

 100 XP

Exercise - Deploying Azure AI services as Azure Functions to the IoT device

15 minutes

Creating Azure Cognitive Service

In the Azure portal, click Create a resource and search for Speech.

Create a Speech Service.

Fill out the form to create the Speech Service.


Name: Enter a name

Subscription: Select your available subscription

Location: Select your location

Pricing tier: Select S0

Resource group: Select your resource group

 Microsoft Azure

Search resources, services, and docs (G+)

Home > Create

Create

Speech

Name *

Subscription *


Location *

Pricing tier (View full pricing details) *


Resource group *


[Create new](#)


Go to the Speech service you just created, click Keys section to copy and note down the Key1. You'll use it for IoT DevKit to access.


 **mxchip-speech | Keys and Endpoint**
Cognitive Services


Search (Ctrl+)

 Overview


 Activity log


 Access control (IAM)


 Tags


 Diagnose and solve problems

RESOURCE MANAGEMENT

 Quick start


 **Keys and Endpoint**


 Pricing tier

 Networking


Regenerate Key1 Regenerate Key2

ENDPOINT




 These keys are used to access your Cognitive Service API. Do not share your keys. Store them securely– for example, using Azure Key Vault. We also recommend regenerating these keys regularly. Only one key is necessary to make an API call. When regenerating the first key, you can use the second key for continued access to the service.

KEY 1



KEY 2



Installing extensions for Visual Studio Code

You need to install some extensions for the Visual Studio Code.

- Launch Visual Studio Code, search for Arduino in the extension marketplace, and install it. This extension provides enhanced experiences for developing on the Arduino platform.
- Search for [Azure IoT Tools](#) in the extension marketplace and install it.

ⓘ Note

The Azure IoT Tools extension pack contains the [Azure IoT Device Workbench](#), which is used to develop and debug on various IoT devkit devices. The [Azure IoT Hub extension](#), also included with the Azure IoT Tools extension pack, is used to manage and interact with Azure IoT Hubs.

- Search for C# in the extension marketplace and install it.

Configure Visual Studio Code with Arduino settings

In Visual Studio Code, click File > Preference > Settings. Then click the ... and Open settings.json.

Add following lines to configure Arduino depending on your platform:

- **Windows:**

JSON

```
"arduino.path": "C:\\Program Files (x86)\\Arduino",  
"arduino.additionalUrls":  
"https://raw.githubusercontent.com/VSCChina/azureiotdevkit_tools/master/package_azureboard_index.json"
```

- **macOS:**

JSON

```
"arduino.path": "/Applications",  
"arduino.additionalUrls":  
"https://raw.githubusercontent.com/VSCChina/azureiotdevkit_tools/master/package_azureboard_index.json"
```

- **Ubuntu:**

Replace the {username} placeholder below with your username.

JSON

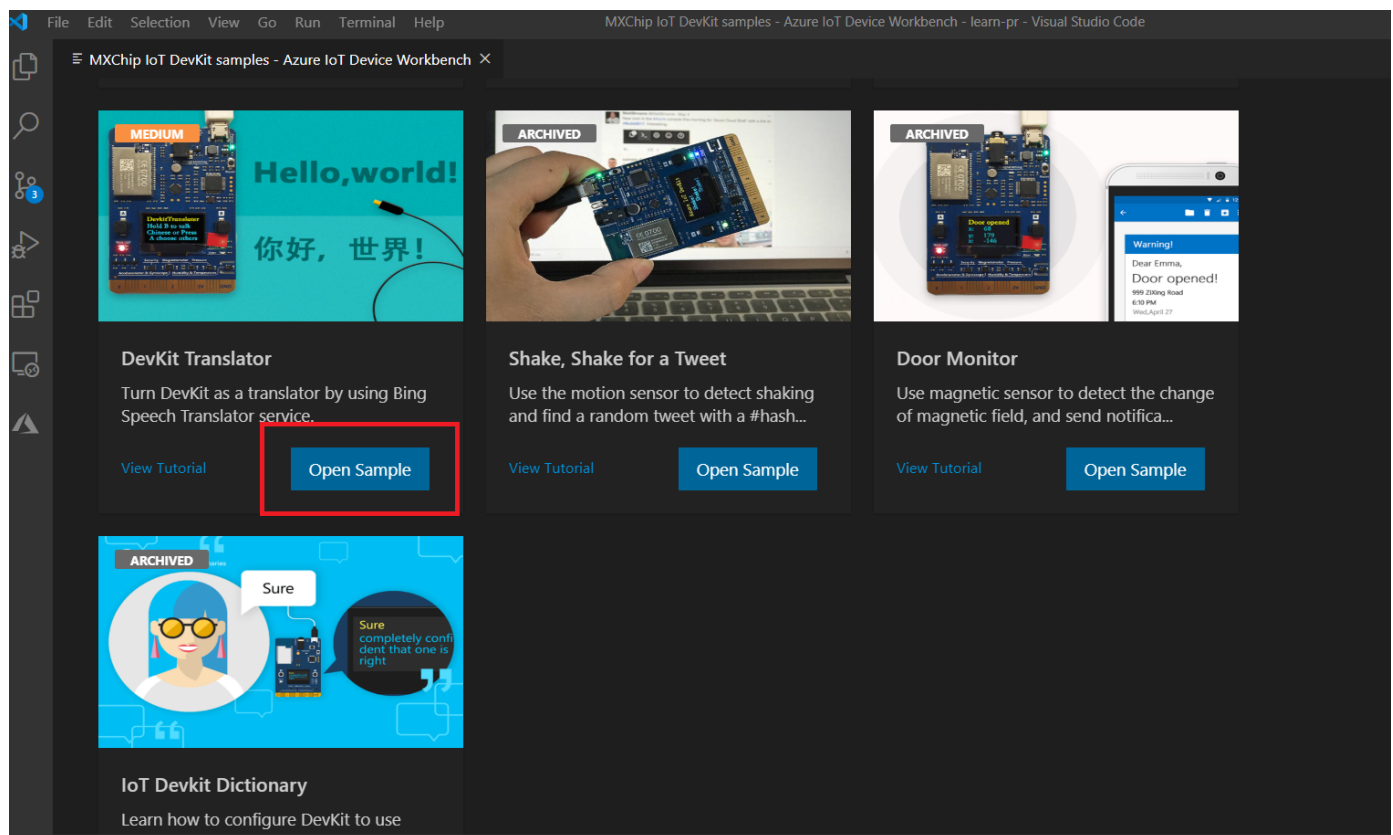
```
"arduino.path": "/home/{username}/Downloads/arduino-1.8.8",  
"arduino.additionalUrls":  
"https://raw.githubusercontent.com/VSCChina/azureiotdevkit_tools/master/package_a  
zur"
```

Opening sample project

First of all, start signing in your Azure account. Click F1 to open the command palette, type, and select Azure: Sign in

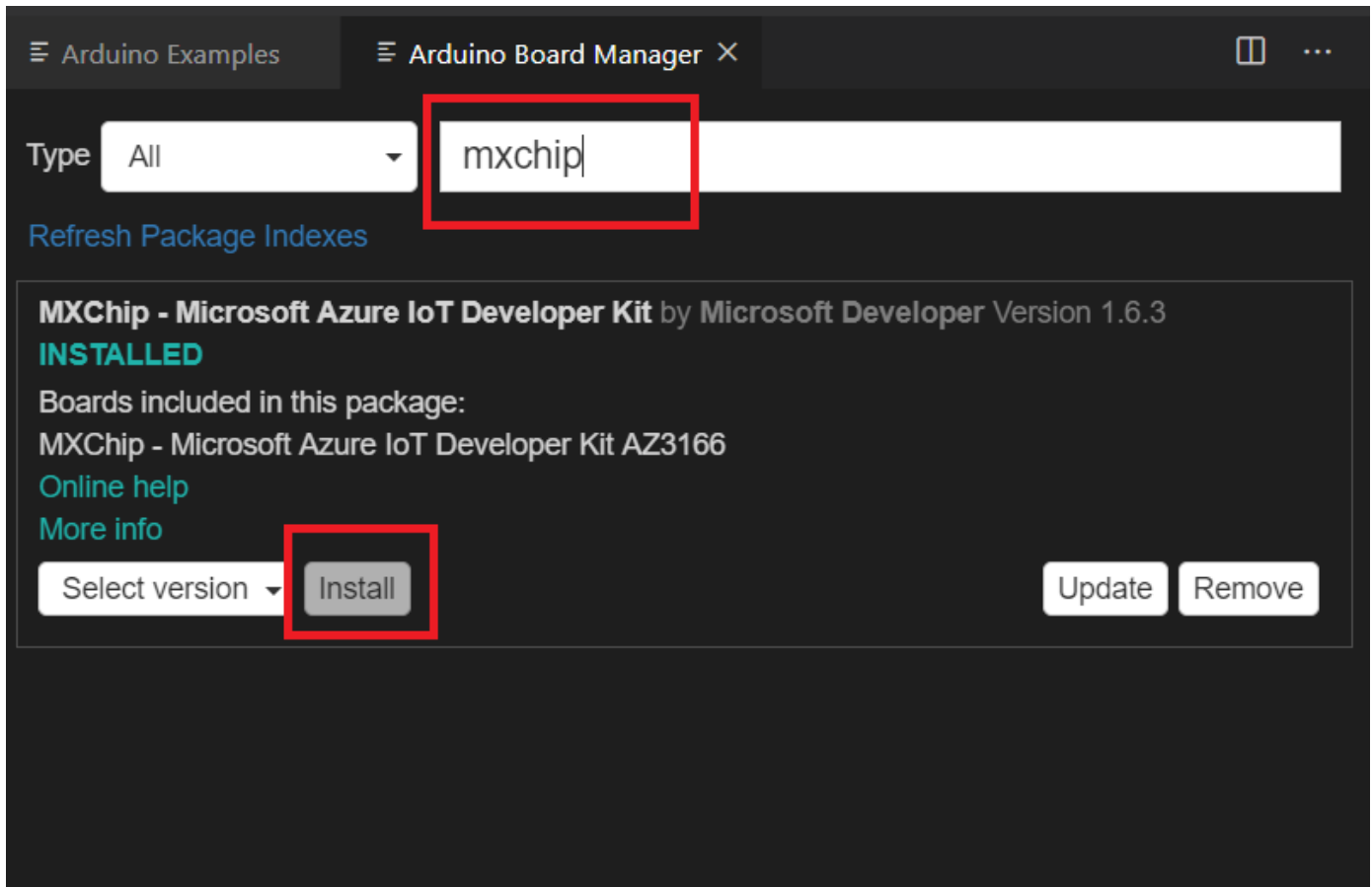
Click F1 to open the command palette, type, and select Azure IoT Device Workbench: Open Examples.... Then select IoT DevKit as the board.

In the IoT Workbench Examples page, find DevKit Translator and click Open Sample. Then select the default path to download the sample code.



Select the board

Click F1 to open the command palette, type, and select Arduino: Board Manager. Search for AZ3166 and install it.



Install Serial Port: ST-Link drivers

- **Windows:** Download and install the USB driver from [STMicroelectronics website](#).
- **macOS:** No driver is required for macOS.
- **Ubuntu:** Run the commands in terminal and sign out and sign in for the group change to take effect:

Bash

```
# Copy the default rules. This grants permission to the group 'plugdev'
sudo cp ~/.arduino15/packages/AZ3166/tools/openocd/0.10.0/linux/contrib/60-
openocd.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules

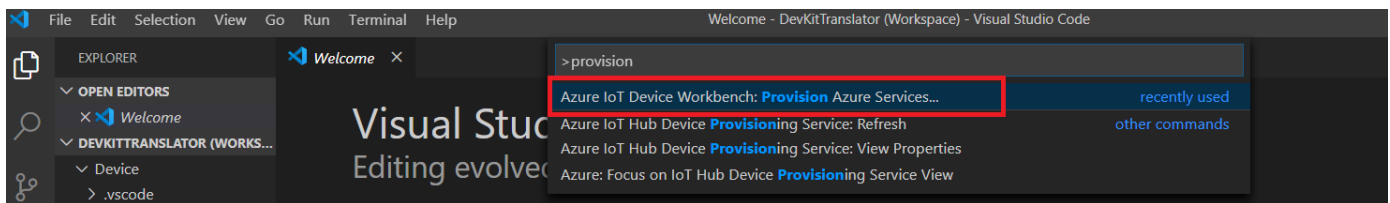
# Add yourself to the group 'plugdev'
# Logout and log back in for the group to take effect
sudo usermod -a -G plugdev $(whoami)
```

ⓘ Note

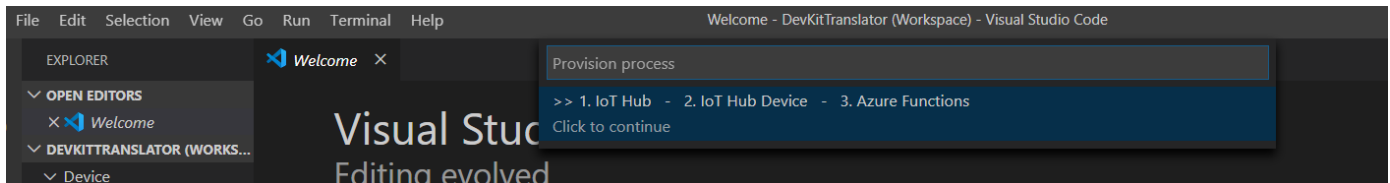
ST-Link/V2 is the USB interface that IoT DevKit uses to communicate with your development machine. You need to install it on Windows to flash the compiled device code to the DevKit. Follow the OS-specific steps to allow the machine access to your device.

Create Azure Function/Provision Azure Services

In Visual Studio Code, click F1, type, and select Azure IoT Device Workbench: Provision Azure Services....

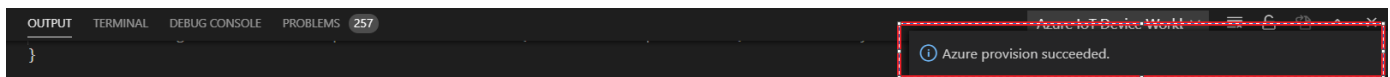


Follow the steps to finish the provisioning of Azure IoT Hub and Azure Functions.



Take a note of the Azure IoT Hub device name you created.

Now you have Azure IoT Hub provisioned and device created in it. Also, the device connection string will be saved in Visual Studio Code for configuring the IoT DevKit later.



Open `Functions\DevKitTranslatorFunction.cs` and update the following lines of code with the device name and Speech Service key you noted down.

```
C#

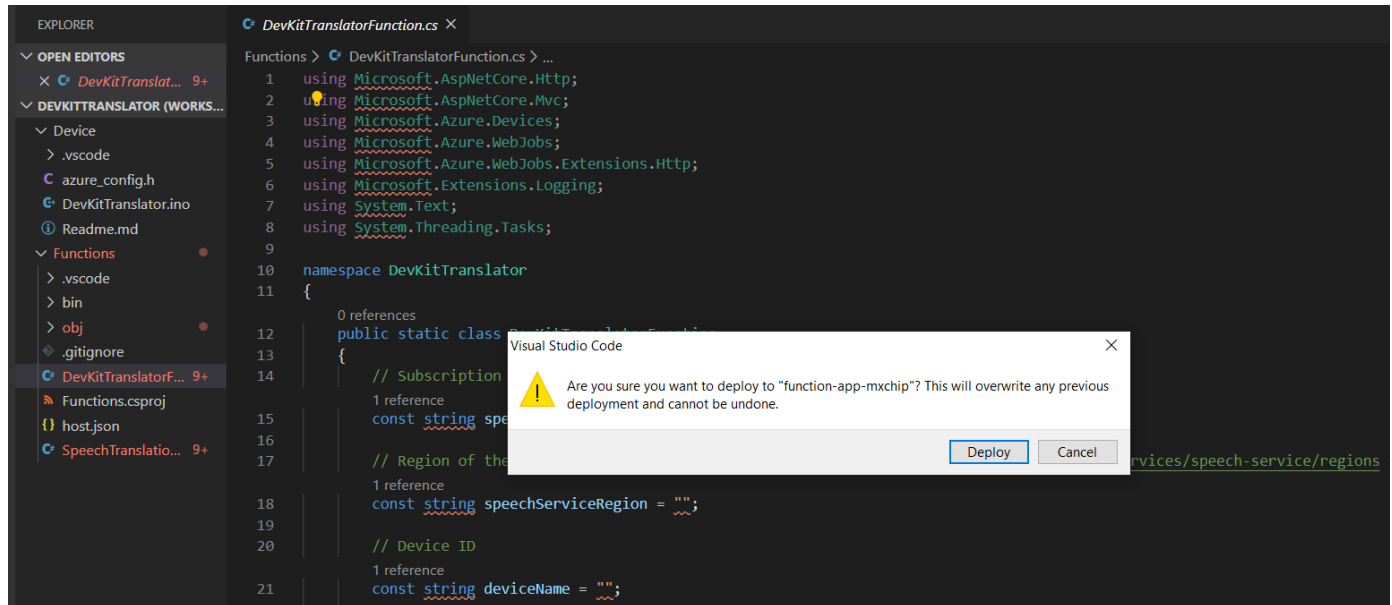
// Subscription Key of Speech Service
const string speechSubscriptionKey = "";

// Region of the speech service, see https://learn.microsoft.com/azure/cognitive-services/speech-service/regions for more details.
```

```
const string speechServiceRegion = "";  
  
// Device ID  
const string deviceName = "";
```

Deploying Azure Function

Click F1, type, and select Azure IoT Device Workbench: Deploy to Azure.... If Visual Studio Code asks for confirmation for redeployment, click Yes.



Make sure the deployment is successful.



In the Azure portal, go to the Functions Apps section, find the Azure Function app created. Click `devkit_translator`, then click **Get Function URL** to copy the URL.

function-app-mxchip - devkit_translator
Function Apps

Pay-As-You-Go

Function Apps

function-app-mxchip

Functions (Read Only)

devkit_translator

Integrate

Manage

Monitor

Proxies (Read Only)

Slots

Your app is currently in read only mode because you have published a generated function.json. Changes made to function.json will not be saved.

function.json

Save

Run

</> Get function URL

```

1 {
2   "generatedBy": "Microsoft.NET.Sdk.Functions-1.0.36",
3   "configurationSource": "attributes",
4   "bindings": [
5     {
6       "type": "httpTrigger",
7       "methods": [
8         "get",
9         "post"
10      ],
11       "authLevel": "function",
12       "name": "req"
13     }
14   ],
15   "disabled": false,
16   "scriptFile": "../bin/Functions.dll",
17   "entryPoint": "DevKitTranslator.DevKitTranslatorFunction.Run"
18 }

```

Paste the URL into `azure_config.h` file.

EXPLORER

OPEN EDITORS 1 UNSAVED

azure_config.h Device

DEVKIT...

Device

.vscode

azure_config.h

Device > C azure_config.h > AZURE_FUNCTION_URL

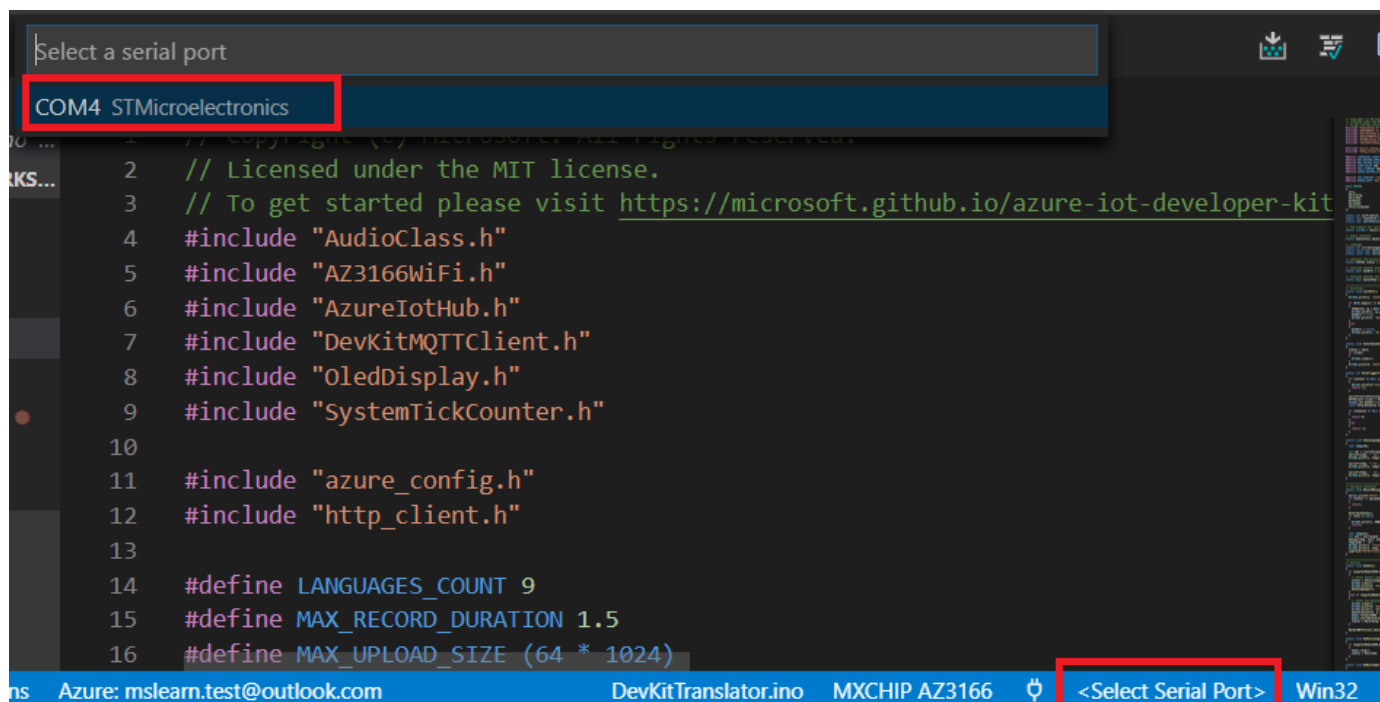
```

1 #define AZURE_FUNCTION_URL "https://function-app-mxchip.azurewebsites.net/api/devkit_translator?code=uaZPdT6Ga/c

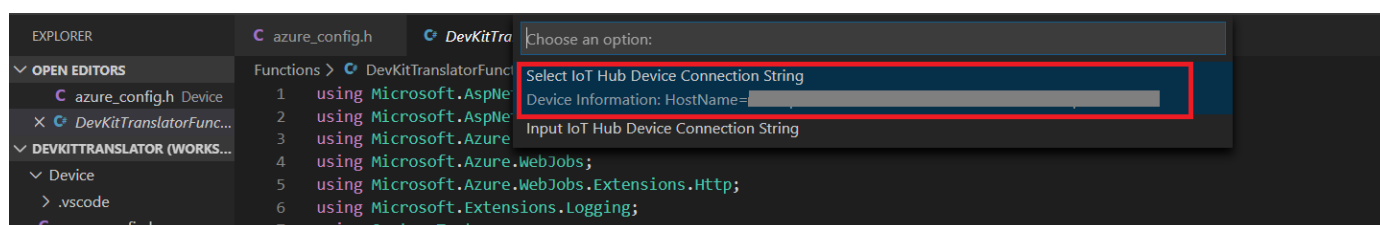
```

Configuring Device Settings

In the bottom-right status bar, check the MXCHIP AZ3166 is shown as a selected board, and serial port with STMicroelectronics is used.

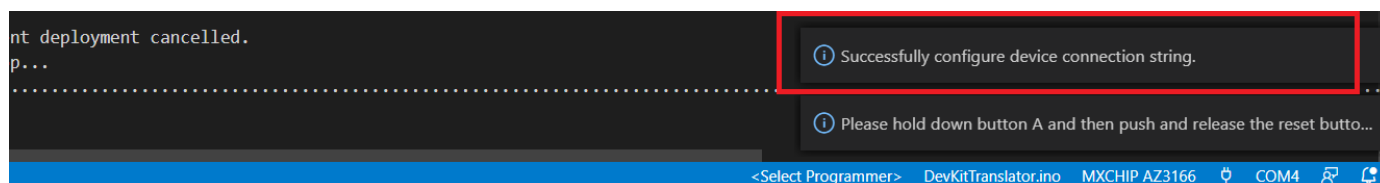


Click F1, type, and select Azure IoT Device Workbench: Configure Device Settings... > Config Device Connection String. Select IoT Hub Device Connection String to configure it to the DevKit.



On DevKit, hold down button A, push and release the reset button, and then release button A. Your DevKit enters configuration mode and saves the connection string.

You'll see the notification once it's done successfully.



Uploading device code

Click F1 again, type, and select Azure IoT Device Workbench: Upload Device Code. It starts to compile and upload the code to DevKit.

```

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS 7
TRACE StatusLogger XmlConfiguration stopping 1 LoggerConfigs.
TRACE StatusLogger XmlConfiguration stopping root LoggerConfig.
TRACE StatusLogger XmlConfiguration notifying ReliabilityStrategies that appenders will be stopped.
TRACE StatusLogger XmlConfiguration stopping remaining Appenders.
DEBUG StatusLogger Shutting down RollingFileManager C:\Users\ajitj\AppData\Local\Arduino15\logs\application.log
DEBUG StatusLogger Shutting down RollingFileManager C:\Users\ajitj\AppData\Local\Arduino15\logs\application.log
DEBUG StatusLogger All asynchronous threads have terminated
DEBUG StatusLogger RollingFileManager shutdown completed with status true
DEBUG StatusLogger Shut down RollingFileManager C:\Users\ajitj\AppData\Local\Arduino15\logs\application.log, all resources released: true
DEBUG StatusLogger Appender RollingFile stopped with status true
DEBUG StatusLogger Shutting down OutputStreamManager SYSTEM_ERR.false.false
DEBUG StatusLogger Shut down OutputStreamManager SYSTEM_ERR.false.false, all resources released: true
DEBUG StatusLogger Appender Console stopped with status true
TRACE StatusLogger XmlConfiguration stopped 2 remaining Appenders.
TRACE StatusLogger XmlConfiguration cleaning Appenders from 2 LoggerConfigs.
DEBUG StatusLogger Stopped XmlConfiguration[location=jar:file:/C:/Program%20Files%20(x86)/Arduino/lib/pde.jar!/log4j2.xml] OK
DEBUG StatusLogger Stopped LoggerContext[name=af3868, org.apache.logging.log4j.core.LoggerContext@5fddc] with status true
[Done] Uploaded the sketch: DevKitTranslator.ino

Azure: mslearn.test@outlook.com  <Select Programmer>  DevKitTranslator.ino  MXCHIP AZ3166  COM4

```

The DevKit reboots and starts running the code.

Test the project

After app initialization, follow the instructions on the DevKit screen. The default source language is Chinese.

To select another language for translation:

1. Press button A to enter setup mode.
2. Press button B to scroll all supported source languages.
3. Press button A to confirm your choice of the source language.
4. Press and hold button B while speaking, then release button B to initiate the translation.
5. The translated text in English shows on the screen.

On the translation result screen, you can:

1. Press buttons A and B to scroll and select the source language.
2. Press the B button to talk. To send the voice and get the translation text, release the B button.

Next unit: Knowledge check

Continue >