



Author

Jianyuan Ma

Phone

+46 10 505 0000

Date

2019-03-01

Project ID

Software Design Description

Azure Sphere



Contents

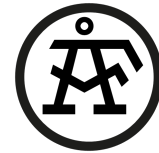
1 Introduction	4
2 Prerequisites	5
3 Install	6
3.1 Set up you board and install SDK	6
3.1.1 Connect the board	6
3.1.2 Install the Azure Sphere SDK Preview for Visual Studio	6
3.2 Update OS	7
3.2.1 Set up an account for Azure Sphere	7
3.3 Claim Device	8
3.4 Configure Wi-Fi	9
4 Simple Application	10
4.1 Build the Blink sample application	10
4.1.1 Prepare your device for development and debugging	10
4.1.2 Build and run the Blink sample	10
4.2 Deploy an application over the air	11
4.2.1 Prepare your device for OTA deployment	11
4.2.2 Link the device to a feed	12
4.3 Update a deployment	13
5 Deploy Azure IoT with Azure Sphere	14
5.1 Step 1: Create an Azure IoT Hub and DPS	14
5.1.1 Create IoT Hub	14
5.1.2 Create IoT Hub Device Provisioning Service	16
5.1.3 Link the IoT Hub and Device Provisioning Service	16
5.1.4 Clean up resources	16
5.2 Step2: Download the tenant authentication	17
5.3 Step3: Upload the tenant CA certificate to DPS and generate a verification code	17
5.4 Step 4: Verify the tenant CA certificate	18
5.5 Step 5: Use the certification to add you device to and enrollment group	20
5.6 Azure Sphere IoT Hub Sample Application	20
5.6.1 Add the device to the IoT hub	21
5.6.2 Start the Sample	22
5.6.3 Send and receive messages	22
5.6.4 Call a direct method on the device	23
5.6.5 Manage a device twin	23
6 Azure Sphere Read AM2302	25
6.1 Connect Azure Sphere to Arduino and AM2302	25
6.2 Clone the code from Github	25
6.3 Some Explanations about the project	25



6.3.1 Read Sensor Data in Arduino	26
6.3.2 Communicate through UART	26
6.4 Test the Output	27
7 Future Work	27



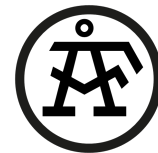
1 Introduction



2 Prerequisites

- An Azure Sphere development kit
- A PC running Windows 10 Anniversary Update or later
- Visual Studio 2017 Enterprise, Professional, or Community, version 15.7 or later

Download the Azure Sphere SDK for Visual Studio Preview. Once you've completed the download, follow the next chapter to install Azure Sphere and set up your board.



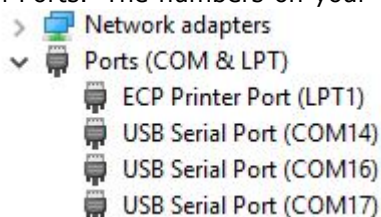
3 Install

3.1 Set up your board and install SDK

3.1.1 Connect the board

The development board connects to a PC through USB. The first time you plug in the board, the drivers should be automatically downloaded and installed. If the drivers are not installed automatically, right-click on the device name in Device Manager and select Update driver. Alternatively, you can download the drivers FTDI. Choose the driver that matches your Windows installation (32- or 64-bit).

To verify installation, open Device Manager. You should be able to see three USB Serial Ports. The numbers on your COM ports may be different from those in the figure.

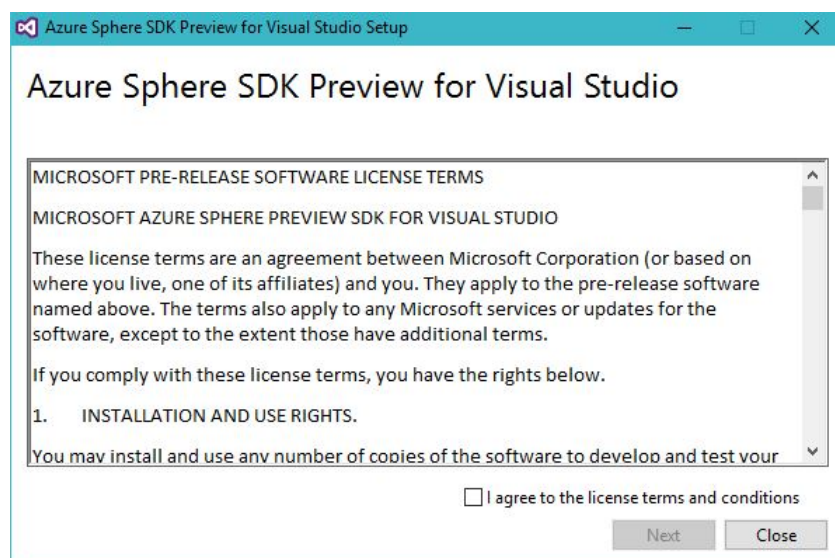


3.1.2 Install the Azure Sphere SDK Preview for Visual Studio

Azure_Sphere_SDK_Preview_for_Visual_Studio.exe installs the complete Azure Sphere software development kit (SDK).

To install the SDK:

1. Download the Azure Sphere SDK Preview for Visual Studio from Visual Studio Marketplace if you have not already done so. Save the downloaded file on your PC.
2. Run Azure_Sphere_SDK_Preview_for_Visual_Studio.exe from the download to install the SDK. Agree to the license terms and select Next.





3. Click Install to begin installation.



If the message "This product requires Visual Studio 2017, Version 15.7 or newer" appears, ensure that Visual Studio 2017 version 15.7 or more recent is installed on your PC.

The message "No product to install SDK on" may appear if you do not have Visual Studio version 15.7 or newer installed, or if you have just installed Visual Studio for the first time. If you see this message, either update your Visual Studio installation if necessary or restart your PC and return to this step.

4. Accept the elevation prompt if one appears.
5. When setup completes, restart your PC if the setup application requests it. The SDK is installed to all compatible editions of Visual Studio on your PC. The SDK requires Visual Studio version 15.7 or later.

3.2 Update OS

If the Azure Sphere device has never been used, it most likely needs to update the OS. Follow the steps to update the Azure Sphere OS.

1. Connect the board to the PC by USB.
2. Open an Azure Sphere Developer Command Prompt. It appears in the Start menu under Azure Sphere.
3. Issue the following command to update the Azure Sphere OS:

```
1 azsphere device recover
```

You should see output similar to this:

```
1 Starting device recovery. Please note that this may take up to 10 minutes.
2 Board found. Sending recovery bootloader.
3 Erasing flash.
4 Sending images.
5 Sending image 1 of 16.
6 Sending image 2 of 16.
7 . . .
8 Sending image 16 of 16.
9 Finished writing images; rebooting board.
10 Device ID: <GUID>
11 Device recovered successfully.
12 Command completed successfully in 00:02:37.3011134.
```

3.2.1 Set up an account for Azure Sphere

Azure Sphere uses Azure Active Directory (AAD) to enforce enterprise access control. Therefore, to use Azure Sphere, you need a Microsoft work or school account (sometimes called an organizational account) that is associated with an AAD.



The AF account is Microsoft work account, to find out you can use the account, open an Azure Sphere Developer Command prompt (on the Start menu under Azure Sphere) and sign in to Azure Sphere with your work or school account:

```
1 azsphere login
```

In response, azsphere prompts you to pick an account. Choose the AF account and type your password if required. If you see a dialog box requesting that an admin grant permission to use the Azure Sphere Utility, you'll need to log in as an administrator or obtain admin approval.

If login succeeds, the command returns a list of the Azure Sphere tenants that are available for you. You should see the similar output to this:

```
1 The selected Azure Sphere tenant 'AFTechnology' (4679af44-ad27-4c1d-9ef8-4f76b540edae) will be retained.
2 Successfully logged in with the selected AAD user. This authentication will be used for subsequent commands.
3 Command completed successfully in 00:00:15.3946315.
```

For more information please visit the Microsoft document on Set up an account.

3.3 Claim Device

Every device must be "claimed" by an Azure Sphere tenant. Claiming the device associates its unique, immutable device ID with your Azure Sphere tenant. The Azure Sphere Security Service uses the device ID to identify and authenticate the device.

We recommend that each company or organization create only one Azure Sphere tenant.

Important

Claiming is a one-time operation that you cannot undo even if the device is sold or transferred to another person or organization. A device can be claimed only once. Once claimed, the device is permanently associated with the Azure Sphere tenant.

Before you claim your device, complete these steps to ensure that you use the right work/school account to create and access your Azure Sphere tenant. Your device must be connected to your PC before you create the tenant, and you can only use the device to create a single tenant.

To claim your device:

1. Connect your device to your PC.
2. Open an Azure Sphere Developer Command Prompt, which is available in the Start menu under Azure Sphere.
3. Claim your device. After you claim your device into a tenant, you cannot move it to a different tenant.

```
1 azsphere device claim
```

You should see the similar output like this:

```
1 Claiming device.
2 Successfully claimed device ID '
   B3E012CE682BA0A6235866AB3A87D838A4817E5C539832A34BF7A715CA8D015FF99C84B909CB2886916259AD186B212E148FC9C4B
3 F8BB6A275A11A2B9495D578' into tenant 'AFTechnology' with ID '4679af44-ad27-4c1d-9ef8-4f76b540edae'.
4 Command completed successfully in 00:00:03.4394424.
```




3.4 Configure Wi-Fi

Before you can configure Wi-Fi, you must:

- Install the SDK and set up the development board
- Update the OS
- Claim the device

Follow these steps to configure Wi-Fi on your Azure Sphere device:

1. Connect your Azure Sphere board to your PC over USB.
2. Open an Azure Sphere Developer Command Prompt.
3. Register the device's MAC address if your network environment requires it. Use the following command to get the MAC address:

```
1 azsphere device wifi show-status
```

4. Add your Wi-Fi network to the device by using the `azsphere device wifi add` command as follows:

```
1 azsphere device wifi add --ssid <yourSSID> --key <yourNetworkKey>
```

Replace `<yourSSID>` with the name of your network and `<yourNetworkKey>` with your WPA/WPA2 key. Azure Sphere devices do not support WEP. Network SSIDs are case-sensitive.

To add an open network, omit the `-key` flag.

If your network SSID or key has embedded spaces, enclose the SSID or key in quotation marks. If the SSID or key includes a quotation mark, use a backslash to escape the quotation mark. Backslashes do not require escape if they are part of a value.

```
1 azsphere device wifi add --ssid "New SSID" --key "key \"value\" with quotes"
```

It typically takes several seconds for networking to be ready on the board, but might take longer, depending on your network environment.

5. Use the `azsphere device wifi show-status` command to check the status of the connection. During update, the `azsphere device wifi show-status` command may temporarily show an unknown configuration state.

```
1 azsphere device wifi show-status
2
3 SSID : azureTest
4 Configuration state : enabled
5 Connection state : connected
6 Security state : psk
7 Frequency : 5180
8 Mode : station
9 Key management : WPA2-PSK
10 WPA State : COMPLETED
11 IP Address : 172.28.22.2
12 MAC Address : 2c:f7:f1:08:72:df
13
14 Command completed successfully in 00:00:01.3747119.
```



4 Simple Application

4.1 Build the Blink sample application

This quickstart shows how to enable application development on an Azure Sphere device and how to build and debug a sample application. It uses the Blink sample, which is part of the Azure Sphere SDK. The Blink sample shows how to access GPIOs and LEDs on the development board.

This section requires:

- Your Azure Sphere device is connected to your PC
- You have completed all the steps to section 3

4.1.1 Prepare your device for development and debugging

By default, Azure Sphere devices are “locked”, which is they do not allow applications to be loaded or debugging on the board from a PC.

The `azsphere device prep-debug` command configures the device to accept applications from a PC for debugging and loads the debugging server onto the device. It also assigns the device to a device group that does not allow over-the-air (OTA) application updates. During application development and debugging, you should leave the device in this group so that OTA application updates do not overwrite the application under development.

To prepare your device:

1. Make sure that your Azure Sphere device is connected to your PC, and your PC is connected to the internet.
2. In an Azure Sphere Developer Command Prompt window, type the following command:

```
1 azsphere device prep-debug
```

You should see output similar to the following:

```
1 Getting device capability configuration for application development.
2 Downloading device capability configuration for device ID '
  B3E012CE682BA0A6235866AB3A87D838A4817E5C539832A34BF7A715CA8D015FF99C84B909CB2886916259AD186B212E148FC9C4BF8BB6A275A11A2B94
  '
3 Successfully downloaded device capability configuration.
4 Successfully wrote device capability configuration file 'C:\Users\A548068\AppData\Local\Temp\
  tmpCCF6.tmp'.
5 Setting device group ID 'cd037ae5-27ca-4a13-9e3b-2a9d87f9d7bd' for device with ID '
  B3E012CE682BA0A6235866AB3A87D838A4817E5C539832A34BF7A715CA8D015FF99C84B909CB2886916259AD186B212E148FC9C4BF8BB6A275A11A2B94
  '
6 Successfully disabled over-the-air updates.
7 Enabling application development capability on attached device.
8 Applying device capability configuration to device.
9 Successfully applied device capability configuration to device.
10 The device is rebooting.
11 Installing debugging server to device.
12 Deploying 'C:\Program Files (x86)\Microsoft Azure Sphere SDK\DebugTools\gdbserver.imagepackage' to
  the attached device.
13 Image package 'C:\Program Files (x86)\Microsoft Azure Sphere SDK\DebugTools\gdbserver.imagepackage'
  has been deployed to the attached device.
14 Application development capability enabled.
15 Successfully set up device '
  B3E012CE682BA0A6235866AB3A87D838A4817E5C539832A34BF7A715CA8D015FF99C84B909CB2886916259AD186B212E148FC9C4BF8BB6A275A11A2B94
  ' for application development, and disabled over-the-air updates.
16 Command completed successfully in 00:00:41.4376552.
```

4.1.2 Build and run the Blink sample

1. Start **Visual Studio 2017** and go to **File>New>Project**. The templates for the Azure Sphere are available in **Visual C++>Cross Platform>Azure Sphere**. Select **Blink Sample for MT3260 RDB(Azure Sphere)**.
2. Enter a name and location for the project, or you can just click OK in default name and location.
3. Open the `main.c` in the **Solution Explorer>Source Files**, if you cannot see the **Solution Explorer** you can find it in **View>Solution Explorer**. Navigate to the line that tests the value of `newButtonState` and press F9 to set a breakpoint:



```
1 if (newButtonState == GPIO_Value_Low) {
```

4. Ensure that your board is connected to your PC by USB. Then select **Remote GDB Debugger** from the menu bar or press F5.



5. If Visual Studio prompt to build the project, select Yes, it will creates an image package, sideloads it onto the board, and starts it in debug mode. Sideloads means the application is delivered directly from the PC over a wired connection, rather than delivered over the air(OTA) by Wi-Fi.

Tip

Note the path in the Build output, which indicates the location of the output image package on your PC. You'll use the image package later for the deploy over Wi-Fi

6. Press button A. Visual Studio stops at the breakpoint that you set. Open **Debug>Windows** and select Autos to display the variables that are used in the current and previous statements. You should see the values like:

Autos		
Name	Value	Type
GPIO_Value_Low	GPIO_Value_Low	enum (...)
buttonState	1 '\001'	GPIO_Value_Type
newButtonState	0 '\000'	GPIO_Value_Type

7. Select **Continue** in the menu bar. Execution pauses again at this breakpoint. Now the value of newButtonState variable is 1, which indicates a button release. Press F9 to remove the breakpoint. Then select **Continue**. Now, each time press and release Button A, the blink rate changes.
8. To see the messages from the debugger, select **Debug>Windows>Output** in the dropdown menu.
9. When you are done debugging, select the Stop icon in the menu bar or press Shift+F5.

4.2 Deploy an application over the air

This section shows how to create your first over-the-air(OTA) application deployment. OTA deployment delivers an application through a feed to the devices in a devices group that match the target stock-keeping unit for the feed.

This section requires:

- Your Azure Sphere device is connected to your PC
- You have completed all the steps to section 3
- You have completed the previous section and retained the image package for the application

4.2.1 Prepare your device for OTA deployment

Before you start the OTA deployment process, your Azure Sphere must be ready to accept OTA application updates which means "lock" the device and enable OTA update so that it able to be operated by the customer site. The **azsphere device prep-field** command is the simplest way to do this. This command:

- Disables the ability for Visual Studio to load applications onto the device, so that only OTA applications can be loaded
- Assigns a new product SKU to the device
- Assigns the device to a new device group that enables OTA application updates



The **azsphere device prep-field** command works on the device that is connected to PC. It has the following form:

```
1 azsphere device prep-field --newdevicegroupname <unique-dg-name> --newsku <unique-sku-name>
```

The `--newdevicegroupname` flag specifies a name for the new device group that the command creates. All device groups created by this command support automatic OTA application updates. Supply a descriptive name that is unique among the device group names in your Azure Sphere tenant.

The `--newsku` flag specifies a name for the new product SKU that the command creates. A product SKU identifies a model of the connected device that contains an Azure Sphere chip. Each SKU in an Azure Sphere tenant must have a unique name.

The Azure Sphere related to the device has created a device group named "AFSphereGroup" and a product SKU named "AFSphereProductSKU". The command created the group and SKU is:

```
1 azsphere device prep-field --newdevicegroupname "AFSphereGroup" --newsku "AFSphereProductSKU"
2
3 Removing applications from device.
4 Component 'e3c4a1ff-8456-49e4-a353-a76235e5136f' deleted or was not present beforehand.
5 Removing debugging server from device.
6 Component '8548b129-b16f-4f84-8dbe-d2c847862e78' deleted or was not present beforehand.
7 Successfully removed applications from device.
8 Locking device.
9 Downloading device capability configuration for device ID '
   B3E012CE682BA0A6235866AB3A87D838A4817E5C539832A34BF7A715CA8D015FF99C84B909CB2886916259AD186B212E148FC9C4BF8BB6A275A11A2B9495D578
   '.
10 Successfully downloaded device capability configuration.
11 Applying device capability configuration to device.
12 Successfully applied device capability configuration to device.
13 The device is rebooting.
14 Successfully locked device.
15 Creating a new device group with name 'AFSphereGroup'.
16 Setting device group ID '5ece1a6b-44b8-40cb-b0e3-623dcd5bc6e' for device with ID '
   B3E012CE682BA0A6235866AB3A87D838A4817E5C539832A34BF7A715CA8D015FF99C84B909CB2886916259AD186B212E148FC9C4BF8BB6A275A11A2B9495D578
   '.
```

You aren't required to create a new device group and SKU every time you prepare a device for field use. Typically, you would create one SKU for each product model, and one device group for each collection of devices that you want to update together. To assign the device to different device group use the following command format:

```
1 azsphere device prep-field --devicegroupid <device-group-id>
```

For example, to change the group to AFSphereGroup which created previously, use group id 5ece1a6b-44b8-40cb-b0e3-623dcd5bc6e in the command.

4.2.2 Link the device to a feed

The next step is to link your device to a feed that delivers the Blink application. You must supply:

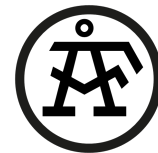
- The ID of Azure Sphere OS feed on which the application depends
- The path to the image package file that Visual Studio created for the Blink application
- A name for the feed that will deliver the application

To link to a feed:

1. Get the feed ID for the Retail Azure Sphere feed, which delivers the Azure Sphere OS.

```
1 azsphere feed list
2 Listing all feeds.
3 Retrieved feeds:
4 --> [3369f0e1-dedf-49ec-a602-2aa98669fd61] 'Retail Azure Sphere OS'
5 --> [82bacf85-990d-4023-91c5-c6694a9fa5b4] 'Retail Evaluation Azure Sphere OS'
6 Command completed successfully in 00:00:03.0017019.
```

Copy the ID of the Retail Azure Sphere OS feed to use in the next step.



2. Issue the azsphere device link-feed command to create a feed and associate it with the Blink image package that you created previously.

```
1 azsphere device link-feed --dependentfeedid 3369f0e1-dedf-49ec-a602-2aa98669fd61 --imagepath "C:\
  Users\A548068\Documents\Azure\Blink\Mt3620Blink1\Mt3620Blink1\bin\ARM\Debug\Mt3620Blink1.
  imagepackage" --newfeedname "Mt3620Blink1"
```

The `--dependentfeedid` flag supplies the ID of the Retail feed.

The `--imagepath` flag provides the path to the image package file for the Blink application. The full path to the image file is displayed in the Visual Studio 2017 Build Output windows. The `azsphere device link-feed` command uploads the image package file to the Azure Sphere Security Service and creates an image set with a unique name.

The `--newfeedname` flag provides a name for the feed that the command creates. Feed names must be unique in an Azure Sphere tenant, so specify a name that distinguishes this feed from any others.

The output could be:

```
1 Getting the details for device with ID '
  B3E012CE682BA0A6235866AB3A87D838A4817E5C539832A34BF7A715CA8D015FF99C84B909CB2886916259AD186B212E148FC9C4BF8BB6A275A11A2B94
  '.
2 Uploading image from file 'C:\Users\A548068\Documents\Azure\Blink\Mt3620Blink1\Mt3620Blink1\bin\ARM
  \Debug\Mt3620Blink1.imagepackage':
3 --> Image ID: 215c72b6-7472-4337-80c8-da98ef536f39
4 --> Component ID: e3c4a1ff-8456-49e4-a353-a76235e5136f
5 --> Component name: 'Mt3620Blink1'
6 Removing temporary state for uploaded image.
7 Create a new feed with name 'Mt3620Blink1'.
8 Adding feed with ID '8354ecb2-5348-4582-9e0c-ae9fc055f3db' to device group with ID '5ece1a6b-44b8
  -40cb-b0e3-623dcd5bc6e'.
9 Creating new image set with name 'ImageSet-Mt3620Blink1-2019.02.22-16.15.07+01:00' for images with
  these IDs: 215c72b6-7472-4337-80c8-da98ef536f39.
10 Adding image set with ID 'e05f2494-a716-48cb-9f41-4d1e1e6fbb7a' to feed with ID '8354ecb2
  -5348-4582-9e0c-ae9fc055f3db'.
11 Successfully linked device '
  B3E012CE682BA0A6235866AB3A87D838A4817E5C539832A34BF7A715CA8D015FF99C84B909CB2886916259AD186B212E148FC9C4BF8BB6A275A11A2B94
  ' to feed with ID '8354ecb2-5348-4582-9e0c-ae9fc055f3db'.
12 Command completed successfully in 00:00:28.3808069.
```

This command creates a feed that is linked to the device group to which the attached device belongs—that is, the “AFSphereGroup” created earlier in this quickstart. It will deliver the “Mt3620Blink1” application to all Azure Sphere devices in the group whose product SKU is “AFSphereProductSKU”.

4.3 Update a deployment

Get the feed ID we created above:

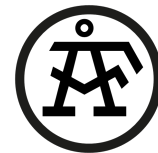
```
1 azsphere feed list
2
3 Listing all feeds.
4 Retrieved feeds:
5 --> [3369f0e1-dedf-49ec-a602-2aa98669fd61] 'Retail Azure Sphere OS'
6 --> [82bacf85-990d-4023-91c5-c6694a9fa5b4] 'Retail Evaluation Azure Sphere OS'
7 --> [8354ecb2-5348-4582-9e0c-ae9fc055f3db] 'Mt3620Blink1'
8 Command completed successfully in 00:00:02.2268023.
```

Copy the ID of Mt3620Blink1 for the application update.

To update a feed with a new version of your application, use the `azsphere component publish` command. This command uploads a new image package, creates a new image set, and adds the new image set to an existing feed.

For example, the following command update the Blink Application to the feed just created named Mt3620Blink1:

```
1 azsphere component publish --feedid 8354ecb2-5348-4582-9e0c-ae9fc055f3db --imagepath "C:\Users\A548068\
  Documents\Azure\Blink\Mt3620Blink1\Mt3620Blink1\bin\ARM\Debug\Mt3620Blink1.imagepackage"
```



5 Deploy Azure IoT with Azure Sphere

To use your Azure Sphere devices with the IoT, you can set up an Azure IoT Hub to work with your Azure Sphere tenant. After you have completed the tasks in this section, any device that is claimed by your Azure Sphere tenant will be automatically enrolled in your IoT hub when it first comes online and connects to the Device Provisioning Service (DPS). Therefore, you only need to complete these steps once.

The following sections will guide you to create an IoT Hub with the Azure Sphere.

5.1 Step 1: Create an Azure IoT Hub and DPS

This section shows how to set up the Azure cloud resources in the portal for provisioning your devices.

5.1.1 Create IoT Hub

This section describes how to create an IoT hub using the [Azure Portal](#)

1. Log in to the [Azure Portal](#)
2. Choose **+Create a resource**, then search for **IoT Hub**.
3. Click **Create** button on the left bottom side of the IoT Hub description.

Subscription: Select the subscription to use for your IoT hub.

Resource Group: You can create a new resource group or use an existing one. To create a new one, click Create new and fill in the name you want to use. To use an existing resource group, click Use existing and select the resource group from the dropdown list.

Region: This is the region in which you want your hub to be located. Select the location closest to you from the dropdown list.

IoT Hub Name: Put in the name for your IoT Hub. This name must be globally unique. If the name you enter is available, a green check mark appears.



- Click **Next: Size and scale** to continue creating your IoT hub.

IoT hub
Microsoft

[Basics](#) [Size and scale](#) [Review + create](#)

Each IoT Hub is provisioned with a certain number of units in a specific tier. The tier and number of units determine the maximum daily quota of messages that you can send. [Learn more](#)

SCALE TIER AND UNITS

* Pricing and scale tier ⓘ S1: Standard tier [Learn how to choose the right IoT Hub tier for your solution](#)

Number of S1 IoT Hub units ⓘ 1
This determines your IoT Hub scale capability and can be changed as your need increases.

Pricing and scale tier ⓘ S1	Device-to-cloud-messages ⓘ Enabled
Messages per day ⓘ 400,000	Message routing ⓘ Enabled
Cost per month 218.43 SEK	Cloud-to-device commands ⓘ Enabled
	IoT Edge ⓘ Enabled
	Device management ⓘ Enabled

Advanced Settings

On this screen, you can take the defaults and just click Review + create at the bottom.

Pricing and scale tier: You can choose from several tiers depending on how many features you want and how many messages you send through your solution per day. The free tier is intended for testing and evaluation. It allows 500 devices to be connected to the IoT hub and up to 8,000 messages per day. Each Azure subscription can create one IoT Hub in the free tier.

IoT Hub units: The number of messages allowed per unit per day depends on your hub's pricing tier. For example, if you want the IoT hub to support ingress of 700,000 messages, you choose two S1 tier units.

Advanced / Device-to-cloud partitions: This property relates the device-to-cloud messages to the number of simultaneous readers of the messages. Most IoT hubs only need four partitions.

- Click Review + create to review your choices. You should see the similar screen as:

[Home](#) > [New](#) > [IoT Hub](#) > IoT hub

IoT hub
Microsoft

[Basics](#) [Size and scale](#) [Review + create](#)

BASICS

Subscription ⓘ	Free Trial
Resource Group ⓘ	AFGöteborgAzureSphere
Region ⓘ	West US
IoT Hub Name ⓘ	IoT-Hub-test-af

SIZE AND SCALE

Pricing and scale tier ⓘ	S1
Number of S1 IoT Hub units ⓘ	1
Messages per day ⓘ	400,000
Cost per month	218.43 SEK

- Click Create to create your new IoT hub. Creating the hub takes a few minutes.



7. In our condition, the IoT Hub has already been created as "azure-temperature-hub", but you can create a new one.

5.1.2 Create IoT Hub Device Provisioning Service

1. Click the **Create a resource** button and searching for the **IoT Hub Device Provisioning Service**. Then click **Create**
2. Filling the following information for your new Device Provisioning service and click **Create**.
 - **Name**: Provide a unique name of the instance. If the name is available, a green check mark appears in the end.
 - **Subscription**: Choose the subscription that you want to use to create this Device Provisioning service instance.
 - **Resource Group**: You can create a new resource group in this field, or choosing an existing one.
 - **Location**: Select the location to your device.
3. Once the service is successfully deployed, the notifications will notify you, then click **Go to resource**
4. In our condition, the name of the provision is "afgoteborg-azure-provision"

5.1.3 Link the IoT Hub and Device Provisioning Service

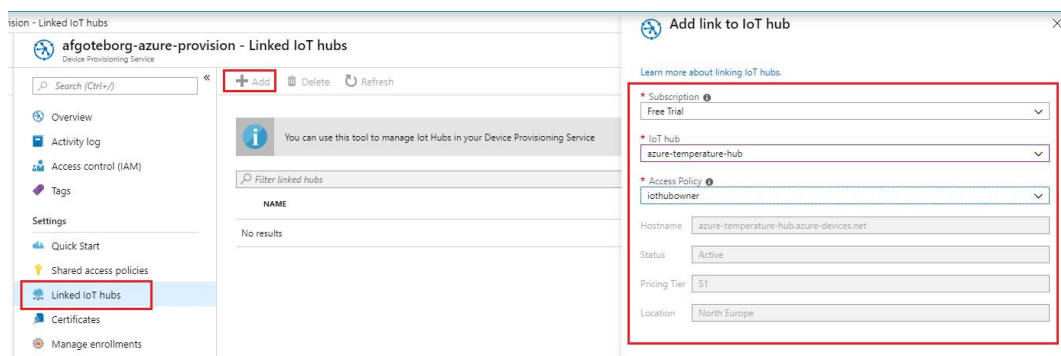
This section describes the configuration in the Device Provisioning Service sets the IoT hub for the devices will be provisioned.

1. Click the **All resources** button from the left-hand menu of the Azure portal. Select the Device Provisioning service that you created in the preceding section.
2. Under the settings, select Linked IoT hubs. Then click **+Add**.
3. In the **Add link to IoT hub**, provide the following information to link your new Device Provisioning service instance to an IoT hub. Then click Save.
 - **Subscription**: Select the subscription containing the IoT hub that you want to link with your new Device Provisioning service instance.
 - **IoT hub**: Select the IoT hub to link with your new Device Provisioning service instance.
 - **Access Policy**: Select iothubowner as the credentials for establishing the link with the IoT hub.
4. You should be able to see the newly created Linked IoT hubs in the list.

5.1.4 Clean up resources

If you want to continue on this tutorial, you do not need to do this step. If you are not, use the following steps to delete the resources just created.

1. From the left-hand menu in the Azure portal, click **All resources** and then select your Device Provisioning service. At the top of the **All resources** blade, click **Delete**.
2. From the left-hand menu in the Azure portal, click **All resources** and then select



your IoT hub. At the top of the All resources blade, click **Delete**.

5.2 Step2: Download the tenant authentication

After finished the previous section, we can set up the Azure Sphere device with the IoT.

1. Open the Azure Sphere Developer Command Prompt in the Start menu.
2. Sign in with the user for your Azure Active Directory:

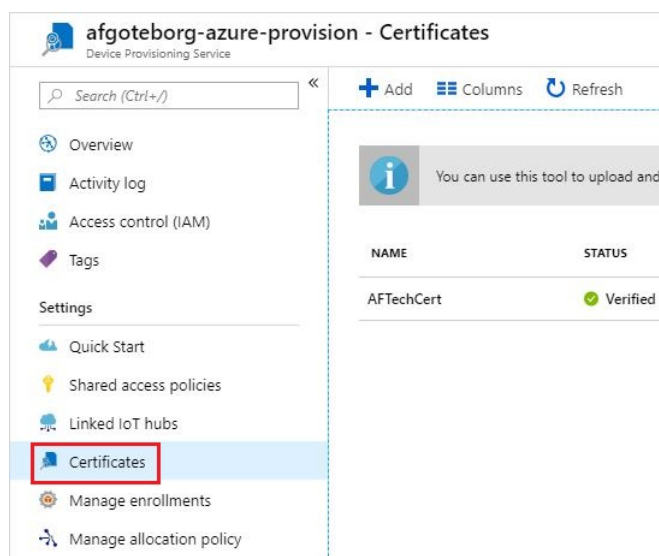
```
1 azsphere login
```

3. Download the Certificate Authority (CA) certificate for your Azure Sphere tenant, the output file must have the .cer extension.

```
1 azsphere tenant download-CA-certificate --output AFTechCert.cer
```

5.3 Step3: Upload the tenant CA certificate to DPS and generate a verification code

1. Log in to the [Azure Portal](#) and navigate to the DPS(Device Provisioning Service) you created in 5.1
2. Open **Certificates** from the menu.





3. Then click **ADD** to add a new certificate and enter a name for your certificate. Upload the .cer file from Step 2.

4. After you are notified that the certificate uploaded successfully, click **Save**.
5. The new certificate is listed under the explorer. Note the STATUS of the certificate is Unverified. Click on this certificate.

NAME	STATUS	EXPIRY	SUBJECT	THUMBPRINT	CREATED
AFTechCert	Verified	Mon Nov 30 2019	Microsoft Azure	9C69FF7C3E99	Tue Feb 25 2019
CertTest	Unverified	Mon Nov 30 2019	Microsoft Azure	9C69FF7C3E99	Fri Mar 01 2019

6. In **Certificate Details**, click Generate Verification Code. The DPS creates a Verification Code that you can use to validate the certificate ownership. Copy the code to your clipboard for use in the next step.

5.4 Step 4: Verify the tenant CA certificate

1. Back to the Azure Sphere Developer Command Prompt. Download the validation certificate based on your own tenant CA certificate. Replace code in the command with the verification code from the previous step.

```
1 azsphere tenant download-validation-certificate --output AFTechValidationCert.cer --  
verificationcode <code>
```

The Azure Sphere Security Service signs the validation certificate with the verification code from the DPS to your own CA.

2. Return to the Azure Portal to upload the validation certificate to DPS. In **Certificate Details**, use the File Explorer icon next to the **Verification Certificate .pem or .cer file** field to upload the signed verification certificate. Then click **Verify**.



Certificate Details

CertTest

Delete

Certificate Name ⓘ
CertTest

ETag ⓘ
AAAAACECd0=

Subject ⓘ
Microsoft Azure Sphere 4579af44-ad27...

Expiry ⓘ
Mon Nov 30 2020 08:23:02 GMT+0100 ...

Thumbprint ⓘ
9C69FF7C3E957C0FD9733546ECCDB3F...

Created ⓘ
Fri Mar 01 2019 09:07:30 GMT+0100 (C...

Updated ⓘ
Fri Mar 01 2019 09:07:30 GMT+0100 (C...

Verification Code ⓘ
2FD99FEFDE308F36D1D645A9A737C78...

[Generate Verification Code](#)

* Verification Certificate .pem or .cer file. ⓘ
"AFTechValidationCert.cer"

- The **STATUS** of the certificate changes to **Verified** in the list. Click **Refresh** if it does not update automatically.



5.5 Step 5: Use the certification to add you device to and enroll-ment group

1. In the same DPS view as previous step. Select **Manage enrollment**, then click **Add enrollment group**.
2. In the Add Enrollment Group panel, create a name for your enrollment group, select CA Certificate as the **Certificate type**, and select the certificate that you validated in the previous step.
3. Click **Save**. You should see the group name appears under the **Enrollment Group** on the successful creation of the enrollment group.

The screenshot shows the 'Add Enrollment Group' form. At the top, there is a 'Save' button highlighted with a red box. Below it, the 'Group name' field contains 'test-enrollment-group'. The 'Attestation Type' is set to 'Certificate'. The 'Certificate Type' is set to 'CA Certificate'. The 'Primary Certificate' dropdown is set to 'AFTechCert'. The 'Secondary Certificate' dropdown is set to 'No certificate selected'. The 'Select how you want to assign devices to hubs' dropdown is set to 'Evenly weighted distribution'. The 'Select the IoT hubs this group can be assigned to' dropdown is set to 'azure-temperature-hub.azure-devices.net'. There is a 'Link a new IoT hub' button at the bottom.

5.6 Azure Sphere IoT Hub Sample Application

The Azure IoT Hub sample application shows the communication between the Azure IoT Hub and the Azure Sphere Device.

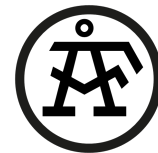
This sample uses the Connected Service for Azure Sphere, which is installed with the Azure Sphere SDK. The Visual Studio provides a template for connecting the Azure Sphere device to your IoT hub and communication with the hub. Application that use the Connected Service functionality can send messages to and receive messages from and IoT Hub, maintain a device twin, and direct method calls from cloud service.

This sample does the following:

1. Displays the currently connected WiFi network
2. Blinks LED 1 constantly. Pressing button A changes the rate. The rate is also stored in the device twin so that a cloud service program can change it.
3. Sends a message to the IoT Hub when you press button B.
4. Lights LED 3 green after start-up to indicate that the device has connected to the IoT Hub and the application has successfully authenticated with the hub.
5. Changes the color of LED 1 in response to a direct method call.

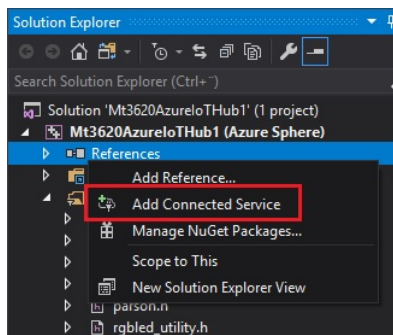
You must set up the board:

1. Connect to a PC and to WiFi network.
2. Change the board to prep-debug.

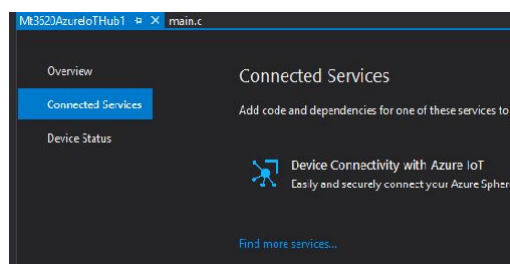


5.6.1 Add the device to the IoT hub

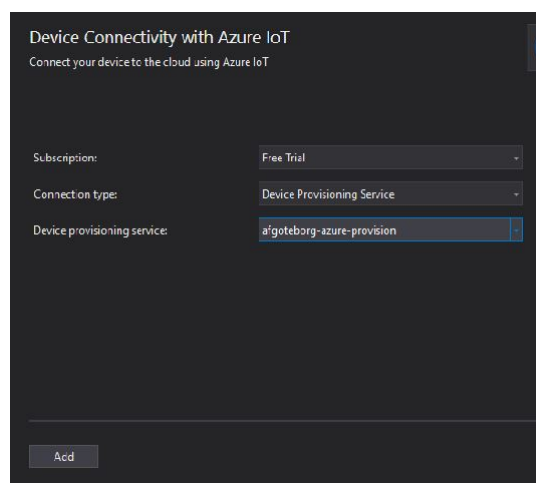
1. Start Visual Studio 2017 and go to **File>New>Project**. Navigate to the template in **Visual C++>Cross Platform>Azure Sphere**. Select Azure IoT Hub Sample for MT3620 RDB (Azure Sphere), specify a name and location or use the defaults, and select OK.
2. In Solution Explorer, right click on **Reference** and then select **Add Connected Service**



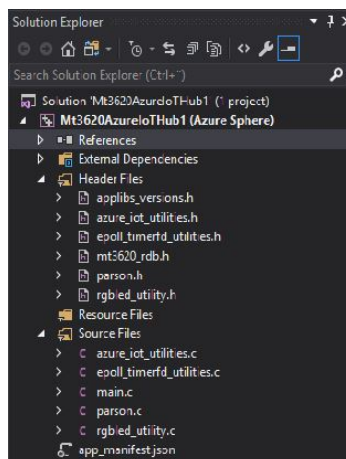
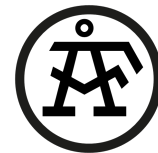
3. Select **Device Connectivity with Azure IoT** from the list of connected services.



4. It may ask you to log in to the account you're using with your Azure Sphere tenant.
5. In the Device Connectivity, select your Azure subscription from the **Subscription** dropdown. Select Device Provisioning Service from the **Connection Type** dropdown. Select your Device Provisioning Service from the **Device Provisioning Service** dropdown.



6. Click Add.
7. In Solution Explorer, you should now see `azure_iot_utilities.h` and `azure_iot_utilities.c` in your solution. In addition, the host name for the Azure IoT Hub has been added to the Capabilities section of the `app_manifest.json` file. An application can connect only to the internet hosts that are specified in the AllowedConnections field.



5.6.2 Start the Sample

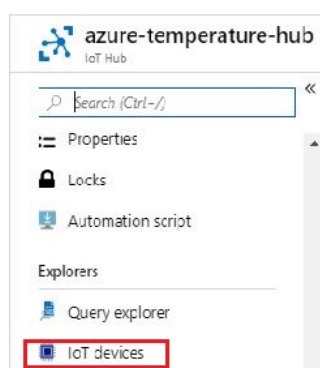
1. Open main.c in the sample application
2. Press F5 to build, load and start the sample.
The sample opens the handles to the buttons and the RGB LEDs, it also displays the connected WiFi network.
Then it starts the main loop. You should see LED 1 start to blink and see output like the following in the Output window:

```
Output
Show output from: Device Output
Remote debugging from host 192.168.35.1
INFO: Azure IoT application starting.
INFO: Opening MT3620_RDB_BUTTON_A.
INFO: Opening MT3620_RDB_BUTTON_B.
INFO: Open RGB LED 0.
INFO: Open RGB LED 1.
INFO: Open RGB LED 2.
INFO: Currently connected Wifi network:
INFO: SSID "azureTest", BSSID 32:52:cb:ca:aa:ff, Frequency 5180MHz.
[Azure IoT] Using HSM cert at /run/daa/4679af44-ad27-4c1d-9ef8-4f76b540edae
[Azure IoT] IoTHubDeviceClient_CreateWithAzureSphereDeviceAuthProvisioning returned 'AZURE_SPHERE_PROV_RE
[Azure IoT Hub client] INFO: AzureIoT_DoPeriodicTasks calls in progress...
```

5.6.3 Send and receive messages

In Azure Portal you can send and receive the messages from your device.

1. Log in to Azure Portal with the IoT Hub we created previously.
2. Select **All Resources** in the left panel. Click on the IoT hub in the resource list.
3. Navigate to **IoT devices** in the Explorers section in the IoT Hub panel. You may need to scroll down a little bit.



4. You should see the device id in the list. Click on the device id to open the **Device**



details panel.

5. Click on the **Message to device**. In the **Message to device** type some messages, for example "Hello from Portal" in the **Message Body** textbox. Then click **Send Message**. Back to Visual Studio 2017, in the output Windows you should see a

message **[Azure IoT] INFO: Received message 'Hello from Portal' from IoT Hub** coming out in the output windows.

5.6.4 Call a direct method on the device

1. Back to Device Details in the portal. You may see the prompt said "Your unsaved edit will be discarded", click **OK**. Then select **Direct method**.
2. In the **Direct method** panel, fill in **LedColorControlMethod** under the **Method Name** box, and **"color": "green"** in the Method payload box. Then click **Invoke Method**.

3. You should see the following output in the Visual Studio output window and LED 1 start blinking green.

```
1 [Azure IoT] INFO: Trying to invoke method LedColorControlMethod
2 INFO: LED color set to: 'green'.
```

In the meantime, check the Result in the Direct Method in the portal, the return payload should be

```
1 {"success": true, "message": "led color set to green"}
```

5.6.5 Manage a device twin

A device twin is a JSON document in the cloud that stores information about a device. Use the device twin to synchronize status information and maintain device properties



that should be accessible to cloud service applications as well as to applications that run on the device.

1. Back to Device Details in the portal. Then select **Device twin**.
2. You should see the JSON document in the panel. Find **LedBlinkRateProperty** under **desired** properties, if you cannot find it, please add it before **metadata**. Set the value 0, 1 or 2.
3. Then click **Save**.

```
"properties": {
  "desired": {
    "azureiot*com^dtracing^1": {
      "sampling_mode": 2,
      "sampling_rate": 3
    },
    "LedBlinkRateProperty": 1,
    "$metadata": {
      "$lastUpdated": "2019-03-01T12:59:02.645035Z",

```

4. In Visual Studio, check the Output window. You should see:

```
1 INFO: Received desired value 1 for LedBlinkRateProperty, setting it to 1.
2 [Azure IoT] INFO: Set reported property 'LedBlinkRateProperty' to value 1.
```

In the meantime, the blinking rate of the LED 1 should be changed.



6 Azure Sphere Read AM2302

This section will explain the functions to read the sensors data from a AM2302. Due to the specific GPIO function in Azure Sphere, it may not able to read the one-wire sensor, such as the AM2302. To have a work around, the sensor connects to an Arduino device, and send the sensor data through UART to Azure Sphere.

This section assume you already have:

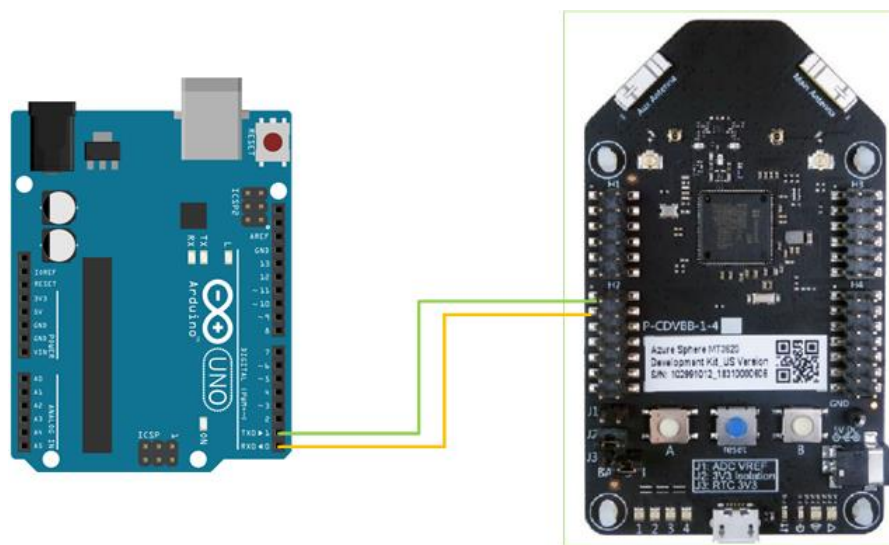
- Your Azure Sphere device is connected to your PC
- You have completed all the steps to section 4
- You have git command installed in a PC.
- You have Visual Studio 2017
- You have Arduino IDE

6.1 Connect Azure Sphere to Arduino and AM2302

The Azure Sphere communicates with Arduino through UART. The UART pins on Azure Sphere are PIN1(RXD0) and PIN3(TXD0) in Header 2. The wirings are:

- Arduino TX(PIN1)->AzureSphere RXD0(Header2, PIN1)
- Arduino RX(PIN0)->AzureSphere TXD0(Header2, PIN3)

The diagram shows how to connect them.



Connecting Arduino to the AM2302 is fairly straight forward, there are three cables to AM2302, red cable to VCC(3.3v/5v), black cable to GND and yellow cable is the signal out, in this project it connects to PIN2. The diagram shows how to connect the Arduino with AM2302:

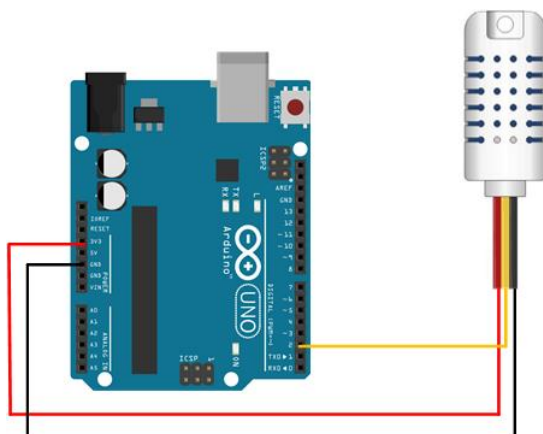
6.2 Clone the code from Github

The code can be download from Github using following command:

```
1 git clone https://github.com/jianyuanMa/AzureSphereDHT22.git
```

6.3 Some Explanations about the project

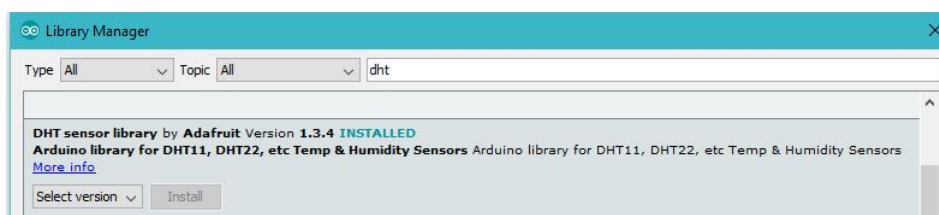
The project has two parts. First part is the Arduino read the sensor data periodically and send it to the UART while the Azure Sphere receives the data as the second part.



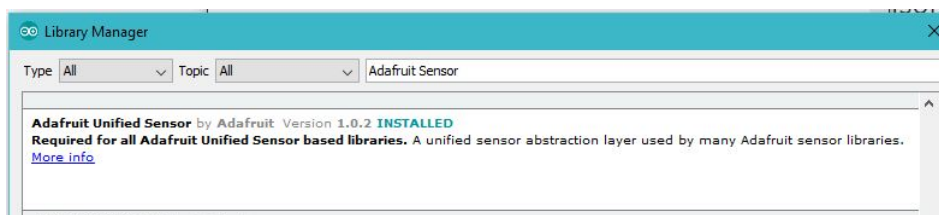
6.3.1 Read Sensor Data in Arduino

Open Arduino IDE in the Start menu and open the file `UartAM2302.ino` in the folder `ArduinoCode`. The sensor reading uses the library from Adafruit. To install the library, select **Sketch>Include Library>Manage Libraries** in the dropdown menu.

Enter "DHT" in the search field and look through the list for "DHT sensor library by Adafruit." Click the "Install" button, or "Update" from an earlier version.



You also need to install the `Adafruit_Sensor` library, which is also available in the Arduino Library Manager.



6.3.2 Communicate through UART


The second part is communication between the Arduino and Azure Sphere through UART. Open Visual Studio 2017 and select **File->Open->Project/Solution**. In the open dialogue, navigate to the Git repository we just downloaded and select the file "`AzureArduino.sln`" to open the Visual Studio Project.

This project is referenced to the **UART Sample for MT3620 RDB(Azure Sphere)**, you find it by select **File>New Project>Visual C++>Cross Platform>Azure Sphere**.

To use the UART read function, it needs to include the "`aplib/uart.h`" library. The function "`UartEventHandler`" reads the messages whenever they have been sent from the Arduino. Since we know the messages are the sensor data from AM2302 in an array format humidity,temperature. We can easily convert each message into the actual numbers and ready for uploading to the cloud. The function "`charArrayToNumbers`" helps to achieve that.



6.4 Test the Output

1. Connect Arduino to a PC and start the Arduino IDE. Then open the UartAM2302.ino.
2. Select **Tools>Port** then looking for the port with the description **Arduino/Genuino Due**.
3. Click Upload  in the tool bar. If you see the output similar to the following, it means the program has been successfully uploaded to the Arduino.

```
1 Sketch uses 5682 bytes (17%) of program storage space. Maximum is 32256 bytes.
2 Global variables use 191 bytes (9%) of dynamic memory, leaving 1857 bytes for local variables. Maximum
  is 2048 bytes.
```

4. Connect the Azure Sphere to a PC. Prepare the board to enable the debug from a PC.

```
1 azsphere device prep-debug
```

5. Open the **AzureArduino.sln** in Visual Studio 2017. Then select the **Remote GDB Debugger** in the tools bar to start the debug process.
6. You should be to see the output in the Output window like:

```
1 Remote debugging from host 192.168.35.1
2 UART application starting.
3 Opening MT3620_RDB_LED2_RED.
4 Humidity 18.70, Temperature 22.10
5 Humidity 18.70, Temperature 22.10
6 .....
```

7 Future Work