19. On the "**Add a Panel"** popup window click **Yes**.
20. Repeat the previous three steps to add another **LED Display** widget on the top right cell.



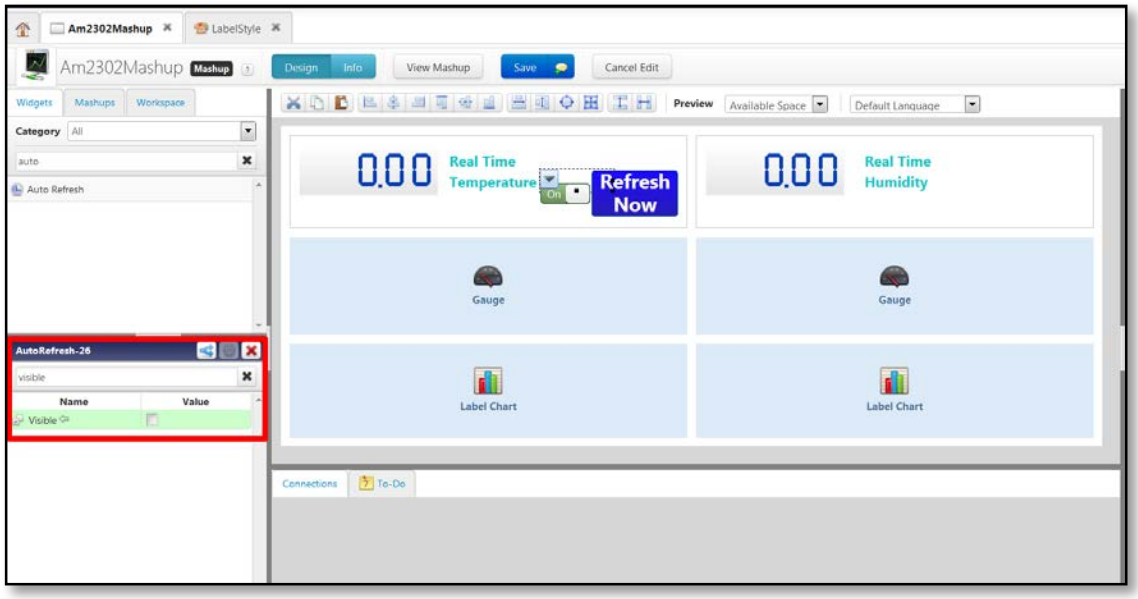
21. On the **Widgets** tab, filter and/or find the **Label** widget.
22. Drag the **Label** widget from the **Widgets** tab and drop it on the top left cell below the **LED Display** widget.
23. While the **Label** widget is still selected, go to the bottom left of the screen and search for **Text** in the Properties list.
24. Type *Real Time Temperature* on the **Text** field and hit Enter.
25. Go back to the **Properties** list and search for **Style.**
26. Click on the **x** next to the color palette on the **Style** field.

27. In the now open field next to Style, start typing *LabelStyle* and select it from the Search Results dialog box.
28. Drag another **Label** widget from the **Widgets** tab and drop it in the top right cell below the **LED Display**.
29. While the **Label** widget is still selected, go to the bottom left part of the screen, search for **Text** in the Properties list.
30. Type *Real Time Humidity* on the empty textbox and hit enter.
31. Go back to the Properties list and search for **Style.**
32. Click on the **x** next to the color palette on the **Style** field.
33. In the now open field next to Style, start typing *LabelStyle* and select it from the Search Results dialog box.

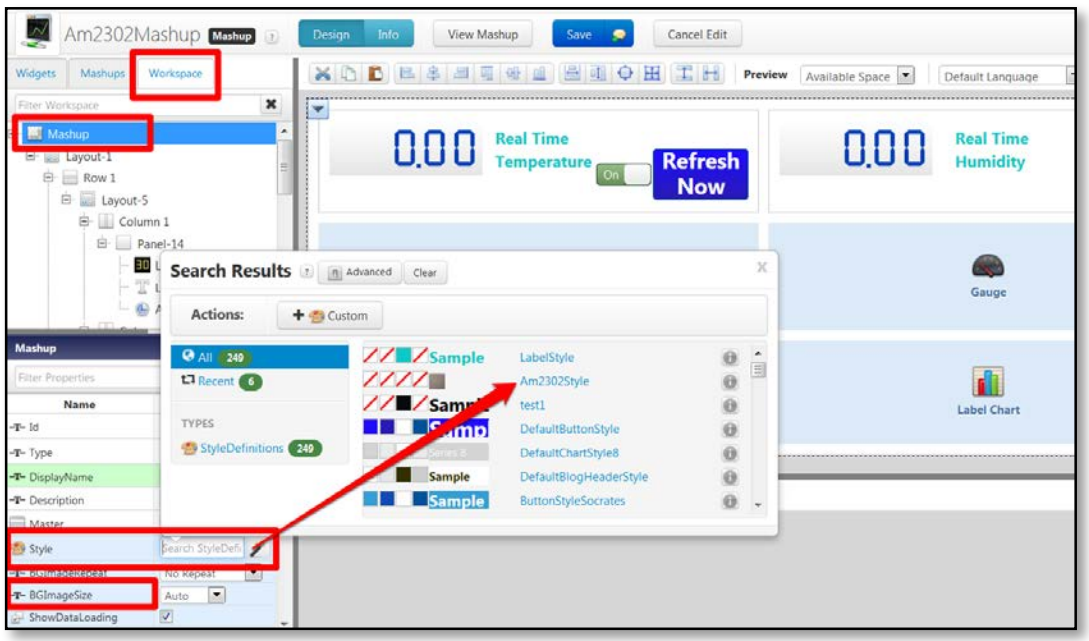


34. Back on the **Widgets** tab, filter and/or find the **Gauge** widget.
35. Drag the **Gauge** widget from the **Widgets** tab and drop it on middle left cell.
36. Drag another **Gauge** widget from the Widgets list and drop it on the middle right cell.
37. On the **Widgets** tab, filter and/or find the **Label Chart** widget.
38. Drag the **Label Chart** widget from the **Widgets** tab and drop it on the bottom left cell.
39. Drag another **Label Chart** widget from the **Widgets** tab and drop it on the bottom right cell.
40. On the **Widgets** tab (left side of the screen), filter and/or find the **Auto Refresh** widget.
41. Drag the **Auto Refresh** widget from the **Widgets** tab and drop it on the top left cell to the left of the **LED Display**.
42. While the **Auto Refresh** widget is still selected, go to the bottom left part of the screen, search for **Visible** in the Properties list.

43. Uncheck the **Visible** field.

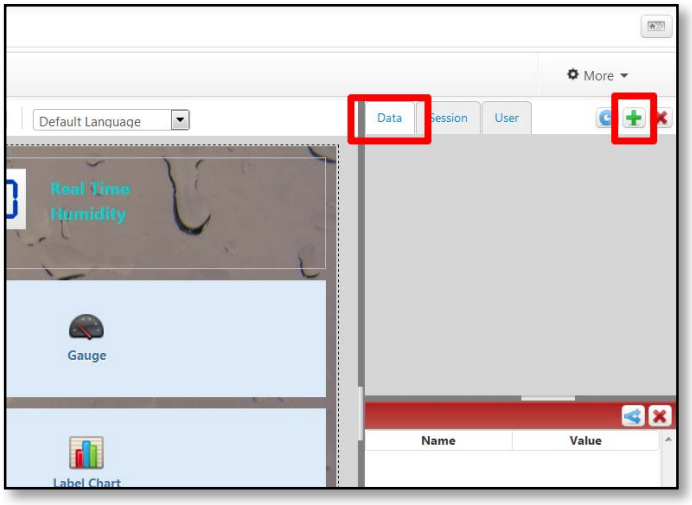


44. To customize your Mashup, go to the top left of your screen and select the **Workspace** tab.
45. Click and select **Mashup** from the **Workspace**.
46. On the bottom left part of the screen, search for **Style** in the Properties list.
47. Click on the **x** next to the color palette.
48. In the now open field next to Style, start typing *Am2302Style* and select it from the Search Results dialog box.
49. Search for **BGImageSize** in the Properties list and select **Cover** from the drop-down menu.
50. Click **Save** on the top of your screen to save your work.

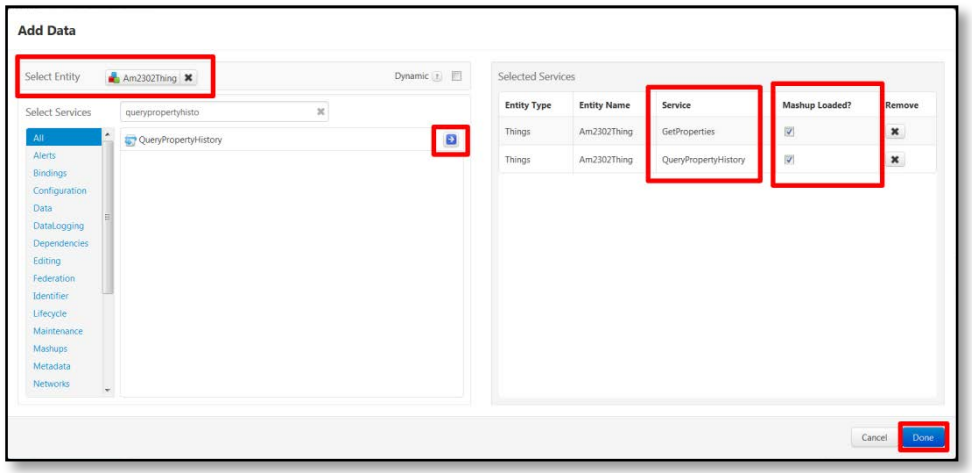


Bind Widgets and Services

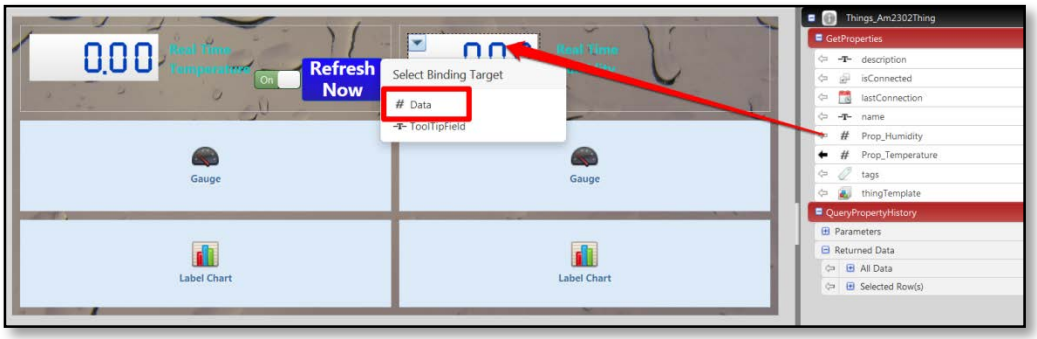
51. On the Data tab (right side of the screen), click the green “plus” sign to the right of the tab.



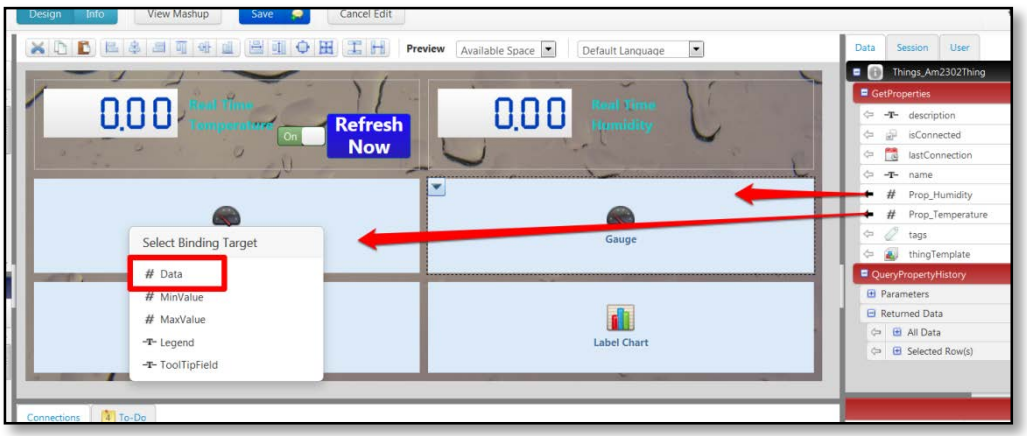
52. On the pop-up window, **Select Entity** section, start typing the name of the thing you created (Am2302Thing) and select it from the **Search Results** box.
53. On the **Select Services** section, type *GetProperties* to filter the **Get Properties** service.
54. Click the blue arrow pointing right to select the **Get Properties** service.
55. Select the **Mashup Loaded?** checkbox.
56. On the **Select Services** section, type *QueryPropertyHistory* to filter the **Query Property History** service.
57. Click the blue arrow pointing right to select the **Query Property History** service.
58. Select the **Mashup Loaded?** checkbox.



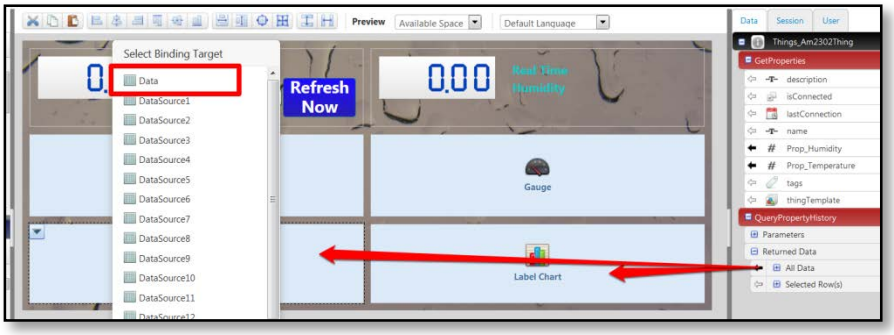
59. Expand both red *service* bars on the right side of the screen by clicking on the + icon next to the name of each service.
60. Bind the **LED Display** widgets:
    a. Select the **LED Display** on the left.
    b. Drag the **Prop_Temperature** property under the **GetProperties** service red bar and drop it on the **LED Display** on the left side of the working area.
    c. Select **Data** as the Binding Target in the pop-up window.
    d. Select the **LED Display** on the right.
    e. Next, drag the **Prop_Humidity** property under the **GetProperties** service red bar and drop it on the **LED Display** on the left side of the working area.
    f. Select **Data** as the Binding Target in the pop-up window.



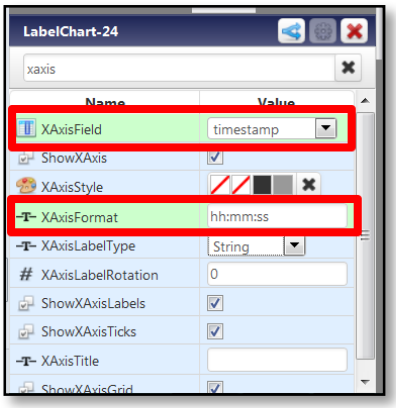
61. Bind the **Gauge** widgets:
    a. Drag the **Prop_Temperature** property under the **GetProperties** service red bar and drop it on the **Gauge** on the left side of the working area.
    b. Select **Data** as the Binding Target in the pop-up window.
    c. Next, drag the **Prop_Humidity** property under the **GetProperties** service red bar and drop it on the **Gauge** on the left side of the working area.
    d. Select **Data** as the Binding Target in the pop-up window.

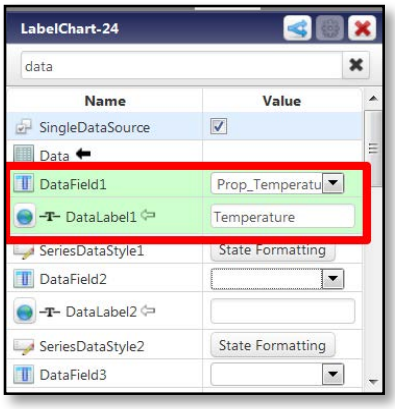


62. Bind the **Label Chart** widgets:
    a. Under the expanded **Query Property History** service red bar expand **Returned Data**.
    b. Drag the **All Data** feature under the Returned Data grey bar and drop it on the **Label Chart** on the left side of the working area.
    c. Select **Data** as the Binding Target in the pop-up window.
    d. Repeat the same process for the Label Chart on the right side

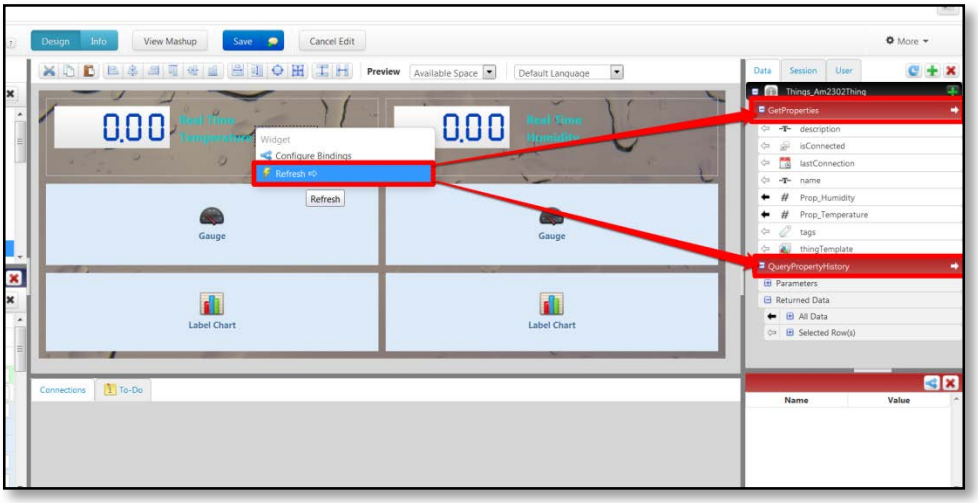


    e. Click to select the Label Chart on the left side of the screen and configure the properties as follows:
        - Chart Type: Line/Marker
        - XAxisField: TimeStamp
        - XAxisFormat: hh:mm:ss
        - XAxisLabelType: Date/Time
        - DataField1: Prop_Temperature
        - DataLabel1: Temperature
    f. Click to select the Label Chart on the right side of the screen and configure the properties as follows:
        - Chart Type: Line/Marker
        - XAxisField: TimeStamp
        - XAxisFormat: hh:mm:ss
        - XAxisLabelType: Date/Time
        - DataField2: Prop_Humidity
        - DataLabel2: Humidity

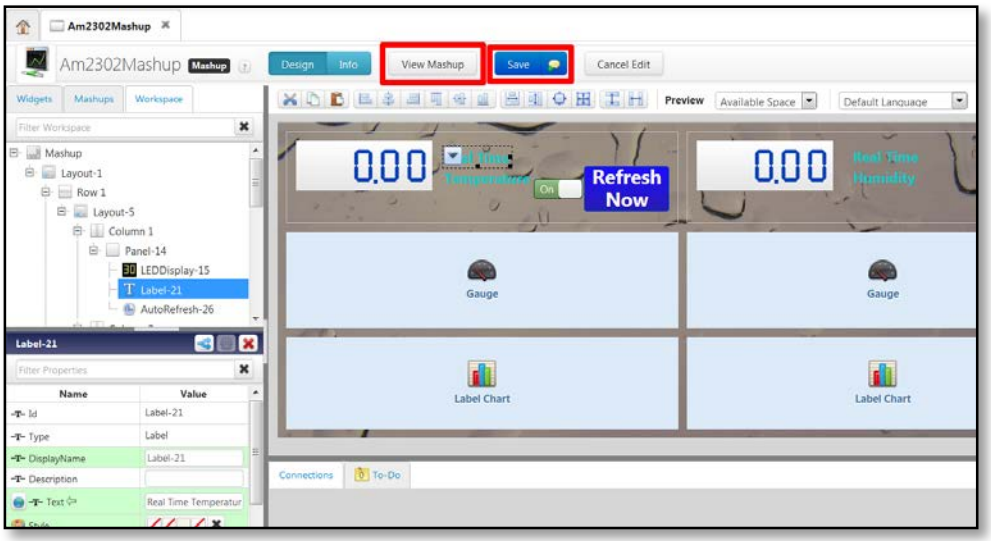




63. Bind the **Auto Refresh** widget:
    a. Click the Auto Refresh button, hover over the left arrow of the widget and select **Refresh** from the dropdown list.
    b. Drag the **Refresh** option and drop it over the **GetProperties** service red bar.
    c. Drag the **Refresh** option and drop it over the **QueryPropertyHistory** service red bar.
    d. Select the **Button** widget, once selected go to the bottom left part of the screen to the list of properties of the selected widget (Button).
    e. Filter and/or search for the **Refresh Interval** property. Type *3* in the text box and hit "Tab" or "Enter", this gives a refresh interval of 5 seconds.



64. Click **Save** on top of the screen



65. Click **View Mashup**

This concludes the Project

Additional Resources

Other Projects:
Build a Basic ThingWorx Mashup

Do you want to learn more about ThingWorx?
Visit our webpage for more information about the Academic Program at PTC.

Other Resources:
Learn the Fundamentals of IoT Development with ThingWorx on Udemy

Questions?
Join us in the ThingWorx Community Site