Provision and Deploy MXCHIP

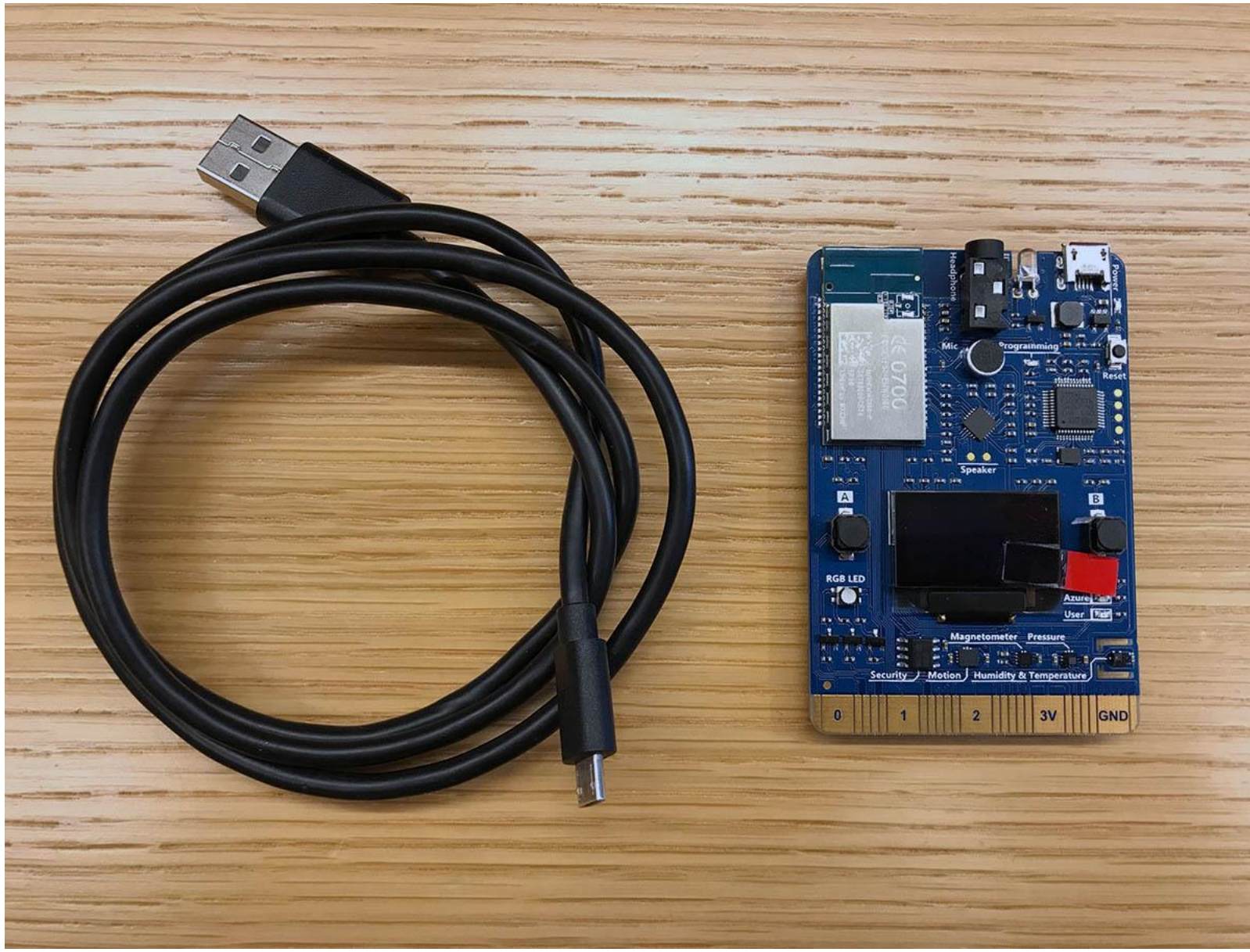
Exercise 5

EXERCISE 5

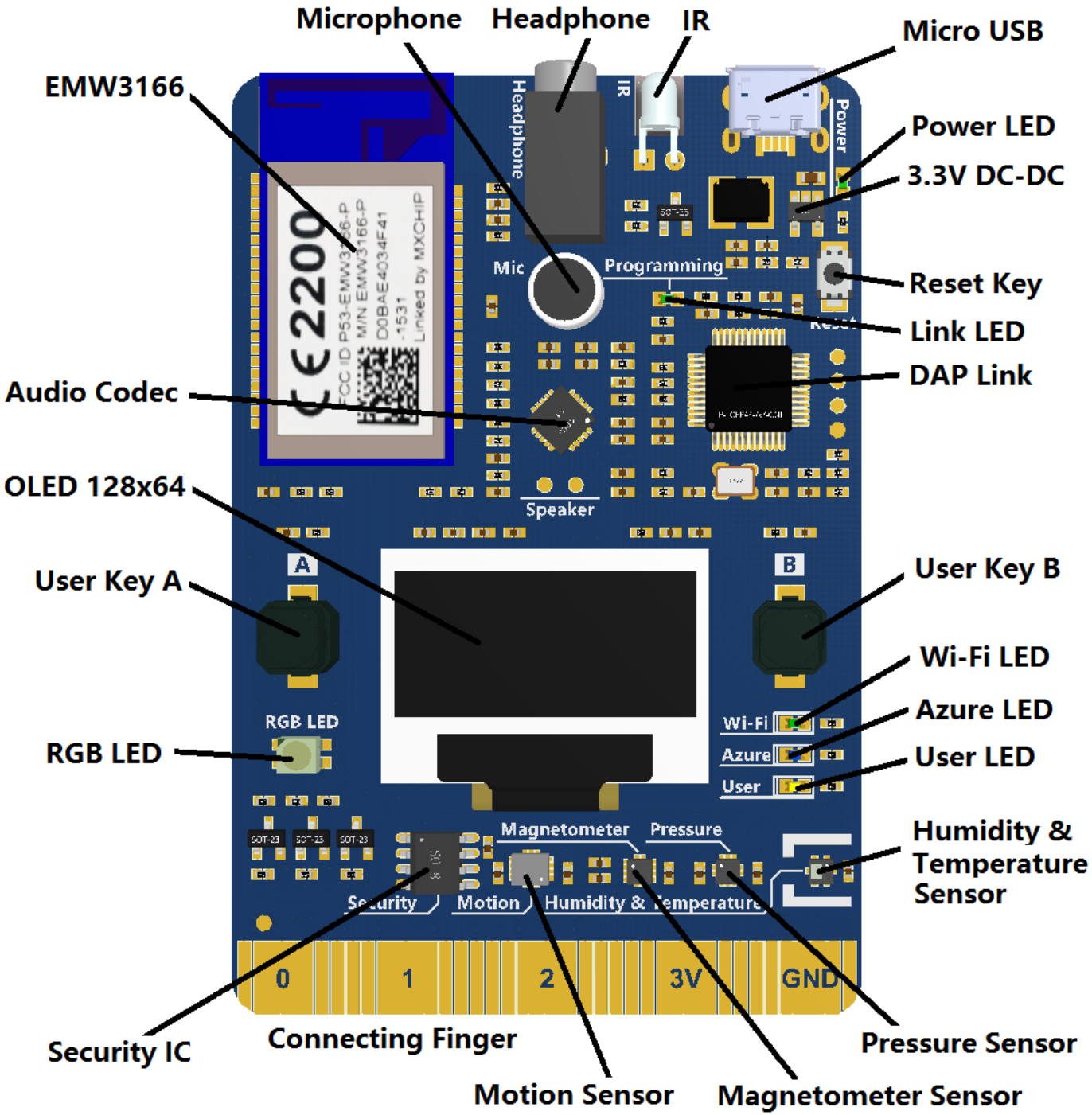
IoT Device MXCHIP

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

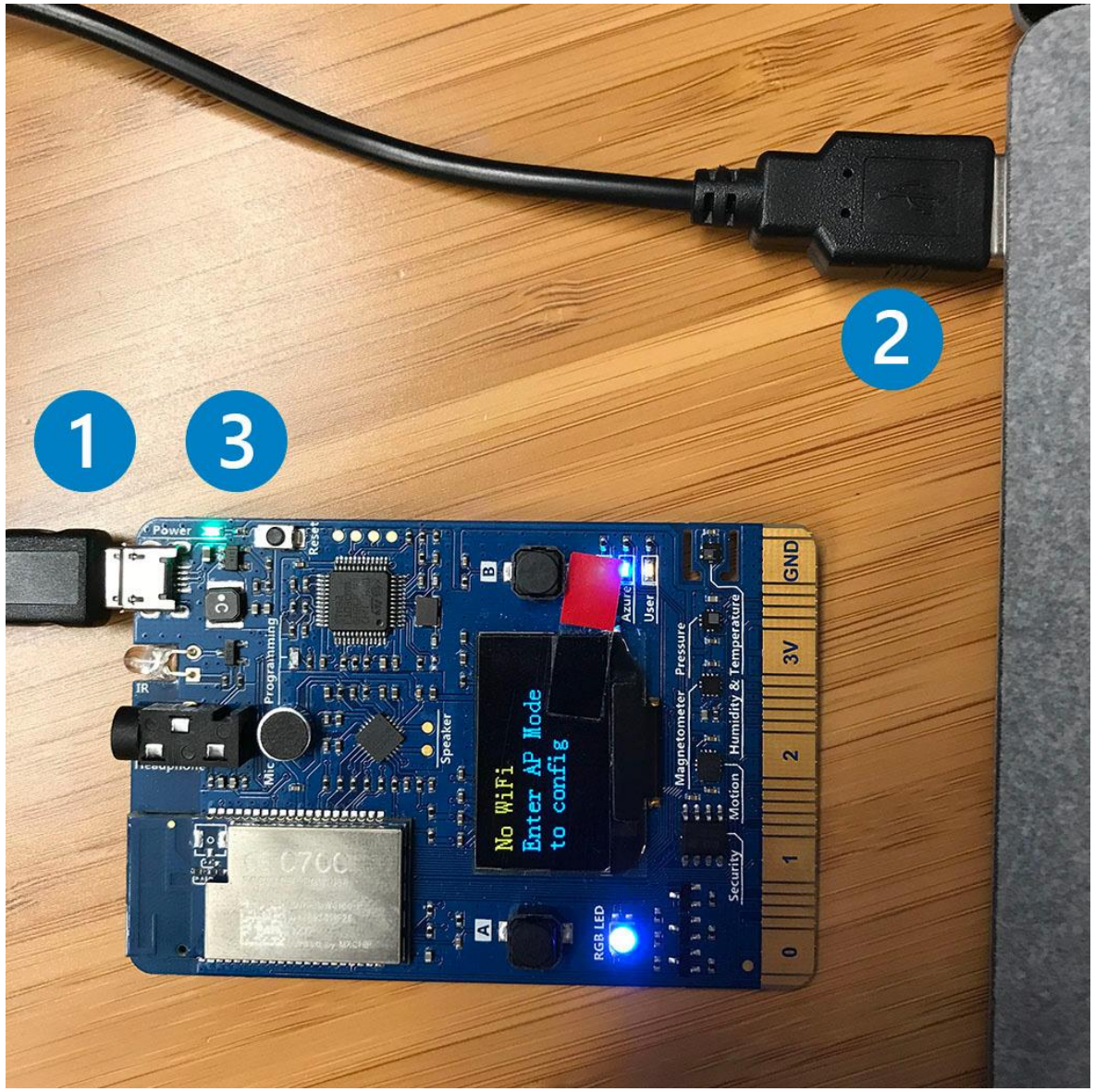


Connect MXCHIP to Computer

- `1` Connect the USB end to your computer.

- `2` Connect the Micro-USB end to the DevKit.

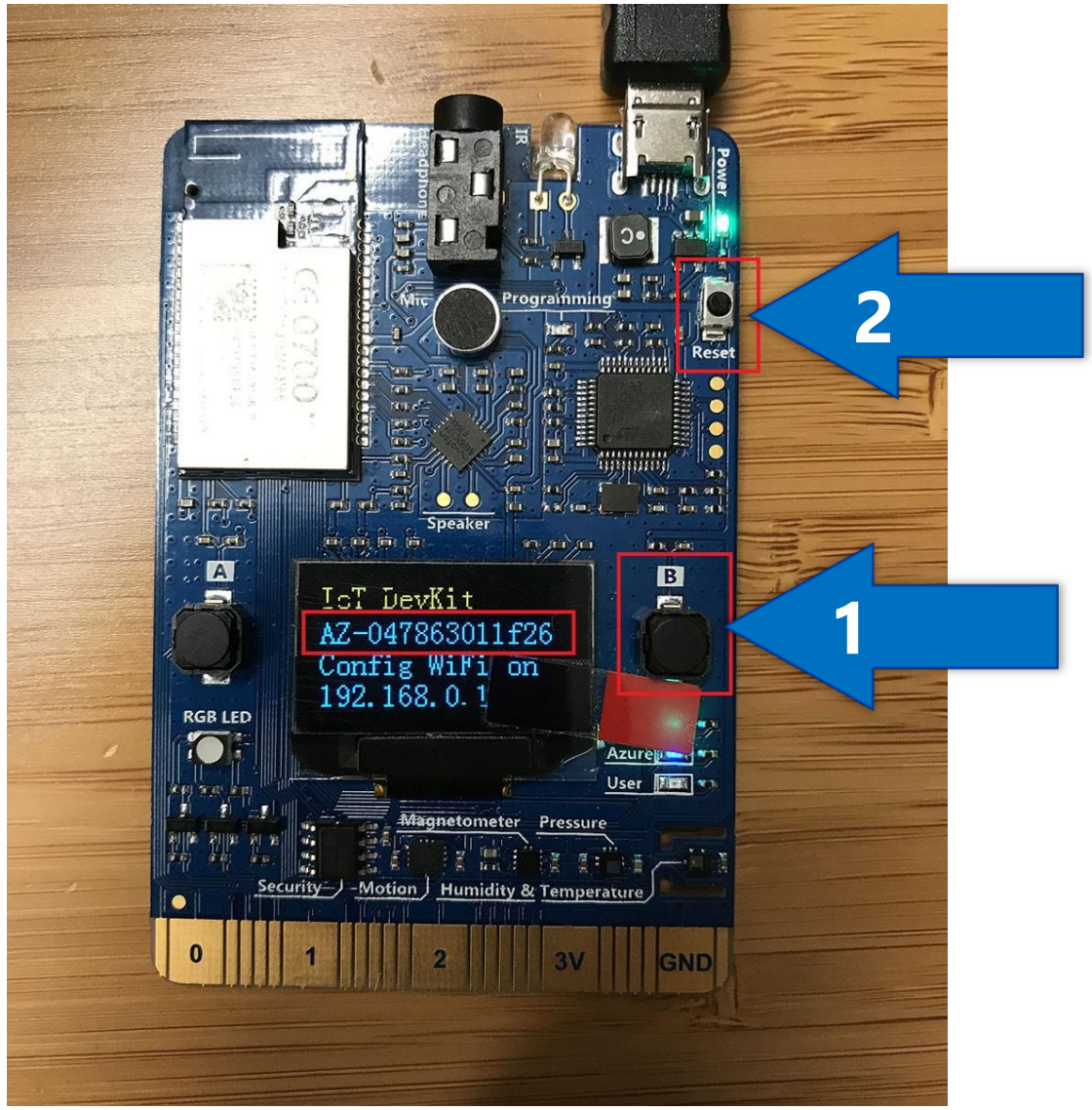
- `3` The green LED for power confirms the connection.



EXERCISE 4

Enter AP Mode for Wifi

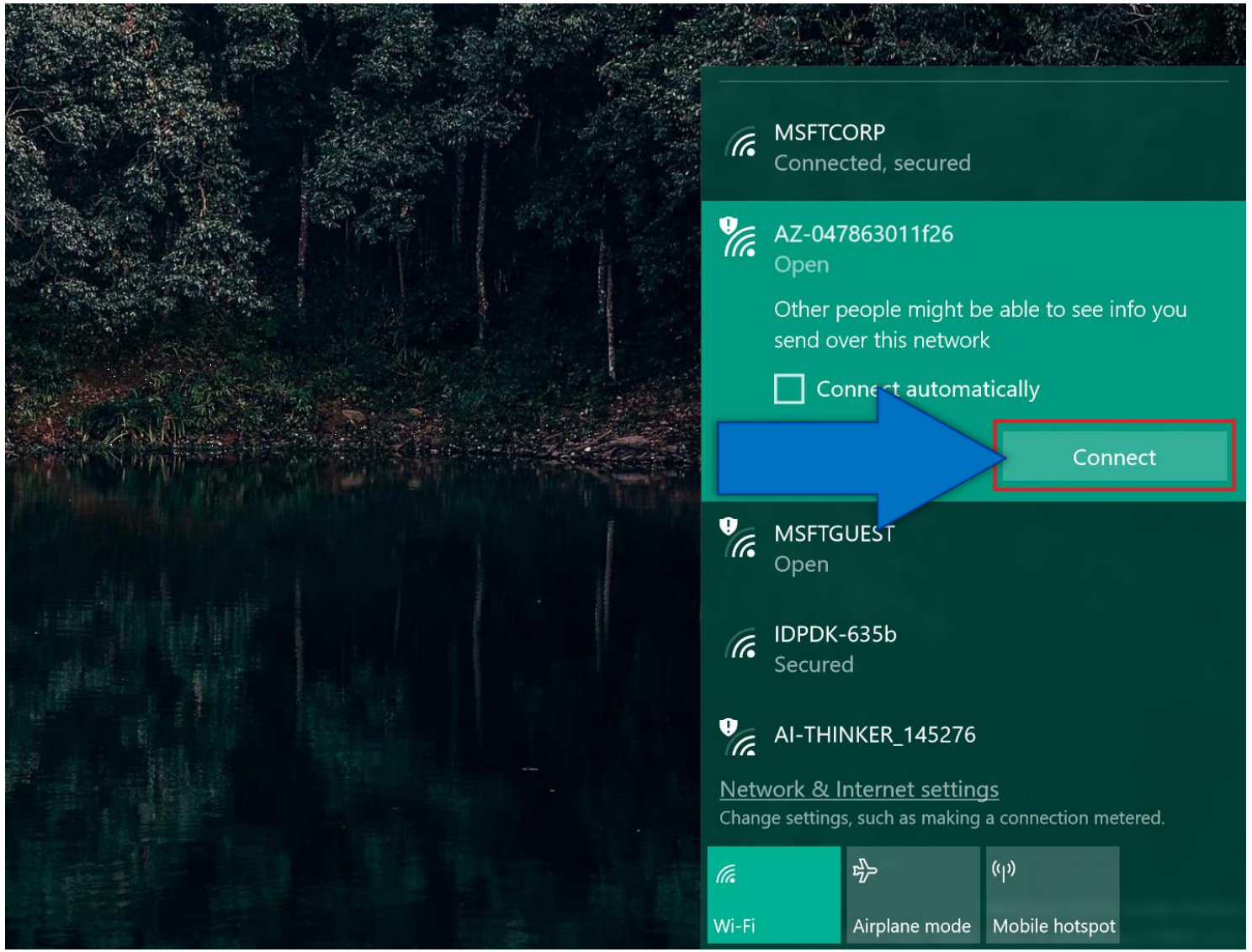
- `1` Hold down button B
- `2` Push and release the reset button, and then release button B. Your MXCHIP enters AP mode for configuring Wi-Fi. The screen displays the service set identifier (SSID) of the MXCHIP and the configuration portal IP address.



EXERCISE 4

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



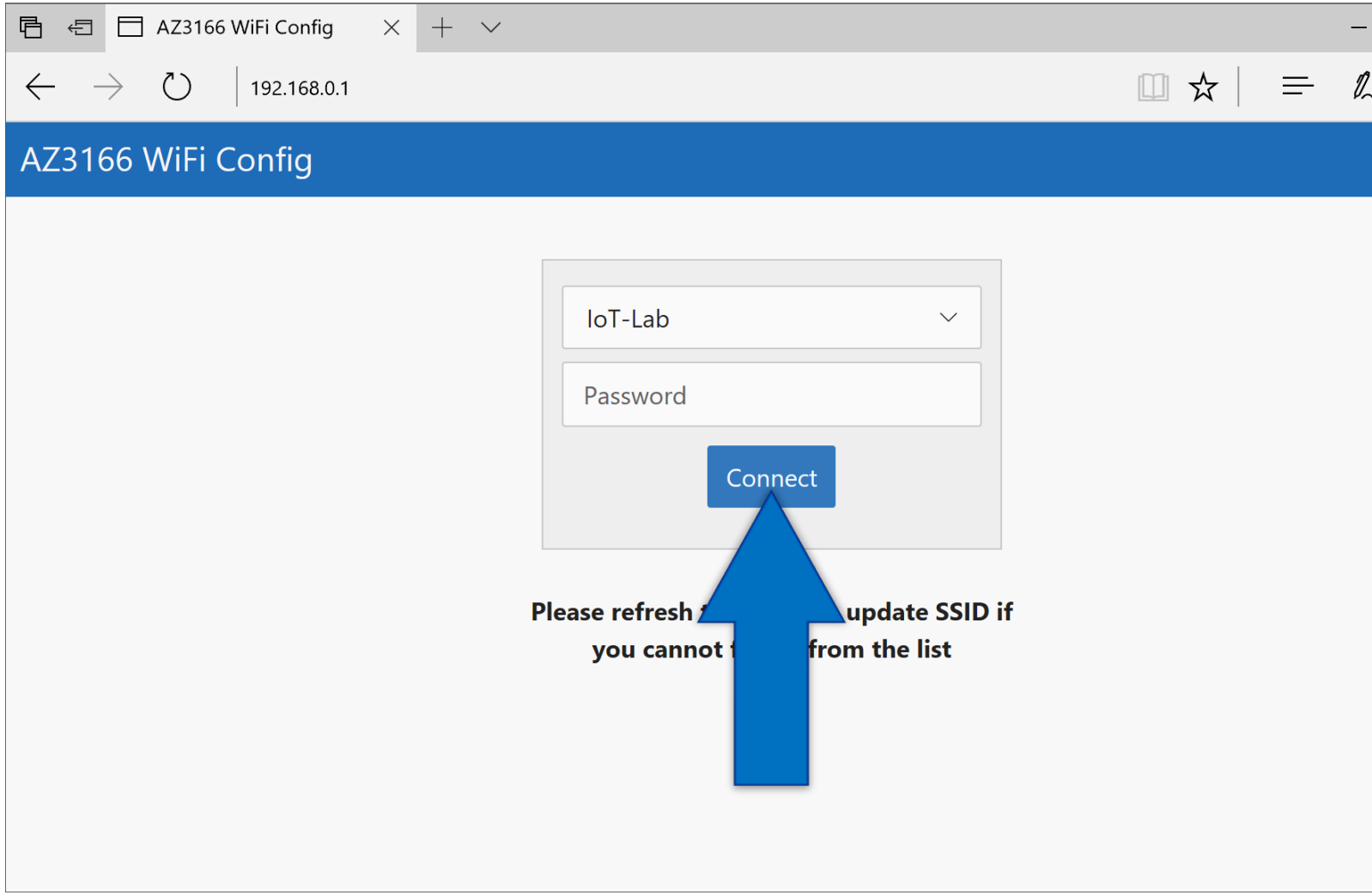
EXERCISE 5

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.

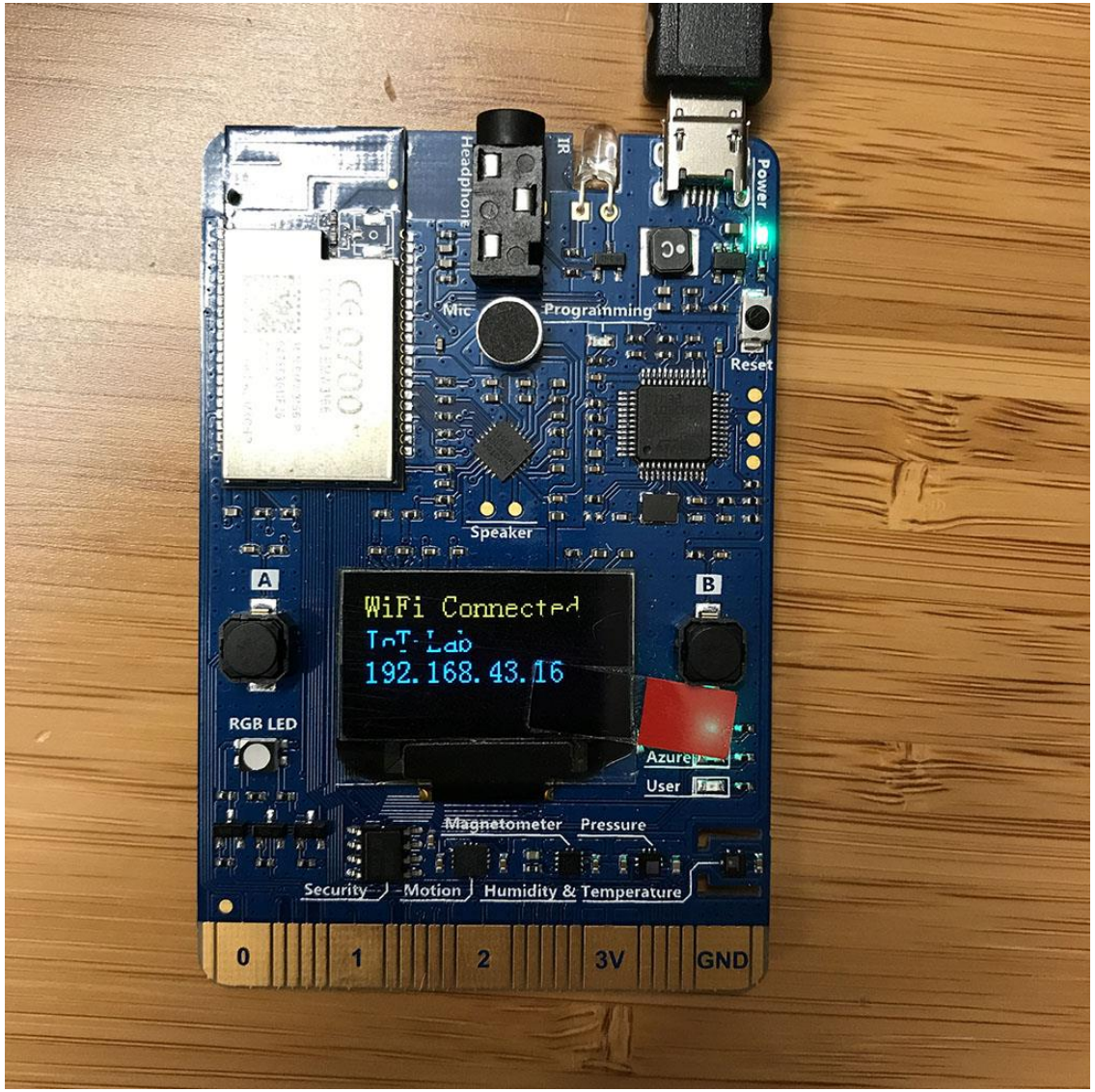
Click Connect.



EXERCISE 4

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.

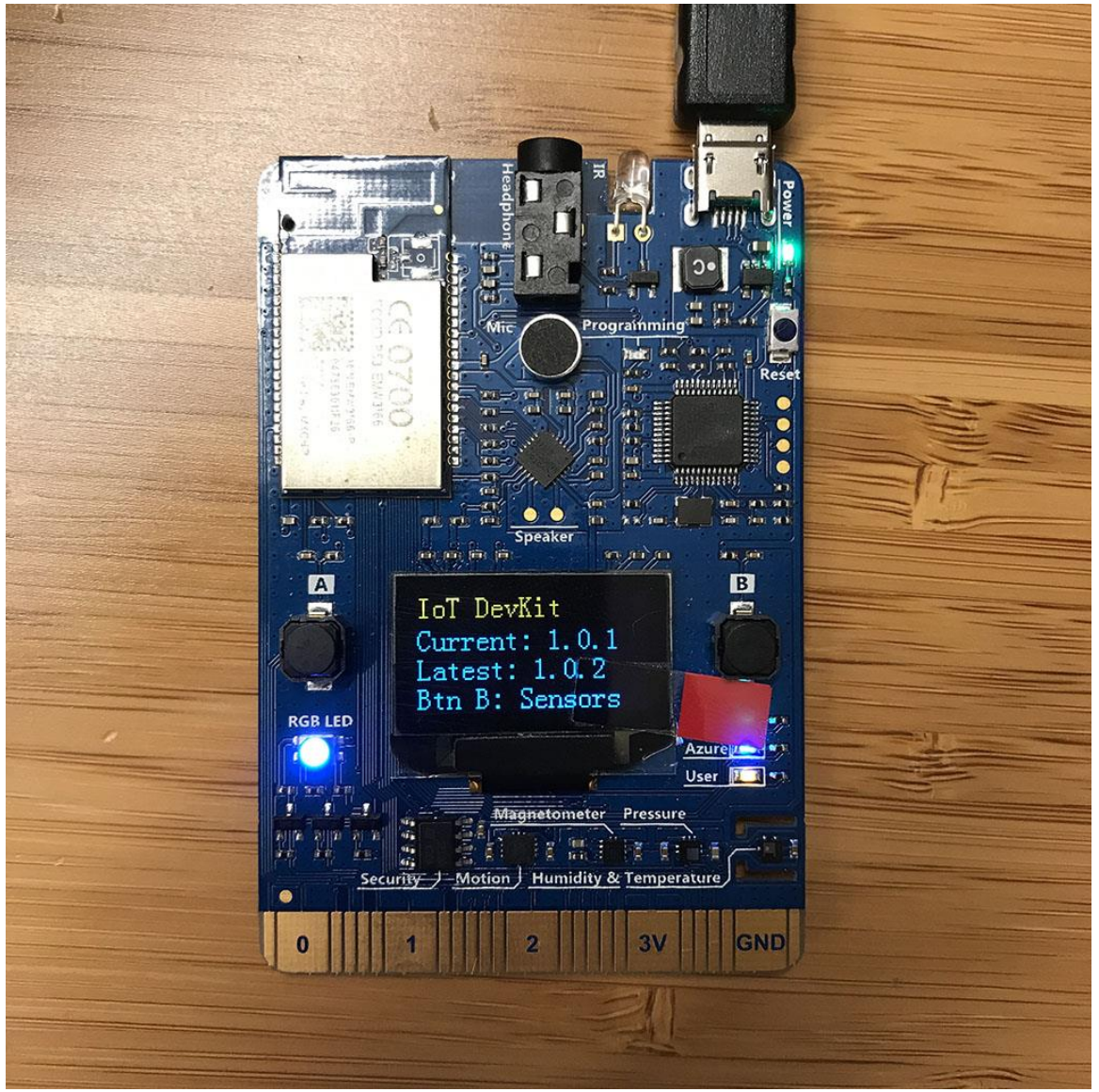


EXERCISE 5

Update firmware

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

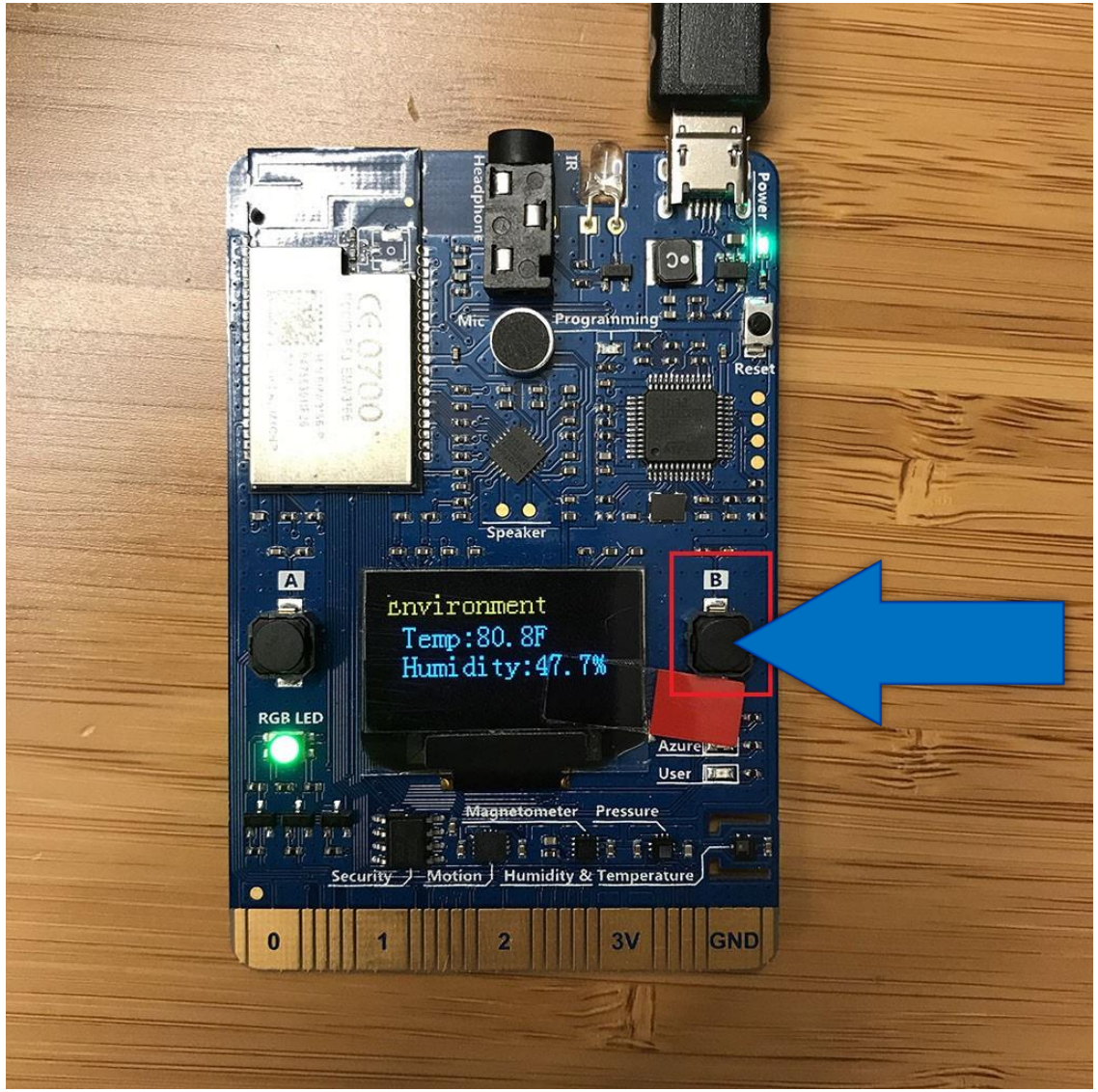
To upgrade, follow the Upgrade firmware guide.



EXERCISE 4

Check out Sensors

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



EXERCISE 5

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.

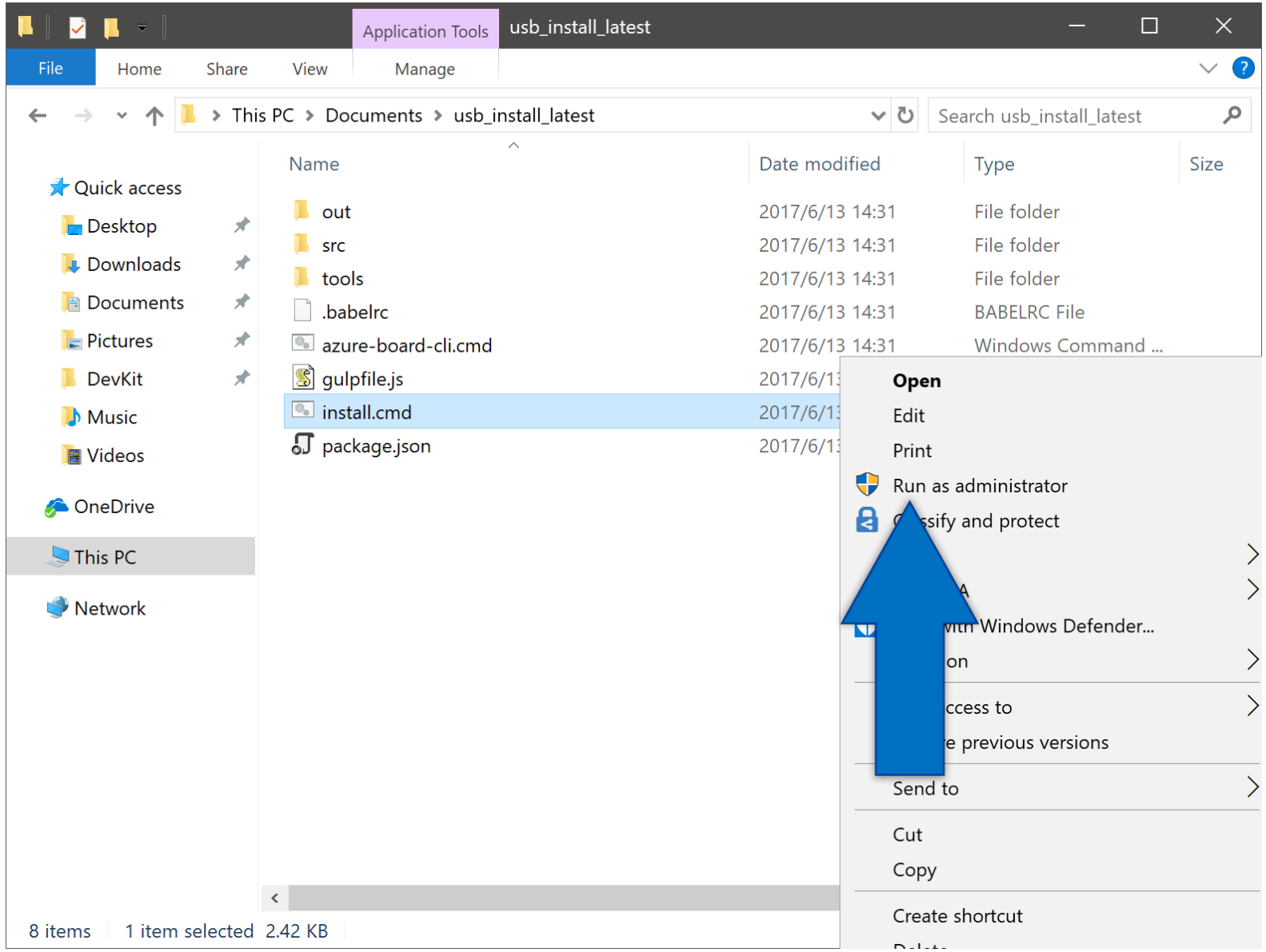
EXERCISE 4

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.



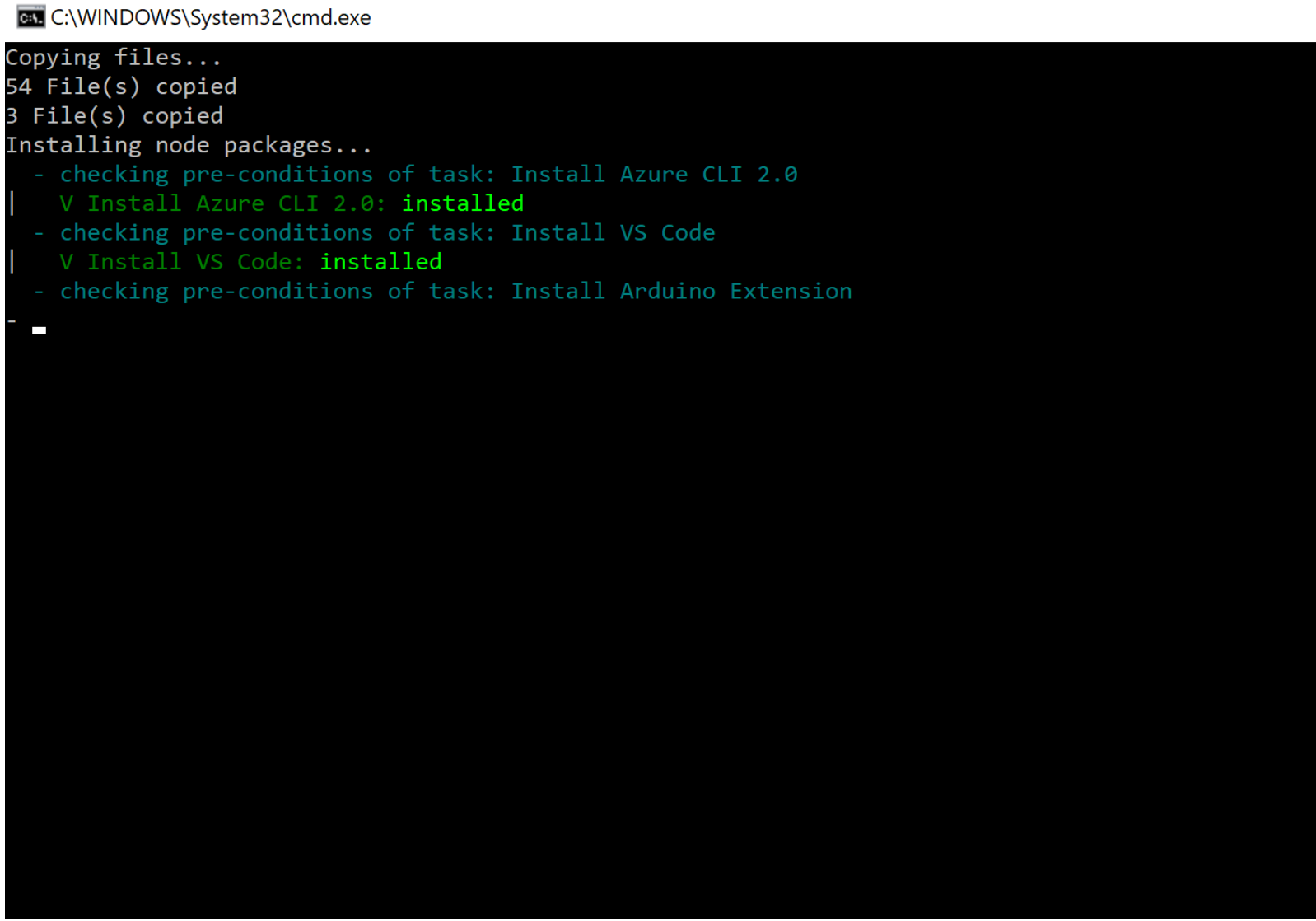
EXERCISE 5

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.



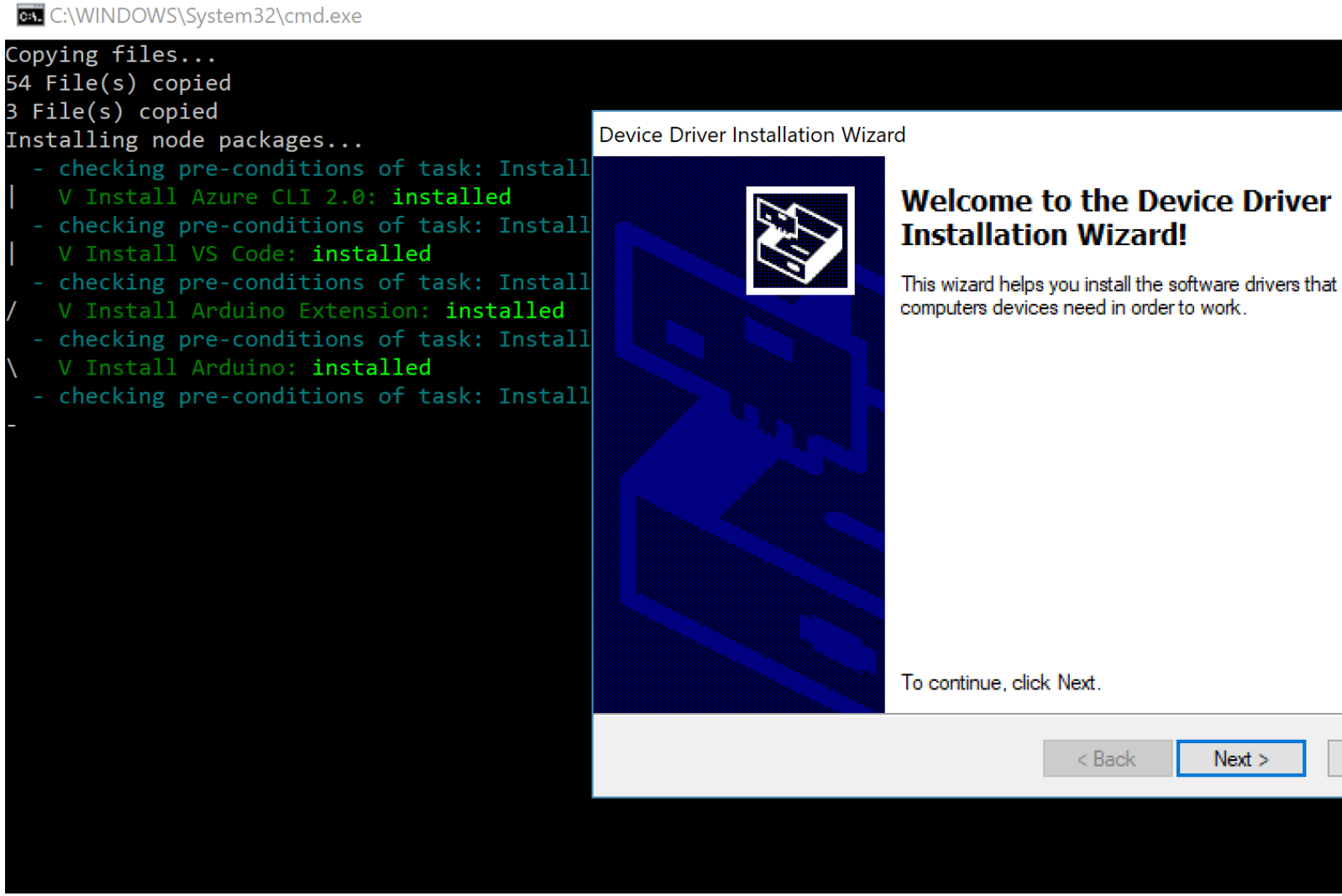
EXERCISE 4

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.



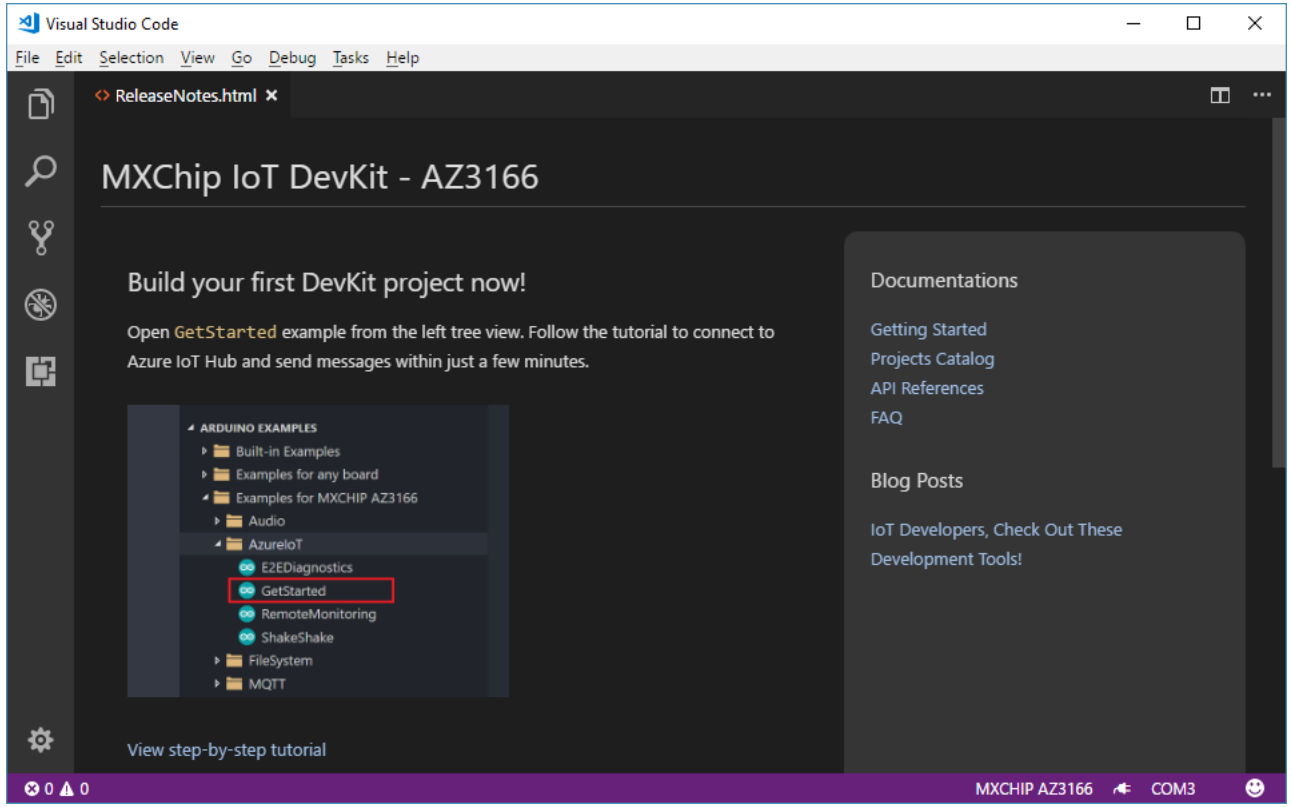
EXERCISE 5

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.



EXERCISE 4

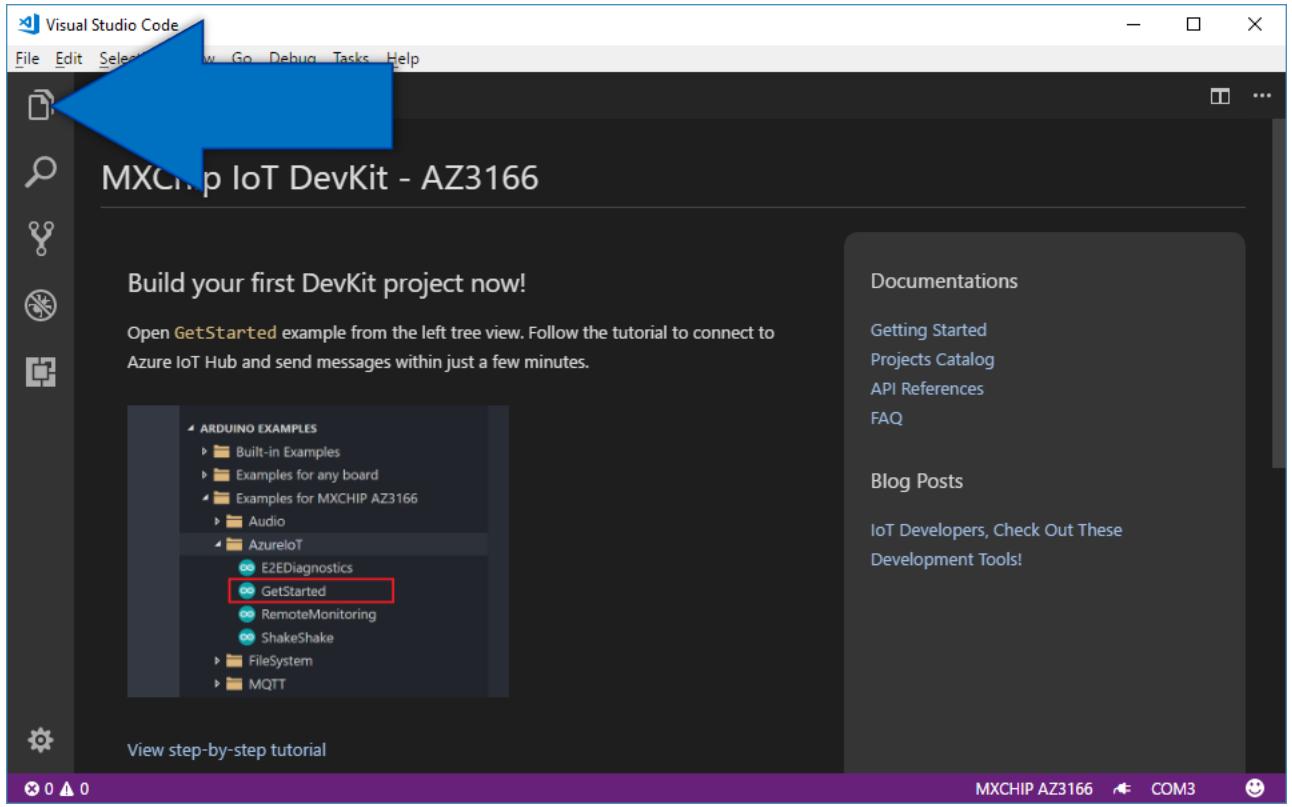
Arduino examples

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**.



EXERCISE 5

Cloud provision MXCHIP in IoT Hub

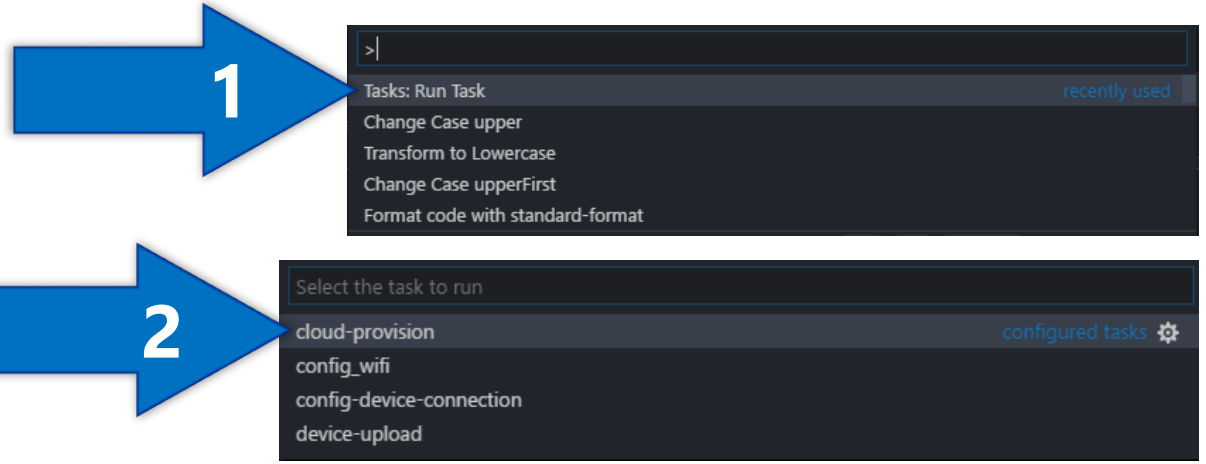
Let's start provisioning your MXCHIP IoT device to communicate with IoT Hub:

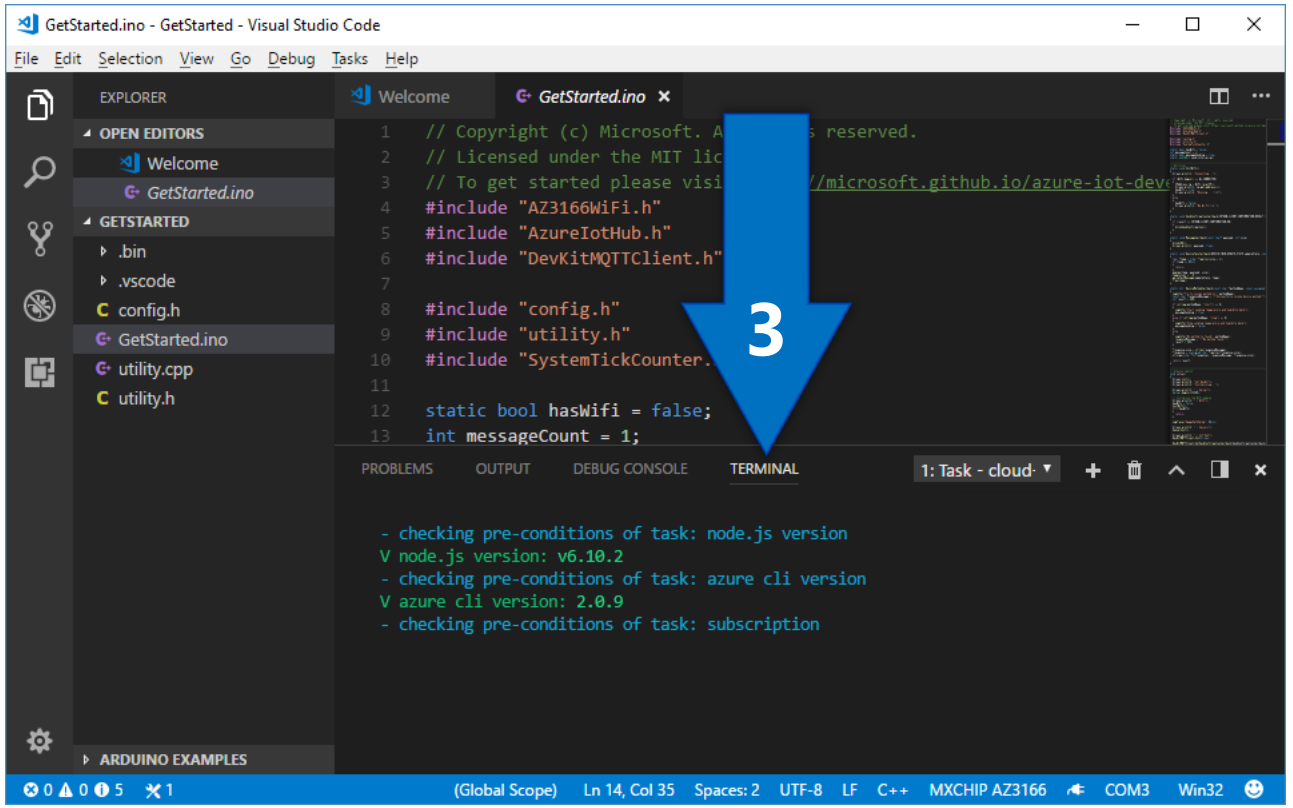
- `1` Run your task through Ctrl+Shift+P (macOS: Cmd+P) and select Tasks:Run Task.

- `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.



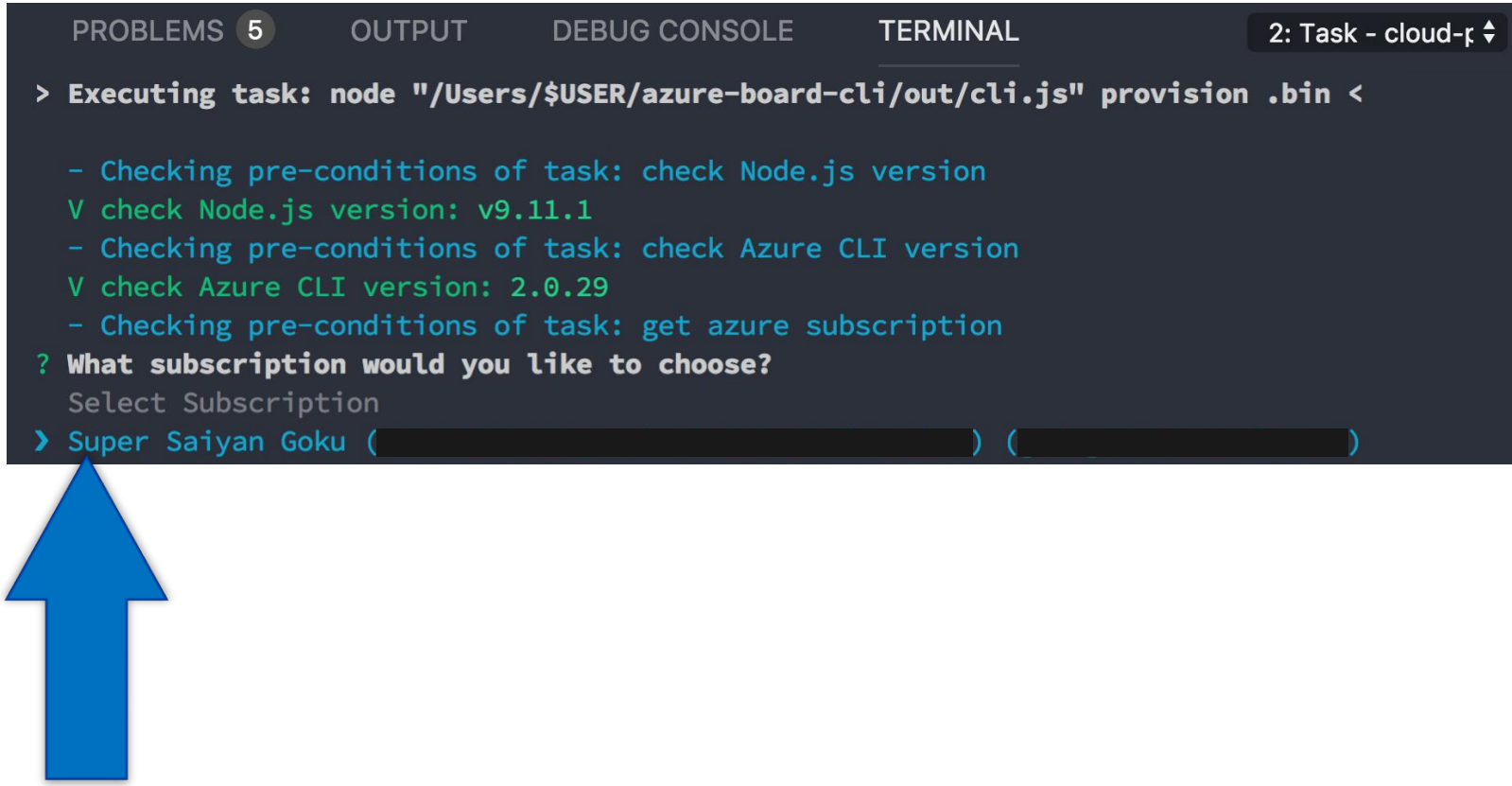


EXERCISE 4

Azure Subscription

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

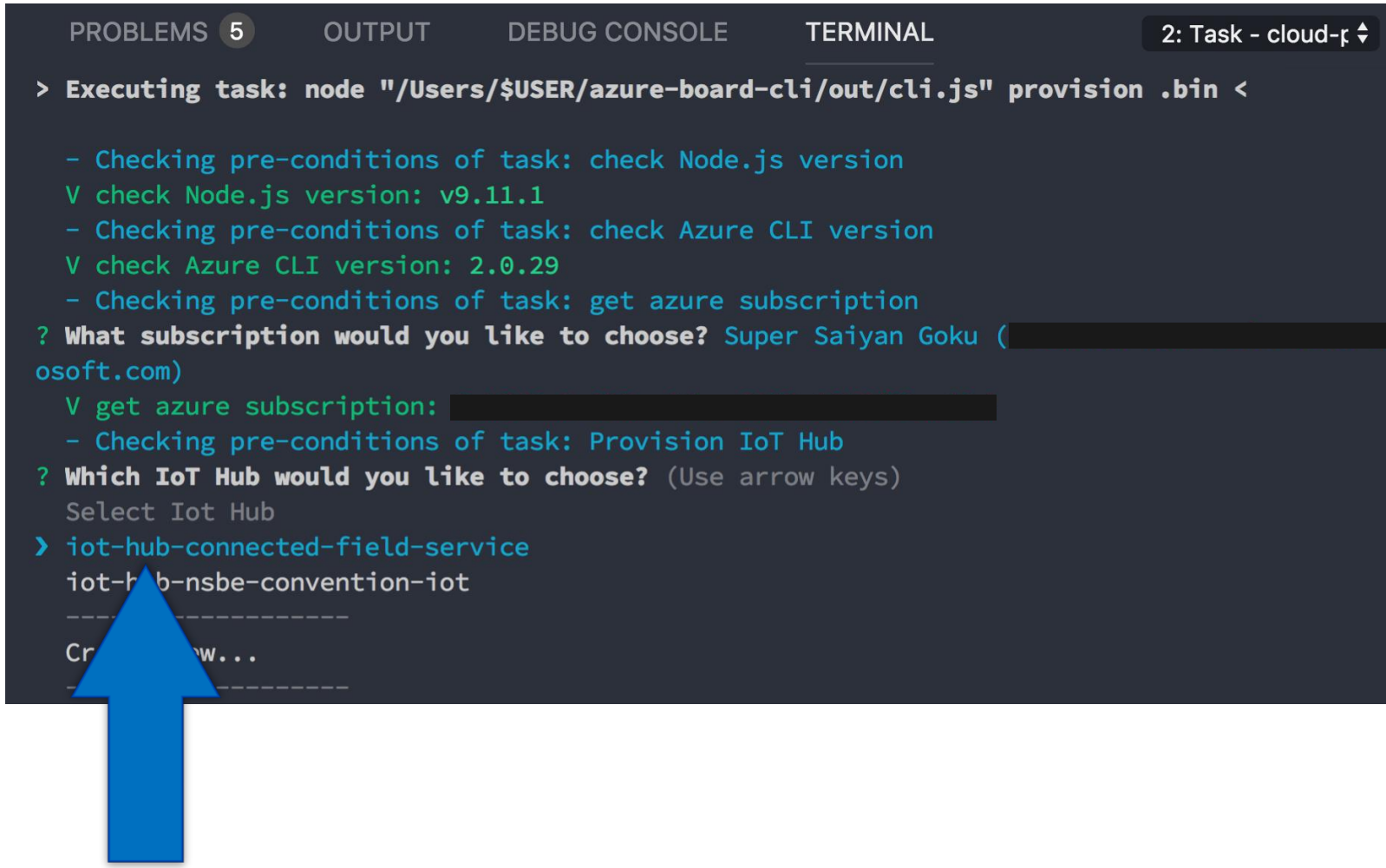
Use your arrow keys, select the subscription and click Enter/Return.



EXERCISE 5

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`.

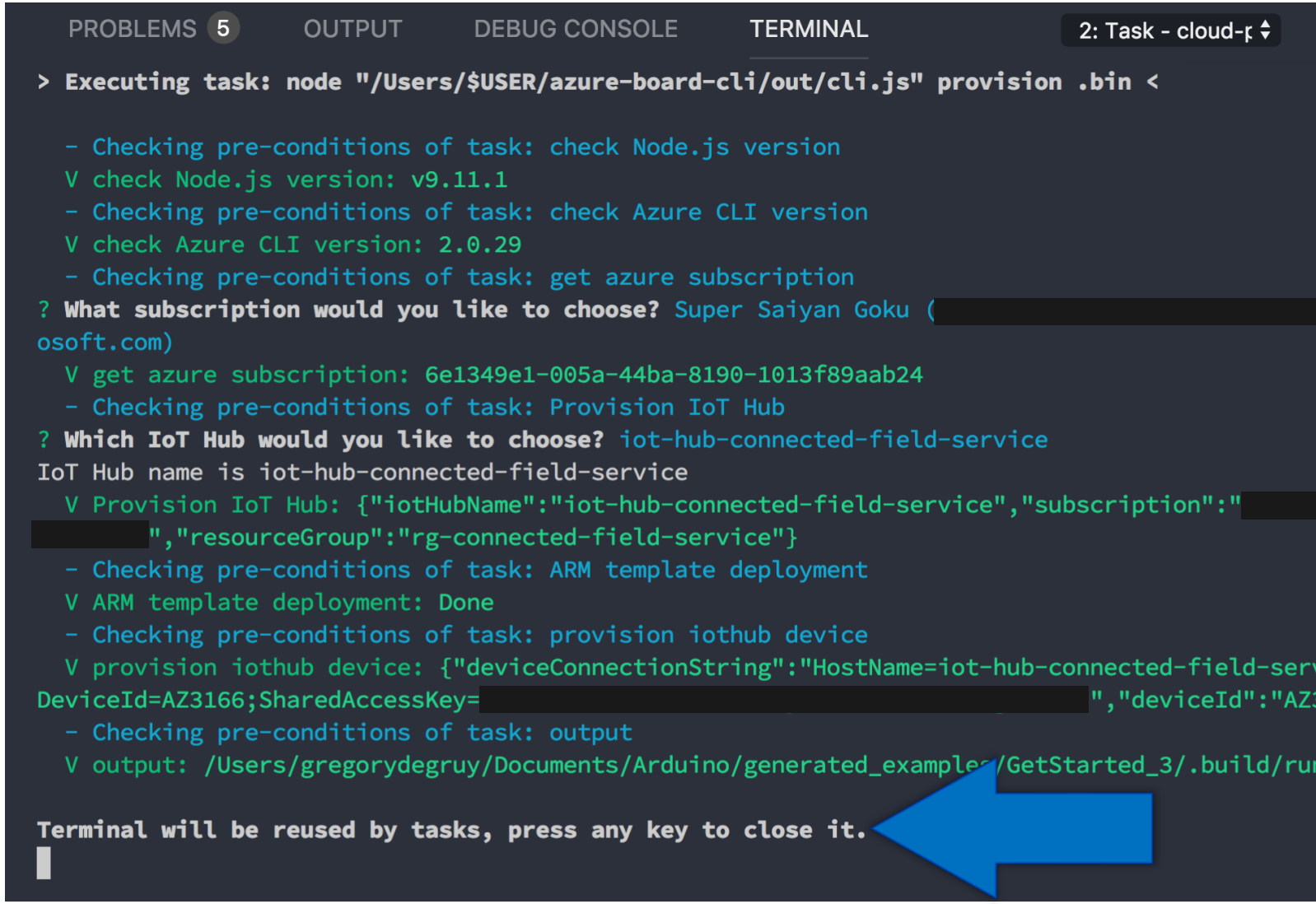


EXERCISE 4

Cloud provision complete

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

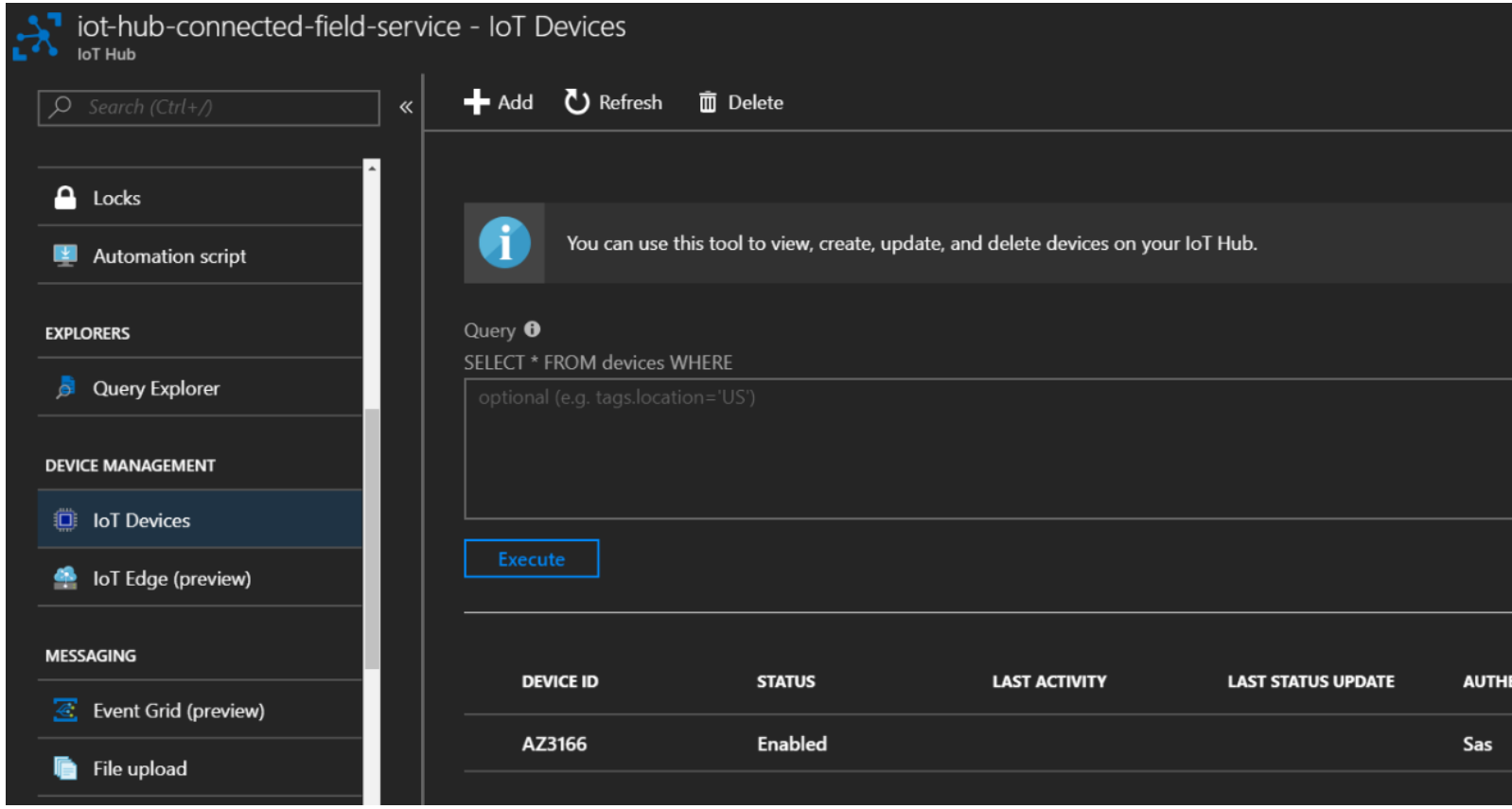
You will know the process is complete once you see the final output "Terminal will be reused by tasks, press any key to close it"



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.



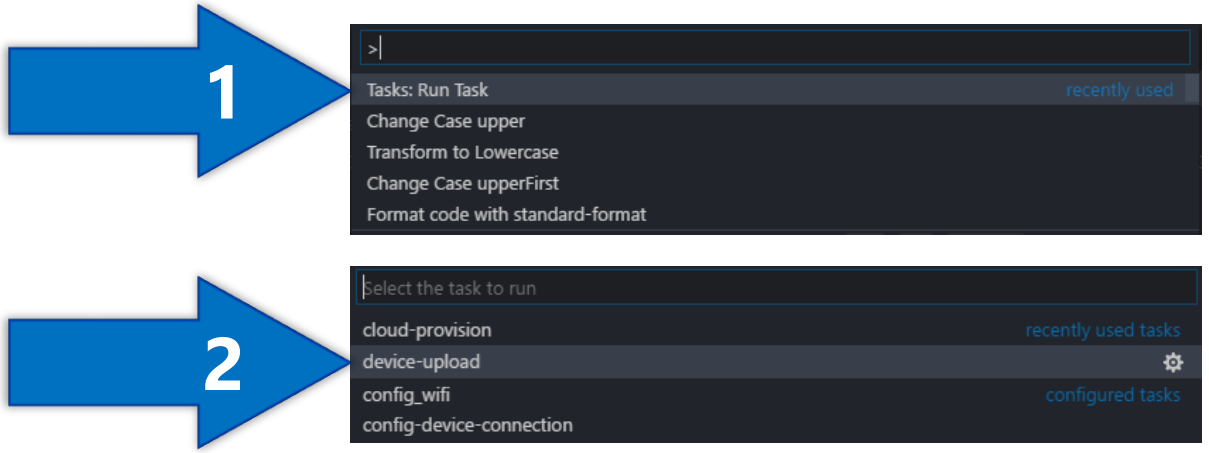
EXERCISE 4

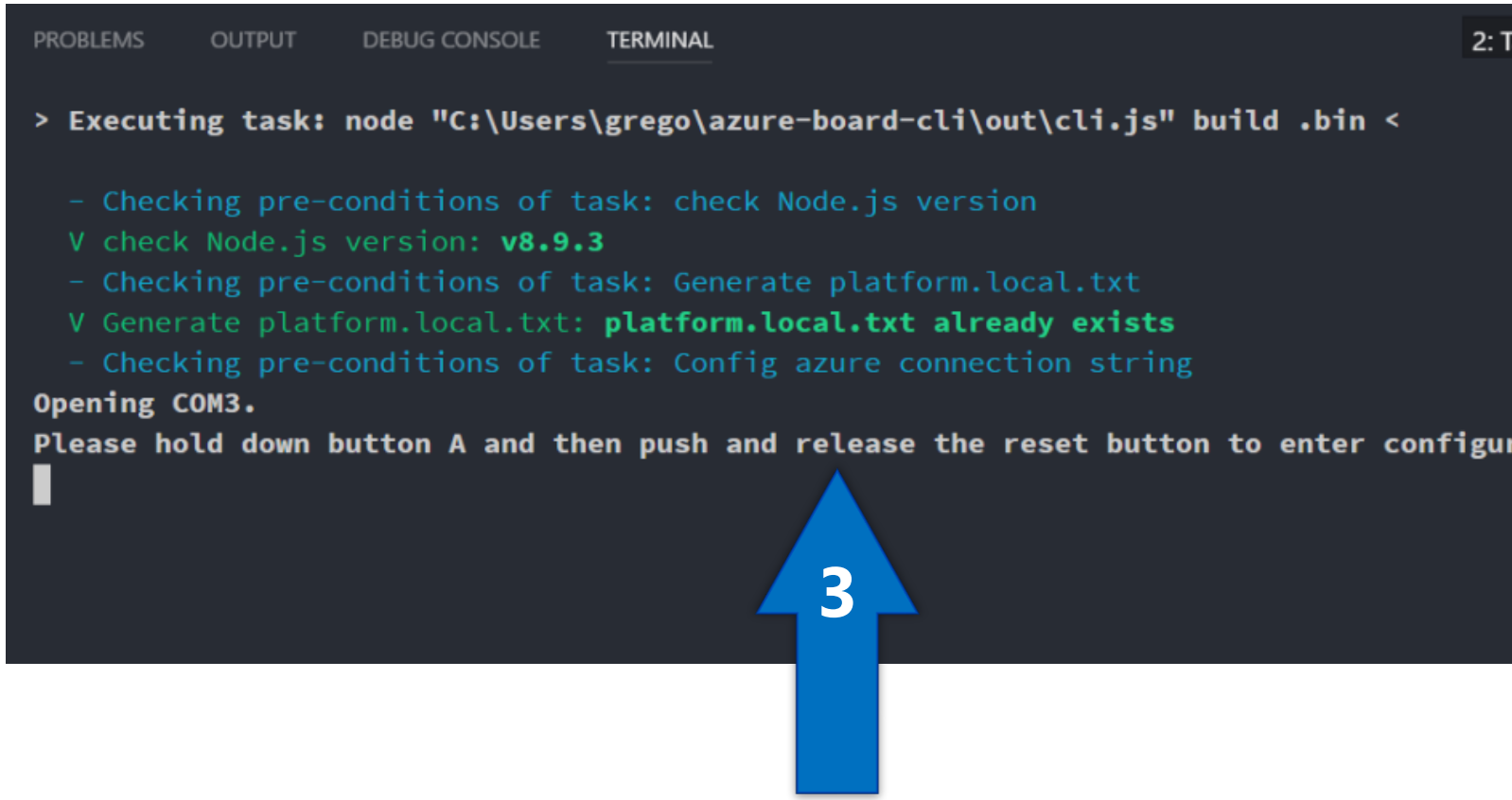
Build and upload the Arduino Code

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.





EXERCISE 5

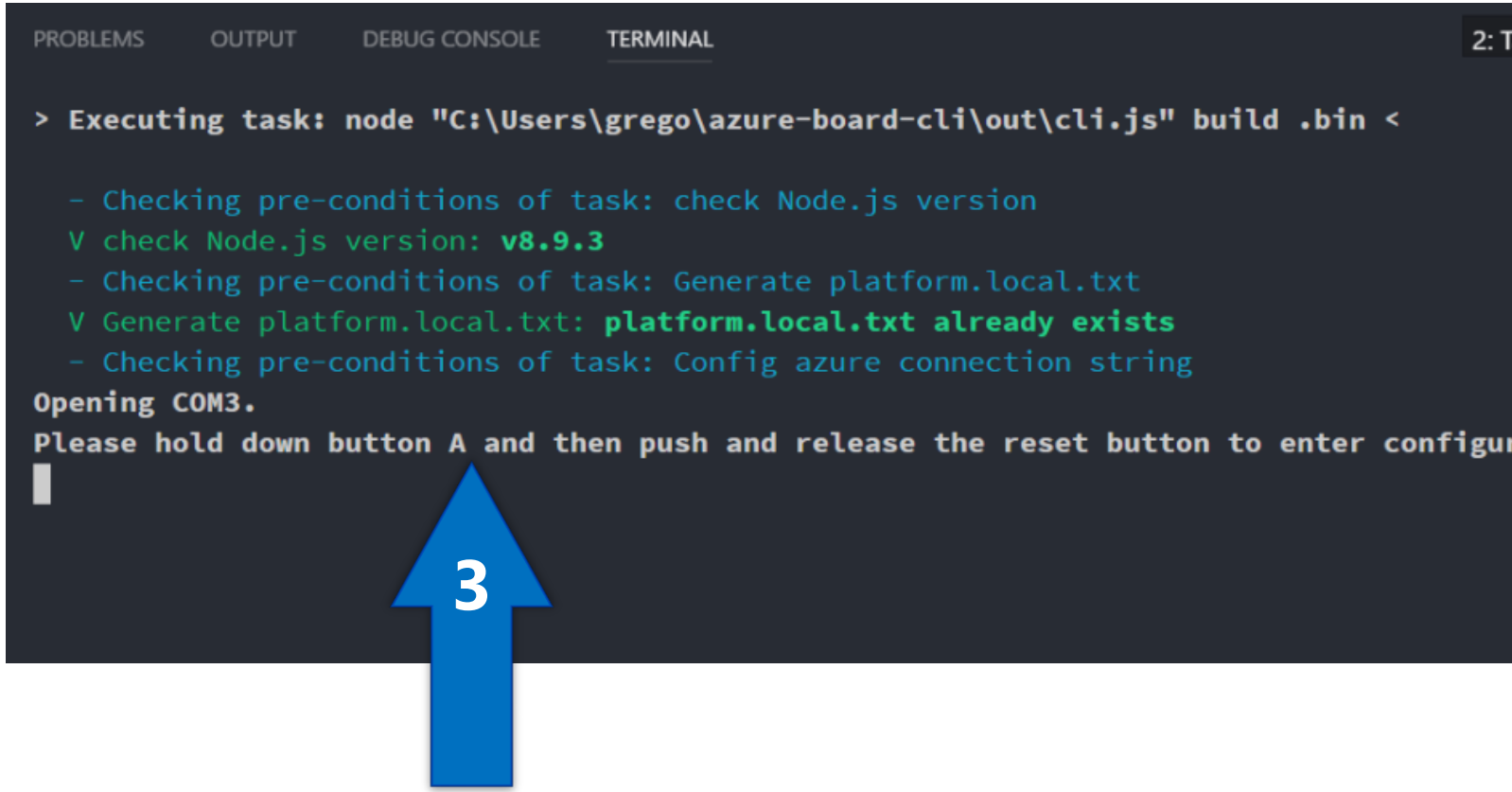
MXCHIP Configuration mode

-`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

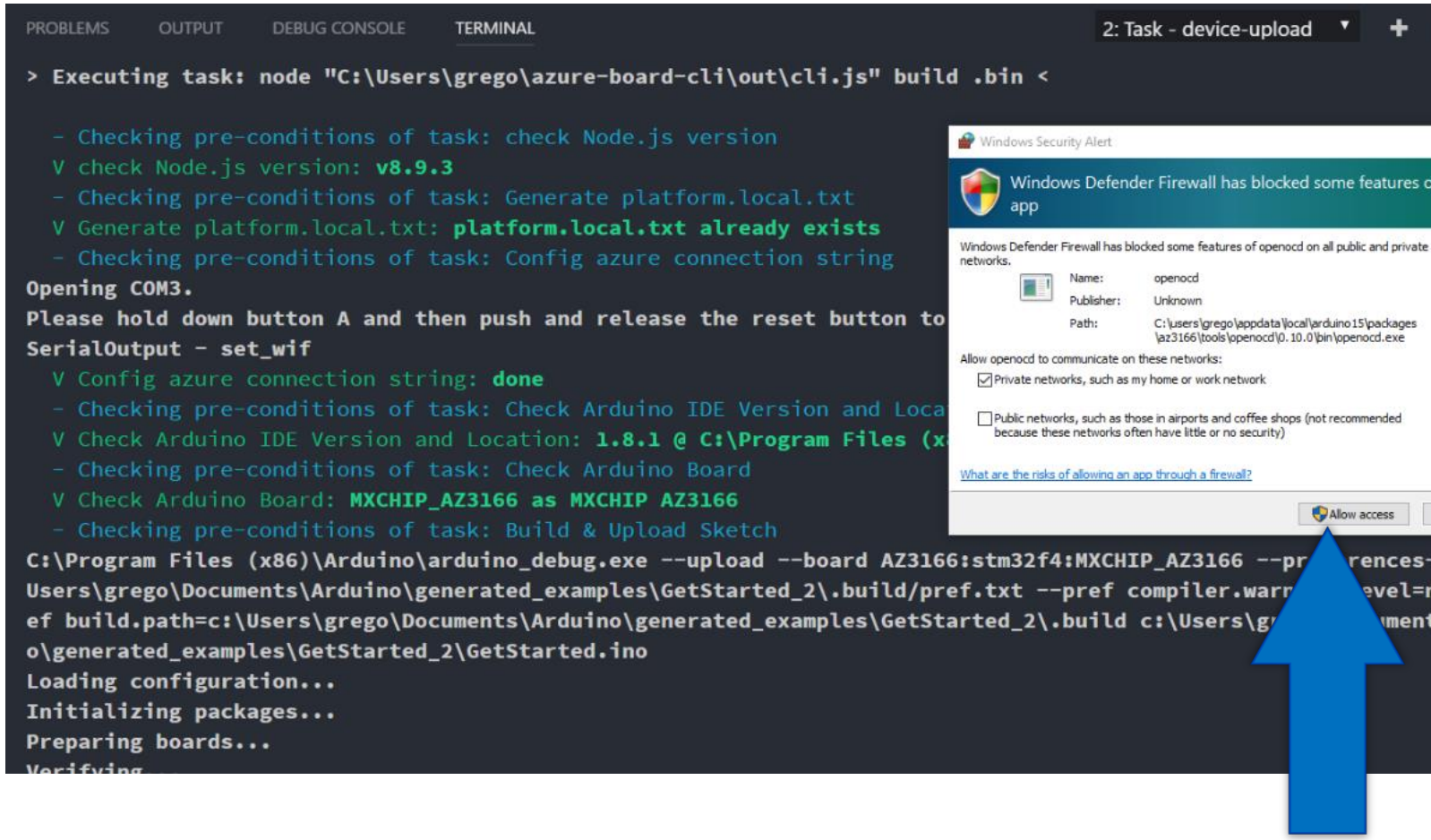
Occasionally, you get error "Error: AZ3166: Unknown package". This is due to the board package index is not refreshed. Check this FAQ steps to solve it.



EXERCISE 4

Firewall

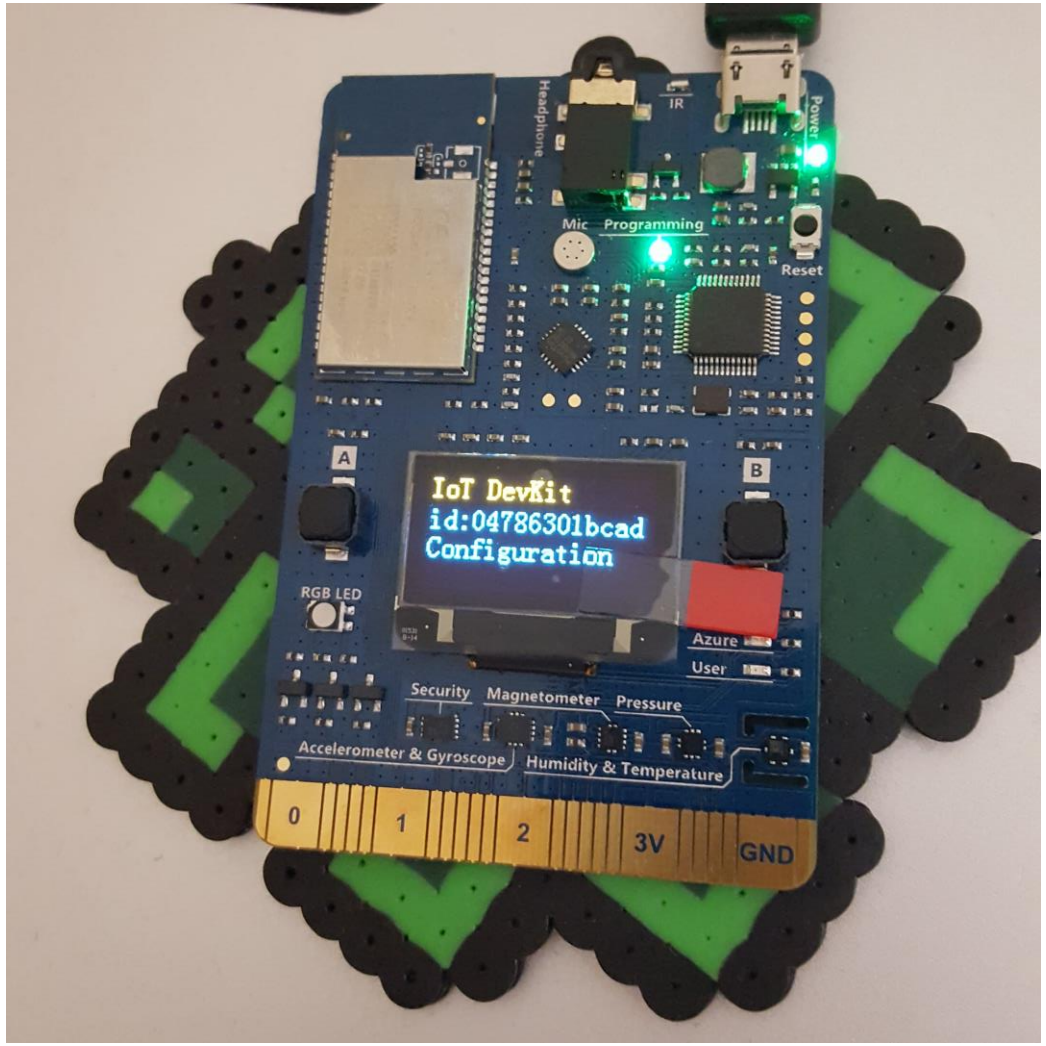
On first run of the **GetStarted.ino** code, you may be prompted to allow Firewall access during the upload process, click Allow access.



EXERCISE 5

Deploying code

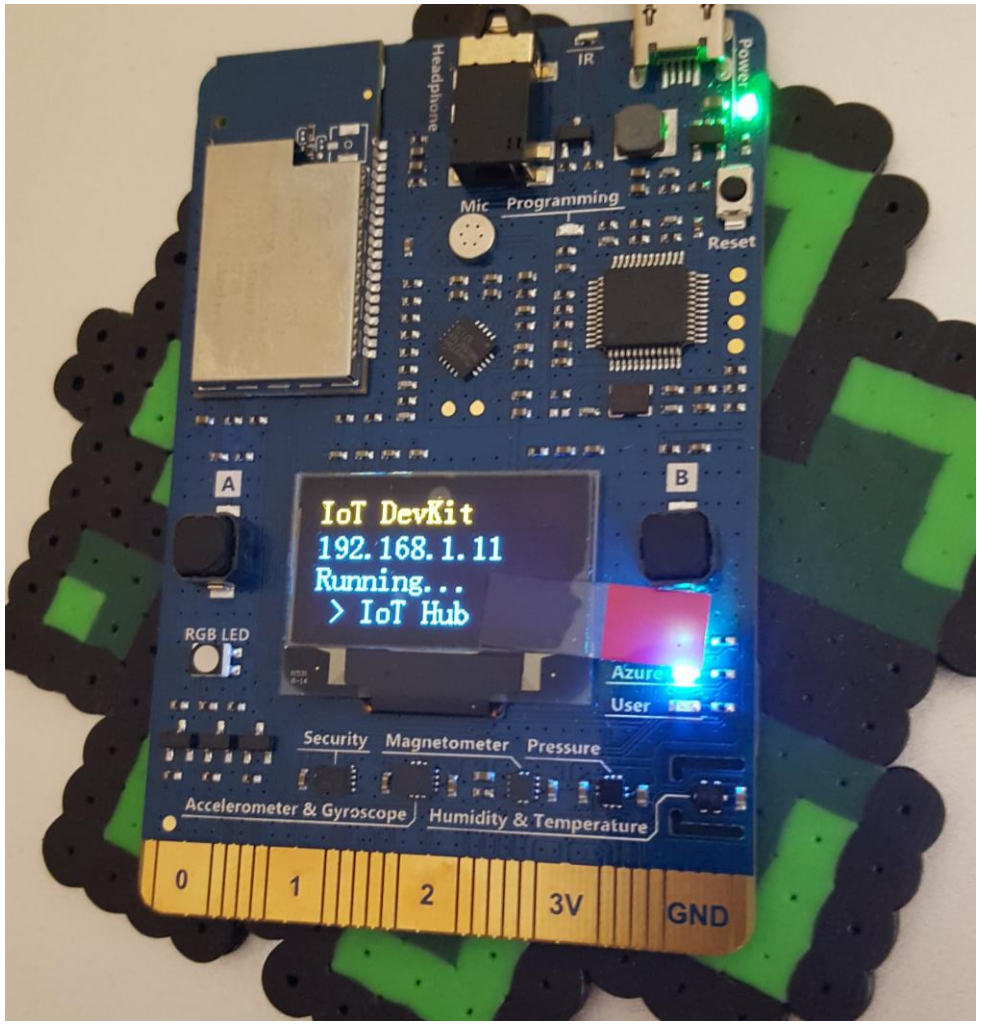
Your device should be in configuration mode during this process briefly.



EXERCISE 4

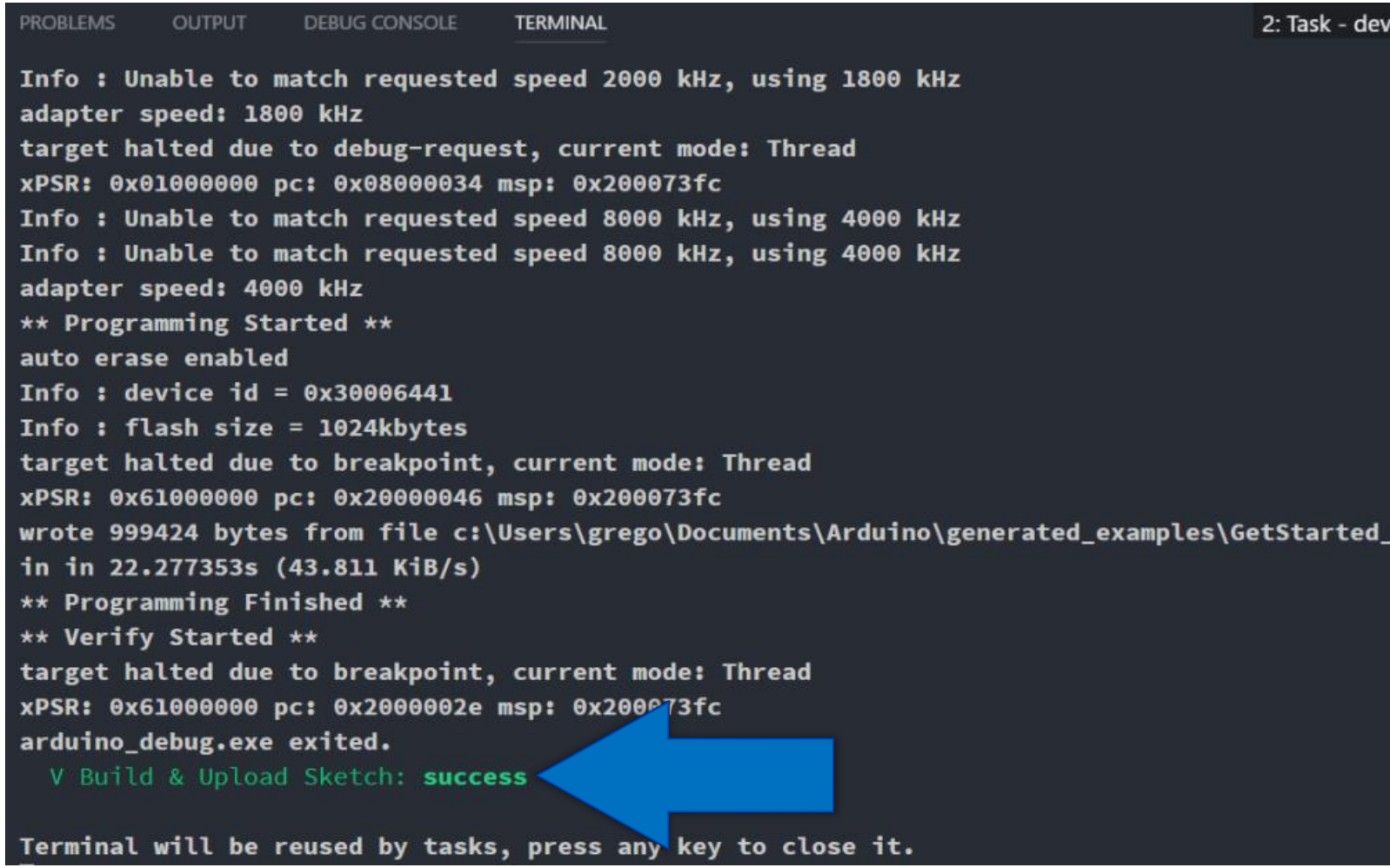
Data sending to IoT Hub

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!



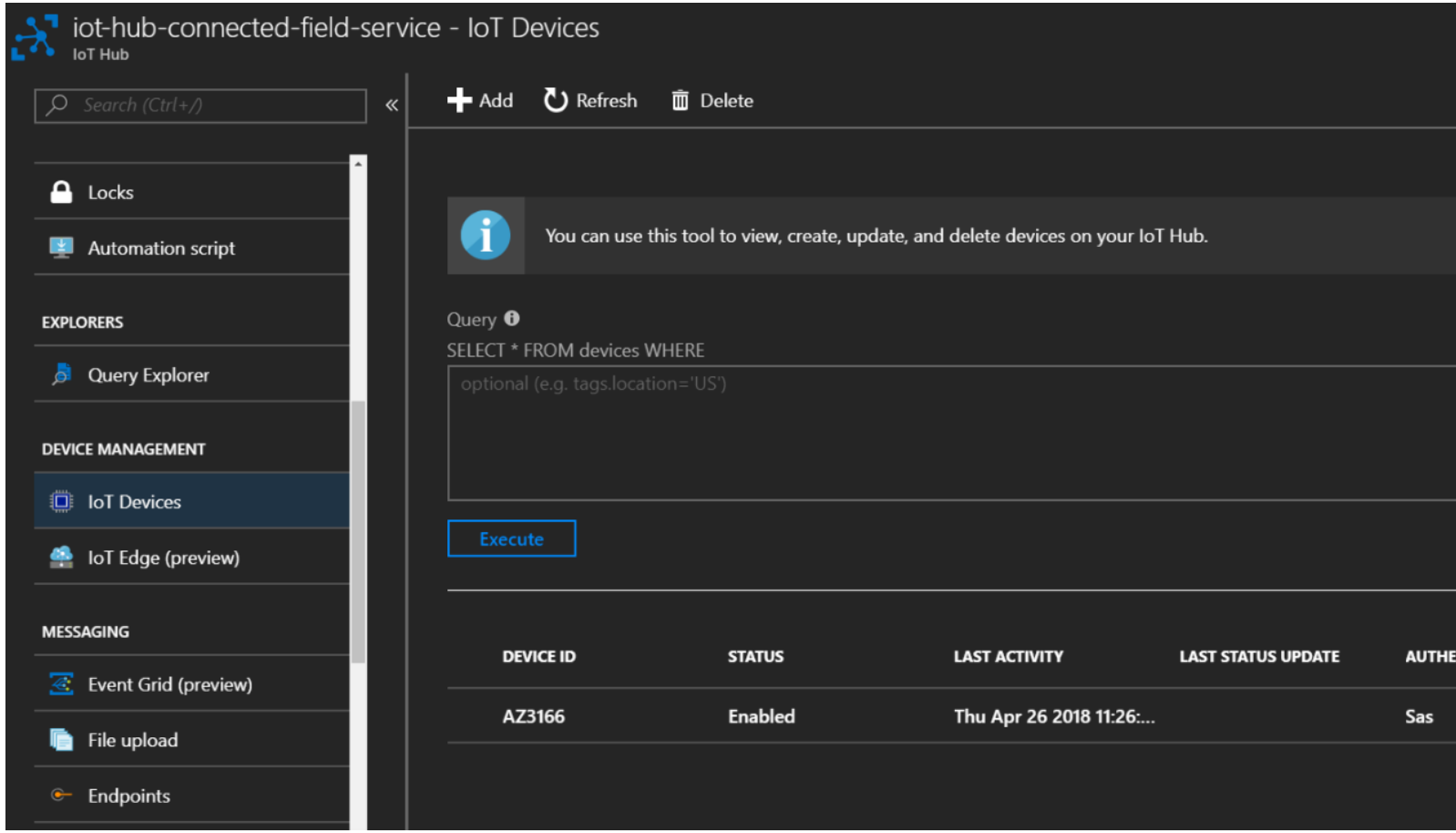
Confirm success

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



Device activity IoT Hub

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



EXERCISE 5

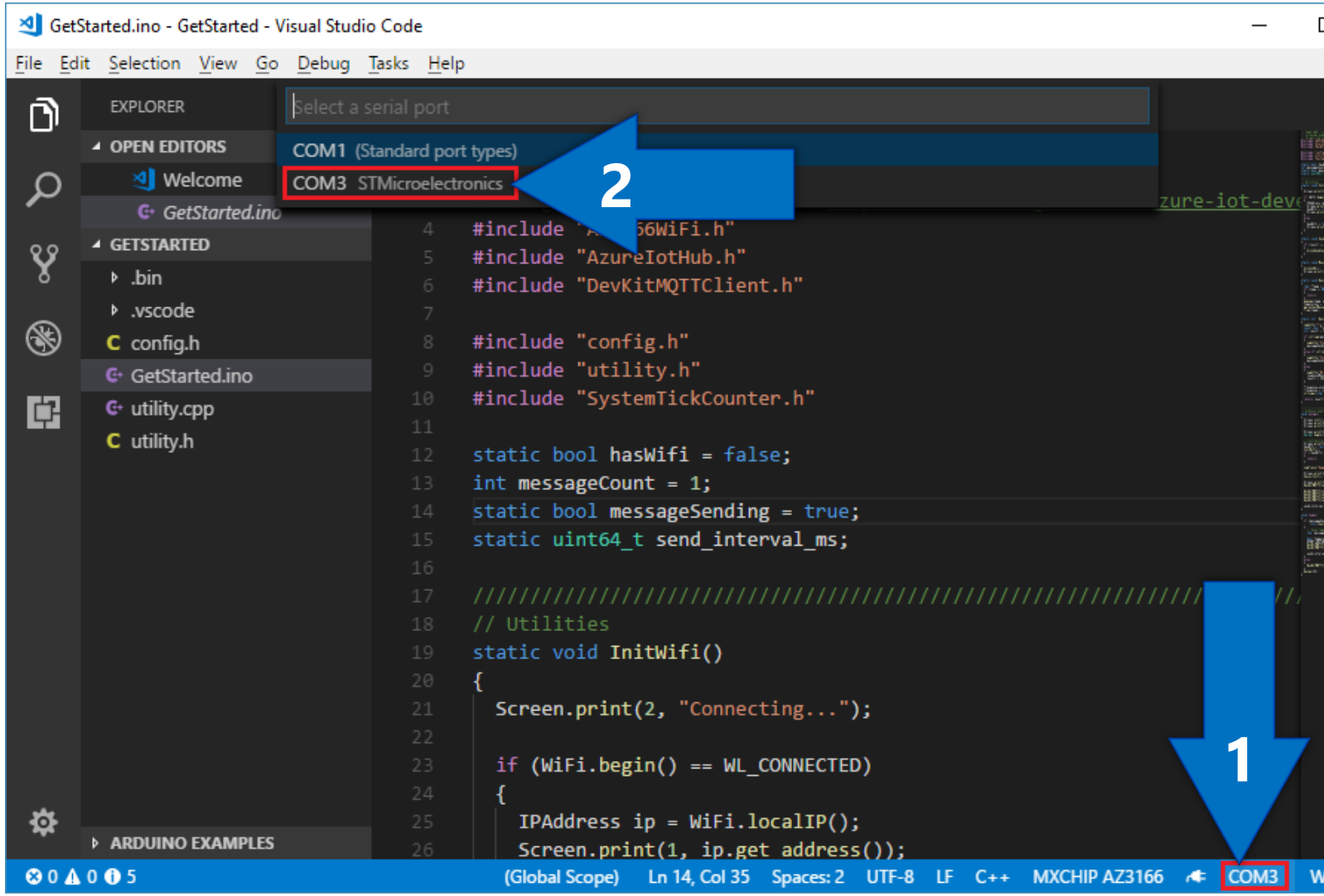
Viewing and testing data upload

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

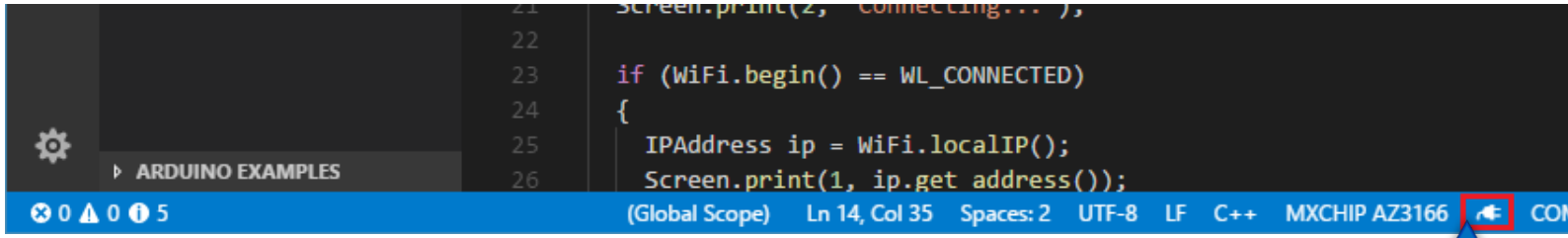


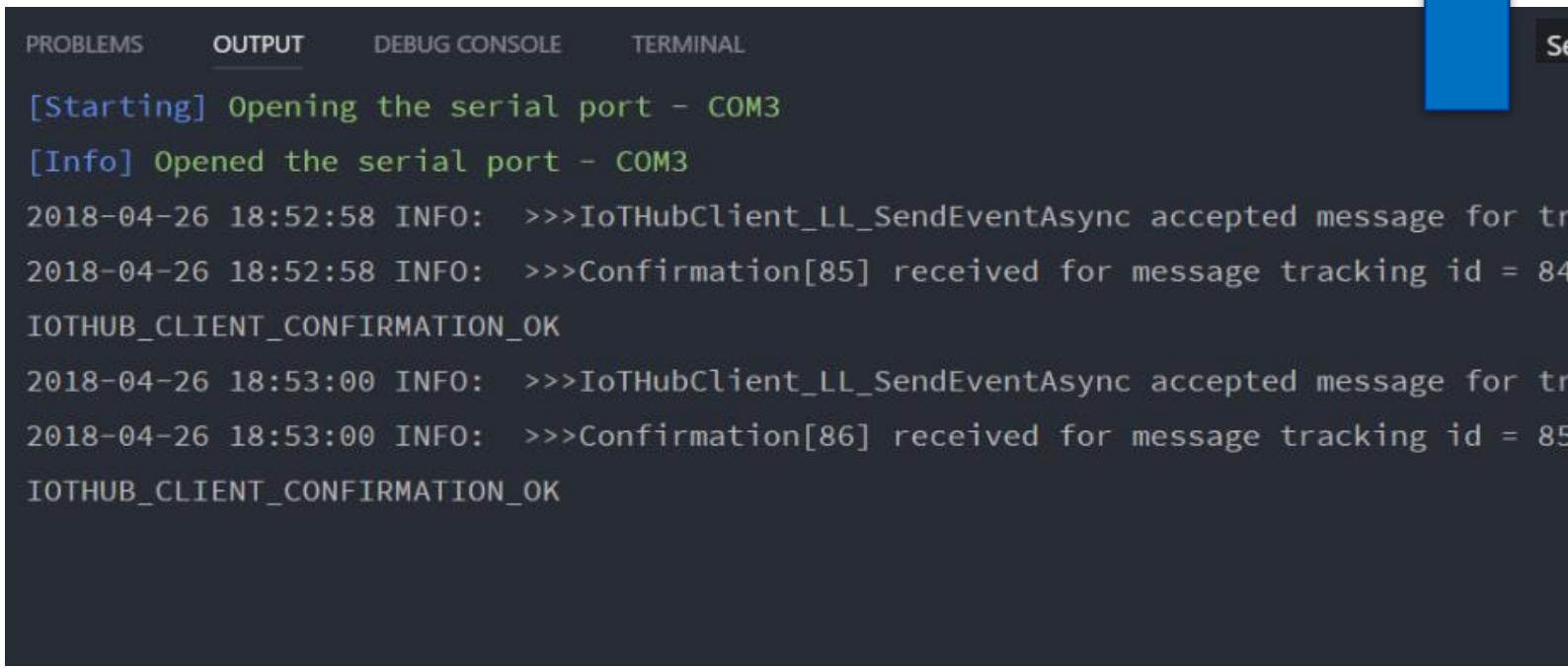
EXERCISE 4

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.

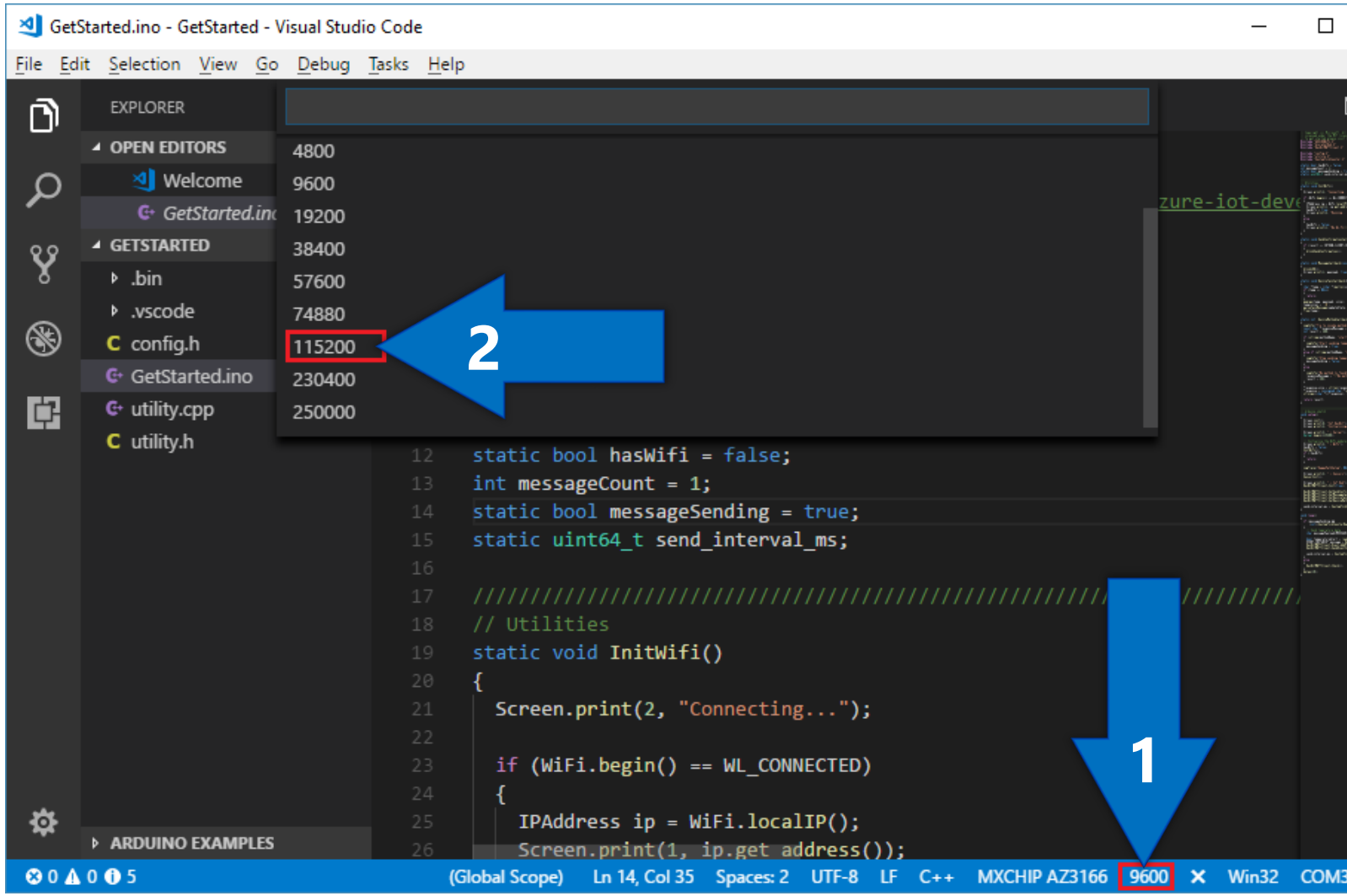




Data upload rate

- `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.



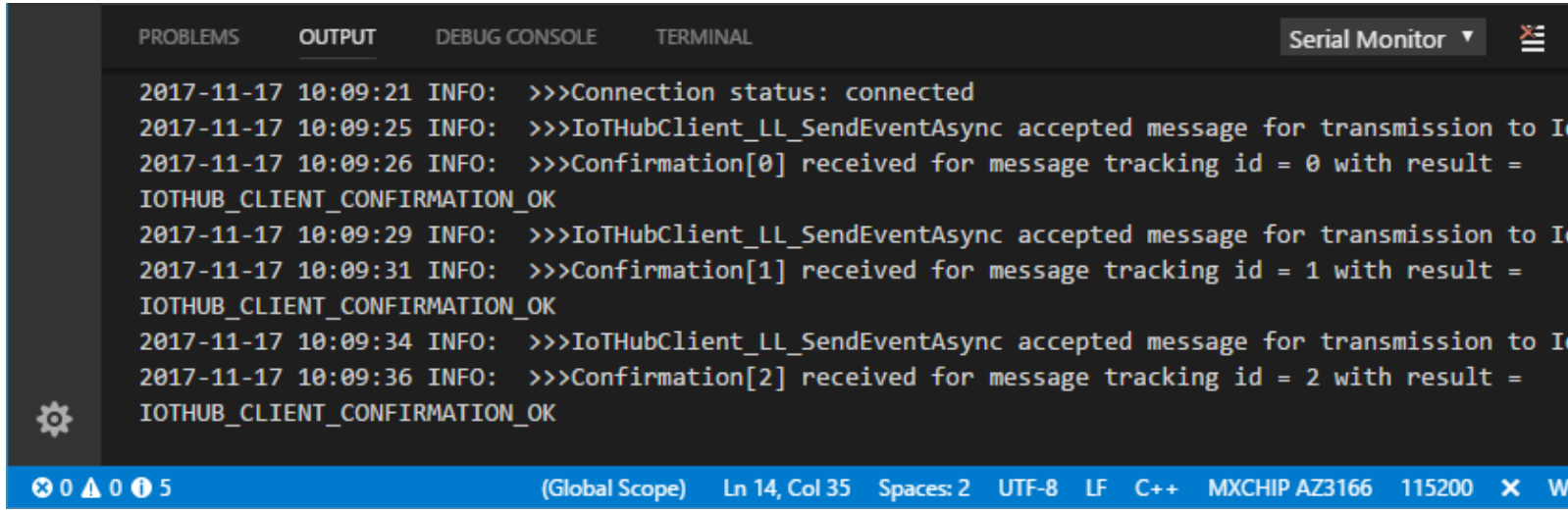
EXERCISE 4

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 as captured sensor data is sent to the IoT Hub.

- The LED on MXChip IoT DevKit is blinking.



EXERCISE 5

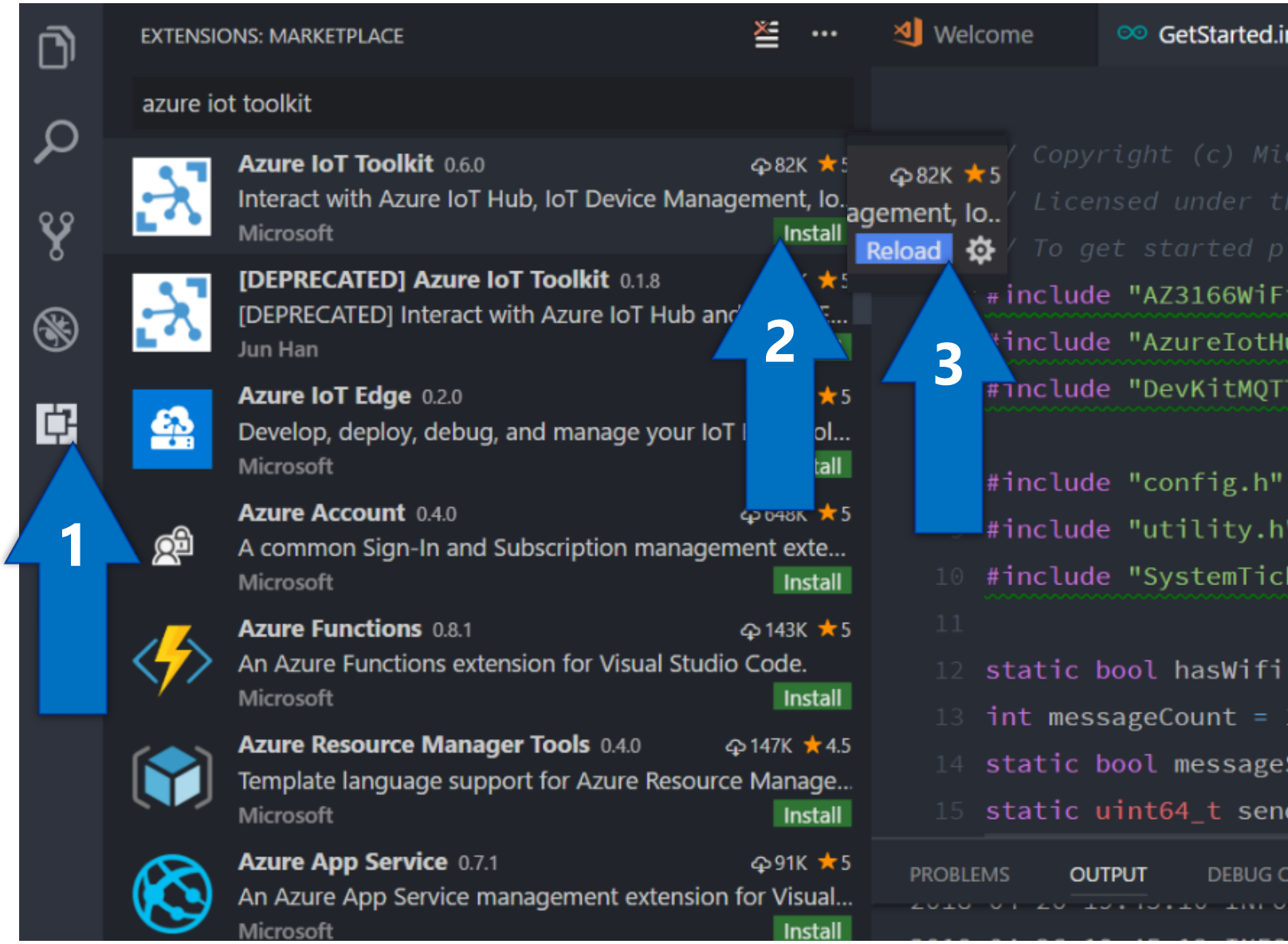
VS Code IoT extension

In VS Code

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

-`2` Click install

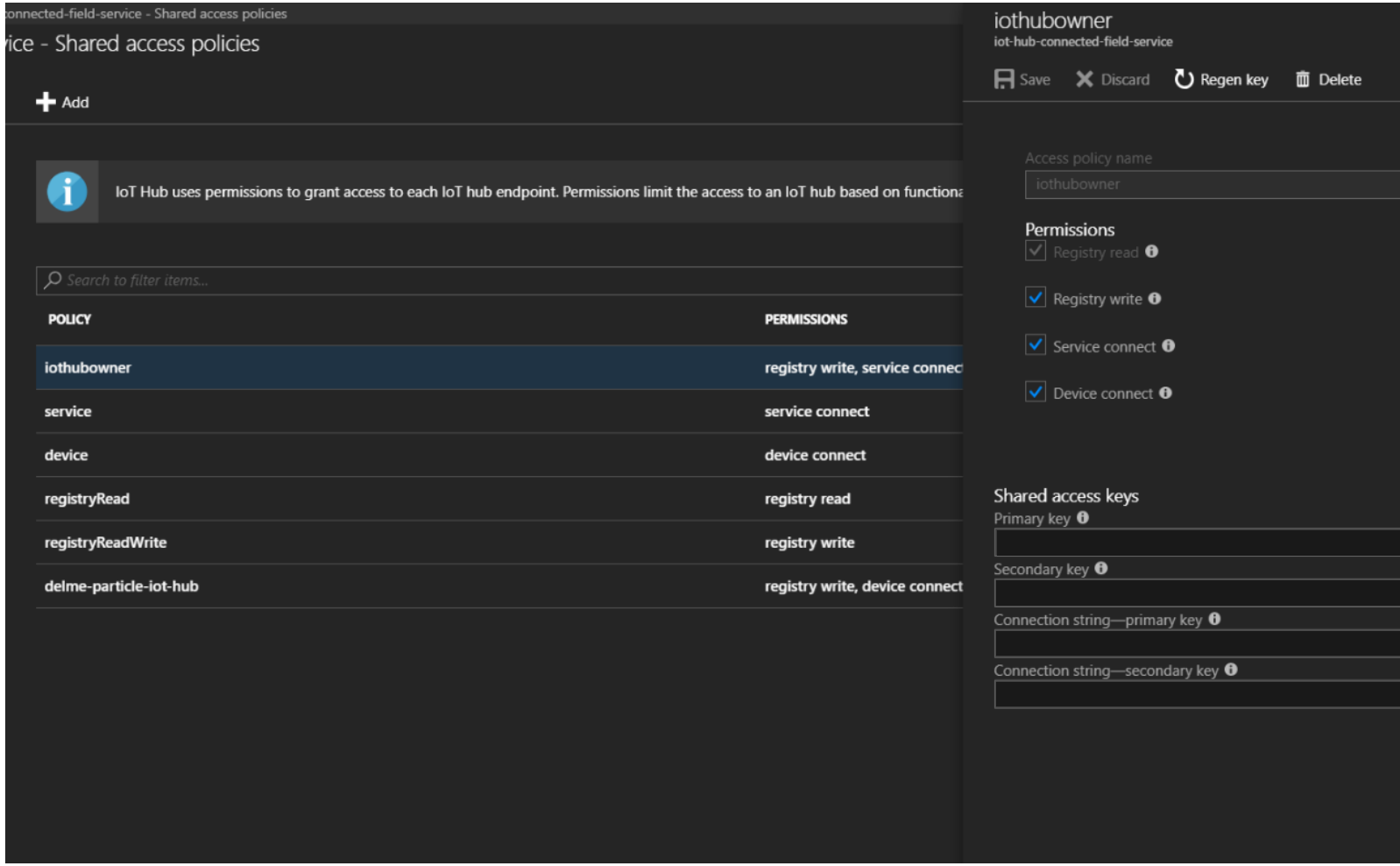
-`3` Click Reload window once installed. Once VS Code reloads make sure to open the **`GetStarted.ino`** up again.



EXERCISE 4

IoT Hub connection string

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



EXERCISE 5

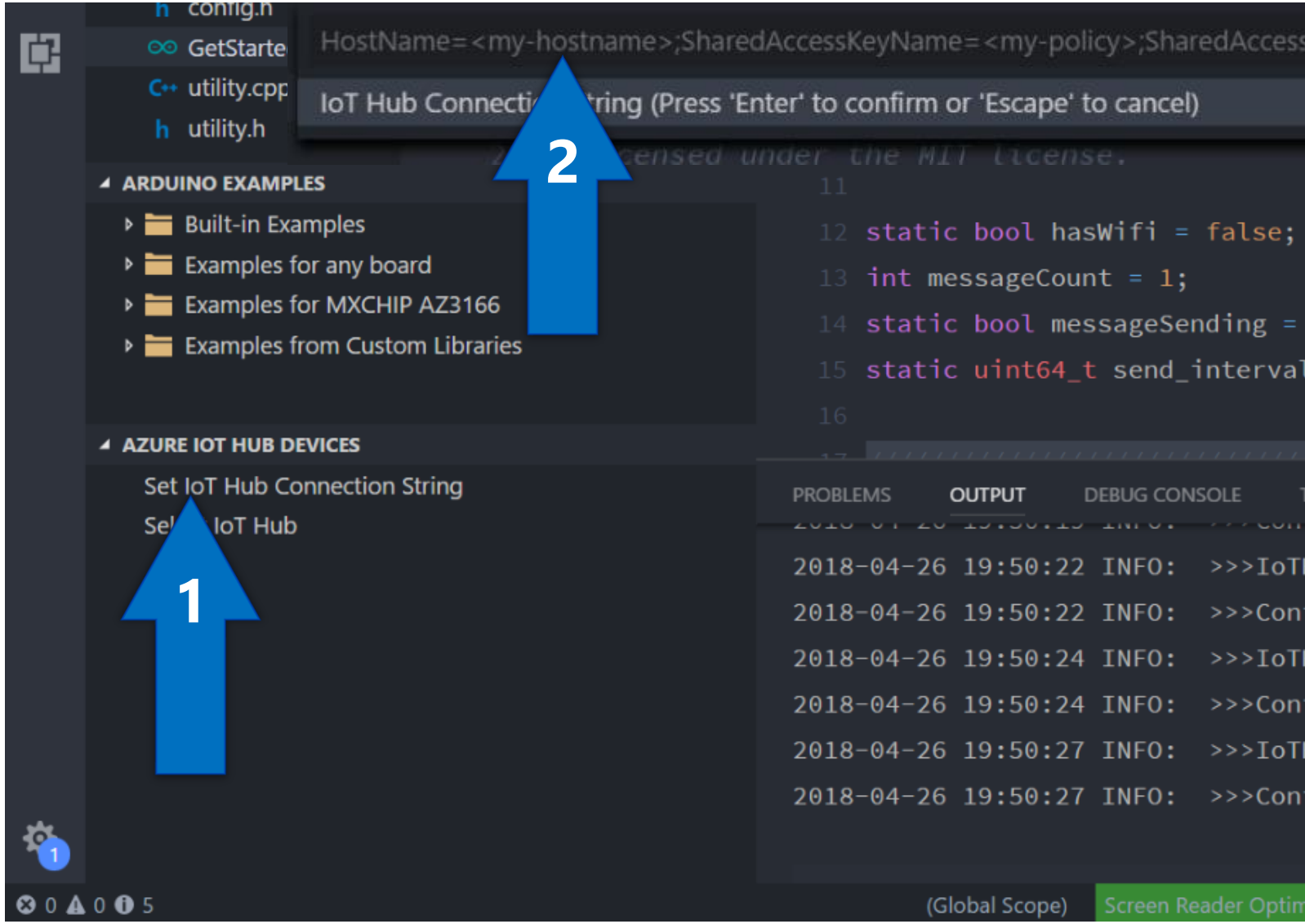
Connection string in VS Code

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”

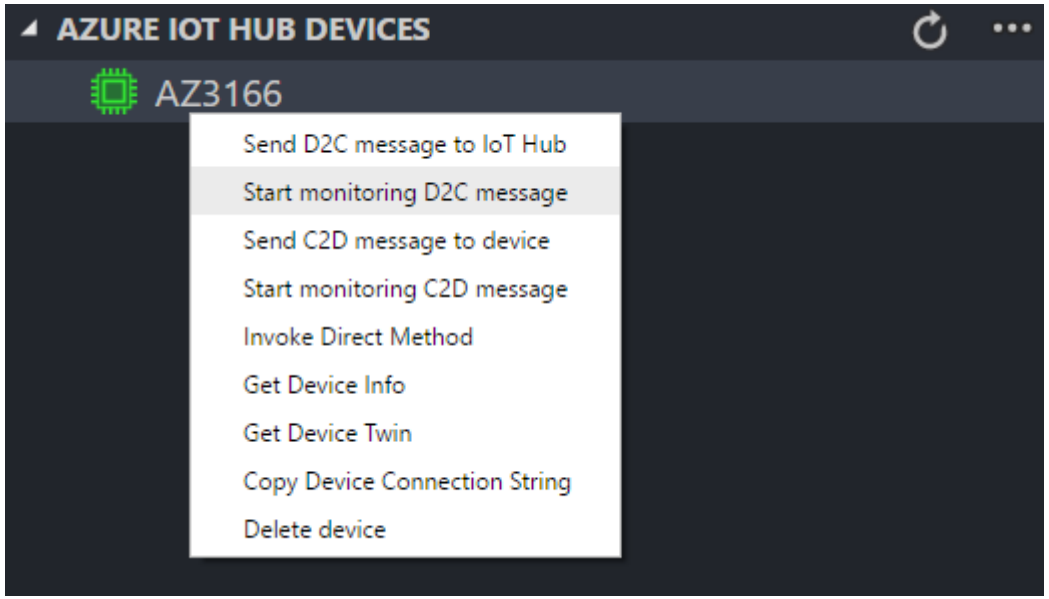
You should now see your device appear in AZURE IOT HUB DEVICES Tab as a green chip.



EXERCISE 5

D2C Message

Right click on the device and choose “Start monitoring D2C message”



EXERCISE 5

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