

Provision and Deploy MXCHIP

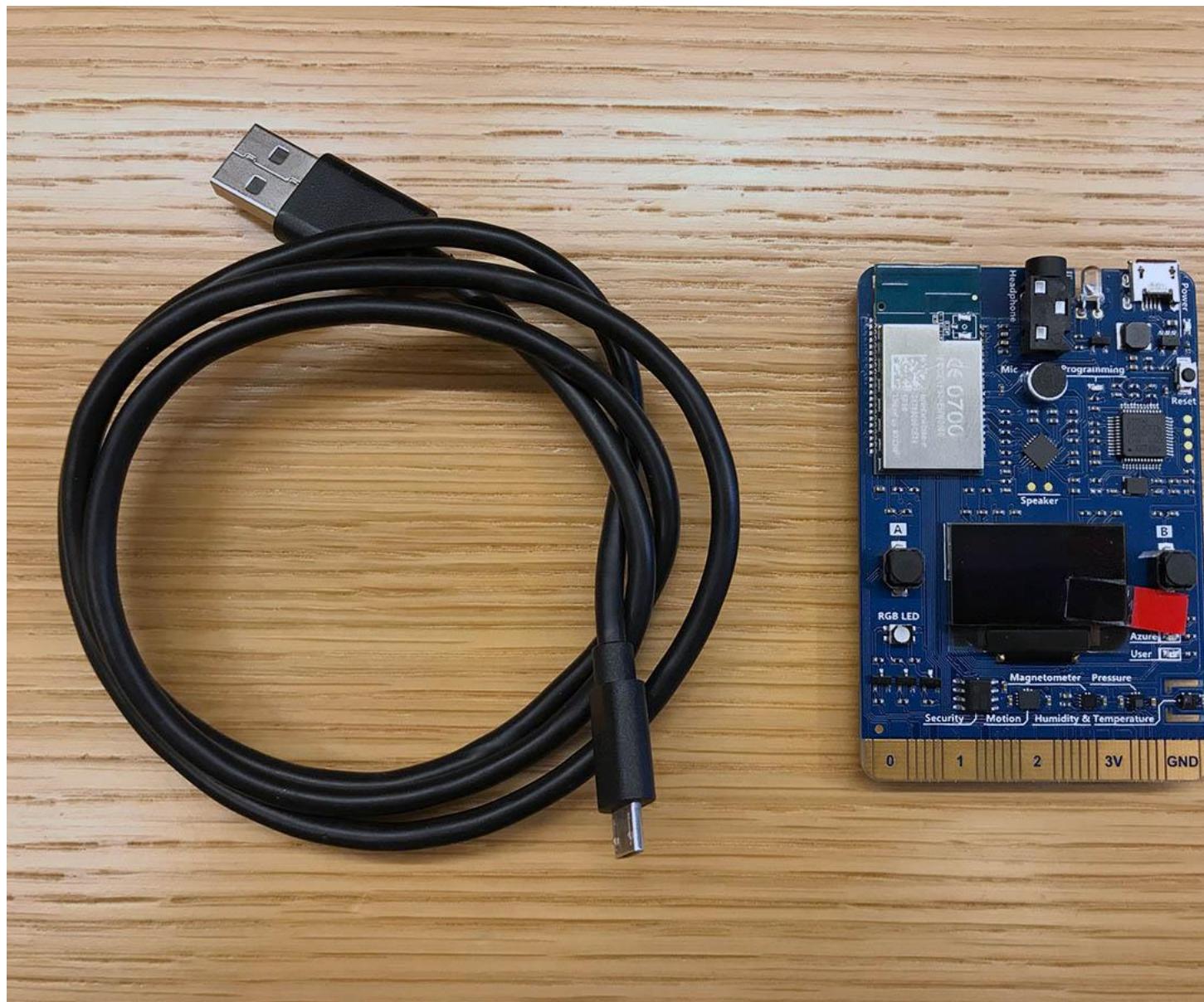
Exercise 5



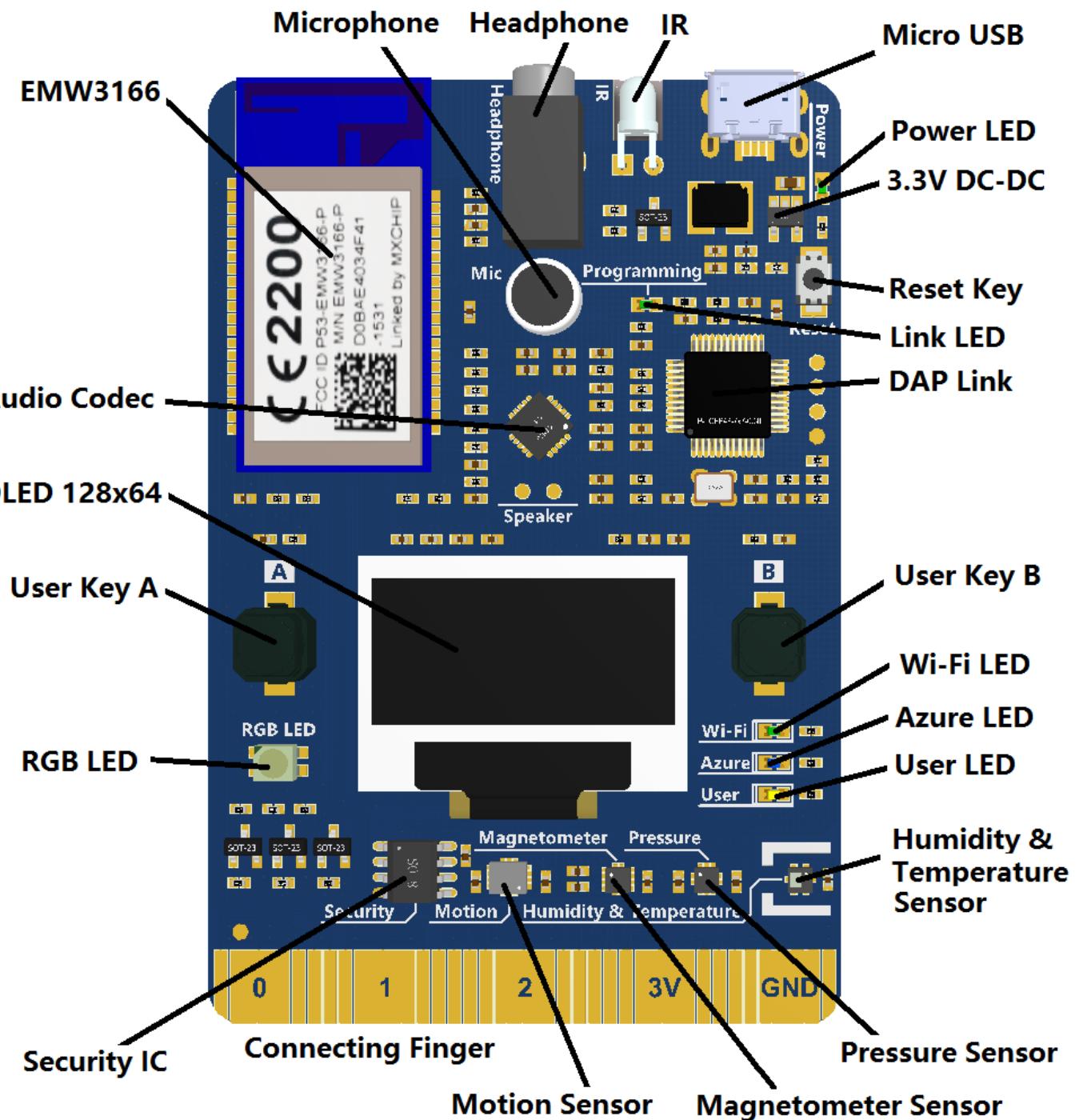
IoT Device MXCHIP

MXChip is an IoT Devkit board with Wifi, OLED display, headphone, microphone, sensors like temperature, humidity, motion, pressure, you.

A Micro-USB Cable is included in the box.

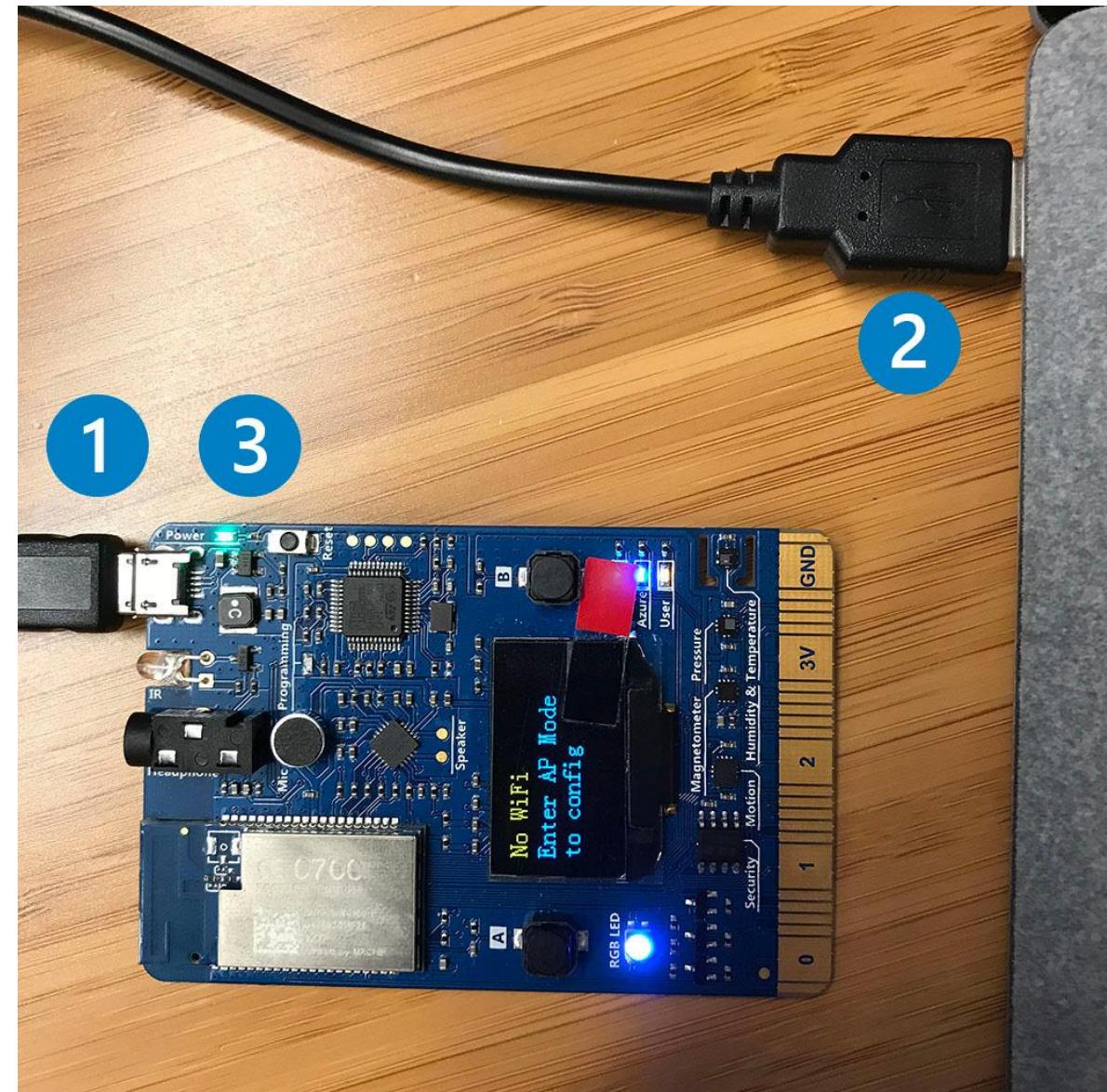


MXCHIP components



Connect MXCHIP to Computer

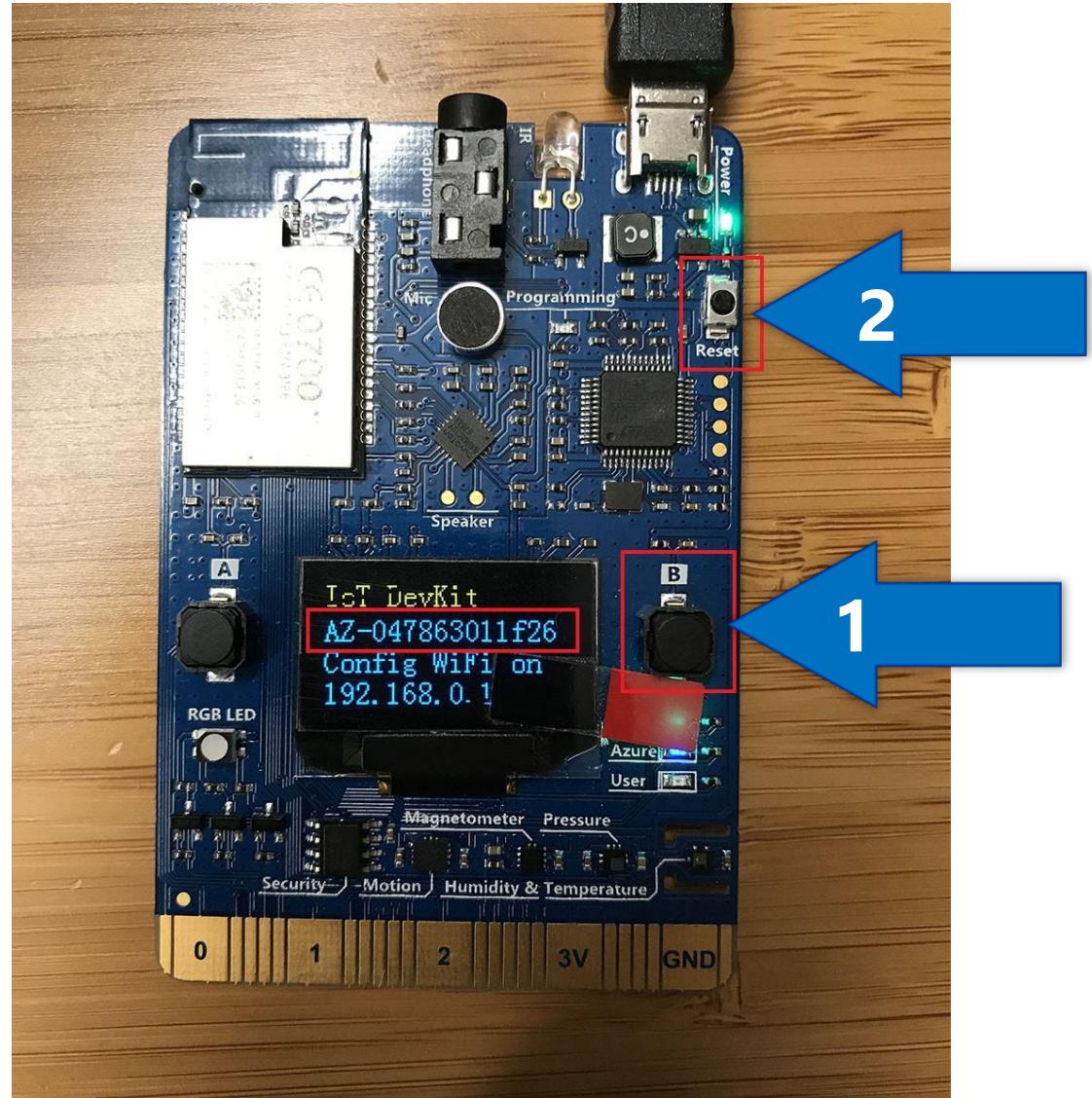
- `1` Connect the USB end to your computer.
- `2` Connect the Micro-USB end to the DevKit.
- `3` The green LED for power confirms the connection.



Enter AP Mode for Wifi

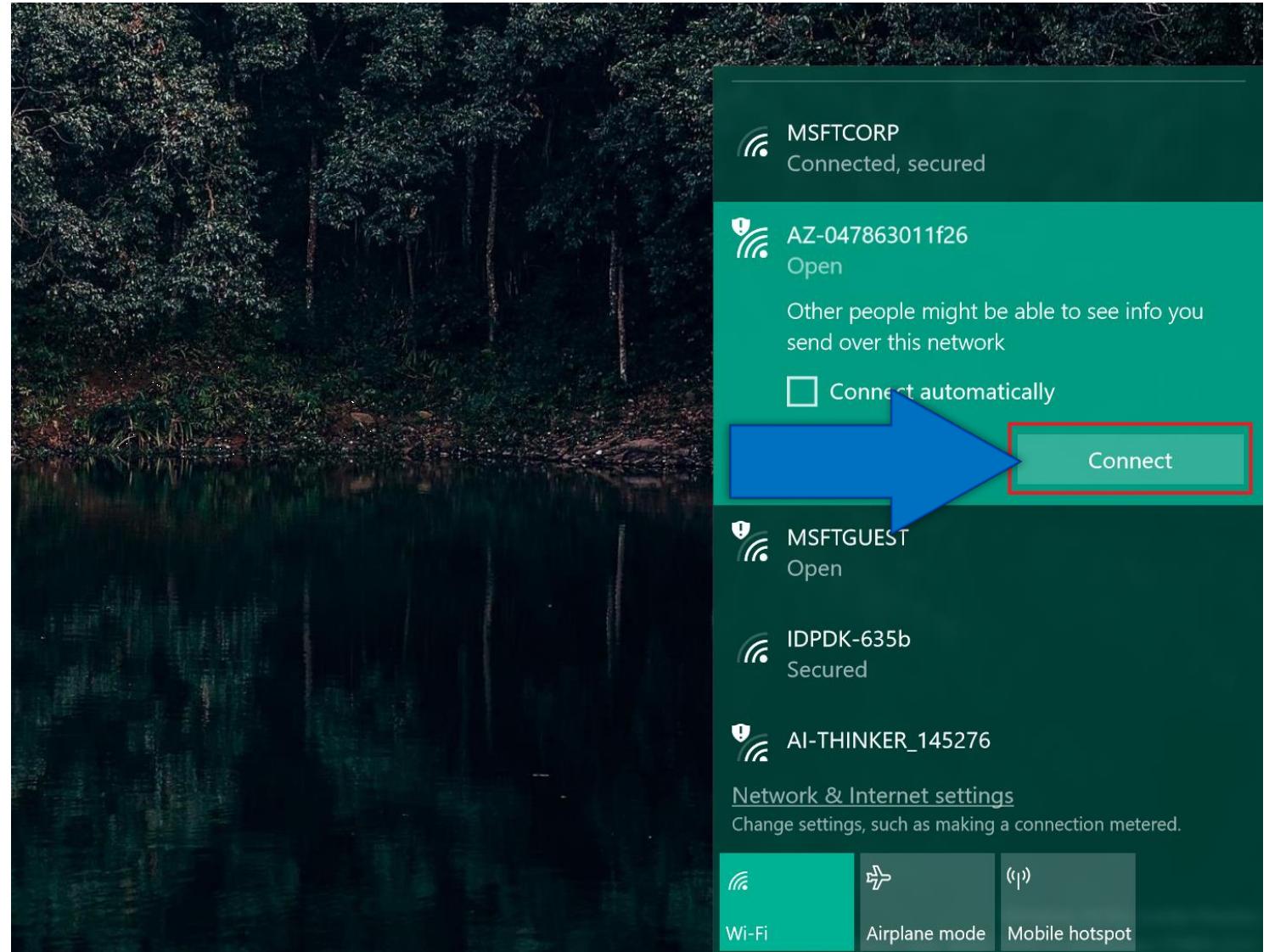
- `1` Hold down button B
- `2` Push and release the reset button, and then release button B. Your MXCHIP enters AP mode for configuring Wi-Fi.

The screen displays the service set identifier (SSID) of the MXCHIP and the configuration portal IP address.



MXCHIP Wifi Network

On your computer to connect to the MXCHIP SSID find the AZ-some number Wifi name for your connected board and click Connect

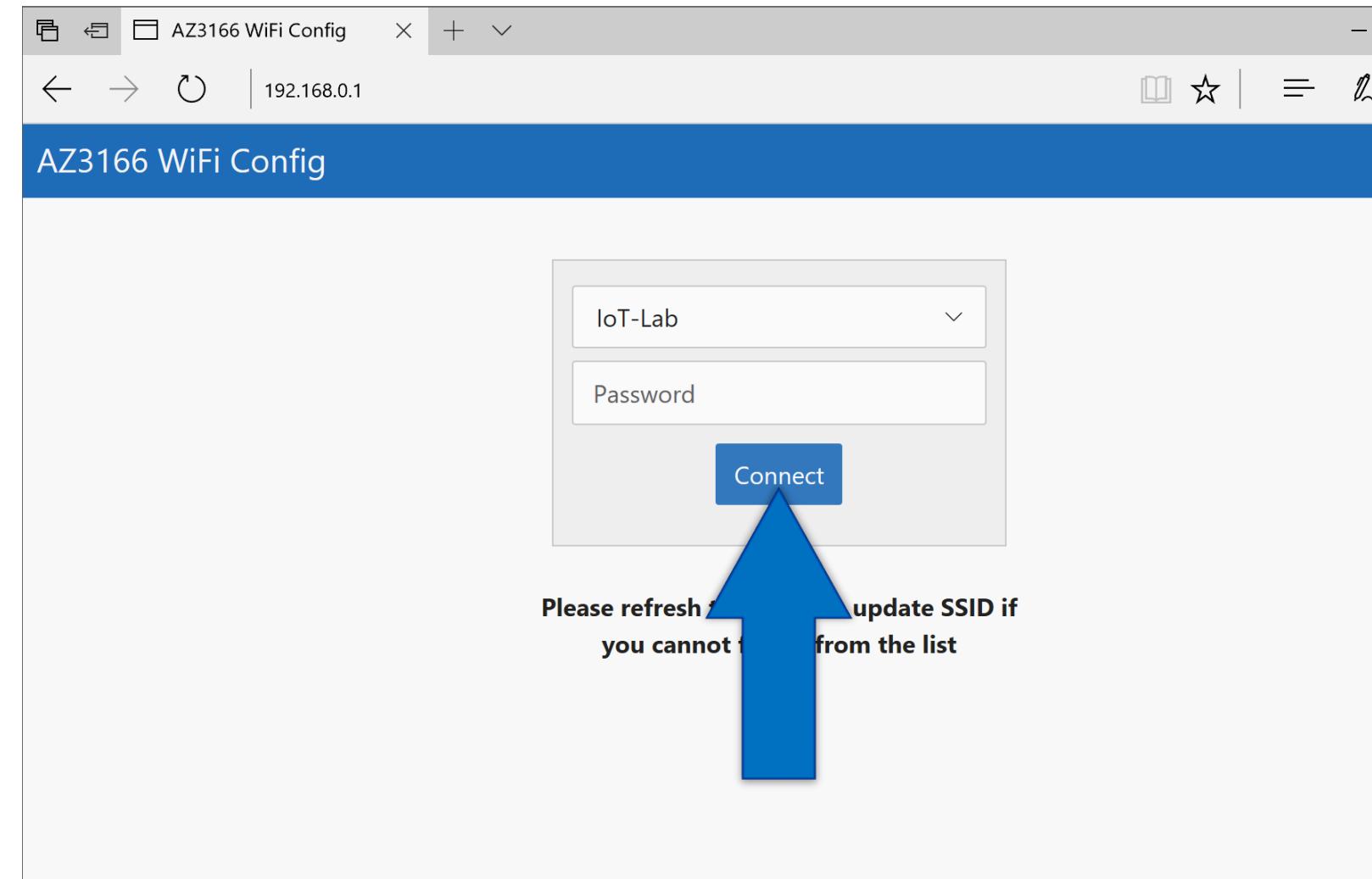


Wifi Config

On your Computer open the IP address shown on the MXCHIP screen on your computer, select the Wi-Fi network that you want the MXCHIP to connect to, in my case I'm using IoT-Lab.

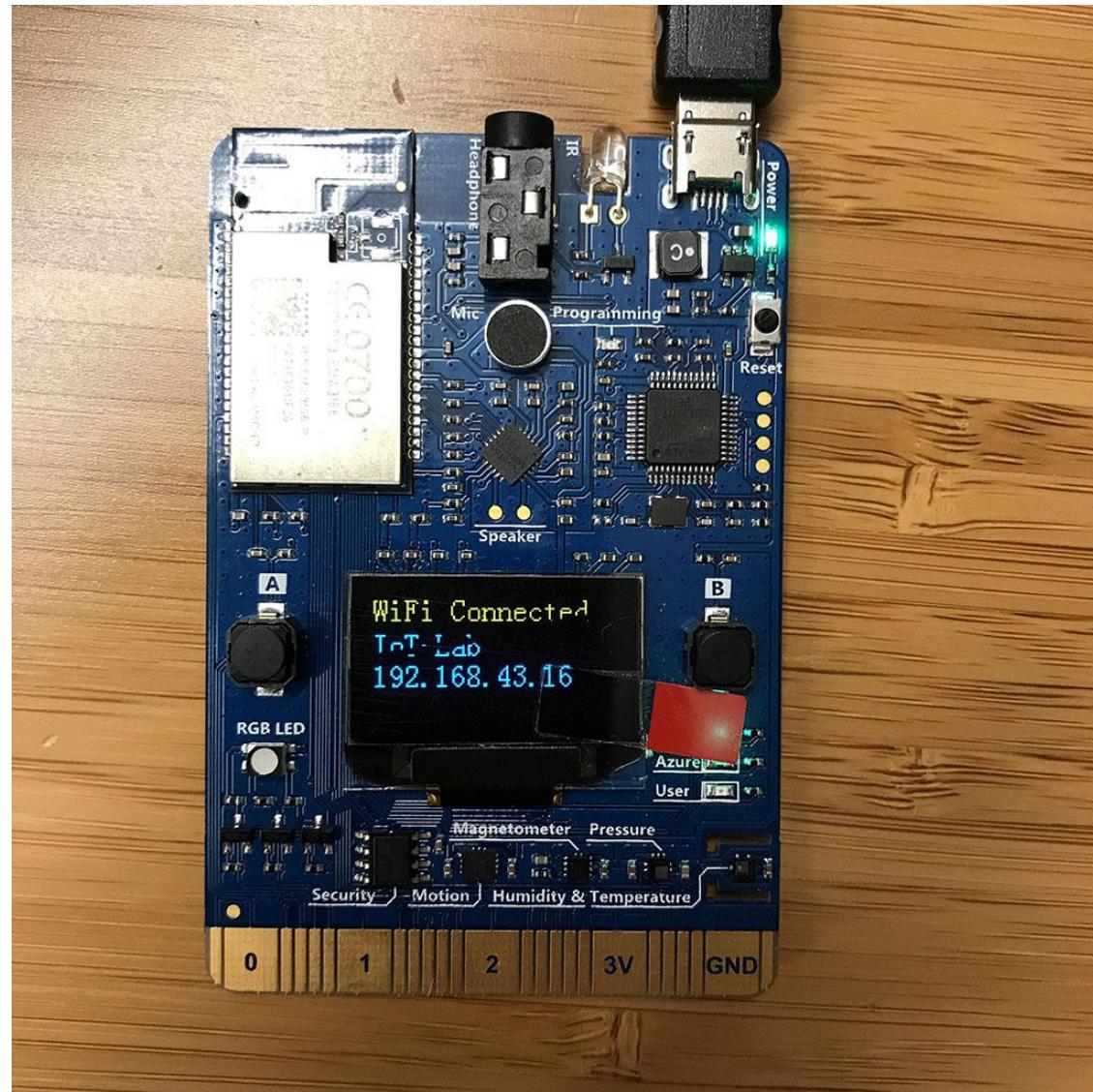
Then type the Wifi password.

Click Connect.



MXCHIP Wifi Connected

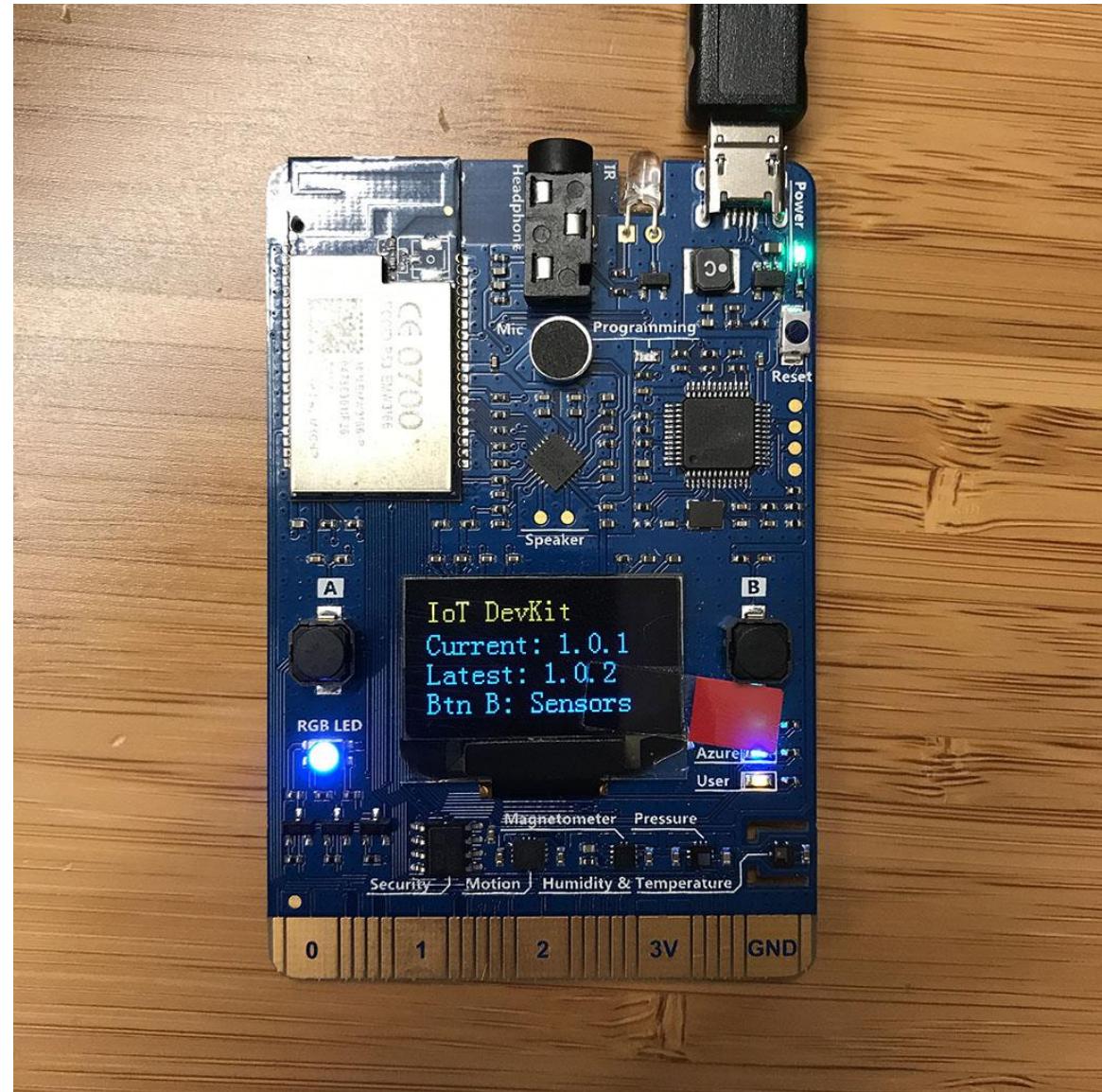
When the connection succeeds, the MXCHIP reboots in a few seconds. You then see the Wi-Fi name and IP address on the screen.



Update firmware

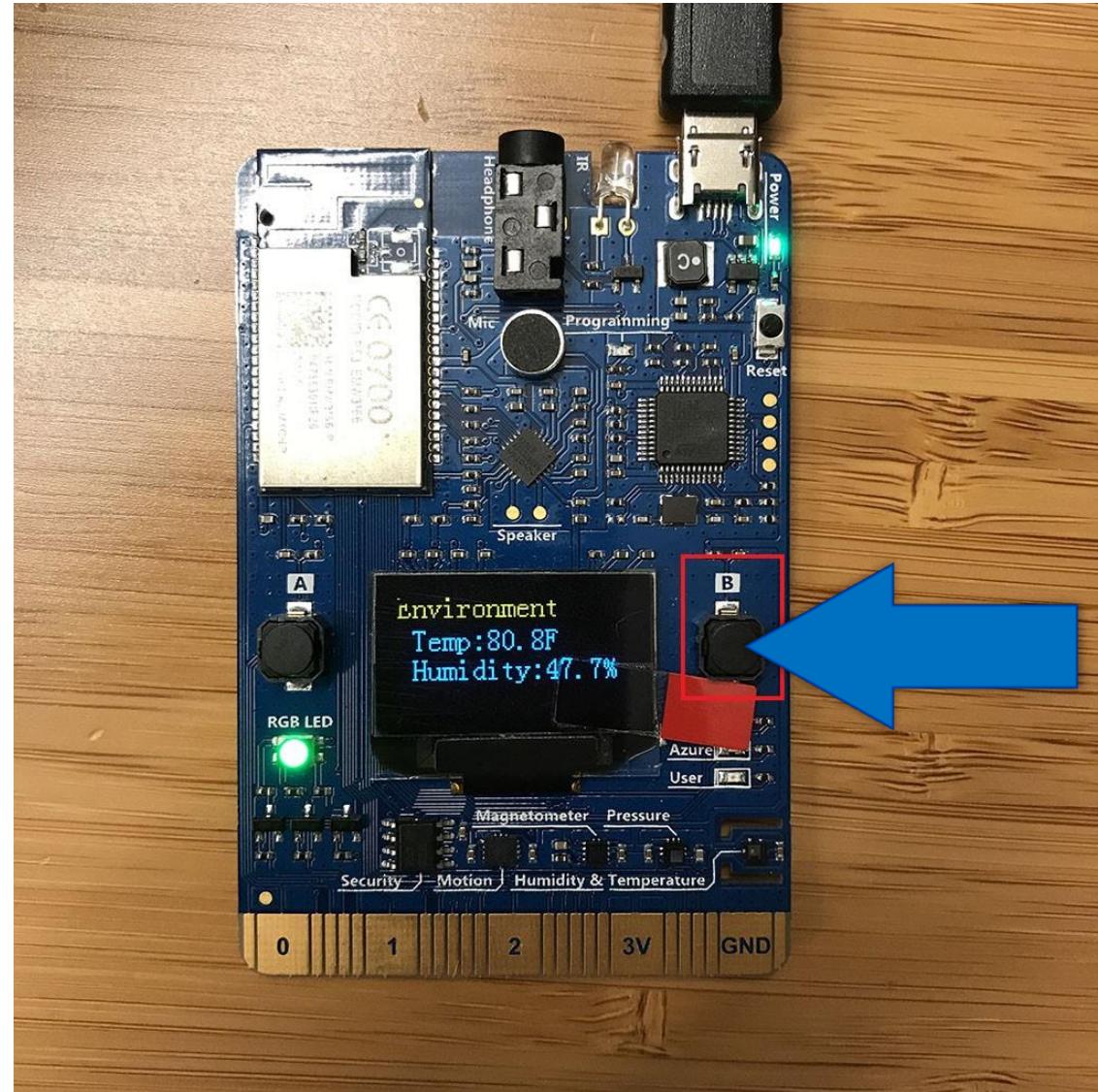
If you need a firmware upgrade, the screen will show the current and latest firmware versions as well as the version difference.

To upgrade, follow the [Upgrade firmware](#) guide.



Check out Sensors

Press button B to test sensors. Continue pressing and releasing the button B to cycle through each sensor.



Dev environment Windows

DOWNLOAD

Download the latest dev environment package.

The .zip file that you download contains all the necessary tools and packages for MXCHIP development.

Docs continue greater detail if you're curious

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-arduino-iot-devkit-az3166-get-started>

The .zip file contains the following tools and packages. If you already have some components installed, the script will detect and skip them.

- [Azure CLI 2.0 MSI](#): Cross-platform command-line experience for managing Azure resources. The MSI contains dependent Python and pip.
- [Visual Studio Code \(VS Code\)](#): Lightweight code editor for MXCHIP development.
- [Visual Studio Code extension for Arduino](#): Extension that enables Arduino development in Visual Studio Code.
- [Arduino IDE](#): Tool that the extension for Arduino relies on.
- DevKit Board Package: Tool chains, libraries, and projects for the MXCHIP.
- ST-Link Utility: Essential tools and drivers.

You may need to Uninstall VS Code before continuing the next step.

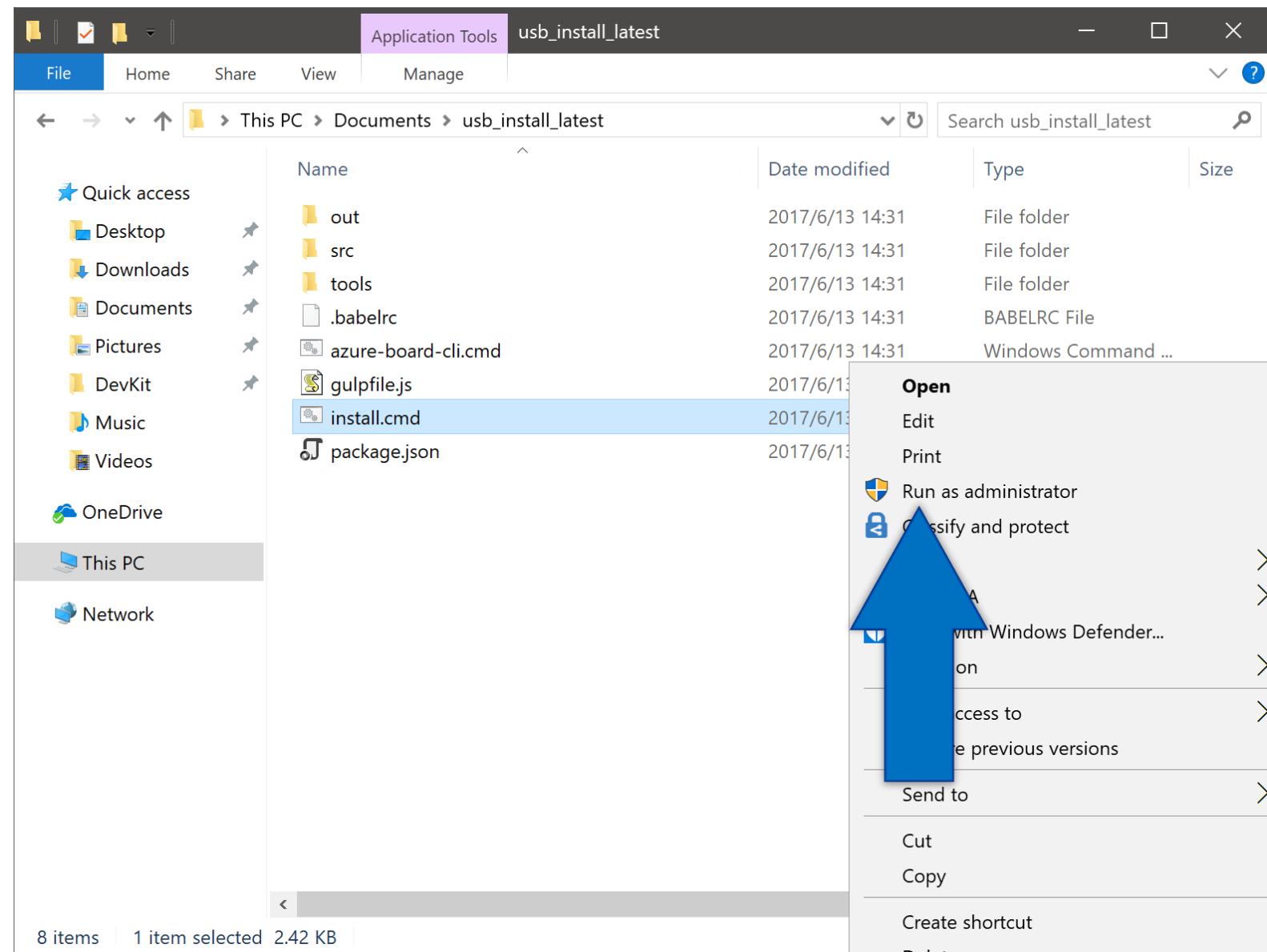


Install script

In Windows File Explorer, locate the .zip file you downloaded for the Windows dev environment and extract it.

Find install.cmd, right-click it, and select Run as administrator.

If you run into any issues during installation this could be due to some of the components already being installed on the system. Uninstalling should fix this and you can Find install.cmd, right-click it, and select Run as administrator again.



Packages installing

During installation, you see the progress of each tool and package.

Depending on your environment, sometimes you will get failure when installing Arduino IDE. In this case, you may try [install Arduino IDE individually](#) and run install.cmd again.

Otherwise, please follow the [manual steps](#) to install all necessary tools and packages.

```
C:\WINDOWS\System32\cmd.exe
Copying files...
54 File(s) copied
3 File(s) copied
Installing node packages...
- checking pre-conditions of task: Install Azure CLI 2.0
| V Install Azure CLI 2.0: installed
- checking pre-conditions of task: Install VS Code
| V Install VS Code: installed
- checking pre-conditions of task: Install Arduino Extension
-
```

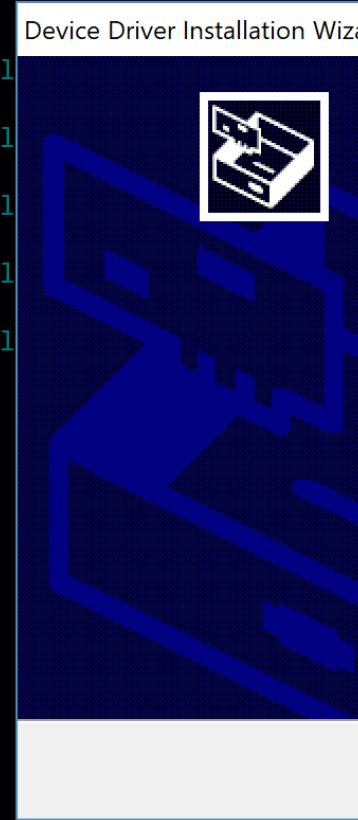


Installing drivers

The VS Code for Arduino extension relies on the Arduino IDE. If this is the first time you are installing the Arduino IDE, you're prompted to install relevant drivers:

After the installation is complete, you should see Visual Studio Code and Arduino IDE shortcuts on your desktop.

Occasionally, when you start VS Code, you're prompted with an error that it cannot find the Arduino IDE or related board package. To solve it, close VS Code and restart the Arduino IDE. VS Code should then locate the Arduino IDE path correctly.



C:\WINDOWS\System32\cmd.exe

```
Copying files...
54 File(s) copied
3 File(s) copied
Installing node packages...
- checking pre-conditions of task: Install
| V Install Azure CLI 2.0: installed
- checking pre-conditions of task: Install
| V Install VS Code: installed
- checking pre-conditions of task: Install
/ V Install Arduino Extension: installed
\ V Install Arduino: installed
- checking pre-conditions of task: Install
-
```

Device Driver Installation Wizard

Welcome to the Device Driver Installation Wizard!

This wizard helps you install the software drivers that computers devices need in order to work.

To continue, click Next.

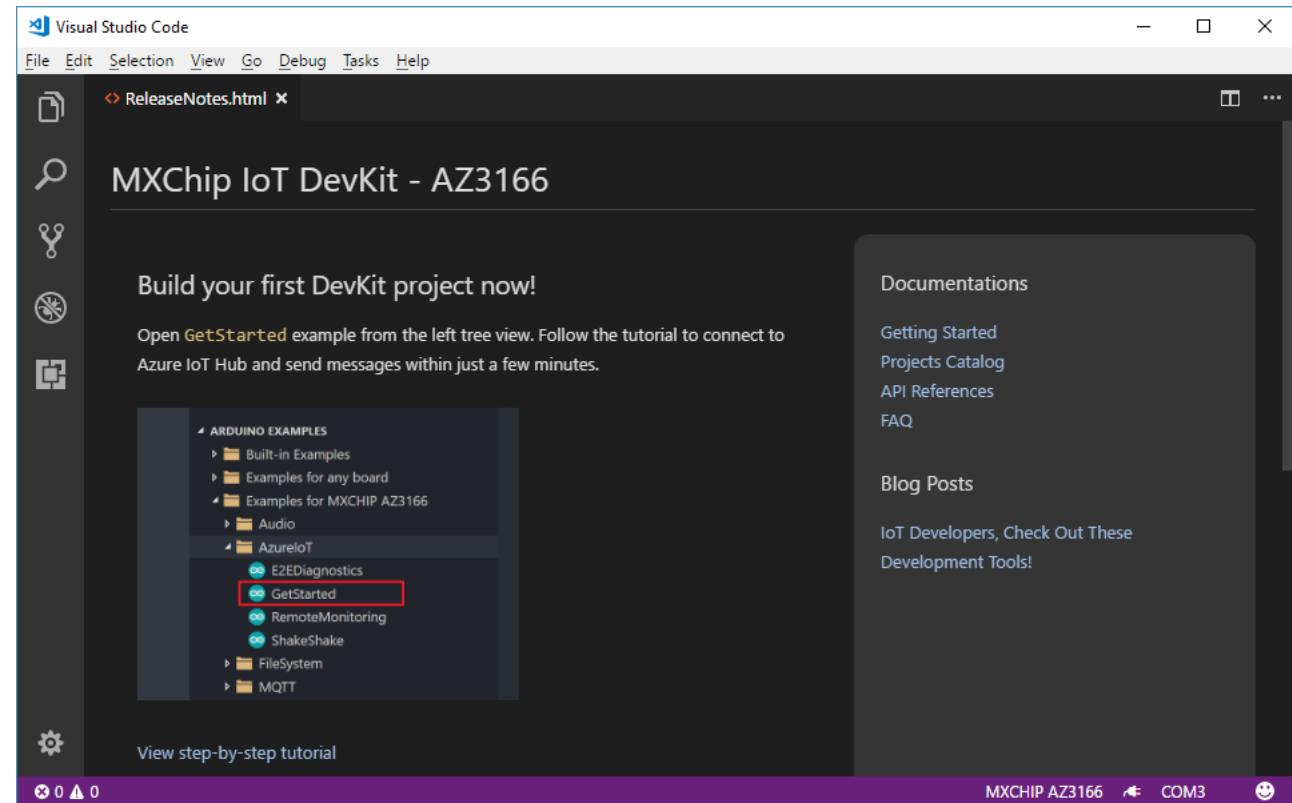
< Back Next >

Open VS Code

Make sure your MXCHIP is not connected to your computer.

Start VS Code first and connect the MXCHIP to your computer. VS Code automatically finds the MXCHIP and opens an introduction page pictured here.

Occasionally, when you start VS Code, you're prompted with an error that it cannot find the Arduino IDE or related board package. Close VS Code and restart the Arduino IDE. VS Code should then locate the Arduino IDE path correctly.



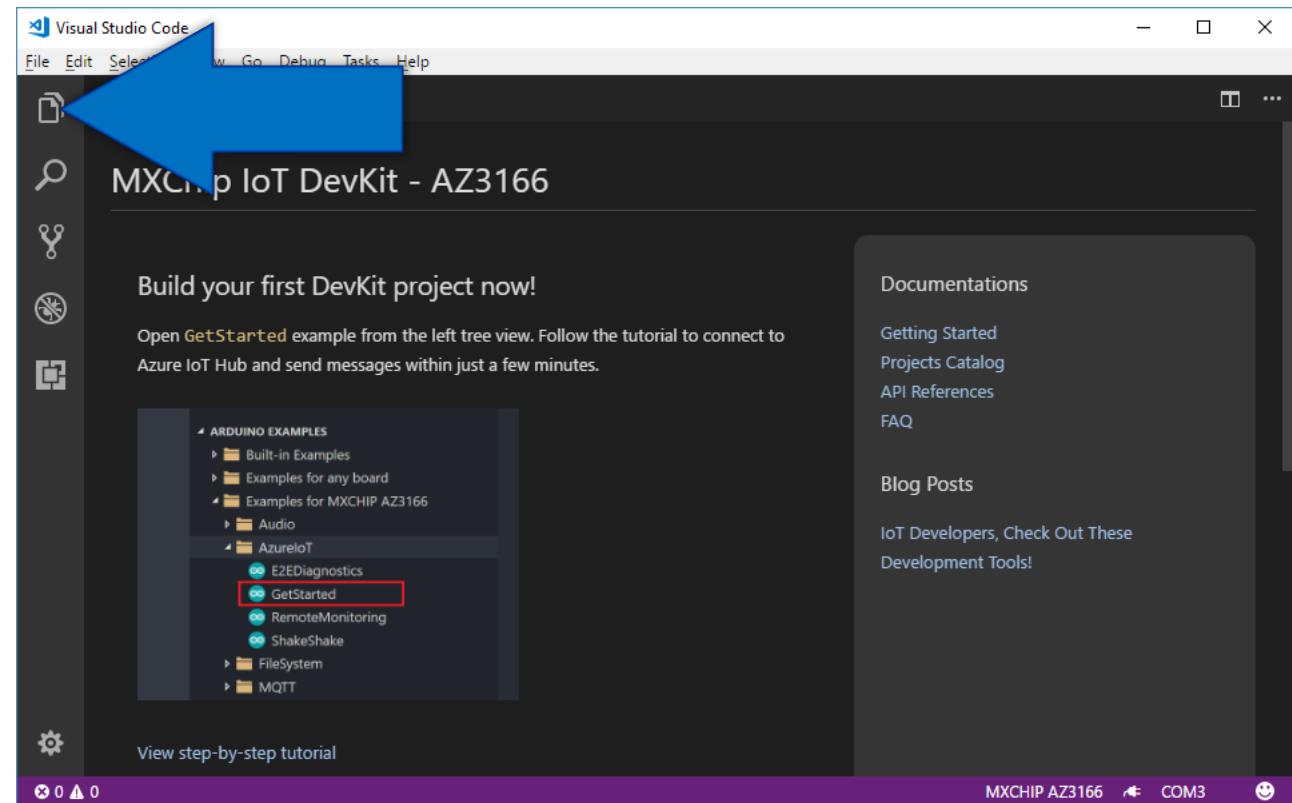
Arduino examples

Click the File Explorer icon to open the files pane.

On the Arduino Examples tab, browse to Examples for MXCHIP AZ3166 > AzureIoT, and select **GetStarted.ino**.

If you happen to close the pane, you can reopen it.

Use Ctrl+Shift+P (macOS: Cmd+Shift+P) to open the command palette, type Arduino, and then find and select Arduino: Examples.

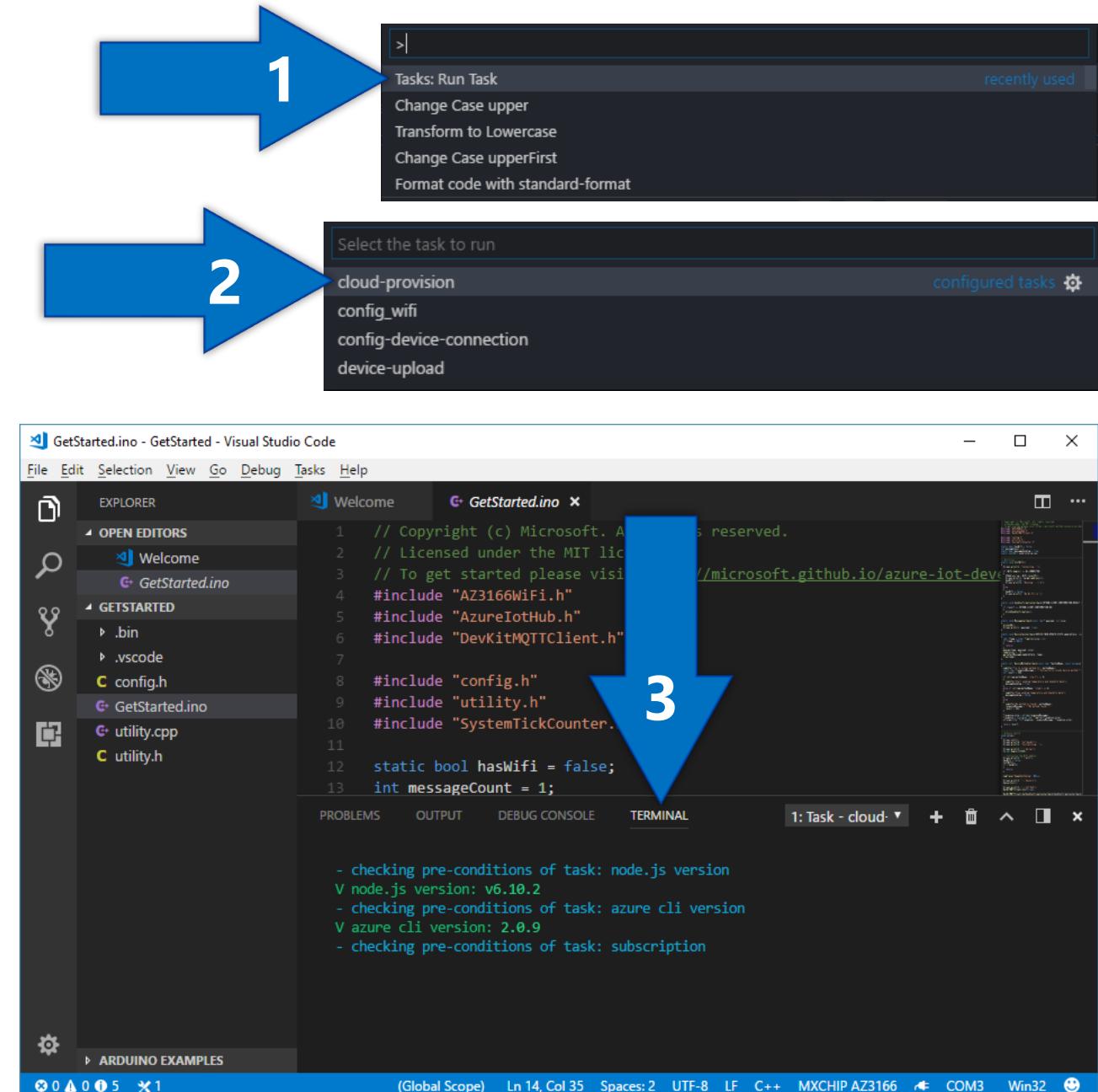


Cloud provision MXCHIP in IoT Hub

Let's start provisioning your MXCHIP IoT device to communicate with IoT Hub:

- `1` Run your task through Ctrl+Shift+P (macOS: Cmd+P) and select Tasks:Run Task.
- `2` Select the cloud-provision task.
- `3` Your browser should then open and asks for you to put in the code given in the VS Code terminal to log into your Azure Account

In the VS Code terminal, an interactive command line guides you through provisioning the required Azure services.



Azure Subscription

Make sure the subscription ID matches the ID shown in your Azure Portal.

Use your arrow keys, select the subscription and click Enter/Return.



```
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
2: Task - cloud-p

> Executing task: node "/Users/$USER/azure-board-cli/out/cli.js" provision .bin <

- Checking pre-conditions of task: check Node.js version
V check Node.js version: v9.11.1
- Checking pre-conditions of task: check Azure CLI version
V check Azure CLI version: 2.0.29
- Checking pre-conditions of task: get azure subscription
? What subscription would you like to choose?
Select Subscription
> Super Saiyan Goku (
```

IoT Hub

Now select the IoT Hub to provision the MXHCIP to and click Enter/Return. In my case I choose `iot-hub-connected-field-service`.

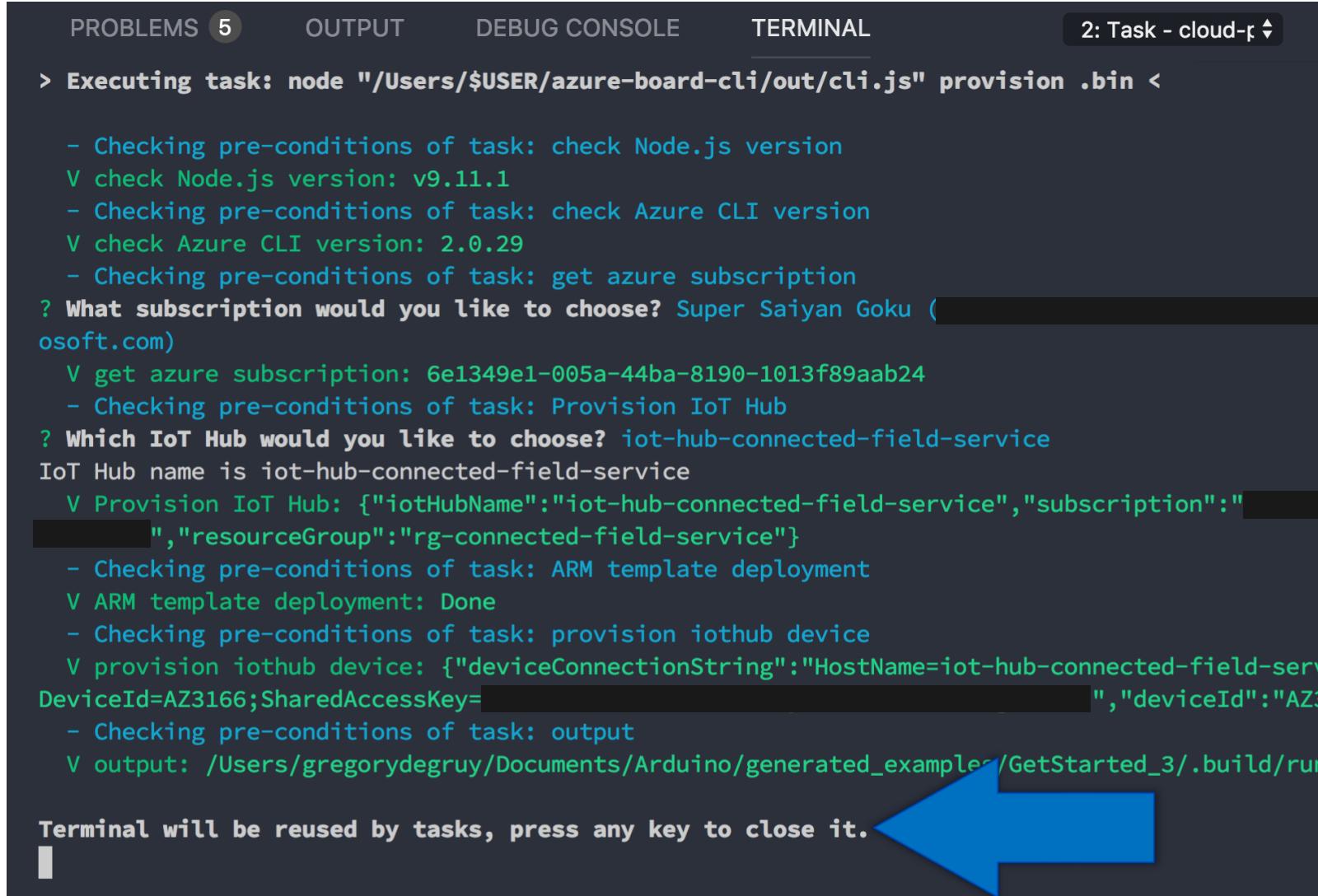


```
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
2: Task - cloud-ρ ▾
> Executing task: node "/Users/$USER/azure-board-cli/out/cli.js" provision .bin <
  - Checking pre-conditions of task: check Node.js version
  V check Node.js version: v9.11.1
  - Checking pre-conditions of task: check Azure CLI version
  V check Azure CLI version: 2.0.29
  - Checking pre-conditions of task: get azure subscription
? What subscription would you like to choose? Super Saiyan Goku (osoft.com)
  V get azure subscription:
  - Checking pre-conditions of task: Provision IoT Hub
? Which IoT Hub would you like to choose? (Use arrow keys)
Select IoT Hub
> iot-hub-connected-field-service
iot-hub-nsbe-convention-iot
-----
Create new...  
...
```

Cloud provision complete

During the IoT device provision process you'll see a series of output steps in the registration process.

You will know the process is complete once you see the final output "Terminal will be reused by tasks, press any key to close it"



```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
2: Task - cloud-p

> Executing task: node "/Users/$USER/azure-board-cli/out/cli.js" provision .bin <

- Checking pre-conditions of task: check Node.js version
V check Node.js version: v9.11.1
- Checking pre-conditions of task: check Azure CLI version
V check Azure CLI version: 2.0.29
- Checking pre-conditions of task: get azure subscription
? What subscription would you like to choose? Super Saiyan Goku (osoft.com)
V get azure subscription: 6e1349e1-005a-44ba-8190-1013f89aab24
- Checking pre-conditions of task: Provision IoT Hub
? Which IoT Hub would you like to choose? iot-hub-connected-field-service
IoT Hub name is iot-hub-connected-field-service
V Provision IoT Hub: {"iotHubName":"iot-hub-connected-field-service","subscription": "", "resourceGroup":"rg-connected-field-service"}
- Checking pre-conditions of task: ARM template deployment
V ARM template deployment: Done
- Checking pre-conditions of task: provision iothub device
V provision iothub device: {"deviceConnectionString":"HostName=iot-hub-connected-field-service.DeviceId=AZ3166;SharedAccessKey=", "deviceId":"AZ3166", "connectionString": "HostName=iot-hub-connected-field-service.DeviceId=AZ3166;SharedAccessKey="}
- Checking pre-conditions of task: output
V output: /Users/gregorydegruy/Documents/Arduino/generated_examples/GetStarted_3/.build/run

Terminal will be reused by tasks, press any key to close it.

```

Device in Azure

If you navigate to your Azure portal now, you'll see your device has been properly registered in Azure!

With the same device ID from the VS Code Terminal output.

The screenshot shows the 'iot-hub-connected-field-service - IoT Devices' blade in the Azure portal. The left sidebar lists 'Locks', 'Automation script', 'EXPLORERS', 'Query Explorer', 'DEVICE MANAGEMENT', 'IoT Devices' (which is selected), 'IoT Edge (preview)', 'MESSAGING', 'Event Grid (preview)', and 'File upload'. The main area displays a query interface with the following text:

```
SELECT * FROM devices WHERE  
optional (e.g. tags.location='US')
```

Below the query is a blue 'Execute' button. A table at the bottom lists device details:

| DEVICE ID | STATUS | LAST ACTIVITY | LAST STATUS UPDATE | AUTH |
|-----------|---------|---------------|--------------------|------|
| AZ3166 | Enabled | | | Sas |

Build and upload the Arduino Code

Let's start configuring the **GetStarted.ino** onto your MXCHIP:

- `1` Run your task through Ctrl+Shift+P (macOS: Cmd+P) and select Tasks:Run Task.
- `2` Select the device-upload task.

1

2

3

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
> Executing task: node "C:\Users\grego\azure-board-cli\out\cli.js" build .bin <
- Checking pre-conditions of task: check Node.js version
V check Node.js version: v8.9.3
- Checking pre-conditions of task: Generate platform.local.txt
V Generate platform.local.txt: platform.local.txt already exists
- Checking pre-conditions of task: Config azure connection string
Opening COM3.
Please hold down button A and then push and release the reset button to enter configuration mode.

```

MXCHIP

Configuration mode

-`3` The terminal prompts you to enter configuration mode. To do so, hold down button A, then push and release the reset button. The screen displays the DevKit id and 'Configuration'. This is to set the connection string that retrieves from task cloud-provision step.

Then VS Code starts verifying and uploading the Arduino sketch. The MXCHIP reboots and starts running the code!

Note

Occasionally, you get error "Error: AZ3166: Unknown package". This is due to the board package index is not refreshed. Check this [FAQ steps](#) to solve it.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

> Executing task: node "C:\Users\grego\azure-board-cli\out\cli.js" build .bin <

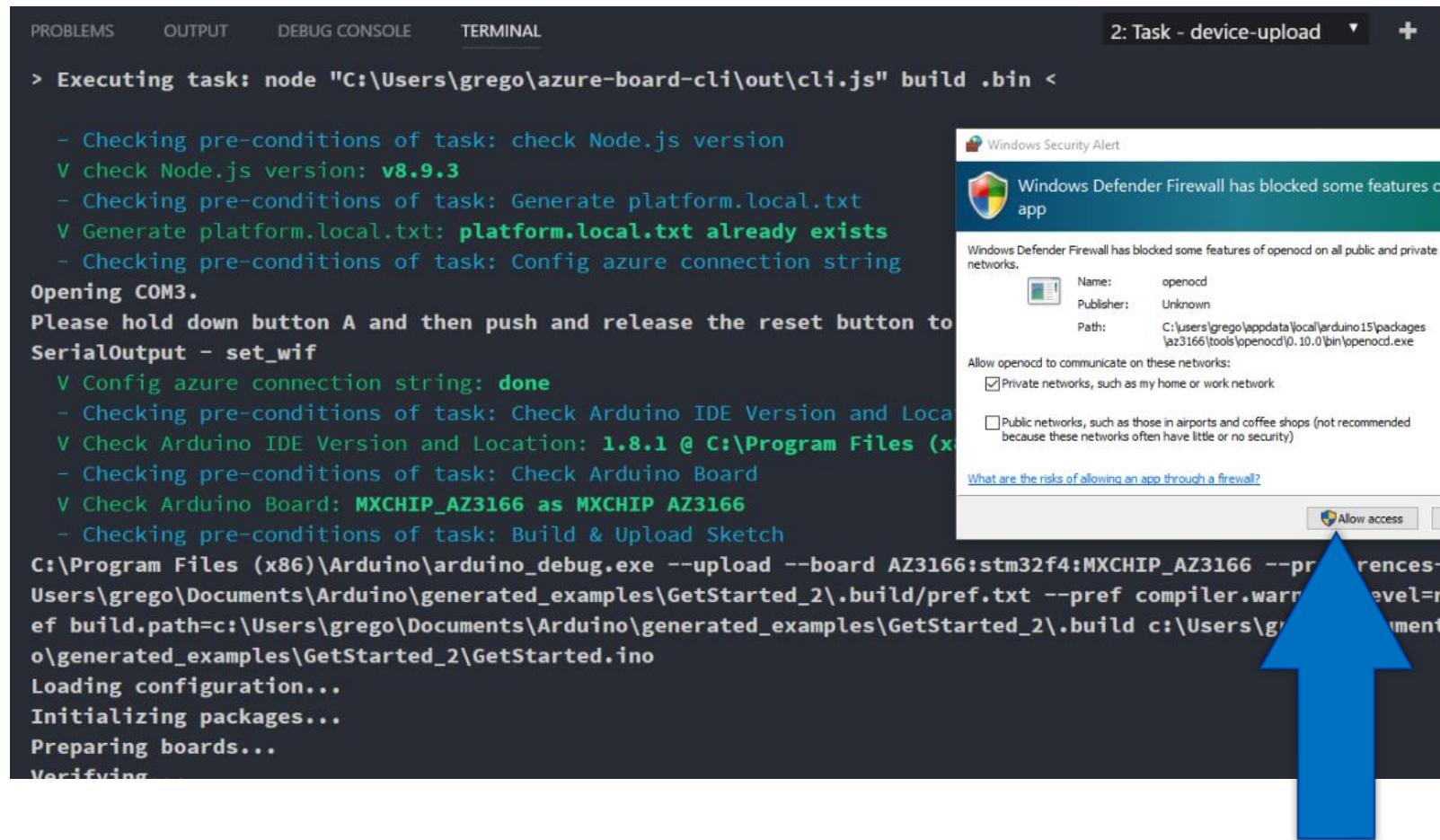
- Checking pre-conditions of task: check Node.js version
- V check Node.js version: v8.9.3
- Checking pre-conditions of task: Generate platform.local.txt
- V Generate platform.local.txt: platform.local.txt already exists
- Checking pre-conditions of task: Config azure connection string

Opening COM3.

Please hold down button A and then push and release the reset button to enter configuration mode.

Firewall

On first run of the **GetStarted.ino** code, you may be prompted to allow Firewall access during the upload process, click Allow access.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: Task - device-upload

```
> Executing task: node "C:\Users\grego\azure-board-cli\out\cli.js" build .bin <
- Checking pre-conditions of task: check Node.js version
V check Node.js version: v8.9.3
- Checking pre-conditions of task: Generate platform.local.txt
V Generate platform.local.txt: platform.local.txt already exists
- Checking pre-conditions of task: Config azure connection string
Opening COM3.
Please hold down button A and then push and release the reset button to
SerialOutput - set_wif
V Config azure connection string: done
- Checking pre-conditions of task: Check Arduino IDE Version and Location
V Check Arduino IDE Version and Location: 1.8.1 @ C:\Program Files (x86)\Arduino
- Checking pre-conditions of task: Check Arduino Board
V Check Arduino Board: MXCHIP_AZ3166 as MXCHIP AZ3166
- Checking pre-conditions of task: Build & Upload Sketch
C:\Program Files (x86)\Arduino\arduino_debug.exe --upload --board AZ3166:stm32f4:MXCHIP_AZ3166 --pref build.path=c:\Users\grego\Documents\Arduino\generated_examples\GetStarted_2\.build\pref.txt --pref compiler.warnings.level=0
ef build.path=c:\Users\grego\Documents\Arduino\generated_examples\GetStarted_2\.build c:\Users\grego\Documents\Arduino\generated_examples\GetStarted_2\GetStarted.ino
Loading configuration...
Initializing packages...
Preparing boards...
Verifying...
```

Windows Security Alert

Windows Defender Firewall has blocked some features of app

Windows Defender Firewall has blocked some features of openocd on all public and private networks.

| | |
|------------|--|
| Name: | openocd |
| Publisher: | Unknown |
| Path: | C:\users\grego\appdata\local\arduino 15\packages\az3166\tools\openocd\0.10.0\bin\openocd.exe |

Allow openocd to communicate on these networks:

Private networks, such as my home or work network

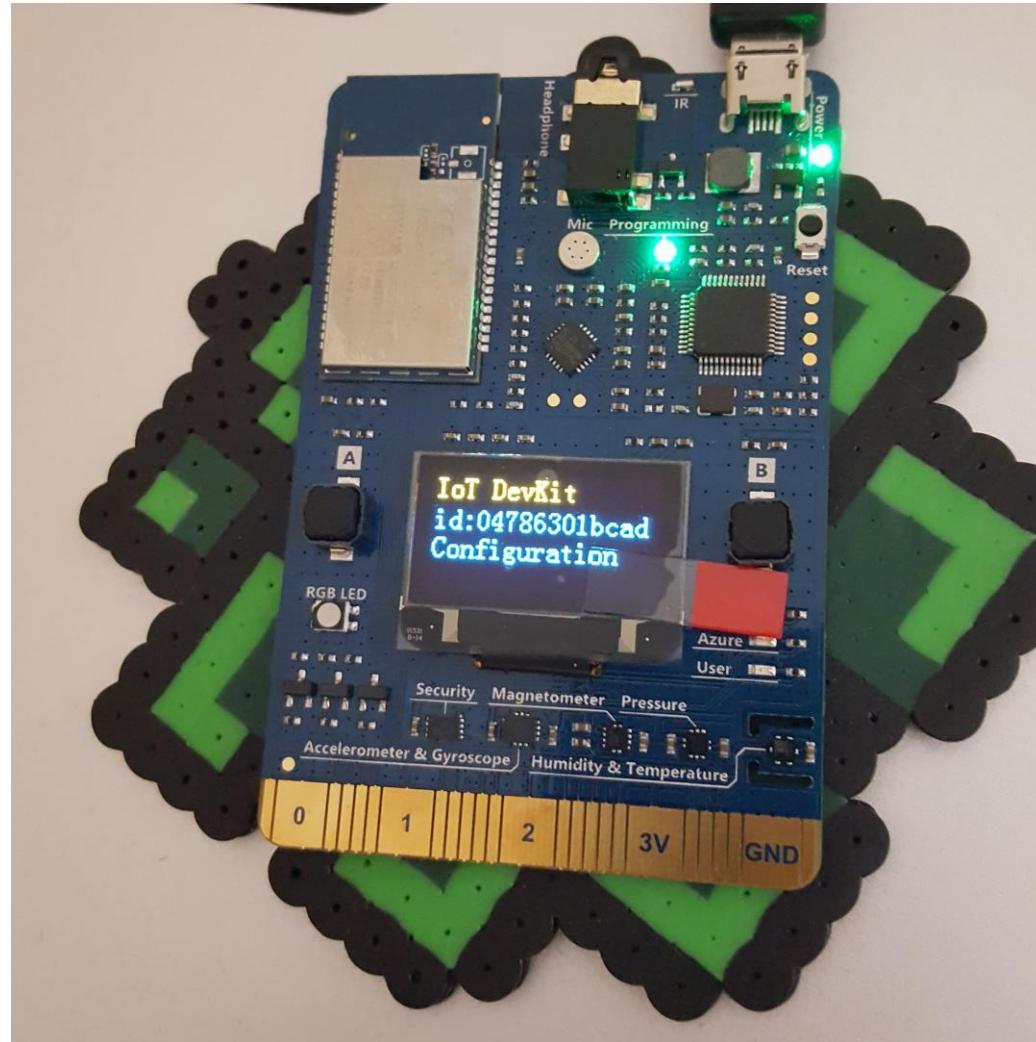
Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

[What are the risks of allowing an app through a firewall?](#)

Allow access

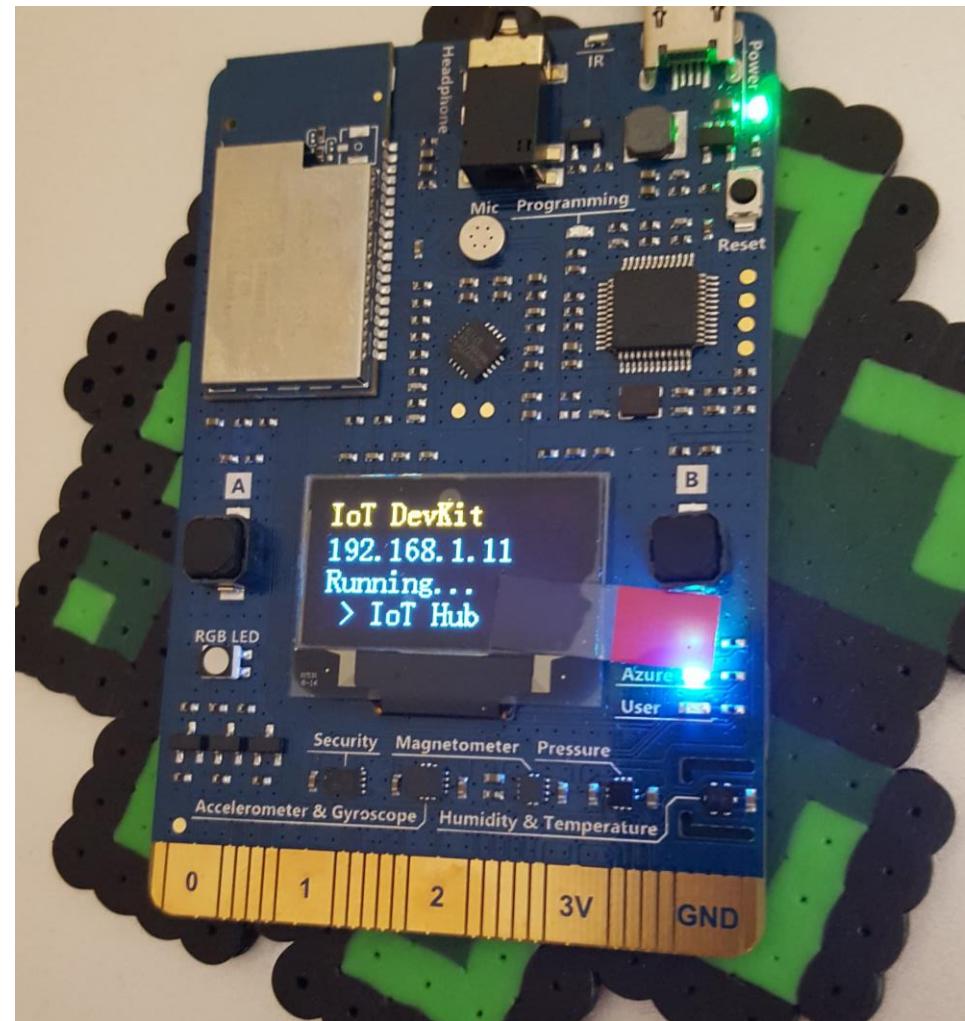
Deploying code

Your device should be in configuration mode during this process briefly.



Data sending to IoT Hub

One configuration and code upload is complete. the device code is now deployed to the MXChip and temperature and humidity data is being sent to IoT Hub!



Confirm success

The upload is safely completed when you see the following Build & Upload Sketch: success response in the Terminal window

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Task - dev

```
Info : Unable to match requested speed 2000 kHz, using 1800 kHz
adapter speed: 1800 kHz
target halted due to debug-request, current mode: Thread
xPSR: 0x1000000 pc: 0x08000034 msp: 0x200073fc
Info : Unable to match requested speed 8000 kHz, using 4000 kHz
Info : Unable to match requested speed 8000 kHz, using 4000 kHz
adapter speed: 4000 kHz
** Programming Started **
auto erase enabled
Info : device id = 0x30006441
Info : flash size = 1024kbytes
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x20000046 msp: 0x200073fc
wrote 999424 bytes from file c:\Users\grego\Documents\Arduino\generated_examples\GetStarted_
in in 22.277353s (43.811 KiB/s)
** Programming Finished **
** Verify Started **
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x2000002e msp: 0x200073fc
arduino_debug.exe exited.
V Build & Upload Sketch: success
Terminal will be reused by tasks, press any key to close it.
```

Device activity IoT Hub

In Azure you'll also see that the device has been activated with a timestamp for Last Activity.

The screenshot shows the 'iot-hub-connected-field-service - IoT Devices' blade in the Azure portal. On the left, there's a sidebar with sections for 'EXPLORERS' (Locks, Automation script), 'DEVICE MANAGEMENT' (IoT Devices, IoT Edge (preview)), and 'MESSAGING' (Event Grid (preview), File upload, Endpoints). The main area has a search bar, 'Add', 'Refresh', and 'Delete' buttons. A message says: 'You can use this tool to view, create, update, and delete devices on your IoT Hub.' Below it is a 'Query' section with a query editor containing:

```
SELECT * FROM devices WHERE  
optional (e.g. tags.location='US')
```

There's a 'Execute' button. At the bottom, a table displays device information:

| DEVICE ID | STATUS | LAST ACTIVITY | LAST STATUS UPDATE | AUTH |
|-----------|---------|---------------------------|--------------------|------|
| AZ3166 | Enabled | Thu Apr 26 2018 11:26:... | | Sas |

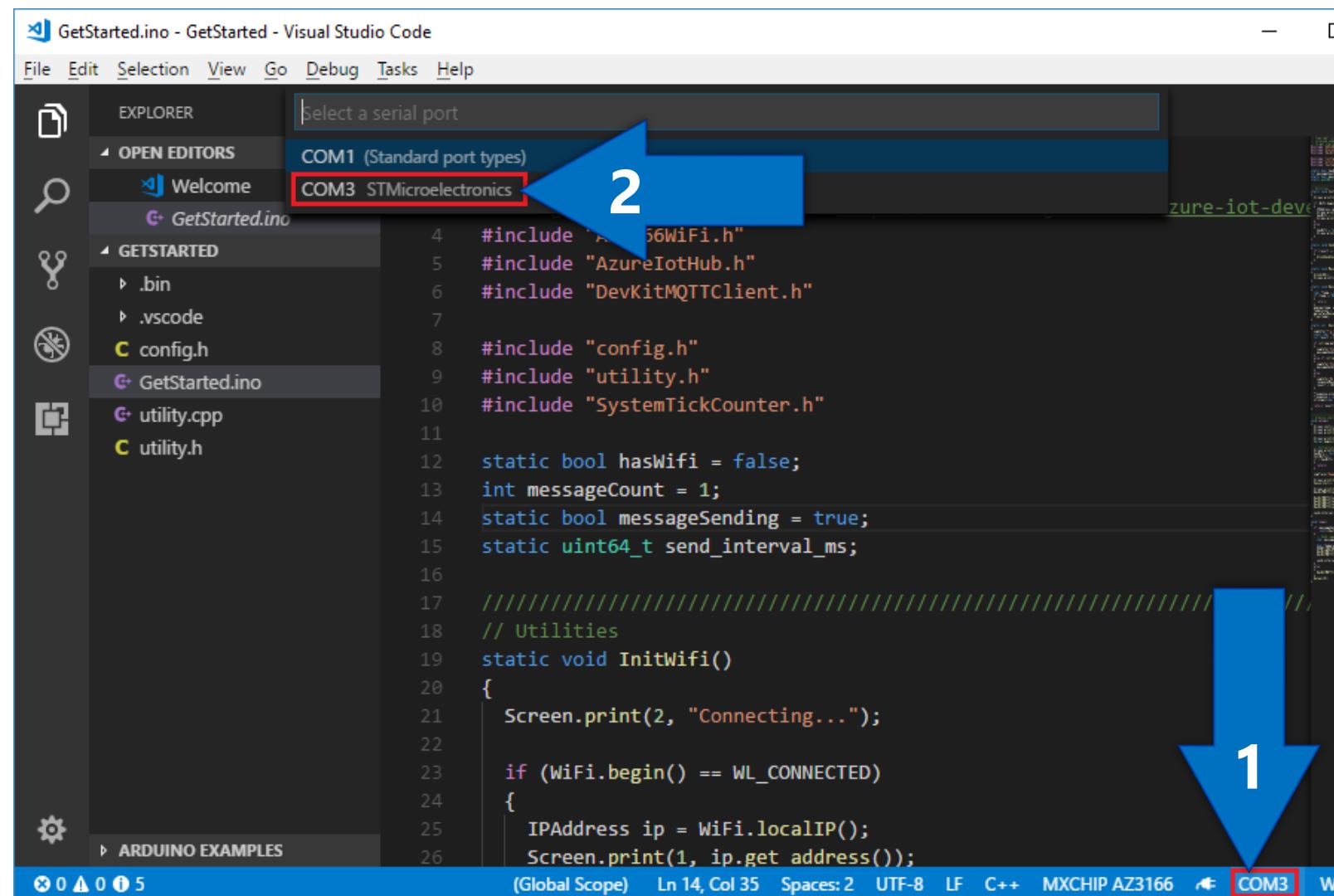


Viewing and testing data upload

Let's properly view the data upload before connecting it to Dynamics. This is important to understand so we can know what data we are able to send to Dynamics Field Service.

In VS Code, follow these steps to open and set up the Serial Monitor:

- `1` Click COM on the status bar.
- `2` Then set the COM port that says STMicroelectronics. In my case it is COM3



GetStarted.ino - GetStarted - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

EXPLORER Select a serial port

OPEN EDITORS COM1 (Standard port types)
Welcome COM3 STMicroelectronics
GetStarted.ino

GETSTARTED .bin
.vscode
config.h
GetStarted.ino
utility.cpp
utility.h

```

4 #include "ESP8266WiFi.h"
5 #include "AzureIoTHub.h"
6 #include "DevKitMQTTClient.h"
7
8 #include "config.h"
9 #include "utility.h"
10 #include "SystemTickCounter.h"
11
12 static bool hasWifi = false;
13 int messageCount = 1;
14 static bool messageSending = true;
15 static uint64_t send_interval_ms;
16
17 // Utilities
18 static void InitWifi()
19 {
20     Screen.print(2, "Connecting...");
21
22     if (WiFi.begin() == WL_CONNECTED)
23     {
24         IPAddress ip = WiFi.localIP();
25         Screen.print(1, ip.get_address());
26     }

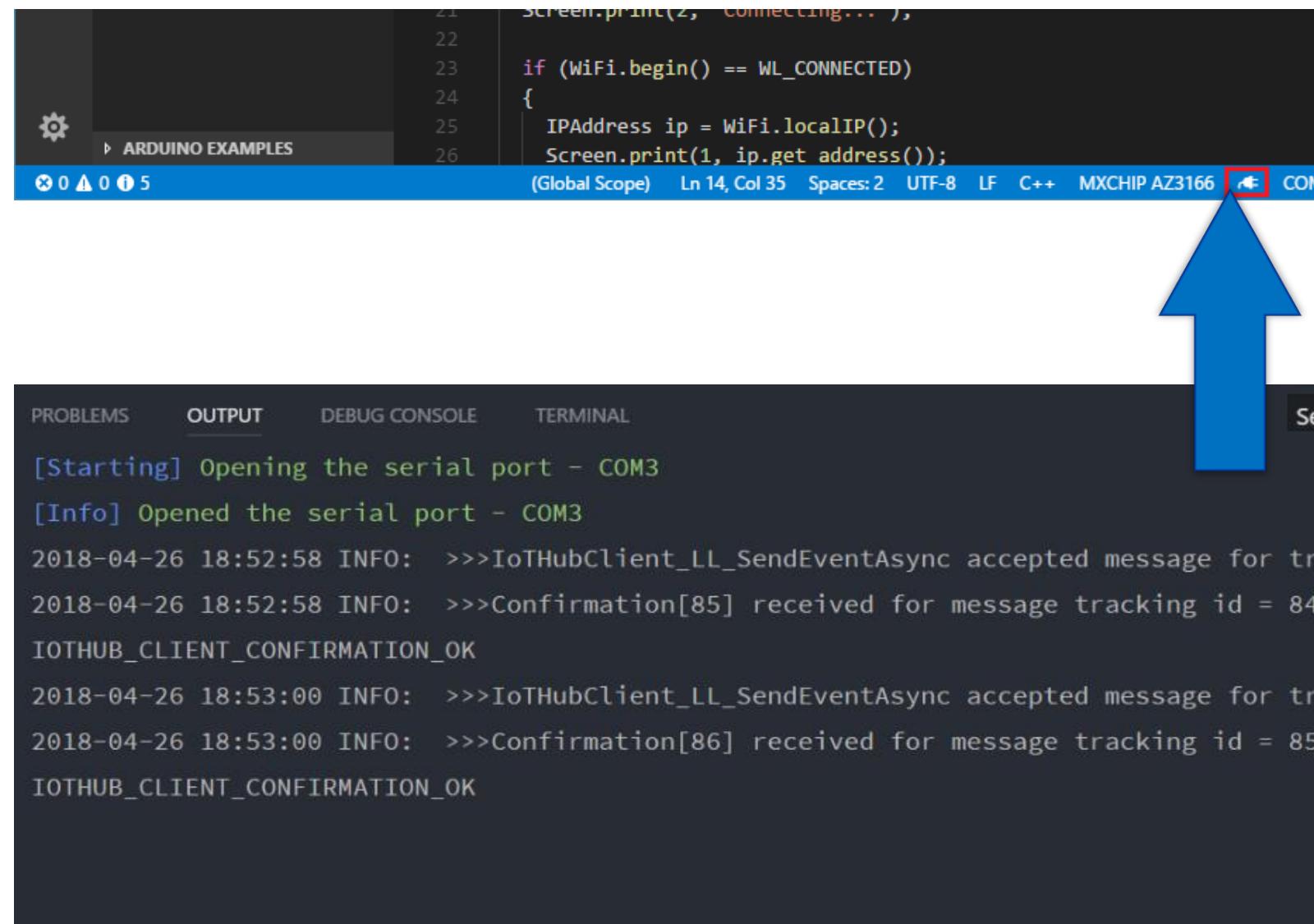
```

(Global Scope) Ln 14, Col 35 Spaces: 2 UTF-8 LF C++ MXCHIP AZ3166 COM3

Serial monitor

Click power plug icon on the status bar to open the Serial Monitor.

The VS Code Output tab will confirm the opening of this port. Meaning messages are being sent to IoT Hub! The LED on the MXCHIP will also blink each time a message is being sent to IoT Hub.



A screenshot of the Visual Studio Code interface. At the top, there's a dark header bar with a gear icon, 'ARDUINO EXAMPLES' text, and a status bar showing '0 0 ▲ 0 0 5'. Below this is a light blue navigation bar with tabs for 'PROBLEMS', 'OUTPUT' (which is selected), 'DEBUG CONSOLE', and 'TERMINAL'. A large blue arrow points upwards from the bottom of the screen towards the 'OUTPUT' tab. The main workspace shows an Arduino sketch with code for connecting to WiFi and printing local IP address. The status bar at the bottom of the code editor shows '(Global Scope) Ln 14, Col 35 Spaces: 2 UTF-8 LF C++ MXCHIP AZ3166'. In the bottom right corner of the code editor, there's a small red square with a white arrow pointing left, and next to it, the text 'Serial'.

```
21 |     Screen.print(2, "Connecting... ");
22 |
23 |     if (WiFi.begin() == WL_CONNECTED)
24 |     {
25 |         IPAddress ip = WiFi.localIP();
26 |         Screen.print(1, ip.get_address());
```

[Starting] Opening the serial port - COM3
[Info] Opened the serial port - COM3
2018-04-26 18:52:58 INFO: >>>IoTHubClient_LL_SendEventAsync accepted message for tracking id = 84
2018-04-26 18:52:58 INFO: >>>Confirmation[85] received for message tracking id = 84
IOTHUB_CLIENT_CONFIRMATION_OK
2018-04-26 18:53:00 INFO: >>>IoTHubClient_LL_SendEventAsync accepted message for tracking id = 85
2018-04-26 18:53:00 INFO: >>>Confirmation[86] received for message tracking id = 85
IOTHUB_CLIENT_CONFIRMATION_OK

Data upload rate

- `1` On the status bar, click the number that represents the Baud Rate
- `2` Set to 115200. This means 115200 bits of data will be sent per second.

The screenshot shows the Visual Studio Code interface with the Arduino extension installed. The title bar reads "GetStarted.ino - GetStarted - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. The Explorer sidebar on the left lists "OPEN EDITORS" (4800, 9600, 19200), "GETSTARTED" (38400, .bin, .vscode, 57600, 74880, config.h, 115200, GetStarted.ino, utility.cpp, utility.h), and "ARDUINO EXAMPLES". The main editor area contains C++ code for an Arduino sketch. The status bar at the bottom shows "Global Scope" and "Ln 14, Col 35" along with other build-related information. Two large blue arrows are overlaid on the image: one pointing to the "115200" entry in the "config.h" list with the label "2", and another pointing to the status bar with the label "1".

```

12 static bool hasWifi = false;
13 int messageCount = 1;
14 static bool messageSending = true;
15 static uint64_t send_interval_ms;
16
17 // Utilities
18 static void InitWifi()
19 {
20     Screen.print(2, "Connecting...");
21
22     if (WiFi.begin() == WL_CONNECTED)
23     {
24         IPAddress ip = WiFi.localIP();
25         Screen.print(1, ip.get_address());
26     }
}

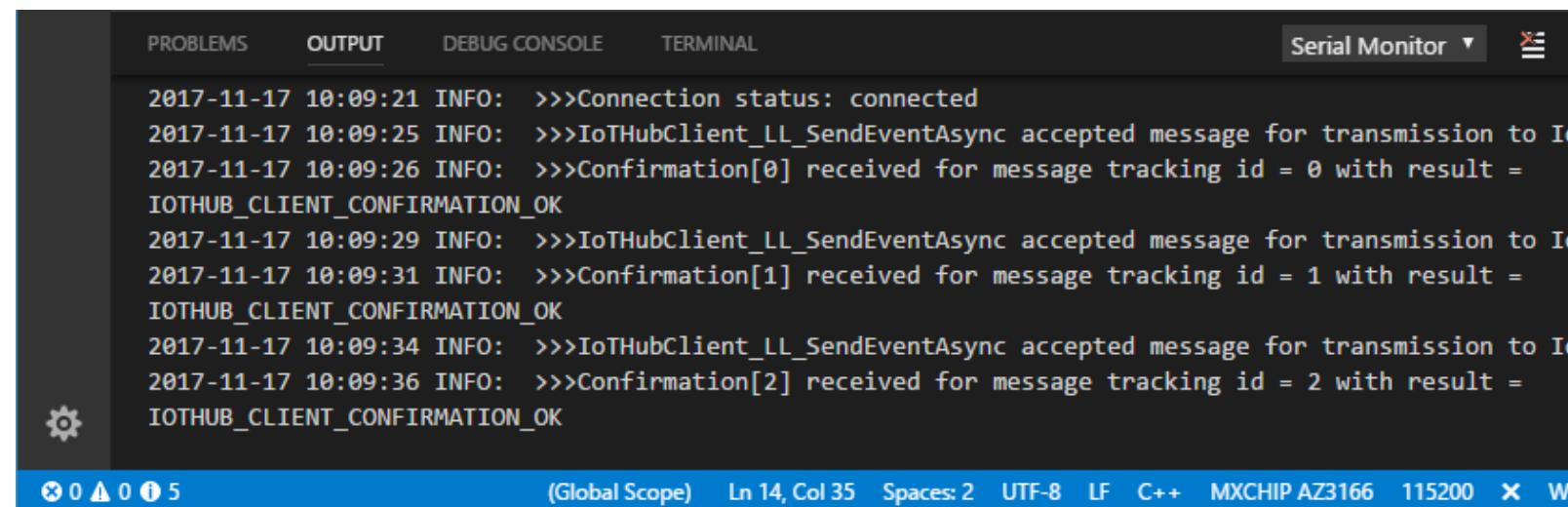
```

(Global Scope) Ln 14, Col 35 Spaces: 2 UTF-8 LF C++ MXCHIP AZ3166 9600 X Win32 COM3

Confirm success

The sample application is running successfully when you see the following results:

- The Serial Monitor displays the same information as the content in the screenshot below and will continue to do so as captured sensor data is sent to the IoT Hub.
- The LED on MXChip IoT DevKit is blinking.



A screenshot of the VS Code interface showing the Serial Monitor tab. The tab bar includes PROBLEMS, OUTPUT (which is selected), DEBUG CONSOLE, and TERMINAL. The Serial Monitor window displays a series of informational log messages from an IoT application. The messages indicate a successful connection to an IoT Hub and the receipt of confirmation for three transmitted messages, each with tracking ID 0, 1, and 2, and result IOTHUB_CLIENT_CONFIRMATION_OK. The log entries are as follows:

```
2017-11-17 10:09:21 INFO: >>>Connection status: connected
2017-11-17 10:09:25 INFO: >>>IoTHubClient_LL_SendEventAsync accepted message for transmission to IoT Hub
2017-11-17 10:09:26 INFO: >>>Confirmation[0] received for message tracking id = 0 with result = IOTHUB_CLIENT_CONFIRMATION_OK
2017-11-17 10:09:29 INFO: >>>IoTHubClient_LL_SendEventAsync accepted message for transmission to IoT Hub
2017-11-17 10:09:31 INFO: >>>Confirmation[1] received for message tracking id = 1 with result = IOTHUB_CLIENT_CONFIRMATION_OK
2017-11-17 10:09:34 INFO: >>>IoTHubClient_LL_SendEventAsync accepted message for transmission to IoT Hub
2017-11-17 10:09:36 INFO: >>>Confirmation[2] received for message tracking id = 2 with result = IOTHUB_CLIENT_CONFIRMATION_OK
```

At the bottom of the Serial Monitor window, there are status indicators for errors (0), warnings (0), and info (5). The footer shows the configuration: (Global Scope), Ln 14, Col 35, Spaces: 2, UTF-8, LF, C++, MXCHIP AZ3166, 115200, and a close button.

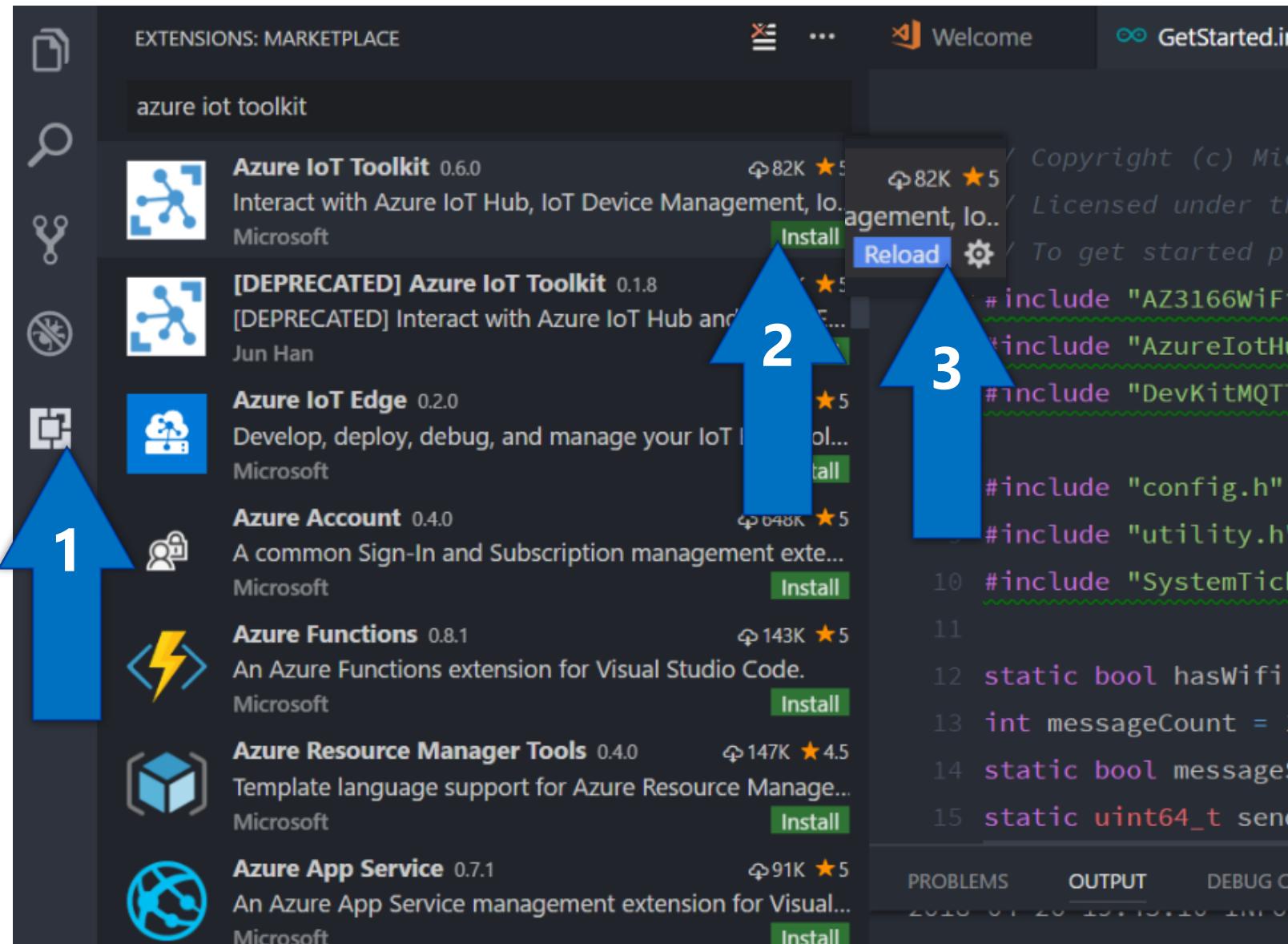
VS Code IoT extension

In VS Code

-`1` Open Extensions and search for the "azure iot toolkit"

-`2` Click install

-`3` Click Reload window once installed.
Once VS Code reloads make sure to open the **GetStarted.ino** up again.



IoT Hub connection string

Go to your IoT Hub, select the iothubowner Policy and copy your "Connection string – primary key" into Notepad

The screenshot shows the 'Shared access policies' section of the Azure IoT Hub configuration. The 'iothubowner' policy is selected, displaying its permissions: Registry read, Registry write, Service connect, and Device connect. Below the table, there are fields for Shared access keys, including Primary key and Secondary key, and their corresponding Connection strings.

| POLICY | PERMISSIONS |
|------------------------|--|
| iothubowner | registry write, service connect, device connect, registry read |
| service | service connect |
| device | device connect |
| registryRead | registry read |
| registryReadWrite | registry write |
| delme-particle-iot-hub | registry write, device connect |

Access policy name: iothubowner

Permissions:

- Registry read ⓘ
- Registry write ⓘ
- Service connect ⓘ
- Device connect ⓘ

Shared access keys:

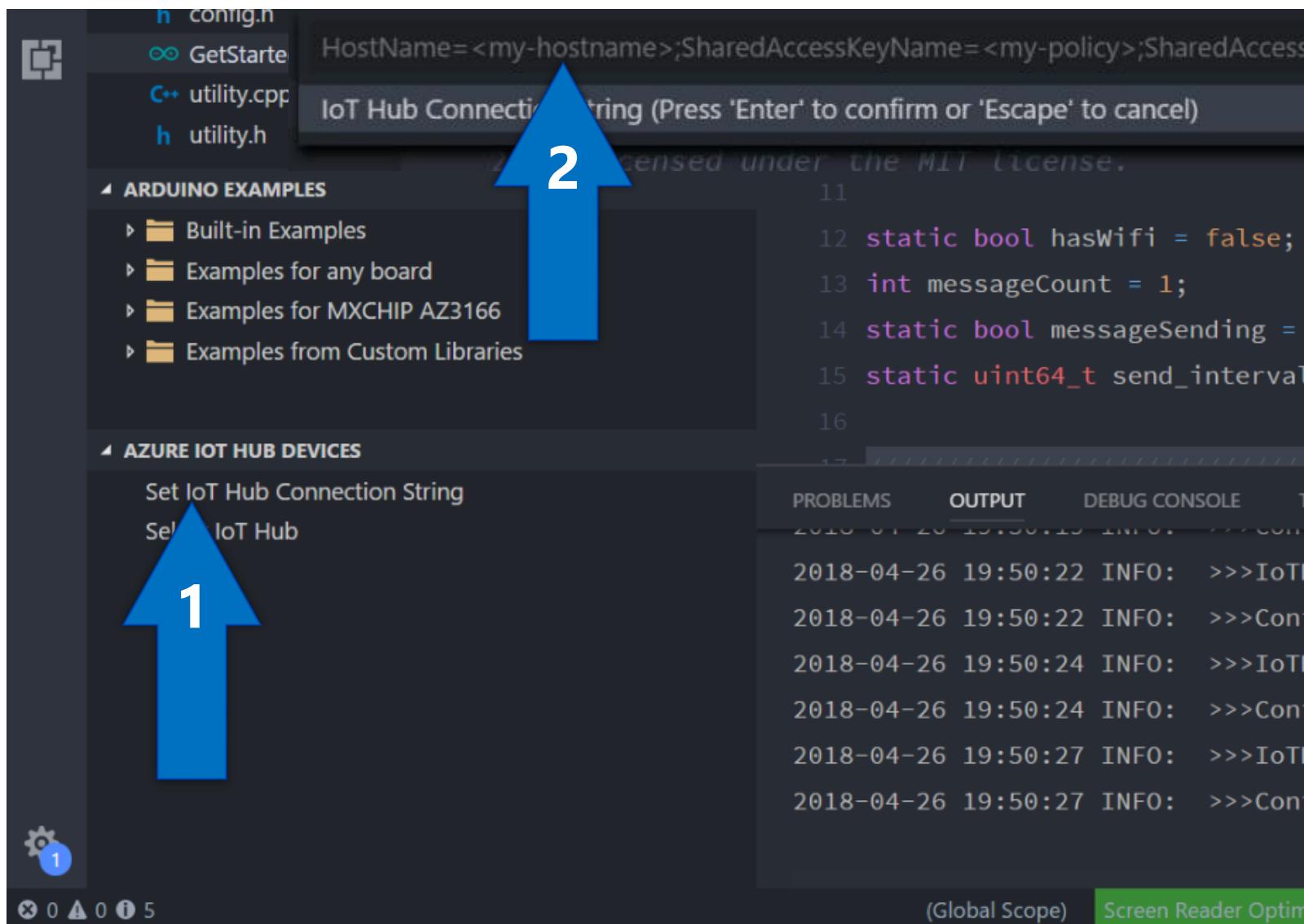
- Primary key ⓘ
- Secondary key ⓘ
- Connection string—primary key ⓘ
- Connection string—secondary key ⓘ

Connection string in VS Code

With the new extension installed.

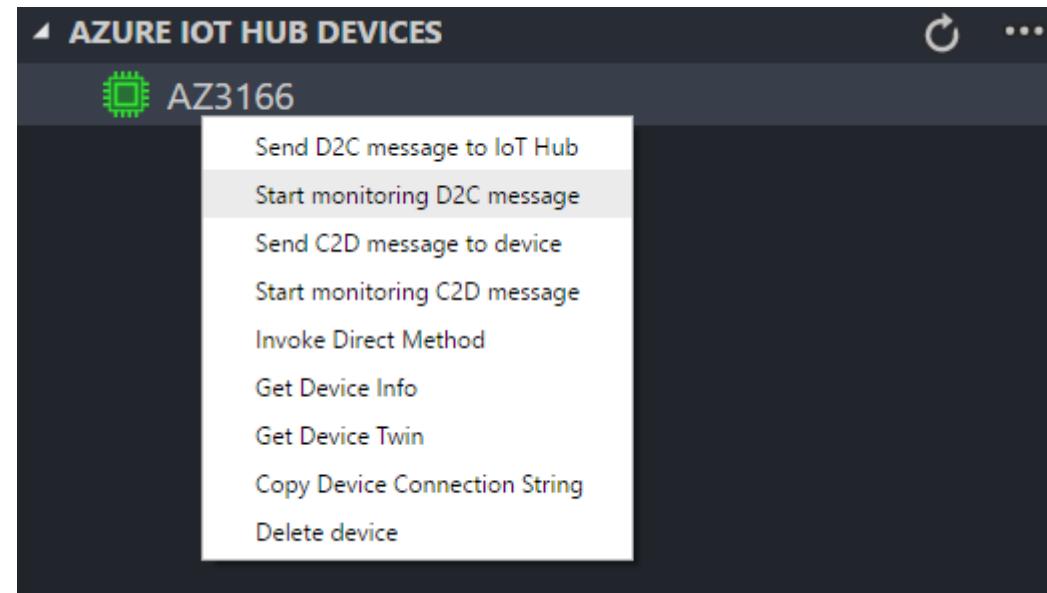
- `1` Go back to VS Code and click the Set IoT Hub Connection String
- `2` Paste your "Connection string – primary key"

You should now see your device appear in AZURE IOT HUB DEVICES Tab as a green chip.



D2C Message

Right click on the device and choose
"Start monitoring D2C message"



JSON Payload

Success!

You should now clearly see the messages being sent to IoT Hub in a JSON payload. Notice it's just temperature and humidity data with some id information. Breath on the device to see the values change.

This message structure tells us exactly how we need to format our Azure Queries responsible for sending this data to Dynamics

The screenshot shows a code editor interface with an Arduino sketch named "GetStarted.ino". The code includes #include "config.h", applicationProperties, and a JSON message body with deviceId, messageId, temperature, and humidity fields. The terminal window shows a received message from AZ3166 with the same structure.

```
#include "config.h"
applicationProperties : {
    "temperatureAlert": "false"
}
[IoTHubMonitor] Message received from [AZ3166]:
{
    "body": {
        "deviceId": "AZ3166",
        "messageId": 2452,
        "temperature": 27.9,
        "humidity": 51.900002
    },
    "applicationProperties": {
        "temperatureAlert": "false"
    }
}
```

(Global Scope) Screen Reader Optimized