Internet of Things (IoT) Workshop



Internet of Things (IoT) Workshop

# Prerequisite: Set up the Network Connection

(This is only required if you do not have a dedicated LAN access)

Once your laptop is connected to the Wi-Fi network, we must ensure that all the components of the kit are connected to the network. Sharing your wireless connection enables this. Complete the following steps to set up the network connection:

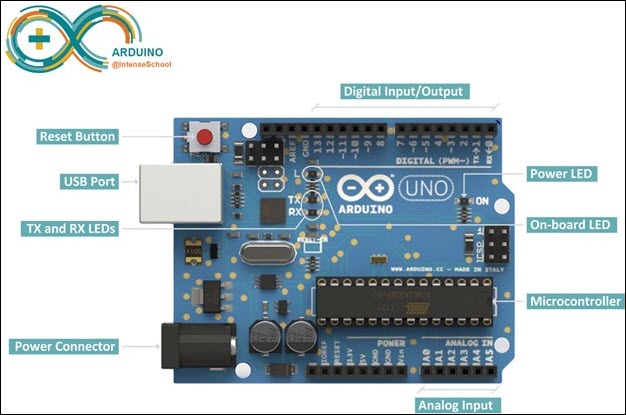
1. Click **Start > View Network Connections**.
2. Select the wireless network that your laptop is connected to.
3. Right click **Wireless** **Network** **Connection**, and select **Properties**.
4. In the **Wireless Network Connection Properties** dialog box, select the **Sharing** tab, and perform the following steps:
   1. Check the **Allow other network users to connect through this computer's internet connection** check box.
   2. In the **Home networking connection** list, select **LAN**.
   3. Click **OK**.

To verify that the wireless network is shared, ensure that you see **Shared** next to your wireless network name.

# **Part A:** Understand and Configure Arduino and the Arduino IDE

## Step 1: Know your Arduino Uno

Before we set up Arduino Uno, let us first try to understand the Arduino board:

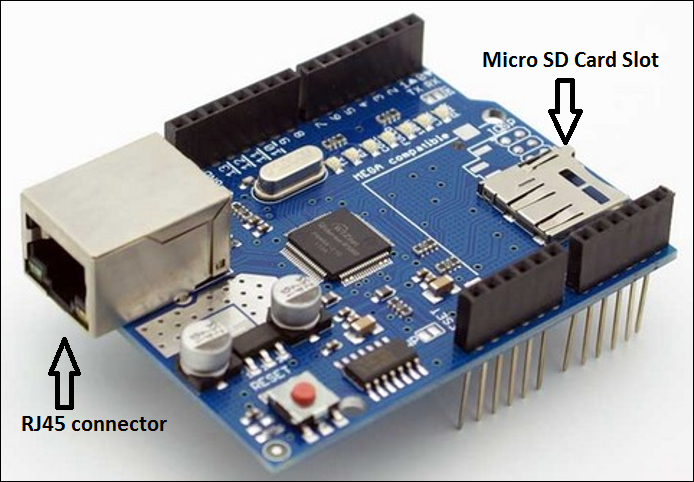


The Arduino Uno is a microcontroller board based on ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or a battery to get started.

Take some time to familiarize yourself with Arduino Uno, and try to identify the different ports!

## Step 2: Know your Ethernet Shield

The Arduino Uno does not have an internet module. The Arduino Ethernet Shield connects your Arduino Uno board to the internet.



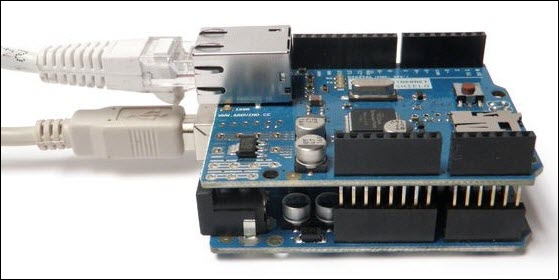
## Step 3: Connect the Arduino Uno and the Ethernet Shield

Complete the following steps to connect the Arduino Uno board, the Ethernet Shield, and your laptop:

1. First, mount the Ethernet shield on top of your Arduino Uno board.

Tip: Take care that you do not break the pins of the Ethernet Shield.

1. Connect the Arduino Uno to your laptop with a USB cable.
2. Connect the Ethernet Shield to your laptop with an RJ45 cable.



## Step 4: Download, Install, and Set up Arduino IDE

**Arduino provides an open-source and easy-to-use programming tool, the Arduino IDE (Integrated Development Environment), to** write code and upload it to your board.

Complete the following steps to download and install the Arduino IDE:

1. Download the Arduino IDE from <https://www.arduino.cc/en/main/software>.

Tip: Use the correct installer according to your operating system.

1. Double**-**click the installer to start the installation.
2. Click **I Agree** to accept the Arduino License Agreement.
3. Retain the default settings in the **Arduino Setup: Installation Options** dialog box.
4. During installation, you might be prompted to install drivers like:
   * Adafruit Industries LLC Ports (COM & LPT)
   * Arduino USB driver
   * Genuino USB driver
   * Linino Ports (COM & LTP)

Click **Install** to install all the drivers.

1. In the **Arduino Setup: Completed** dialog box, click **Close** to complete the installation.
2. Double-click the Arduino Desktop icon to launch the Arduino IDE.
3. Go to **Sketch > Include Library > Manage Libraries**.

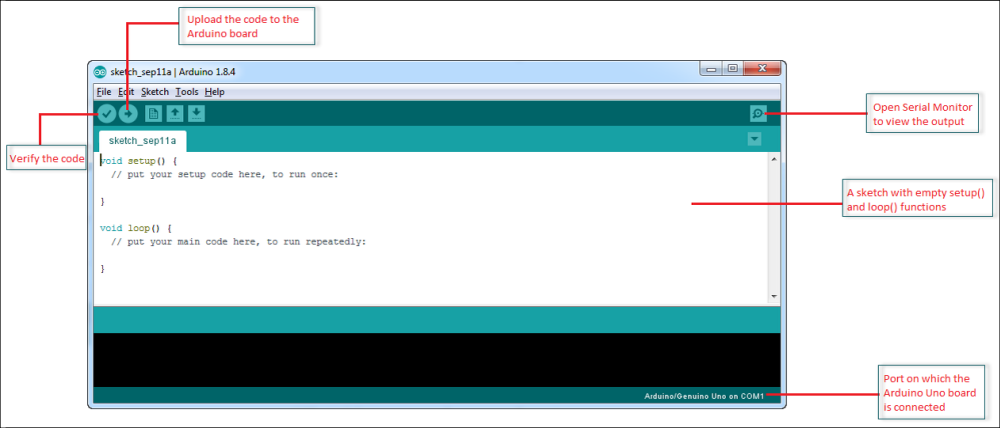
The **Library Manager** dialog box is displayed.

1. Ensure that the following libraries are installed:
   * Ethernet Built-In – Version 1.1.2
   * DHT sensor library – Version 1.3.0
   * SimpleDHT – Version 1.0.6
   * Adafruit Unified Sensor – Version 1.0.2

If they are not installed, in the **Library** **Manager** dialog box, search for the library, select it, and click **Install**.

Take some time to familiarize yourself with the Arduino IDE. Explore the **File**, **Edit**, **Sketch**, and **Tools** menus.

Note that all options will be available in the Arduino IDE only if you have connected the Arduino to your laptop.



## Step 5: Understand a Sketch

The Arduino IDE is used to write a sketch. A sketch is a C program with setup and loop functions that are compiled in the IDE and loaded on to the Arduino board.

A sketch has the following two methods:

* Setup() – This function is called once when the program is run. It is used to define the initial properties such as pin mode (INPUT or OUTPUT), starting the serial port, etc., before the main program, loop() starts executing.

The variables that are declared within setup() are not accessible within loop().

* Loop() – This function executes code until the program is stopped. The loop() function is used in conjunction with setup(). The delay() and the delayMicroseconds() control the number of times that a loop() function executes every second.

## Step 6: Verify the Connection

Before we continue, let us verify that the connection between the Ethernet Shield, Arduino Uno, and the laptop is done. To do this, complete the following steps:

1. In Arduino IDE, go to **File > Examples > 01.Basics > Blink**.

The Blink sketch is displayed in a new window.

1. Click the  icon to compile the sketch.
2. Once the sketch is successfully compiled, click  to upload the sketch to the Arduino Uno board.

After the sketch compiles, the LED must start blinking.

# **Part B:** The Problem – Smart Parking

A shopping center wants a smart car parking system which would automatically identify all those parking slots that are empty. No one is required to monitor the parking.

The smart parking system is capable of sending back data of a parking space being empty or occupied. The person at the entrance just needs to direct the car owner to the parking slot that is available.

The smart parking system should also be able to identify any anomalies present in the parking area.

## Step 1: The Solution

Place an ultrasonic sensor at the ceiling of the parking slot, and calculate the distance between the sensor and the floor. This distance is constant.

If a car is parked in this slot, the calculated distance is less than this constant distance.

**Calculation**:

Speed of sound = 0.0343cm/microsecond @ 20°

Divide duration by 2 as the total duration is the time required for the ultrasonic signal to go and come back.

## Step 2: Understand the Electronic Components

Apart from understanding the Arduino and the Ethernet Shield, it is a good idea to understand the ultrasonic sensor and the bread board.

**Ultrasonic sensor**

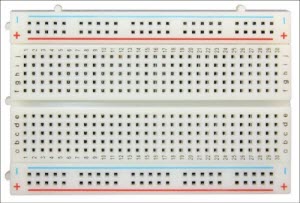


The ultrasonic sensor has 4 pins:

|  |  |
| --- | --- |
| **Pin** | **Function** |
| VCC | Power supply |
| Trig | Output |
| Echo | Input |
| GND | Ground |

**Bread** **board**

A bread board is a device that is used to make a temporary electronic circuit.



Here is a cheat sheet for the breadboard connections:

* The red line is the +5 V line
* The blue line is the 0 V line.
* Internally, holes a1, b1, c1, d1, and, e1 are connected. Let us call this Group 1.
* Internally, holes f1, g1, h1, i1, and, j1 are connected. Let us call this Group 2.

However, Group 1 and Group 2 are not connected to each other.

## Step 3: Connect the Electronic Circuit

In the previous sections, we have already mounted the Ethernet Shield on the Arduino board, and connected them to the laptop.

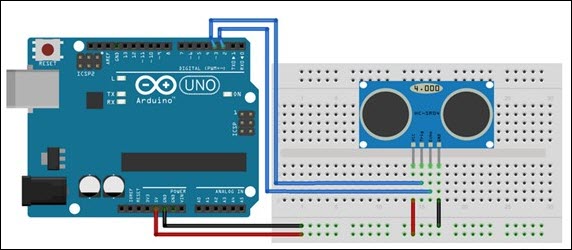
Now, let us connect the following components:

* Arduino board and Ethernet Shield
* Ultrasonic Sensor
* Bread Board
* Jumper wires (Male to Male)

Connect the ultrasonic sensor to the bread board directly.

Tip: Ensure that all the wire connections are behind the ultrasonic sensor. Thus, there will be no interference in the distance calculations.

Connect the circuit according to the circuit diagram below:



## Step 4: Test the Arduino Sketch to Calculate the Distance

Before we move on to the next section, let us verify that the electronic circuit has been connected properly:

1. Go to **Tools > Board**, and verify that **Arduino/ Genuino Uno** is selected.
2. Go to **Tools** **>** **Port**, and verify that the serial port on which the Arduino Uno board is connected must be selected.
3. Copy and paste the following piece of code:

const int trigPin = 3;

const int echoPin = 2;

int duration;

float distance;

void setup() {

Serial.begin(9600);

Serial.print("initializing arduino pins");

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration\*.0343)/2;

Serial.print("Distance: ");

Serial.println(distance);

delay(500); // waiting for .5 sec

}

1. Save the sketch to a folder on your laptop.
2. Click the  icon to compile the sketch.
3. Once the sketch is successfully compiled, click  to upload the sketch to the Arduino Uno board.
4. Once the sketch is successfully compiled, go to **Tools > Serial Monitor**, to view the result.

# **Part C:** Set up ThingWorx

Now, let us send the distance data to the ThingWorx server. Before, we do that we need to set up and configure different entities in ThingWorx.

## Step 1: Launch ThingWorx Composer

Complete the following steps to launch ThingWorx Composer:

1. In your browser (preferably, Google Chrome), enter the ThingWorx URL and log in with the credentials that have been provided to you.
2. On the License Agreement page, click **I Accept**.

The ThingWorx Composer is displayed.

## Step 2: Create a Project

During the course of this exercise, we will create multiple entities. Creating a project will allow us to group all these entities together. Later, you can easily import or export the project. This ensures that all the entities that are associated with the project are imported or exported together.

In this section, we will create the **MindSpark17\_*<user-name>***project.

Complete the following steps to create a project:

1. In ThingWorx Composer, click the **Explorer** icon ().
2. In the left navigation panel, under **MODELING**, hover your cursor over **Projects**, and click the  icon.
3. In the **Name** field, enter **MindSpark17\_*<user-name>***.

Ensure that you use the same *<user-name>* that you used to log in to ThingWorx.

1. Click **Save**.

## Step 3: Create a Value Stream

ThingWorx value streams store property data of a thing. You need to associate the value stream with a thing so that the value stream can store data of a property on the thing. We will do that in the next step.

In this section, we will create the **DistanceSensorValueStream\_*<user-name>*** valuestream**.**

Complete the following steps to create a value stream:

1. In ThingWorx Composer, click the **Explorer** icon ().
2. In the left navigation panel, under **DATA STORAGE**, hover your cursor over **Value Streams**, and click the  icon.
3. In the **Choose Template** dialog box, select **Value** **Stream**, and click **Choose**.
4. In the **Name** field, enter **DistanceSensorValueStream\_*<user-name>***.

Ensure that you use the same *<user-name>* that you used to log in to ThingWorx.

1. In the **Project** field, search for and select the project, **MindSpark17\_*<user-name>*** that you created.
2. Click **Save**.

## Step 4: Create a Thing

Things are ThingWorx entities that represent actual physical devices. Things are based on thing templates. A Thing can have its own properties, services, events, and subscriptions and can inherit other properties, services, events, and subscriptions from its thing template.

In this section, we will create the **DistanceSensor\_*<user-name>*** thing that represents the ultrasonic distance sensor.

Complete the following steps to create a thing:

1. In ThingWorx Composer, click the **Explorer** icon ().
2. In the left navigation panel, under **MODELING**, hover your cursor over **Things**, and click the  icon.
3. In the **Name** field, enter **DistanceSensor*\_<user-name>***.

Ensure that you use the same *<user-name>* that you used to log in to ThingWorx.

1. In the **Project** field, search for and select the project, **MindSpark17\_<user-name>**that you created.
2. In the **Base Thing Template** field, search for and select **GenericThing.**
3. In the **Value** **Stream** field, search for and select the value stream, **DistanceSensorValueStream\_*<user-name>*** that you created.
4. Click **Save**.

## Step 5: Add a Property to the Thing

Properties are used to provide values of attributes of a thing. They let us know the current conditions of a thing. Properties can either be static (constant) or dynamic (changing).

Making the property as persistent ensures that the property values are persisted or kept in the ThingWorx database even after a system restart.

Logging the property ensures that every value change of the property is logged to the value stream that we have already associated with the thing.

In this section, we will create the **distance** property on the **DistanceSensor\_*<user-name>*** thing that we created in the previous step.

Complete the following steps to add a property to the thing:

1. In ThingWorx Composer, click the **Explorer** icon ().
2. In the left navigation panel, under **MODELING**, select **Things**.
3. Click the **View** icon  next to the **DistanceSensor\_*<user-name>***thing that you just created.
4. Click **Edit**.
5. Under **ENTITY INFORMATION**, select **Properties**.
6. Click **Add My Property**.
7. In the **Name** field, enter **distance**.
8. In the **Base Type** list, select **NUMBER**.
9. In the **Units** field, enter **cm**.

In our case, the unit of distance is centimeter.

1. Select the **Persistent** check box.
2. Select the **Logged** check box.
3. Click **Done**.
4. Click **Save**.

Tip: In a text editor like Notepad, copy and paste the name of the property. Going forward, you will need it.

## Step 6: Create a Service on the Thing

Services help us define functions that a thing can perform. When you create a new service, you can define input properties and an output.

In this section, we will create the **setDistance** service on the **distance** property that we created in the previous step.

Complete the following steps to create a service:

1. Under **ENTITY INFORMATION**, select **Services**.
2. Click **Add My Service**.
3. Under the **Service Info** tab, in the **Name** field, enter **setDistance**.
4. Click **Inputs/Outputs** tab.
5. Under the **Inputs** section, click **Add**.

The **Input** **Parameter** dialog box is displayed.

1. In the **Name** field, enter **distance.**
2. In the **Base Type** list, select **NUMBER**.
3. In the **Units** field, enter **cm**.

In our case, the unit of distance is centimeter.

1. Click **Done**.
2. Under the **Script** section towards the right, enter the following line of code:

me.distance=distance;

1. Click **Done**, and click **Save**.

Tip: In a text editor like Notepad, copy and paste the name of the service. Going forward, you might need it.

## Step 7: Execute the Service

Executing the **setDistance** service updates the value of the **distance** property. Let’s verify that the service works as expected.

To do this, complete the following steps:

1. On the **Services** page, click Test next to the **setDistance** service.

The **setDistance – Test Service** dialog box is displayed.

1. In the **distance** field, enter a value, say 123.45, and click **Execute Service**.

The **Results** section does not display any result. This is the expected behavior.

1. Click **Close.**
2. Under **ENTITY INFORMATION**, select **Properties**.
3. Click the Refresh icon  and check the value of the **distance** property. It should reflect the value that you entered in Step 2.

## Step 8: Create an Application Key

In order to successfully connect Arduino to ThingWorx, it important that the Arduino authenticates with the ThingWorx server. An application key is a security token that is used to log in to ThingWorx directly.

Complete the following steps to create an application key:

1. In ThingWorx Composer, click the **Explorer** icon ().
2. In the left navigation panel, under **SECURITY**, hover your cursor over **Application Keys**, and click the  icon.
3. In the **Name** field, enter ***<user-name>*\_App\_Key**.

Ensure that you use the same *<user-name>* that you used to log in to ThingWorx.

1. In the **Project** field, search for and select the project, **MindSpark17*\_<user-name>*** that you created.
2. In the **User Name Reference** field, enter the ***<user-name>*** with which you have logged in to ThingWorx.
3. Click **Save**.

You can view the application key in the **keyId** field.

Tip: In a text editor like Notepad, copy and paste the application key ID. Going forward, you will need it.

# **Part D:** Update the Sketch

You need to update the sketch so that it can send its data to the ThingWorx server. Ensure that you have the following values with you:

* server – Specifies the ThingWorx server host name that has been provided to you.
* port – Specifies the ThingWorx server port number.
* appKey – Specifies the ThingWorx application key that you created in Part D: Step 9.
* thingName – Specifies the thing name that you created in Part D: Step 5.

Complete the following steps to update the sketch in Arduino IDE:

1. Copy this code snippet to Arduino IDE.

#include <SPI.h>

#include <Ethernet.h>

const int trigPin = 3;

const int echoPin = 2;

int duration;

float distance;

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; //Make sure your mac address is unique in your network

EthernetClient client;

char server[] = "server"; //ThingWorx server hostname. Change this to your server

int port=8080; //ThingWorx server port. Change as per your server port

char appKey[] = "<your\_app-key>"; //App key you created

char thingName[] = "<DistanceSensor\_<username>>"; //Thing Name you created for getting sensor value

char serviceName[] = "setDistance"; //service name

char propertyName[]="distance";

int timeBetweenRefresh = 1000; //refresh interval in milliseconds

unsigned long lastConnectionTime = 0;

void setup() {

Serial.begin(9600);

Serial.print("initializing ethernet");

Serial.println("Trying to get an IP address using DHCP");

Ethernet.begin(mac); //Initialize the ethernet with the provided MAC. Ethernet sheild will also get an IP thought DHCP

Serial.print("My IP address: ");

Serial.print(Ethernet.localIP());

Serial.println();

Serial.print("initializing arduino pins");

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop() {

if (millis() - lastConnectionTime > timeBetweenRefresh){

updateValues(); //uploading data to ThingWorx

}

else{

updateSensorsValue(); //updating senror data locally

}

}

void updateSensorsValue(){

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration\*.0343)/2;

Serial.print("Distance: ");

Serial.println(distance);

delay(500); // waiting for .5 sec

}

void updateValues()

{

if (client.connect(server, port)) {

if(client.connected()){

// Sending a header of a network packet

Serial.println("Sending data to ThingWorx Server...\n");

Serial.print("POST /Thingworx/Things/"); client.print("POST /Thingworx/Things/");

Serial.print(thingName); client.print(thingName);

Serial.print("/Services/"); client.print("/Services/");

Serial.print(serviceName); client.print(serviceName);

Serial.print("?appKey="); client.print("?appKey=");

Serial.print(appKey); client.print(appKey);

Serial.print("&method=post&x-thingworx-session=true"); client.print("&method=post&x-thingworx-session=true");

Serial.print("&"); client.print("&");

Serial.print(propertyName); client.print(propertyName);

Serial.print("="); client.print("=");

Serial.print(distance); client.print(distance);

Serial.println(" HTTP/1.1"); client.println(" HTTP/1.1");

Serial.println("Accept: application/json"); client.println("Accept: application/json");

Serial.print("Host: "); client.print("Host: ");

Serial.println(server); client.println(server);

Serial.println("Content-Type: application/json"); client.println("Content-Type: application/json");

Serial.println(); client.println();

client.stop();

lastConnectionTime = millis();

}

}

else{

Serial.println("Connection could not be established");

client.stop();

}

}

1. Modify the values of the following parameters:
   * server – Replace this with the provided ThingWorx server host name/ IP.
   * appKey – Replace *<your\_app-key>* **ONLY** with the application key that you created.
   * thingName – Replace *<DistanceSensor\_<user-name>>* **ONLY** with the name of the thing that you created.
2. Save the sketch.

# **Part E:** Send Data to Thingworx

Now, let us send data from Arduino to the ThingWorx server. We have already set up the Arduino sketch and also configured the ThingWorx server as per our requirement.

Complete the following steps to send data to ThingWorx server:

1. In Arduino IDE, click the  icon to compile the sketch.
2. Once the sketch is successfully compiled, click  to upload the sketch to the Arduino Uno board.
3. In ThingWorx Composer, view the thing, **DistanceSensor\_*<user-name>*** thing that you created.
4. Under **ENTITY INFORMATION**, select **Properties**.
5. Click the  **Refresh** icon and check the value of the **distance** property. It should reflect the value that the sensor is sending.

# **Part F:** Create the ThingWorx Mashup

## Step 1: Create the Mashup

Mashups are web pages. You can configure mashups as per your requirement.

In this section, we will create the **DistanceMashup\_*<user-name>*** mashup.

Complete the following steps to create a mashup:

1. In ThingWorx Composer, click the **Explorer** icon ().
2. In the left navigation pane, under **VISUALIZATION**, hover your cursor over **Mashups**, and click the  icon.
3. In the **New Mashup** dialog box, select the following options:
   * For **Mashup Type**, select **Page**.
   * For **Layout** **Options**, select **Responsive**.
4. Click **Done**.

The **New** **Mashup** page is displayed.

1. Click the **Info** tab.
2. In the **Name** field, enter **DistanceMashup\_*<user-name>***.

Ensure that you use the same *<user-name>* that you used to log in to ThingWorx.

1. In the **Project** field, search for and select the project, **MindSpark17\_*<user-name>*** that you created.
2. Click **Save.**
3. Click **Edit** to open the **Mashup Builder**.

## Step 2: Add a Service to the Mashup

In order to obtain data in a mashup, you can add services to the mashup.

In this section, we will add the **QueryNumberPropertyHistory** service to query the value of the **distance** property on the **DistanceSensor\_*<user-name>*** thing.

Complete the following steps to obtain data in a mashup:

1. On the right-side of the Mashup Builder window, click **Data**.
2. Click the  **Add** **Entity** icon.
3. In the **Add** **Data** dialog box, do the following:
   1. In the **Select** **Entity** list, select **Things**.
   2. In the Search Things field, enter the thing, **DistanceSensor\_*<user-name>*** that you created.
   3. Under the **Select Services** section, in the **Filter** field, enter **QueryNumberPropertyHistory**, and click the  icon.
   4. Under the **Selected Services** section, select the **Mashup Loaded** check box.
   5. Click **Done**.

On the **Data** tab, under **Things\_DistanceSensor\_*<user-name>***, the **QueryNumberPropertyHistory** service is displayed.

* 1. Under the **QueryNumberPropertyHistory** service, click **Parameters**.
  2. In the **propertyName** field, enter **distance**.

## Step 3: Add Widgets to the Mashup

You can visualize data in the mashup by adding graphs or charts.

In this section, let us add the **Grid** widget and the **Time** **Series** **Chart** widget to the mashup.

Complete the following steps to visualize data in a mashup:

1. On the left-side of the Mashup Builder, on the **Widgets** tab, search for the **Layout** widget.
2. Drag and drop the **Layout** widget to the mashup builder area.

The **Configure Widget** dialog box is displayed.

1. Retain the default settings, and click **Done**.
2. Again, drag and drop the **Layout** widget to the mashup builder area.

The **Configure Widget** dialog box is displayed.

1. Select the **Vertical** layout, and click **Done**.
2. Select the upper section of the left layout.
3. On the **Widgets** tab, search for the **Grid** widget.
4. Drag and drop the **Grid** widget on to the upper left layout section.
5. Select the lower section of the left layout.
6. On the **Widgets** tab, search for the **Time Series Chart** widget.
7. Drag and drop the **Time Series Chart** widget on to the lower left layout section.
8. On the right-side of the Mashup Builder window, click the **Data** tab.
9. Under the **QueryNumberPropertyHistory** service, click the  next to **Returned Data**.
10. Under **Returned** **Data**, drag and drop **All Data** to the **Grid** section of the layout.
11. Under **Select Binding Data**, select **Data**.
12. Now, drag and drop **All Data** to the **Time Series Chart** section of the layout.
13. Under **Select Binding Data**, select **Data**.
14. Select the **Time Series Chart**, and in the lower left section, under the **Properties** section, set the following values:
    1. In the **XAxisField** list, select **timestamp**.
    2. Inthe **DataField1** list**,** select **value**.
15. Select the right layout section.
16. On the **Widgets** tab, search for the **Auto Refresh** widget.
17. Drag and drop the **Auto Refresh** widget to the right layout section.
18. In the **Add a Panel** dialog box, click **Yes**.

The Auto Refresh widget is placed on the right layout section.

1. Click the **Auto** **Refresh** widget.
2. Hover over the  icon of the widget, and drag **Refresh** to **QueryNumberPropertyHistory**.

Ensure that the widget binds to the data.

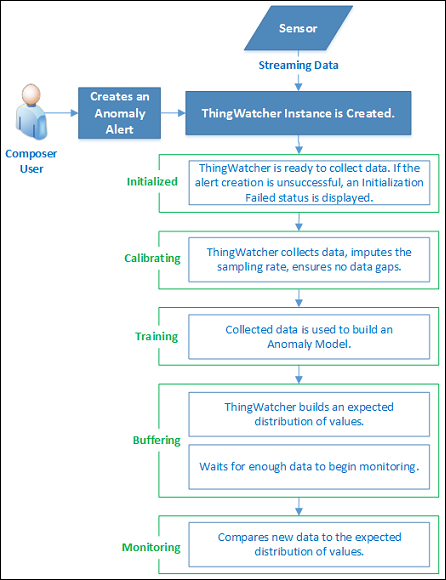
1. In the **Properties** section on the lower left part, set the value of **RefreshInterval** property of the **Auto Refresh** widget to 1.

Thus, the data will refresh every 1 second.

1. Click **Save**.
2. Click **View** **Mashup** to view the mashup in a new tab.

# **Part G:** Set up Anomaly Detection

Anomaly Detection in ThingWorx is implemented via the in-built ThingWatcher functionality. ThingWatcher detects anomalies by monitoring a data stream from a device, calculating an expected distribution of data, and validating that the current data point is a member of the expected distribution. Without any coding, you can attach an Anomaly alert to any numeric property and train the alert during normal operation with regularly timed property value updates.



## Step 1: Enable ThingWorx New Composer

Anomaly detection can be enabled in the ThingWorx New Composer only.

Complete the following steps to enable the ThingWorx New Composer:

1. On the upper-right corner of the ThingWorx Composer page, click the *<user-name>* with which you logged in, and click **Preferences**.
2. In the Preferences window, select the **Turn on New Composer Features** check box.
3. Click **Done**.

The **New Composer** link appears next to your *<user-name>*.

1. Click **New Composer** to launch the ThingWorx composer in a new tab.

## Step 2: Create an Anomaly Alert

Complete the following steps to start detecting anomalies:

1. In ThingWorx New Composer, open the **DistanceSensor\_*<user-name>*** that you created.
2. In the Entity Header list, select **Properties and Alerts**.
3. Click **Alerts** to switch to the Alerts view.
4. Click **Add** to add an alert on the **distance** property.

The **New Alert** panel is displayed on the right.

1. In the **Name** field, enter **Anomaly\_Alert\_*<user-name>****.*

Ensure that you use the same ***<user-name>*** that you used to log in to ThingWorx.

1. In the **Property** list, select **distance**.
2. In the **Alert Type** list, select **Anomaly**.
3. Keep the default values for the remaining fields.
4. Click the  icon to save the changes.

Internally, a ThingWatcher is created for the property alert. You can check that you have configured the alert correctly by verifying that the status of the anomaly alert that you created is **Initialized**.

Once the value of the distance property starts updating, the status of the anomaly alert changes from **Initialized** to C**alibrating.** It remains in the **Calibrating** state until approximately 500 property value updates have been received.

Then, the status changes to the **Training** state to indicate that an anomaly model has been created internally and incoming data is synchronized to this model.

When this model receives enough property value updates, it builds an expected distribution of values. Then the status of the anomaly alert changes to **Monitoring** state. Now, the anomaly model compares new data with the expected distribution of values.

## Step 3: Launch the Old ThingWorx Composer

You can create mashups in the old ThingWorx Composer only. Let us navigate to the old ThingWorx Composer before we proceed with the next step.

On the upper-right corner of the ThingWorx New Composer page, click **Composer** next to your *<user-name>*.

The old ThingWorx Composer opens in a new tab.

Tip: It is a good idea to keep the old ThingWorx Composer and the ThingWorx New Composer open in two consecutive tabs. This will enable us to switch between the two as and when required.

## Step 4: View the Anomaly in the Mashup

Complete the following steps to view an anomaly in the mashup:

1. In the old ThingWorx Composer, click the **Explorer** icon ().
2. In the left navigation pane, under **VISUALIZATION**, click **Mashups** icon.
3. In the mashups list, click **Anomaly\_Dashboard**.
4. Click **View Mashup** to open the mashup in a new window.

Note: Based on your browser pop-up blocker settings, you might need to **Allow pop-ups**.

1. In the **Thing** field, search for and select the thing on which you created the anomaly alert.

In our case, you can select the **DistanceSensor\_*<user-name>*** thing.

Once the Anomaly Alert has been trained and reached the **Monitoring** state, the mashup is shown as below:



Here is a quick reference of what the mashup depicts:

|  |  |
| --- | --- |
| **Section** | **Description** |
| 1 | Displays the static data that was used to train the anomaly alert. |
| 2 | Displays live data from the ultrasonic sensor.  Blue line: Normal data  Red line: Anomalous value |
| 3 | Displays the anomaly alert life cycle. |

# **Part H:** Set up Anomaly Notification with Twitter

## Step 1: Create the Twitter thing

Complete the following steps to create the Twitter thing:

1. In the old ThingWorx Composer, click the **Explorer** icon ().
2. In the left navigation panel, under **MODELING**, hover your cursor over **Things**, and click the  icon.
3. In the **Name** field, enter **TwitterThing\_*<user-name>***.

Ensure that you use the same *<user-name>* that you used to log in to ThingWorx.

1. In the **Project** field, enter the project, **MindSpark17\_<user-name>**that you created.
2. In the **Base Thing Template** field, search for and select **Twitter.**
3. Click **Save**.

## Step 2: Create a New Twitter Application

Prerequisites:

* Ensure that you have a valid Twitter account, and you remember the credentials to log in to that account.
* Ensure that you link and validate your mobile number to your Twitter account. This is required because we are going to create a new Twitter Application access key.

Complete the following steps to create a new Twitter application:

1. Go to <https://apps.twitter.com/>.
2. Click **sign in** to log in to your Twitter account.
3. Click **Create New App** to create a new application.
4. On the **Create** **an** **application** page, provide the following details:
   1. In the **Name** field, enter **ThingWorx Demo\_*<user-name>***.

Ensure that you use the same *<user-name>* that you used to log in to ThingWorx.

* 1. In the **Description** field, enter **ThingWorx Demo\_*<user-name>***.

Ensure that you use the same *<user-name>* that you used to log in to ThingWorx.

* 1. In the **Website** field, enter a valid URL. For example, enter <https://www.google.com>.

1. Select the **Developer Agreement** check box.
2. Click **Create your Twitter application**.

Your application is successfully created.

1. Click the **Keys and Access Tokens** tab.
2. Under the **Your Access Token** section, click **Create my access token**.

## Step 3: Connect your New Twitter Application to the Twitter Thing in ThingWorx

Complete the following steps to connect your new Twitter application to the Twitter thing in ThingWorx:

1. In this step, ensure that you have the **Keys and Access Tokens** tab of your new Twitter application and the **TwitterThing\_*<user-name>*** thing open simultaneously in 2 consecutive tabs. This will make it easier for us to copy different values from the Twitter application to the Twitter thing in ThingWorx. Refer to the following table to understand what you need to copy:

|  |  |
| --- | --- |
| **Twitter Credentials** | **Twitter Configuration in ThingWorx** |
| Twitter application - **Keys and Access Tokens** tab | **TwitterThing\_*<user-name>*** - Under **ENTITY** **INFORMATION**, select **Configuration**. |
| 1. Under the **Application Settings** section, copy the value of **Consumer Secret (API Secret).** | In the **consumerSecret** field, click **Change Password**. In the **Enter New** Password field, paste this value.  In the **Retype Password** field, paste the same value. |
| 2. Under the **Application Settings** section, copy the value of **Consumer Key (API Key).** | In the **consumerKey** field, paste this value. |
| 3. Under the **Your Access Token** section, copy the value of **Access Token**. | In the **accessToken** field, paste this value. |
| 4. Under the **Your Access Token** section, copy the value of **Access Token Secret**. | In the **accessTokenSecret** field, click **Change Password**. In the **Enter New** Password field, paste this value.  In the **Retype Password** field, paste the same value. |

1. Click **Save.**

## Step 4: Verify that your Twitter Application is correctly connected to the Twitter Thing

### Execute the **Search** service

1. Under **ENTITY** **INFORMATION**, select **Services.**
2. Click **Test** next to the **Search** service to execute the service.
3. In the **searchString** field, enter a term that you want to search through your Twitter account. For example, you can search for #IoT, @PTC, @life@PTC, @COEPMindSpark, @LiveWorx, and so on.
4. Click **Execute Service**.

If you have followed the steps correctly, you can see search results from Twitter under the **Results** section.

### Execute the **UpdateStatus** service

1. Under **ENTITY** **INFORMATION**, select **Services**.
2. Click **Test** next to the **UpdateStatus** service to execute the service.
3. In the **status** field, enter a tweet that you want to post to Twitter.

Let us try to be innovative. Don’t forget to add these hashtags: #PTC or #COEPMindSpark

For example, you can enter “My First IoT Project #PTC #COEPMindSpark” or “ThingWorx-the best IoT application #PTC”.

1. Click **Execute Service**.

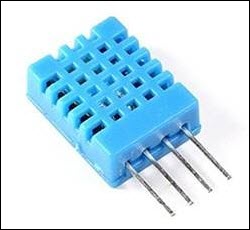
**Note**: Executing this service posts a public tweet from your Twitter account.

1. Log in to your Twitter account, and check the last tweet under the **Tweets** section.

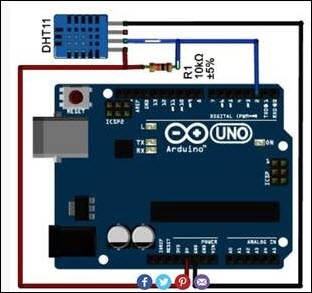
-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-

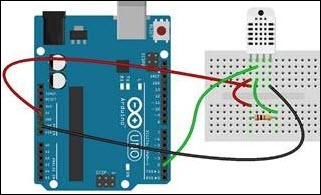
Project: Temperature and Humidity Sensor

**DHT11 temperature and humidity sensor**



**Electronic Circuit**





**Code**

#include <DHT.h>

// DHT11 Temperature and Humidity Sensors Example

#include "DHT.h" //include DHT library --> Goto Sketch--> Include Library -->

//Manager Library--> search for DHT (In Filter your search).

//Select the DHT moreinfo and install the library if it is not installed.

#define DHTPIN 7 //define as DHTPIN the Pin 7 used to connect the Sensor

#define DHTTYPE DHT11 //define the sensor used(DHT11)

DHT dht(DHTPIN, DHTTYPE);//create an instance of DHT

/\*setup\*/

int delayPeriod = 3000;

void setup() {

Serial.begin(9600); //initialize the Serial communication

delay(delayPeriod); //wait 3 seconds

Serial.println("Temperature and Humidity are");//Debug prints. See on Serial monitor. Got to Tools-->Serial Monitor in Ardiuno IDE. Select 9600

Serial.println("T(C) \tH(%)"); //

dht.begin(); //initialize the Serial communication

}

/\*loop\*/

void loop() {

float h = dht.readHumidity(); // reading Humidity

float t = dht.readTemperature(); // read Temperature as Celsius (the default)

// check if any reads failed and exit early (to try again).

if (isnan(h) || isnan(t)) {

Serial.println("Failed to read from DHT sensor!"); //too low values. is this an issue with the sensor.

return;

}

Serial.print(t, 2); //print the temperature

Serial.print("\t");

Serial.println(h, 2); //print the humidity

delay(delayPeriod); //wait 3 seconds

}

-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-

Experience AR

Download the ThingWorx View Mobile App from either Google Play Store or Apple App Store, and then scan the following Thing Mark:



-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-