

---

# IoT Learning Lab

*A Hands-On Workshop*

*O'Reilly Media, Inc. and PTC, Inc.*

Beijing • Boston • Farnham • Sebastopol • Tokyo

**O'REILLY®**

## **IoT Learning Lab**

by O'Reilly Media, Inc. and PTC, Inc.

---

# Table of Contents

<b>Preface.....</b>	<b>v</b>
<b>1. From Thing to Cloud.....</b>	<b>7</b>
Connect to Your Raspberry Pi with SSH	9
Download the Agent	10
Compile the Agent	11
Configure the Agent's Properties	12
Create Your User In ForgeRock and Claim Your Thing	13
Watch the Data Stream into ThingWorx	17
<b>2. Mashing Up the Data.....</b>	<b>21</b>
A Quick Tour of the Agent	21
Creating the Mashup	25
Clone the Starter Mashup	25
Adding the Gauge	27
Labeling the Gauge	29
Viewing the Mashup	30
Adding data to the Graph	30
<b>A. Answer to the SHT21 Exercise.....</b>	<b>33</b>
<b>B. Configuring and Connecting the Device.....</b>	<b>35</b>



---

# Preface

O'Reilly and ThingWorx are teaming up to teach everyone about the Internet of Things. It's not just a buzzword any more....it is the future. As the leading IoT Platform, ThingWorx is uniquely positioned to offer its expertise in the emerging market. O'Reilly has long been a name that engineers trust to provide the right knowledge about new technology at the right time. The two companies combined create a powerhouse to help everyday folks learn about the emerging trend and walk away with practical hands-on knowledge that can be taken back to the day job.

## Welcome to the IoT Learning Lab

The inaugural IoT Learning Lab was held at LiveWorx 2016 in Boston, Massachusetts. In the workshop, participants learned to build solutions based on ThingWorx, Analog Devices' sensor modules, ForgeRock's digital identity and access management, and Raspberry Pi.

## Who This Book Is For

Complete beginners; anyone new to the IoT, Raspberry Pi, or even electronics. Strategists, sales folks, decision makers can all have fun with this project. Our goal was to make this course as accessible as possible.

# What You Need

## *Raspberry Pi 3 Model B*

\$35. Sold by many vendors, such as **Newark**.

## *Micro SD Card (8GB or more)*

\$4 or more. Available from many sources, including **CDW**.

You will need a micro SD card with the **Raspbian** operating system installed on it. The Raspberry Pi foundation has **instructions** for installing Raspbian on a micro SD card. Workshop attendees will receive a micro SD card with the operating system preinstalled.

## *USB Micro B cable*

\$5 or more. Available from many places such as **CDW**.

## *ADI FAE Board*

Although this is not yet commercially available, you can obtain many of the sensors that are on this board.

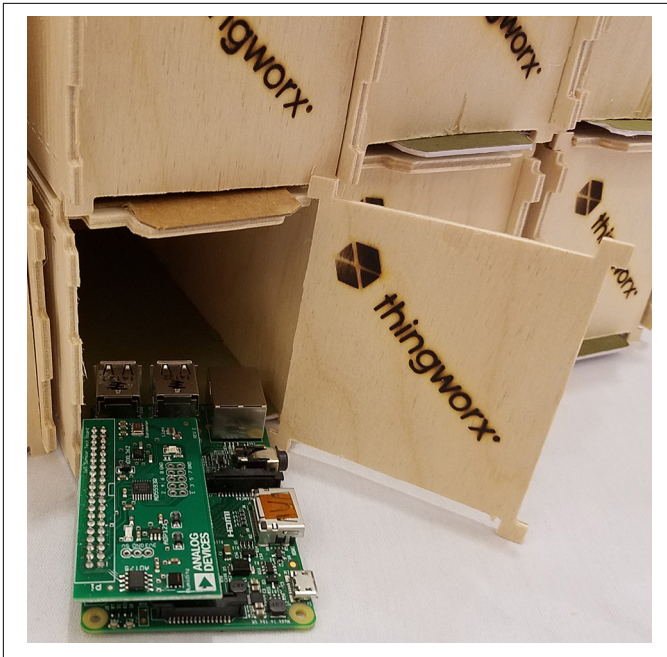
## *Container Box model*

This is a CNC-cut, customized design just for the IoT Learning Lab based on a design by **Iron Mammoth** and used with permission. You can download and cut your own version from <http://www.thingiverse.com/thing:1611873>

# From Thing to Cloud

Here's where we put the pieces of the IoT Learning Lab together. At the beginning of the workshop, you should have received a Raspberry Pi, a micro SD card, and some other components. Let's get talking to ThingWorx!

If it's not installed already, install the Analog Devices FAE board by plugging it into the Raspberry Pi header block. The board should extend over the Raspberry Pi, not away from it, as shown in **Figure 1-1**.



*Figure 1-1. The Raspberry Pi sporting an ADI FAE board*

There are three parts to the software stack here, and we'll be touching various parts of these throughout the process:

#### *The Agent*

This is a program that runs on an IoT device (like your Raspberry Pi) and reports on the status of some physical asset or of the environment around the device.

#### *ThingWorx*

This is PTC's IoT technology platform that allows developers to quickly bring together a constellation of IoT devices and deploy applications.

#### *ForgeRock*

This user and identity management platform lets workshop participants instantly create a user account and provision their IoT device in the ThingWorx platform.



# Connect to Your Raspberry Pi with SSH

Boot up your Pi by connecting it to USB power. The lights will blink for a while, and after about a minute, it should be connected to the Wi-Fi network. Once it's connected to the network, you'll be able to connect to it using SSH (Secure SHell). Mac and Linux come with an SSH client, `ssh`, that you can run from the Terminal by typing `ssh <HOSTNAME>`. On Windows, you'll need to install an SSH client, such as PuTTY, which you can download from [PuTTY](#).

You'll need to know your Raspberry Pi's hostname. If you're using one of the workshop Pis, your instructor can tell you its name. If you configured the Pi yourself, use that hostname. On some network configurations, you'll need to append `.local` to the hostname to connect to it. But try it without the `.local` first.

## *Windows*

Run PuTTY from the Start Menu. If it's already running, click the icon in the top-left corner and choose New Session from the menu that appears.

Under Connection Type, select SSH. Type your Raspberry Pi's hostname (for example, `lwpi01` or `lwpi01.local`) into the Host Name field. If you want to make it easy to connect again, type a name under Saved Sessions (such as LiveWorx Pi) and click Save.

Click the Open button (at the bottom of the screen).

## *Mac or Linux*

Open up a Terminal window, and type the command `ssh pi@HOSTNAME` (replace `HOSTNAME` with the name you gave the Pi earlier), for example `ssh pi@lwpi01`.



The first time you connect, you'll see a warning indicating that the Raspberry Pi is unknown. You'll see a fingerprint code, which you could **match to** that of your Raspberry Pi if you'd like. You'll need to accept the fingerprint before you can connect.

If you're prompted for a user name (`login as:` in PuTTY), use `pi`). Supply the password (the default is `raspberry`) and you're now connected!

The remaining instructions will take place within an SSH session.

## Download the Agent

We've published a repository with the source code for this project. It includes the agent source code and supporting libraries.



This repository does not contain ThingWorx or any other software for which you must obtain a license. If you don't have access to ThingWorx, look for the ThingWorx Developer Trial Edition, which you can use to deploy your own ThingWorx instance for up to 120 days. Visit <http://developer.thingworx.com/>, log in, and visit the dashboard to obtain the Developer Trial Edition.

If you're not already connected to the Raspberry Pi directly with a keyboard/mouse, or remotely over SSH or PuTTY, do so now, and run the following command to check out the source code into your current working directory:

```
git clone https://github.com/TWTPGPUB/OreillyLearningLab.git
```



Unless otherwise directed, you will need to type commands such as this one into your SSH client (PuTTY on Windows, `ssh` on Mac/Linux). Your SSH client sends those commands over the network to the Pi, and after the Pi executes the command, it will display the results in your SSH client. In PuTTY, you can paste into the window by right-clicking (Control-V won't paste into the window).

Next, `cd` into the `OreillyLearningLab` directory and type `ls` to have a look around:

```
cd OreillyLearningLab
ls
```

You'll see some files and subdirectories there.



We will occasionally make updates to the code, so you should periodically issue the command `git pull` to pull down the latest version of the source code.

## Compile the Agent

Change directory into the `OLL_Agent` directory, and type `make`. Then use `ls -l twc_agent` to confirm the agent was compiled successfully:

```
cd OLL_Agent
make
ls -l twc_agent
```

You should see something similar to the following output:

```
-rwxr-xr-x 1 pi pi 1155048 Jun  7 10:10 twc_agent
```

# Configure the Agent's Properties



The *agent.properties* file should come pre-populated with the host, port, and AppKey for your agent to connect to the ThingWorx instance. If any changes are needed, your instructor will supply you with the host, port, and AppKey for the ThingWorx instance. But, if you're running ThingWorx on your own, you can use the ThingWorx instance you set up.

Still working in your SSH client, use one of the Pi's built-in plain text editors (such as nano or vi) to edit the *agent.properties* file. For example, you can open it in the nano editor by typing *nano agent.properties* and pressing Enter. Next, make the following changes and then save the file:



The nano editor displays a list of commonly-used commands at the bottom of the screen. To save a file, press Control-X, answer Y when it asks "Save modified buffer?", then press Enter (don't change the name of the file before pressing Enter).

## *tw\_host*

If you are in the workshop, you probably won't need to change this. Set this to the host name of your ThingWorx instance. Your instructor will supply this, or you can use the Developer Trial Edition. Visit <http://developer.thingworx.com/>, log in, and visit the dashboard to obtain the Developer Trial Edition.

## *tw\_port*

Set this to the port that your instance uses. If you are in the workshop, you probably won't need to change this.

### *AppKey*

Set this to the application key from your ThingWorx instance. If you are in the workshop, you probably won't need to change this.

### *tw\_name*

This should be a unique name that you choose (or that your instructor supplies). Note that this is different from the Raspberry Pi host name. Make note of this name.

Save the file, and then run the agent with this command:

```
sudo ./twc_agent
```

If everything is working correctly, you should be seeing a lot of text scroll by, and you may notice an error message, Entity <NAME> does not exist or is not yet associated with a Thing, which is expected at this point. Let's address that right now.

To stop the agent, press Control-C. Keep it running for now.



If you are running this on your own instance (such as a ThingWorx Developer Trial Edition), you will need to create the Thing yourself. If you don't know how to create a Thing, please visit the ThingWorx Raspberry Pi QuickStart at <http://developer.thingworx.com/> to learn how.

## Create Your User In ForgeRock and Claim Your Thing

If you're in the workshop, you'll have access to the ForgeRock Identity Platform where you can create a user account, take ownership of your shipping container, and share data from it with your Harbor Master.

### Step 1

Open a web browser and navigate to <https://forgerock-shipping.forgerocklabs.org:2443>. You will need internet connectivity for this to work. See Figure 1-2. Click the Register button.

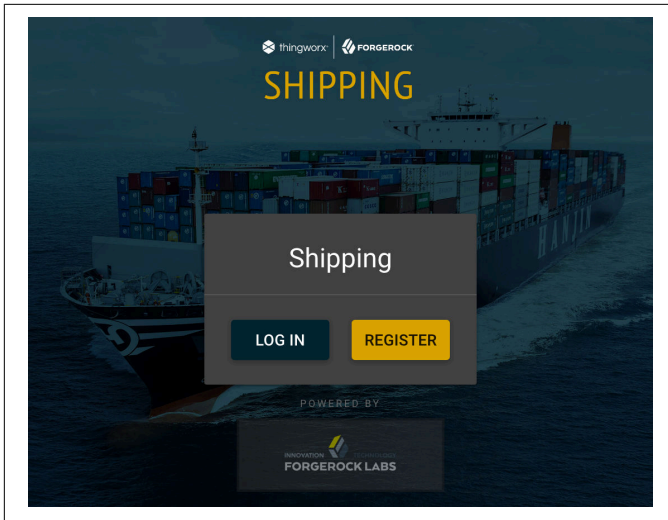
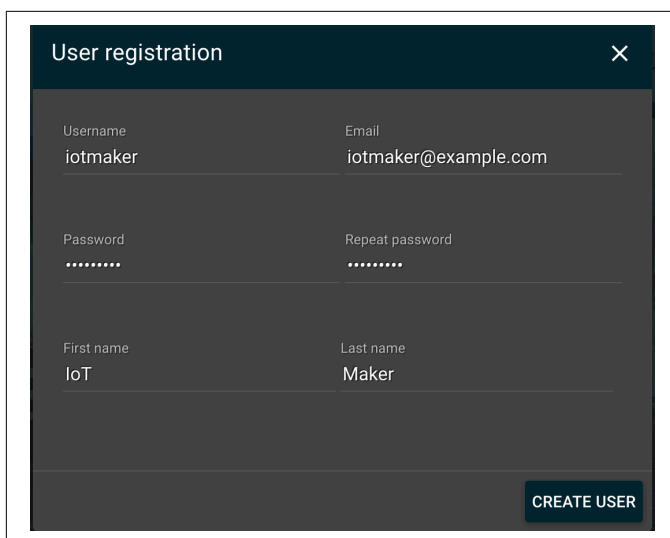


Figure 1-2. The ForgeRock landing page

### Step 2

The user registration page appears, as shown in Figure 1-3. Select a unique username for the first field, and supply all the information for the remaining fields. Choose a unique password that you don't use elsewhere.

Next, click the CREATE USER button. You'll see a message in the lower-left corner of the screen notifying you that the account creation was successful. If not, try using a different user ID or check with your instructor for help.

A dark-themed user registration form titled "User registration" with a close button (X) in the top right corner. The form contains six input fields arranged in three rows. The first row has "Username" (filled with "iotmaker") and "Email" (filled with "iotmaker@example.com"). The second row has "Password" (filled with ".....") and "Repeat password" (filled with "....."). The third row has "First name" (filled with "IoT") and "Last name" (filled with "Maker"). A "CREATE USER" button is located at the bottom right of the form.

User registration

Username: iotmaker

Email: iotmaker@example.com

Password: .....

Repeat password: .....

First name: IoT

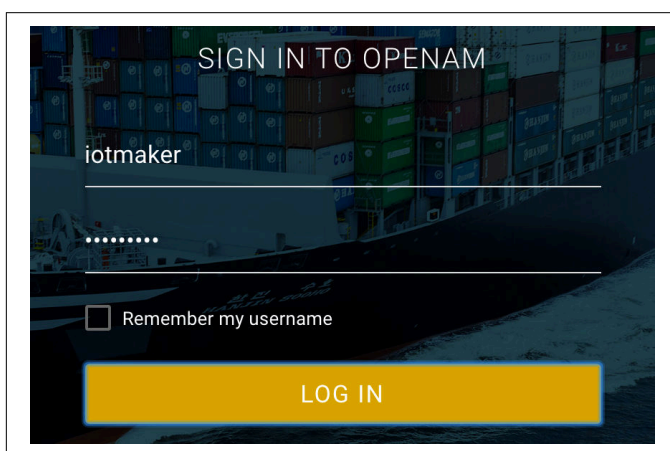
Last name: Maker

CREATE USER

Figure 1-3. Registering in ForgeRock

### Step 3

You'll be back at the landing page shown back in [Figure 1-2](#). Click the LOG IN button, and supply the user name and password you used when you registered your account, as shown in [Figure 1-4](#).

A log in form titled "SIGN IN TO OPENAM" overlaid on a background image of a cargo ship. The form has two input fields: "Username" (filled with "iotmaker") and "Password" (filled with "....."). Below the password field is a checkbox labeled "Remember my username". A large yellow "LOG IN" button is at the bottom of the form.

SIGN IN TO OPENAM

Username: iotmaker

Password: .....

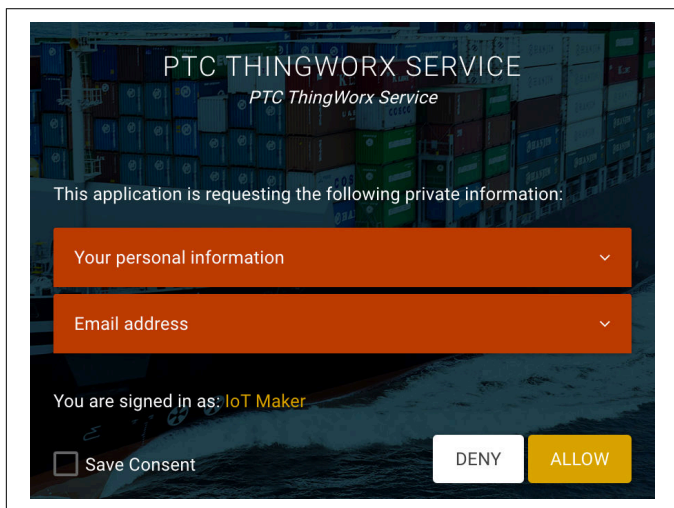
☐ Remember my username

LOG IN

Figure 1-4. Log in

#### Step 4

Now you'll be asked to allow the ThingWorx Service to have access to your ForgeRock account. Click ALLOW as shown in [Figure 1-5](#).



*Figure 1-5. Grant permission to connect ThingWorx to your ForgeRock account*

#### Step 5

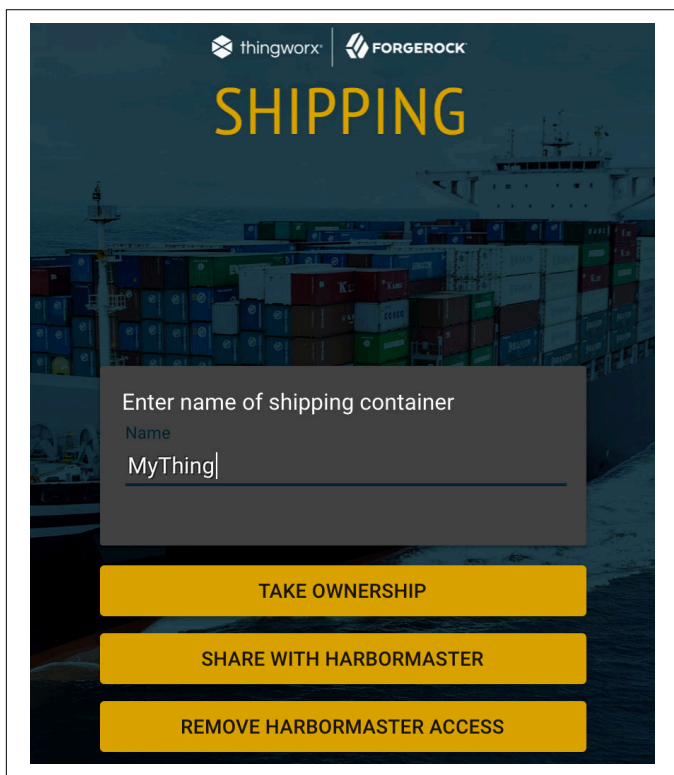
The screen shown in [Figure 1-6](#) will appear. Now you're ready to take ownership of your shipping container. Your shipping container name needs to match the `tw_name` in your `agent.properties` file (see [“Configure the Agent's Properties”](#) on page 12).

Type in the shipping container name, and click the TAKE OWNERSHIP button. Upon taking ownership, the ForgeRock Identity Platform will provision your account name and password inside ThingWorx, create an ownership association between your user identity and your device (the shipping container). It will also create a Thing inside of ThingWorx where you can see data streaming in.





If you don't take ownership of a Thing, your user won't be created in ThingWorx.



*Figure 1-6. Taking ownership of your device*

## Watch the Data Stream into ThingWorx

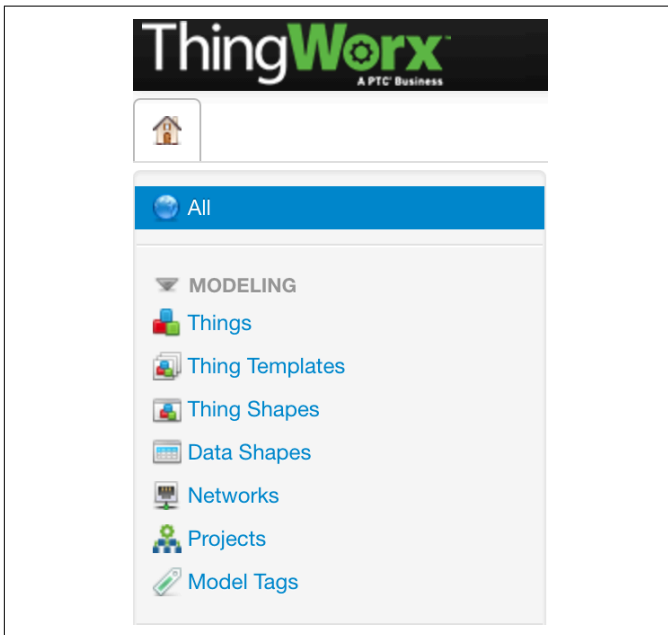
Let's find the data that's streaming from your Raspberry Pi into ThingWorx:

Now, visit the ThingWorx instance we've set up for the workshop. It's at <https://thingworx.forgerocklabs.org/Thingworx/Composer/>. Log in with the username and password you created in ForgeRock.



You are likely to see a security warning for this site indicating that the web site's certificate is invalid. This is because the site we are using for testing is temporarily using a self-signed certificate and may be resolved in a future workshop. For now, it's OK to proceed to the site. In FireFox, click I Understand The Risks, then click Add Exception. In Chrome, click Advanced, then click "Proceed to thingworx.forgerock-labs.org (unsafe)".

1. On the left side of the ThingWorx window, make sure the Home tab is selected, and click Things as shown in [Figure 1-7](#).



*Figure 1-7. Click Things*

2. A list of things will appear to the right. Locate your Thing in the list and click it. The Thing's General Information will appear (see [Figure 1-8](#)).

**General Information**

Name	HeresTheThing
Description	<div></div>
Project	<div>Search Projects or + to </div>
Tags	<div>Search ModelTags or + to </div>
Thing Template	<div> RPIwFAE</div>
Implemented Shapes	<div>Search ThingShapes or + </div>

*Figure 1-8. Your Thing's general information*

3. Click Properties and check out the data values. You can load the latest values by clicking the blue refresh icon in the Value column header ([Figure 1-9](#)).

Name	Type	Value	
# temperature_...	temperature_s...	86.0	<div>Set</div>
# temperature_...	temperature_a...	86.3	<div>Set</div>
-T- Name	Name		<div>Set</div>
Location	Location	-71.5400 : 41...	<div>Set</div>
# light	light	7.0	<div>Set</div>

*Figure 1-9. Data is streaming into ThingWorx*



# Mashing Up the Data

Now that you've got your Thing talking to ThingWorx, let's dig in a little deeper and show you how to add a new data item, and also how to create your own Mashup.

## A Quick Tour of the Agent

Let's have a look at the source code to the agent. If you aren't already in the `OLL_Agent` subdirectory, use the `cd` command to go there, and then type the following command. This will let you view the source code using the *less* system pager to scroll through the text of the file:

```
less src/twc_oll_agent.c
```

Press `f` to move forward one screen, `b` to move backwards one screen, the arrow keys to move one line at a time, and `q` to quit.

You don't need to read the whole source file, but look for the following key sections:

### *INCLUDES*

This is where we pull in all the header files from the standard C library, and application-specific support

functions, and external libraries (including the ThingWorx C SDK).

## MACROS

This is where we define C macros used by the agent. A macro is a piece of text (such as `APPLICATION NAME`) that is replaced by a value (such as “O’Reilly Learning Lab Agent”) wherever it appears in the agent program.

## STRUCTS

These are the data structures used throughout the agent program. Data structures are contiguous blocks of memory that hold data values in a structured format.

## GLOBAL VARIABLES

These are variables that are used throughout the agent program. This includes the `properties` data structure, which is of the type `local_properties`, the data structure we defined earlier.

## *queryFAE()*

This function accesses the sensors on the FAE board and stores them into the `properties` data structure. That data structure is a global variable, so it can be accessed anywhere in the program.



If you were to add a new sensor reading to your agent, you’d also need to add some code to `queryFAE()` to read the value and insert it into the `properties` data structure. Have a look at the function calls that end with `_measure`, and note the naming conventions they use for variables and properties. The `_measure` function definitions can be found in the sensor type’s corresponding header file (which you can find in the `src/Headers` subdirectory under the `OLL_Agent` directory).

### *writeProperty()*

This function is called by `propertyHandler()` (described shortly) any time a value is set for a property by the ThingWorx platform.

### *sendPropertyUpdate()*

This is the function that takes care of moving data from the `properties` data structure to the ThingWorx platform. The `queryFAE()` function you saw earlier takes care of putting sensor values into that data structure. This function takes care of pushing those values up to the cloud.



If you were to add a new sensor reading to your agent, you'd also need to add a call to `twApi_AddPropertyToList` in `sendPropertyUpdate()` to send it to the ThingWorx platform.

### *propertyHandler()*

This function is called when the server sends a request to write or read a property.

### *resetTask()*

This function is invoked when the server wants to stop the agent from executing and have it perform a clean exit.

### *multiServiceHandler()*

A catch-all for multiple services that might be requested by the server. This hands off control to a separate task to handle the request.

### *blinkPin()*

If the server asks the agent to “blink” a pin (toggle it on and off a specified number of times, which is useful for flashing an LED), this function handles that request.

### *dataCollectionTask()*

This controls how often the agent invokes the `sendPropertyUpdate()` function.

### *ThingWorxTask()*

This is where the ThingWorx magic starts. It initializes the SDK, declares the properties the agent will use, and opens a connection to the server.



If you were to add a new sensor reading to the agent, you'd need to add a call to `twApi_RegisterProperty()` in `ThingWorxTask()`.

Now that you've read through it, quit out of the file by pressing `q`. Make a copy of the file for safe keeping, and open it up for editing in a text editor:

```
cp src/twc_oll_agent.c src/twc_oll_agent.orig
nano src/twc_oll_agent.c
```

There is one sensor that is defined in the `local_properties` struct, the SHT21 temperature sensor. It's also defined as an FAE construct in the `STRUCTS` section of the code (`sht21_t`), and it's correctly initialized in `fae_init()`. However, it's not being queried—nor is it stored in the properties data structure, and it's not being sent to the ThingWorx platform.

Try adding it to the `twc_oll_agent.c` file, then save the file, and recompile the agent with the `make` command.

If the agent is still running, you'll need to shut it down (press `Control-C`) and start it again with `sudo ./twc_agent`.

You'll know you succeeded if you see a value for this sensor appear in your Thing's properties (see [“Watch the Data Stream into ThingWorx” on page 17](#)).





If you need hints, look for the notes like this one earlier in this chapter. If you get really stuck, check [Appendix A](#) for spoilers.

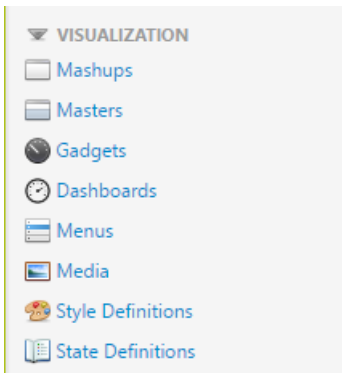
## Creating the Mashup

Now that you have data coming into ThingWorx and you've added a new property, it's time to display all the properties in an attractive and functional mashup.

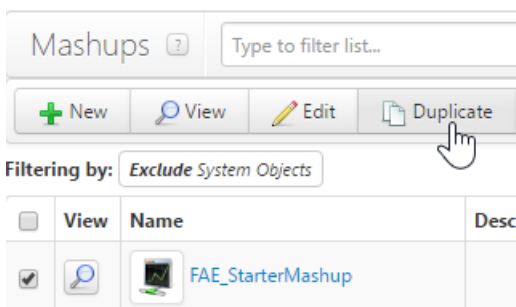
### Clone the Starter Mashup

Make sure you've got the ThingWorx Composer open (see [“Watch the Data Stream into ThingWorx” on page 17](#)). Next:

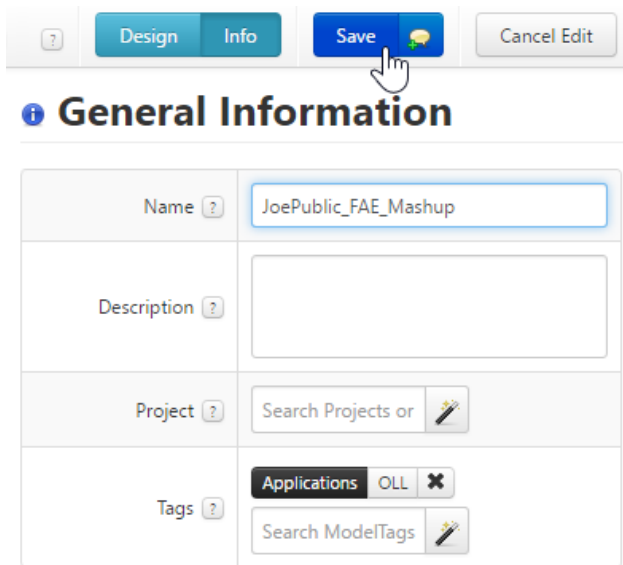
1. Go to the Home tab and select Mashups from the Visualization section:



2. The list of available Mashups will appear on the right. Look for FAE\_StarterMashup, Select the checkbox, and click the Duplicate button up at the top:



3. The Mashup editor opens up. Click on the Info Button if it's not already selected.
4. Give your Mashup a new name that is unique to you, for example JoePublic\_FAE\_Mashup, then click Save:



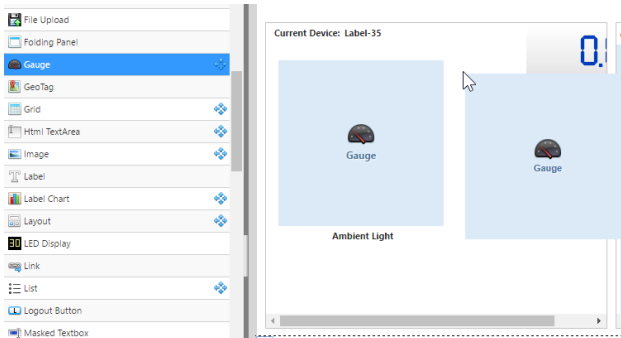
5. You have now created a copy of the mashup and can now edit it as you please! Now, click on the Edit button then make sure the Design button is selected.

The Mashup editor is fully WYSIWYG to make displaying data as easy as possible. Mashups are also dynamic and support different size screens. Most of your data is already

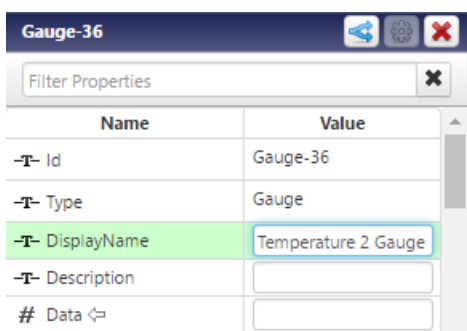
showing in the Mashup but you still need to add something for the temperature sensor you just added. You are going to add two things that represent the same data in different ways: a gauge for instantaneous readout and a graph for long term trending.

## Adding the Gauge

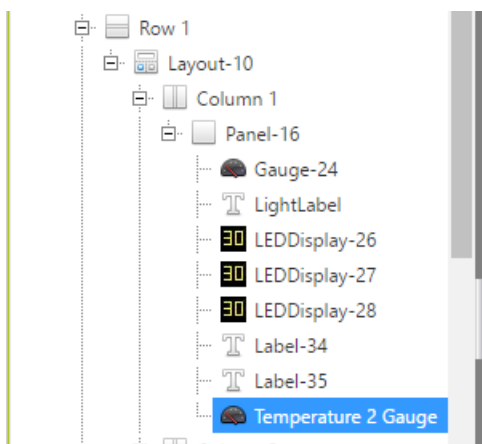
1. In the widgets section on the left, find the Gauge widget.
2. Click and drag it to a spot on the Mashup:
3. Once you release your mouse, the gauge will appear on the Mashup editor and the box in the lower left hand corner will show information about the gauge:



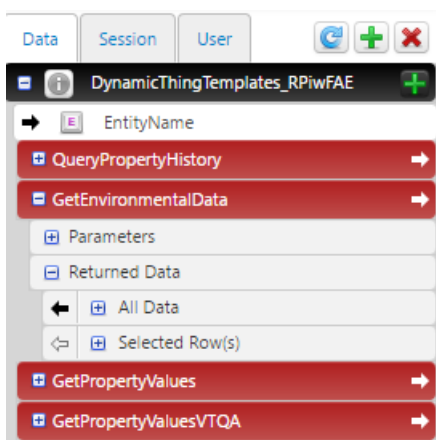
4. Set a Display Name so that you can interact with it more easily. Select the DisplayName box and type in a name such as “Temperature 2 Gauge”. Press Enter when you’re done:



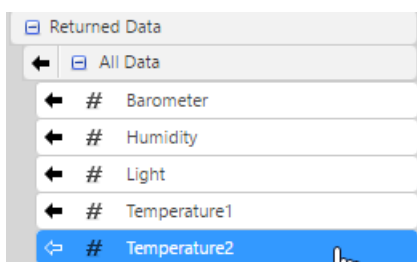
5. If you click on the workspace tab, you will see a tree with all the mashup components and your new gauge will appear with the name you assigned it:



6. Now you need to give it data. Look for the Data tab on the right of the screen, and you'll see that several services have already been added for you. Select the GetEnvironmentalData service and expand it:



7. Next, select Returned Data and make sure it's expanded. Expand the All Data node under it.
8. Notice that you can now see a few of the properties that the service returns. One of them is Temperature2.
9. Click the white arrow next to the Temperature2 and drag that over to the lower left hand properties palette where you see the Data Field. You can also drag it directly on to the gauge and select Data when prompted:



10. Now the gauge is able to display data. Click Save

## Labeling the Gauge

1. Having a gauge is all well and good but how do we know what data it is showing? A simple label will solve

the problem. From the Widgets box, find Label and drag and drop it below your new gauge.

2. With the label selected, scroll down the lower left properties box and select the Text box
3. Type in a name that you would like displayed, such as Temperature 2.

## Viewing the Mashup

1. From the Mashup Editor there is a convenient View Mashup button, click it and the Mashup will open up in a new tab. (You may need to enable popups for this site for it to display).
2. This Mashup interacts with the Thing Template rather than directly with a Thing. We need to tell it what Thing to pull data from. In the browser URL, add the following `&Entity=MYTHINGNAME` (replace MYTHINGNAME with the name of your Thing) and press Enter. Note: Capitalization matters, so the thing name has to match exactly.
3. If done correctly, the Current Device field below the logo will show your device name and you will start to see data showing up.

## Adding data to the Graph

1. The hard part is done! You've gotten data into the platform and have associated it with a gauge. Now we just need to add data to the graph. Return to the Mashup design, then select the graph named WeatherChart
2. On the lower left hand properties box scroll down until you see DataField4.
3. Click on the menu and select the "temperature sht21" option.

4. Click Save and reload the window with your Mashup view, and you should now see the SHT21 sensor added to the legend and populating with data!

Try adding another type of display for the SHT21 sensor. You can follow the earlier instructions, but instead of selecting a Gauge widget, select, for example, an LED Display.





# Answer to the SHT21 Exercise

If you're completely stuck, here is how the various functions need to be modified to support the SHT21 sensor on the ADI FAE board. This is not a complete listing; new lines are shown in bold.

## queryFAE()

```
adxl_measure(&s_accel);
tsl_measure(&s_lux);
adc_measure(&s_adc);
sht21_measure(&s_temprh);

// Add the values to the properties data structure.
properties.temp_adt75 = s_temp.temperature*(9.0/5.0)+32.0;
properties.humidity   = s_temprh.humidity;
properties.barometer  = s_baro.pressure * 0.000295301f;
properties.altitude   = s_baro.altitude;
properties.light       = s_lux.lux;
properties.x = s_accel.x;
properties.y = s_accel.y;
properties.z = s_accel.z;
properties.temp_sht21 = s_temprh.temperature_f;
```

## sendPropertyUpdate()

```
twApi_AddPropertyToList(proplist,"accel_z",
    twPrimitive_CreateFromNumber(properties.z), 0);
```

```

twApi_AddPropertyToList(propList, "temperature_sht21",
    twPrimitive_CreateFromNumber(properties.temp_sht21), 0);

//FAE Analog ADC Pins
twApi_AddPropertyToList(propList, "ADC1",
    twPrimitive_CreateFromNumber(properties.adc[0]), 0);

```

## ThingWorxTask()

```

twApi_RegisterProperty(TW_THING, progSets.tw_name,
    "accel_z", TW_NUMBER, NULL,
    "ALWAYS", 0, propertyHandler, NULL);
twApi_RegisterProperty(TW_THING, progSets.tw_name,
    "temperature_sht21", TW_NUMBER, NULL,
    "ALWAYS", 0, propertyHandler, NULL);

// Register our properties: FAE Analog ADC Pins
twApi_RegisterProperty(TW_THING, progSets.tw_name,
    "ADC1", TW_NUMBER, NULL,
    "ALWAYS", 0, propertyHandler, NULL);

```

# Configuring and Connecting the Device

In the workshop, your Raspberry Pi was preconfigured. If you're reading this now, you're probably trying this on your own. Here's how to get a Raspberry Pi set up (without a keyboard, monitor, or mouse). If you're using a keyboard, monitor, and mouse, you can use the standard [Raspberry Pi configuration instructions](#) and then skip ahead to “[Power On and Configure the Pi](#)” on page 37.

Make sure the Raspberry Pi is powered off. Insert the included micro SD into the Raspberry Pi, and connect the FTDI cable as shown in [Figure B-1](#) (do not plug the USB cable into a computer or power source yet). [Table B-1](#) summarizes the connections.

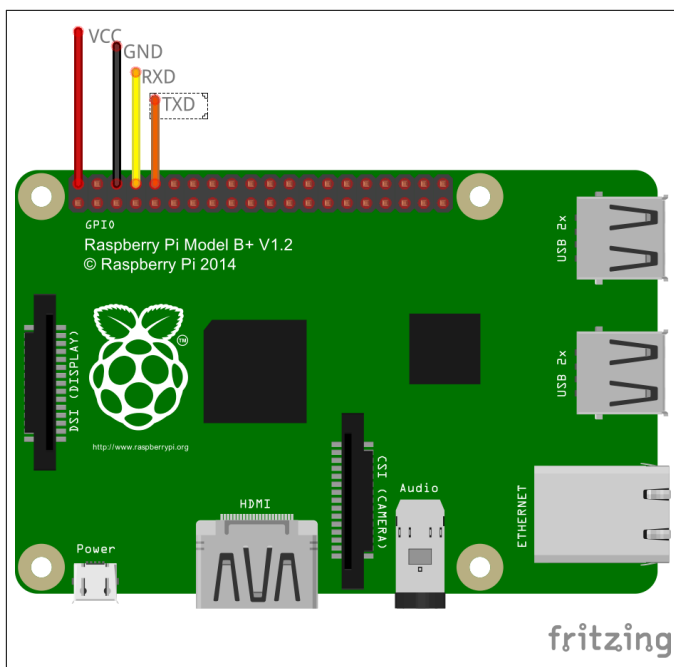


Figure B-1. Connecting the FTDI cable to the Raspberry Pi

Table B-1. FTDI cable and Raspberry Pi connections

FTDI Cable	Raspberry Pi
VCC	5V
GND	GND
RXD	TXD
TXD	RXD



If the Analog Devices FAE board is attached to your Pi, you'll need to remove it before attaching the cable.

## Install Software (Windows Only)

Because Windows doesn't include a suitable serial terminal program, and because it doesn't come with the drivers needed to connect to the cable, you need to install two software packages:

You'll need a way to connect to the Pi over the serial connection. Download **PuTTY** and install it.

You'll also need to download the **latest virtual COM port drivers from FTDI**. Install those and you're ready for the next step.

## Power On and Configure the Pi

Now it's time to plug the FTDI cable into your computer's USB port. The lights on the Pi will start flashing. Fire up your terminal program and connect:



Don't plug in the micro USB cable yet; you'll be powering the Raspberry Pi from the FTDI cable for now.

### *Windows*

Before you try to connect, open up Device Manager (Windows Key→R, type `devmgmt.msc`, and press Enter/Return). Expand the section labeled Ports (COM&LPT) and look for the USB Serial Port and make a note of its COM port number (such as COM3).

Run PuTTY from the Start Menu.

Under Connection Type, select Serial. Type the COM port (for example COM3) into the Serial Line field, and change Speed from 9600 to 115200. If you want to make it easy to connect again, type a name under Saved Sessions (such as Pi Over USB) and click Save.

Click the Open button (at the bottom of the screen).

## Mac OS X

Open a Terminal window, then type the following (but don't press Return or Enter):

```
screen /dev/tty.usb
```

Press the Tab button. You should see the command line expanded to something like:

```
screen /dev/tty.usbserial-AL02A1K2
```

Add the connection speed (115200) to the end of the line. For example (note that your actual USB port name will differ):

```
screen /dev/tty.usbserial-AL02A1K2 115200
```

Press Enter or Return.



When you are done using screen, press Control-a followed immediately by k to close the session.

## Linux

First of all, make sure the screen utility is installed. (On Ubuntu or Debian-based Linux distributions, you can install it with `apt-get install screen`). Next, open a Terminal window, then type the following and press Return or Enter:

```
ls /dev/ttyUSB*
```

You should see something like the following:

```
/dev/ttyUSB0
```

Connect to the Raspberry Pi like this, but replace `/dev/ttyUSB0` with the USB port that corresponds to your Pi:

```
screen /dev/ttyUSB0 115200
```

Press Enter or Return.

You may be looking at a blank screen. If so, press Enter or Return. Now the Raspberry Pi is displaying a login prompt:

```
Raspbian GNU/Linux 8 raspberrypi ttyAMA0
```

```
raspberrypi login:
```

## Customizing the Raspbian Environment

Log in as username `pi` with the password `raspberry`. Now you're ready to configure the pi:

### *Configure the Time Zone*

Type the command `sudo raspi-config` to bring up the Raspberry Pi configuration menu.

Select Internationalisation Options and go into Change Timezone. Next, choose US (press enter), then Eastern (if you are at LiveWorx 2016, that is—otherwise, choose your time zone), and press enter again.

### *Configure the Host Name*

Using `raspi-config`, choose Advanced Options, Host-name, and give your Pi a unique hostname that no one else in the workshop is using. Press enter, then press Tab until the Finish button is highlighted. Press enter and reboot when asked to do so.

### *NOTE*

Remember the host name; you'll need it later!

### *Enable SPI and I2C*

Using `raspi-config`, choose Advanced Options, and enable both the SPI and I2C options. You'll need to go into Advanced Options once for each option because `raspi-config` returns you to the main menu each time.



**SPI** and **I2C** are both serial computer buses that the Raspberry Pi uses to communicate with peripherals such as the FAE board.

### *Change the Password*

There will be a lot of Pis on the same network, and to avoid any potential mishaps, you should use this option to change your password so no one else can log into your Pi without knowing your password. You can use the Linux `passwd` command to change it from the default raspberry password, or use `raspi-config`.

### *Configure Wi-Fi*

This won't be necessary if you've configured Wi-Fi using the graphical user interface (requires keyboard and monitor).

You'll need to edit a couple of files. First, type:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Now, add these lines at the bottom to specify the SSID and shared key for the network you want to connect to (replacing NETWORK and PASSWORD with the name/password for the network):

```
network={ ssid="NETWORK" psk="PASSWORD" }
```

Press Control-X, and choose Yes when prompted, then press enter to save the file and exit.

Type `sudo reboot` and then press enter to restart the pi.

When your Pi is done rebooting, it should connect to the network. Log in again using PuTTY (Windows) or screen (Mac or Linux) as directed in the preceding section.

Log in, and use the `ifconfig` command to confirm whether your Wi-Fi network interface (`wlan0`) has an IP address:

```
ifconfig wlan0
```

You should see output similar to the following:

```
wlan0 Link encap:Ethernet HWaddr 00:9e:95:9b:45:43 inet addr: 10.0.0.36 Bcast:10.0.0.255 Mask:255.255.255.0 inet6 addr: fe80::ad9c:88f6:82b5:874f/64 Scope:Link
```



If you have an IP address (`inet addr: 10.0.0.36` in the preceding example), let's see if you can connect to the outside world. Type the following command:

```
curl -I http://www.oreilly.com
```

This command visits `oreilly.com` and prints the response code (not the whole response, which would be a lot of HTML and JavaScript), and should begin with `HTTP/1.1 200 OK`. If that test worked, you're on the network. The Raspberry Pi community has a [troubleshooting guide](#) that can help you diagnose problems you are having with Wi-Fi.

With Wi-Fi configured, you can SSH into your Raspberry Pi, as described in [“Connect to Your Raspberry Pi with SSH” on page 9](#).



If you are using a Mac or Linux machine, you should now be able to use `ssh` to connect to your Raspberry Pi using its hostname (or its hostname followed by `.local`). Some Windows machines may be configured to do so as well, but if you can't connect to it using its hostname, install Samba on the pi by typing the command `sudo apt-get install samba winbind` and follow the prompts. This not only installs Windows file sharing, but makes your Raspberry Pi discoverable from Windows machines.

Now that you can connect to the Pi over the network, you don't need the FTDI USB cable. Type the command `sudo halt` to shut down the Pi, and unplug the FTDI USB cable.

From here on out, you're going to need more power, so use the external USB power supply to provide power to the Pi rather than the FTDI cable.