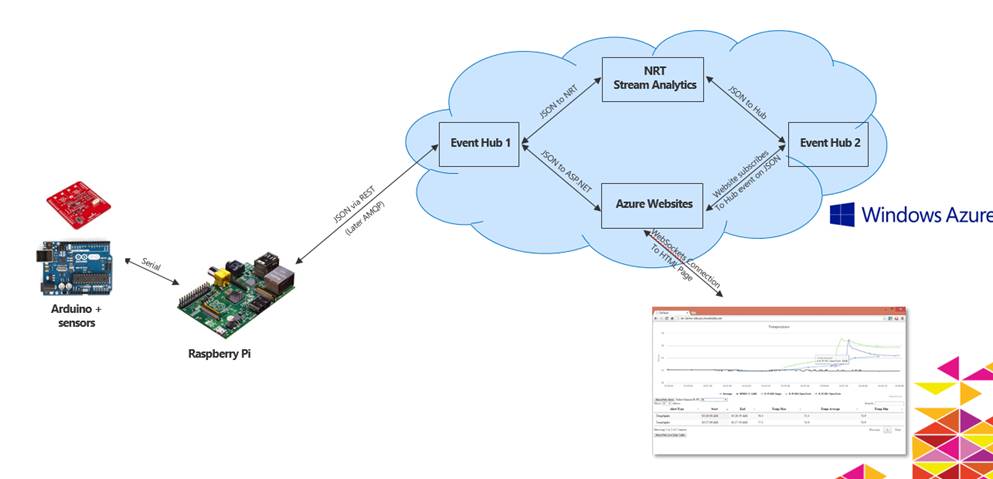
Connect the Dots – Quick Start

# Intro

Connect the Dots illustrates how to connect sensors and devices to the Azure Cloud, und use Azure to analyze and visualize the resulting data streams.

In a typical topology, a sensor (here an Arduino Uno with Weather Shield) connects to a local IoT Gateway (here a Raspberry PI), which relays the data to Azure Event Hubs. Once in the cloud, the data streams are fed into Azure Streaming Analytics (in this case to generate averages and alerts across all devices). The data is then visualized in an Azure Website, which feeds the real-time data to HTML5 browsers via a WebSocket connection.



Connect the Dots provides you with all the code and step-by-step instructions to build and deploy such an end-to-end solution.

# Prerequisites:

1. Microsoft Azure subscription ([free trial subscription](http://azure.microsoft.com/en-us/pricing/free-trial/) is sufficient)
2. Access to the [Azure Streaming Analytics Preview](https://account.windowsazure.com/PreviewFeatures)
3. Visual Studio 2013 [– Community Edition](http://go.microsoft.com/fwlink/?linkid=517106)

## Hardware:

1. [Raspberry PI B/B+](http://www.raspberrypi.org/products/model-b-plus/) with Internet access
2. [Arduino Uno R3](http://arduino.cc/en/Main/ArduinoBoardUno)
3. [SparkFun Weather Shield](https://www.sparkfun.com/products/12081) for Arduino (make sure you also have the required headers etc. as specified on the SparkFun site)

Note: Only the models above have been tested. The Weather Shield for example is known to not work reliably on Arduino Due or Arduino Uno R2.

# Step-by-step

1. Download and Build “Connect the Dots”:
   1. Download or clone the “Connect the Dots” depot from <https://github.com/MSOpenTech/connectthedots-pr>
   2. Open ConnectTheDots.sln in Visual Studio and build the solution
   3. If the AMQP.Net Lite package cannot be found, you may need to build it yourself:
      1. Download or clone from <http://amqpnetlite.codeplex.com>
      2. Open amqp.sln in Visual Studio and build the solution
         1. To avoid errors for platforms for which you don’t have tooling, comment out or remove all <file> elements except for “bin\Release\Amqp.Net\\*.\*” in Amqp.Net.nuspec
      3. Add a private NuGet location:
         1. Open Tools, Options, NuGet Package Manager, Package Sources and add “<yourpath>\amqpnetlite\Build\Packages” as a new package source.
2. Create Azure resources for Event Hub:
   1. Download publishsetting file
      1. Go to <https://manage.windowsazure.com/publishsettings/index?client=powershell>) and save to local disk <publishsettingsfile> (contains keys to manage all resources in your subscriptions, so handle with care).
      2. If you have access to multiple subscriptions, make sure the file only contains the subscription that you want to use, otherwise, remove the other XML elements for the other subscriptions.
   2. Run ConnectTheDotsCloudDeploy from a command prompt, passing a name to be used for all cloud resources, and the publishsettings file
      1. Chose a <name> that has only letters and number
      2. ConnectTheDotsCloudDeploy\bin\debug\ ConnectTheDotsCloudDeploy.exe –n <name> -ps <publishsettingsfile>
   3. Note the device connection strings displayed by the tool, as you will need them to provision the devices later, i.e.

Device AMQP address strings (for Raspberry PI/devices):

amqps://D1:xxxxxxxx@yyyyyyyy.servicebus.windows.net

amqps://D2:xxxxxxxx@yyyyyyyy.servicebus.windows.net

amqps://D3:xxxxxxxx@yyyyyyyy.servicebus.windows.net

amqps://D4:xxxxxxxx@yyyyyyyy.servicebus.windows.net

1. Create Azure Stream Analytics jobs:
   1. Make sure you have access to the preview: if not sign up at <https://account.windowsazure.com/PreviewFeatures>
   2. Create a new job “**Aggregates**”:
      1. “+” in lower left corner -> Data Services -> Stream Analytics -> Quick Create -> Job name “Aggregates”.
      2. Add Input
         1. Inputs tab -> Add an Input -> Data Stream, Event Hub
         2. Input Alias: “DevicesInput”
         3. Choose the namespace <name>-ns,
         4. Event Hub “ehdevices”
         5. Policy Name: “StreamingAnalytics”
         6. Serialization: JSON, UTF8
      3. Query:
         1. Query tab
         2. Copy/paste contents “Queries\Aggregates.sql”
         3. Save
      4. Create output
         1. Output tab -> Add an Output, Event Hub,
         2. Choose the namespace <name>-ns,
         3. Event Hub “ehalerts”
         4. Policy name “StreamingAnalytics”
         5. Serialization “JSON”, UTF8
      5. Start the Job
         1. Dashboard, Start
   3. Create a new job “**Alerts**”: as above, but use “alerts.sql” contents for the query (in step iii/2.)
2. Publish the Azure Websites
   1. In VS: Right-click on WebClient, Publish.
   2. Select Azure Web Sites, create new one.
   3. Publish
   4. Enable WebSockets for the new Azure Web site
      1. Browse to <https://manage.azure.com> and select your Azure Web Site.
      2. Click on the Configure tag. Then set WebSockets to On
   5. Open the site in a browser to verify it has deployed correctly.
3. Provision the Raspberry PI:
   1. Get a Raspbian NOOB SD Card or download a NOOBS image as per the instructions on <http://www.raspberrypi.org/downloads/>
   2. Connect to the Raspberry PI via SSH, either via a [USB-Serial adapter](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-5-using-a-console-cable/connect-the-lead) or via the network (enable once as per [these instructions](http://www.raspberrypi.org/documentation/remote-access/ssh/README.md) while booting via a monitor on HDMI and a USB keyboard).
      1. For Windows, download PuTTY and PSCP from [here](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html).
   3. Install mono and required SSL root certs:
      1. Run the following from a shell (i.e. via SSH):
         1. sudo apt-get update
         2. sudo apt-get upgrade
         3. sudo apt-get install mono-complete
         4. mozroots --import --ask-remove
   4. On your PC, update Devices\RaspberryPiGateway\scripts\autorun.sh with one of the amqp address strings returned by ConnectTheDotsCloudDeploy.exe, i.e. amqps://D1:xxxxxxxx@yyyyyyyy.servicebus.windows.net.
   5. Copy the required files from Devices\RaspberryPiGateway\bin\Debug to Raspberry’s /usr/pi/RaspberryPiGateway/
      1. Files: RaspberryPiGateway.exe, Amqp.Net.dll, Newtonsoft.Json.dll, scripts/autorun.sh
         1. You can use Devices\RaspberryPiGateway\scripts\scprpi.cmd to copy via SSH, or copy directly to the SD card.
   6. On the Raspberry PI, modify /etc/rc.local by adding one line to start the gateway program on every boot:
      1. /home/pi/RaspberryPiGateway/autorun.sh &
   7. On the Raspberry PI Make /usr/pi/CloudPI/autorun.sh executable:
      1. chmod 755 /usr/pi/RaspberryPiGateway/autorun.sh
4. Prepare the Arduino Uno R3:
   1. Install and run the [Arduino IDE](http://arduino.cc/en/Main/Software) (we recommend the 1.5.8 version, with the Windows Installer).
   2. Download the Weather Shield libraries from [here](https://dlnmh9ip6v2uc.cloudfront.net/assets/b/5/9/7/f/52cd8187ce395fa7158b456c.zip) (as per the instruction in the [Weather Shield Hookup guide](https://learn.sparkfun.com/tutorials/weather-shield-hookup-guide)), then import them in the IDE by clicking Sketch, Import Library, Add Library and selecting the downloaded ZIP file.
   3. In the Arduino IDE open Devices\Arduino\Weathershield.ino (it is modified from the original Spark fun sample to send data in JSON format)
   4. Compile and upload the weather shield sketch to the Arduino. You should now see temperature and other data on the serial monitor (shift-crtl-m).
5. Run Raspberry PI + Arduino
   1. Plug Arduino’s USB cable into one of the Raspberry PI USB ports.
   2. (Re-)start Raspberry PI

You should now see average temperature measurements showing up in the web browser every minute.

If you select “All”, you should see raw readings from the device coming in every second.

If the temperature exceeds 75 degrees (F), you should see an alert showing in the alerts table, once per minute while the temperature on any of the devices exceeds 75 degrees (F).

# Additional things to try:

To add **more devices**, you can modify the parameters to RaspberryPiGateway.exe in autorun.sh before copying the file to additional Raspberry PI: generate a new GUID as the device id, chose a new device display name and chose one of the other AMQP connection strings (so can turn off rogue devices in the cloud later).

**Multi-environment deployment** (i.e. test, demo, production): Use ASP.Net Web.config transforms to keep separate settings for each of your deployments. You can associate a transform with each publisher profile, and the deployment tool will generate a transform file if you specify the -transform command line parameter, and use the profile name as the -n parameter.

**Running without hardware:** The RaspberryPiGateway is a C# executable that can also run on Windows. Specify a COM port on the command line (i.e. -serial COM10) and you can receive serial data from an Arduino Uno.

If you do not have an Arduino or no Weather Shield, you can compile the RaspberryPiGateway to generate random temperature data by uncommenting the following line in Devices\RaspberryPIGateway\Program.cs:

//#define SIMULATEDATA