.NET IoT and AI Hack Days - Session 1

@CPTMSDUG

What is Session 1 about?

- Learning about .NET and running .NET locally and remotely on a Raspberry Pi
- Automating debugging with VS Code
- Getting started with Raspberry Pi and basic electronics
- Connecting your Raspberry Pi to Azure IoT and IoT Central

Foundation for more complex sessions

.NET – A unified development platform

CLOUD

Azure

WEB

ASP.NET

Blazor

DESKTOP

.NET MAUI

WPF

WinForms

MOBILE

.NET MAUI

Xamarin

GAMING

Unity

IoT

ARM32

ARM64

AI

ML.NET

.NET for Apache Spark

ECOSYSTEM

NuGet

GitHub

Components, tools, library vendors

TOOLS

Visual Studio

Visual Studio for Mac

Visual Studio Code

CLI

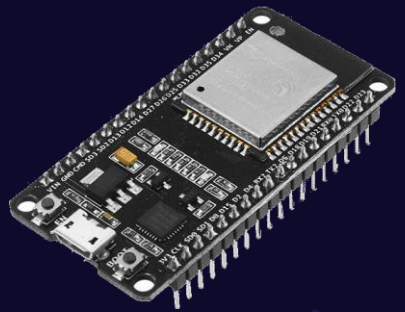
.NET 7

COMMON BASE LIBARIES/APIs

INFRASTRUCTURE

RUNTIME COMPONENTS | **COMPILERS** | **LANGUAGES**

Let’s talk about devices



Devices

Microcomputers

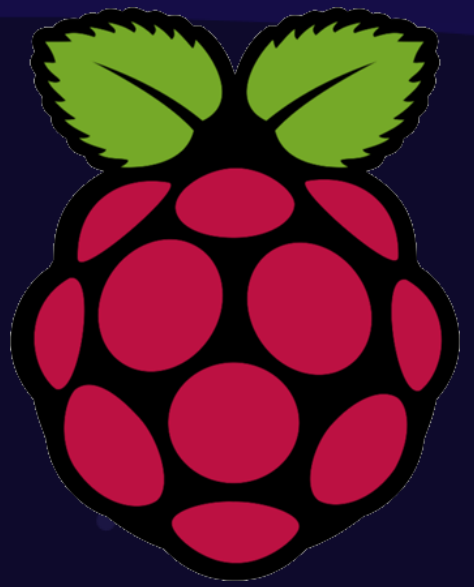
- Single board computers (SBC)
- X86 / 64 or ARM architecture
- RAM 512 MB+
- Operating System
  - Linux
  - Windows IoT
- High Power requirements
  - AI
  - Compute
- Interface with Hardware

Microcontrollers

- Small Single Chip Computer
- Dedicated Processor
- +- 64 K RAM
- No Operating System
- Low Power requirements
  - Long battery life
  - Remote locations
- Extremely Low Cost
  - High volume
- Interface with Hardware

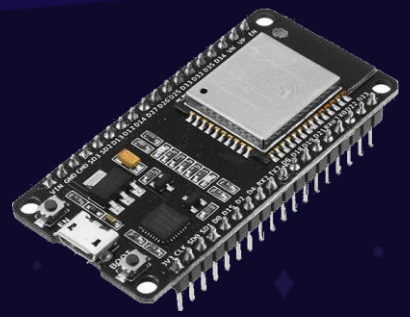
Microcomputers

- “Low-Cost” Computer
- Linux, Android, Windows 10 IoT
- Interact with hardware and electronics
  - GPIO, I2C, SPI
  - HATs
- IoT Device (Internet of Things!)



Microcontrollers

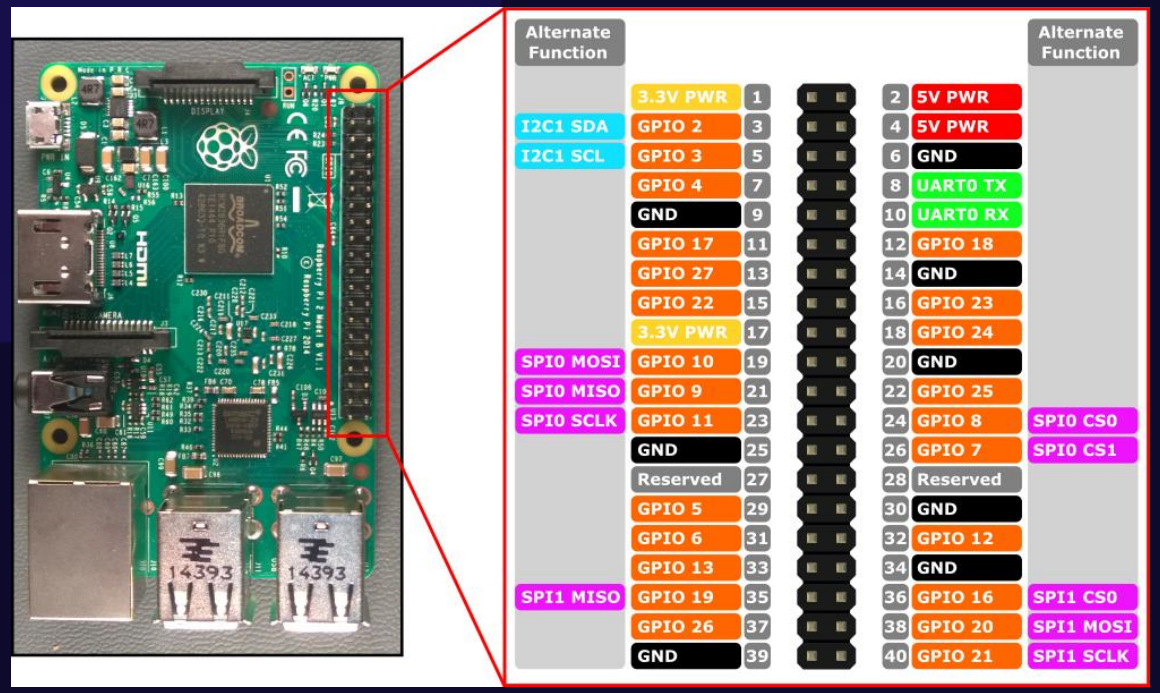
- Extremely low cost
- Long battery life
- Solutions run on the metal
- Interact with hardware and electronics
  - GPIO, I2C, SPI
  - HATs
- IoT Device (Internet of Things!)

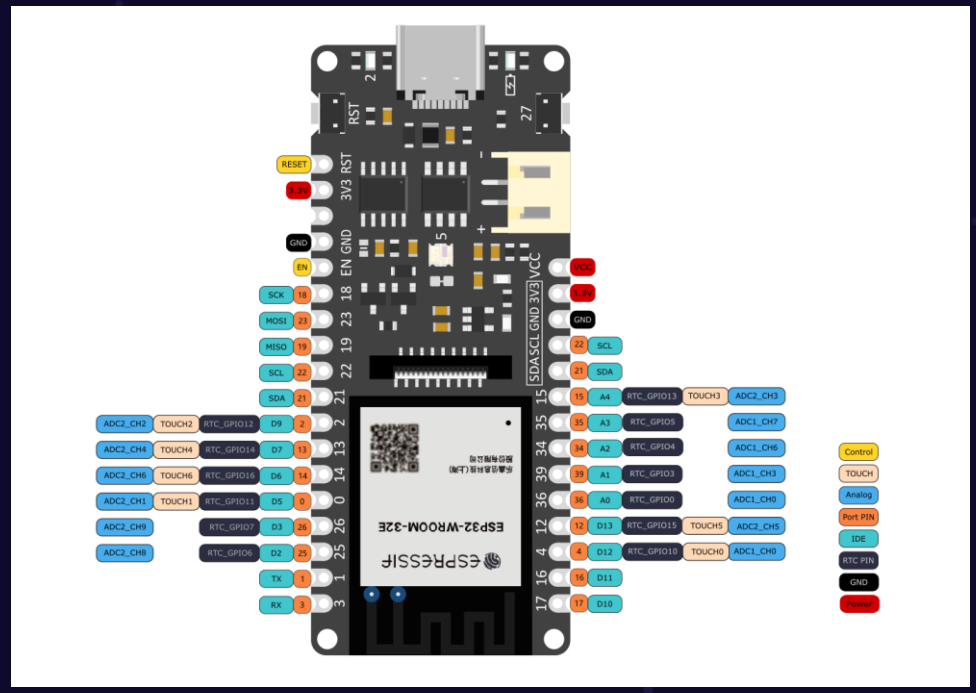


ARDUINO

M5STACK

Hardware Interfacing





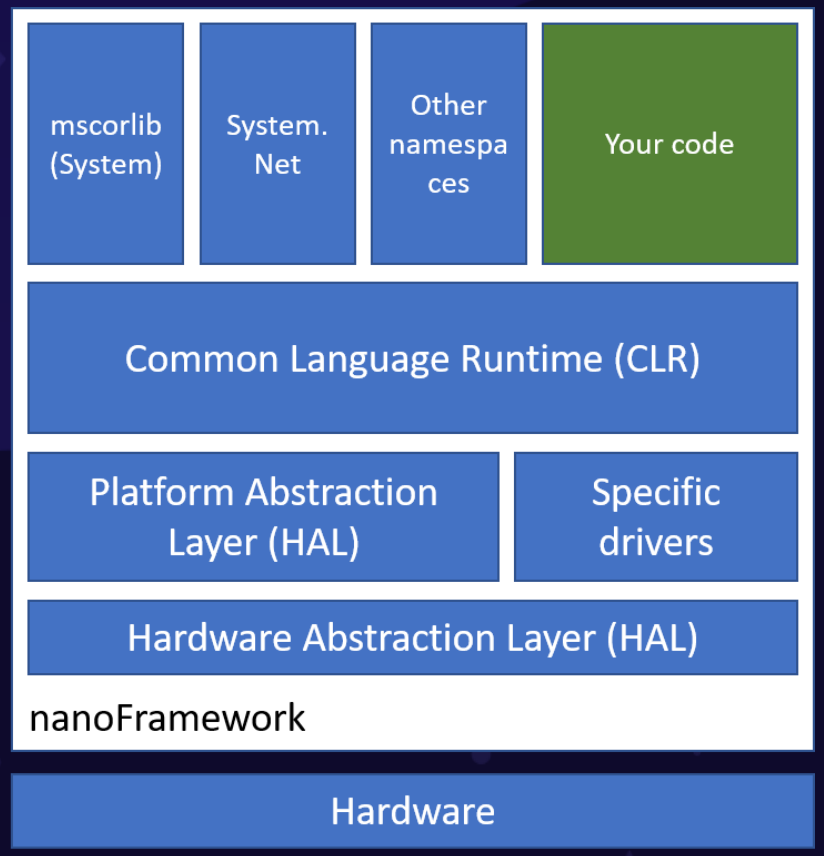
.NET for IoT

.NET 6 for ARM (v7+) or ARM64

- System.Device.Gpio
- Iot.Device.Bindings
- GPIO, UART, SPI, I2C

https://dotnet.microsoft.com/apps/iot

.NET Nanoframework



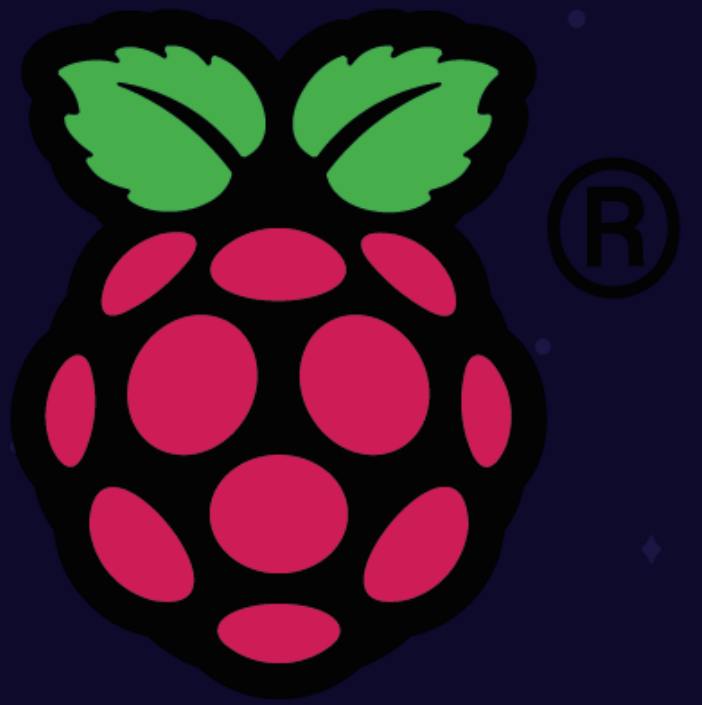
Reduced version of the .NET Common Language Runtime (CLR)
Subset of the .NET base class libraries
Resource-constrained devices with as low as 256kB of flash and 64kB of RAM
Runs directly on bare metal

GPIO, UART, SPI, I2C

Espressif ESP32
OrgPal
STMicroelectronics
TI SimpleLink
NXP boards
Netduino boards

https://docs.nanoframework.net/

Module 1



Running a .NET Application locally and remotely on a Raspberry Pi

Installing .NET 7

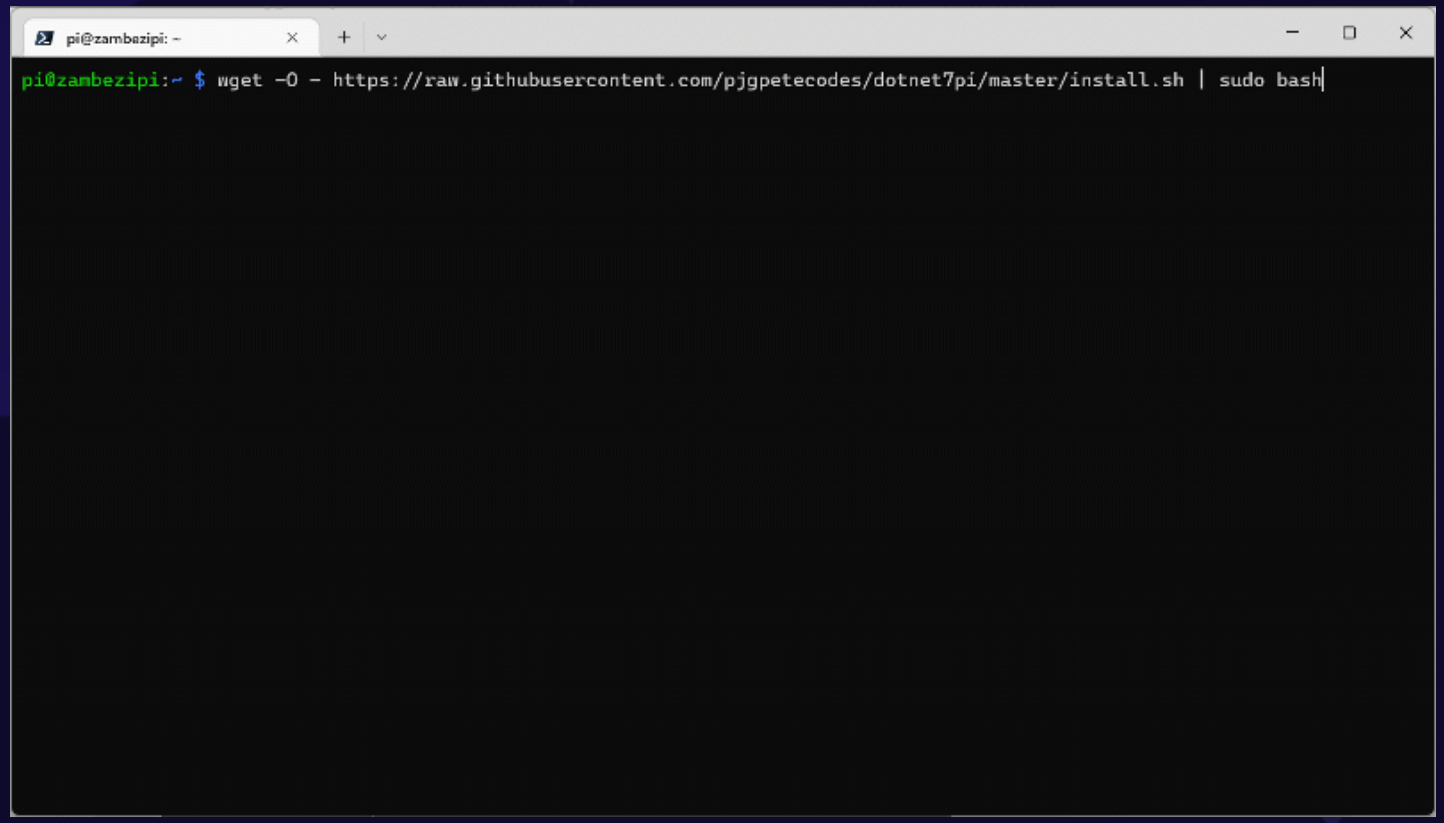
Build apps - SDK ⓘ

SDK 7.0.100

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS	Arm64 | x64	Arm64 | x64
Windows	Arm64 | x64 | x86 | winget instructions	Arm64 | x64 | x86
All	dotnet-install scripts	

https://dotnet.microsoft.com/en-us/download/dotnet/7.0

Installing .NET 7 on Raspberry Pi

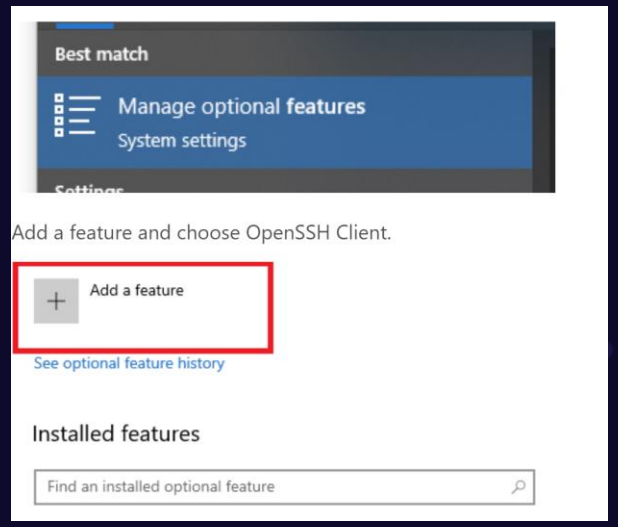


wget -O - https://raw.githubusercontent.com/pjgpetecodes/dotnet7pi/master/install.sh | sudo bash

Development Tools

SSH

SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, a secure way to access a computer over a network.



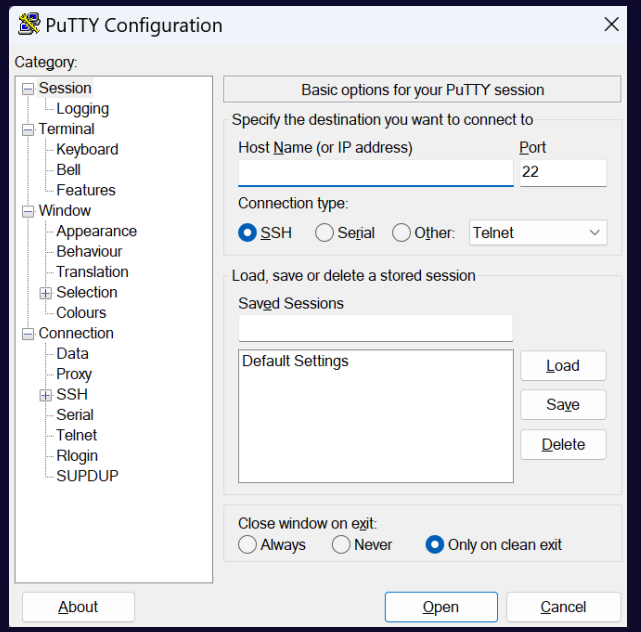
Development Tools

SSH

SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, a secure way to access a computer over a network.

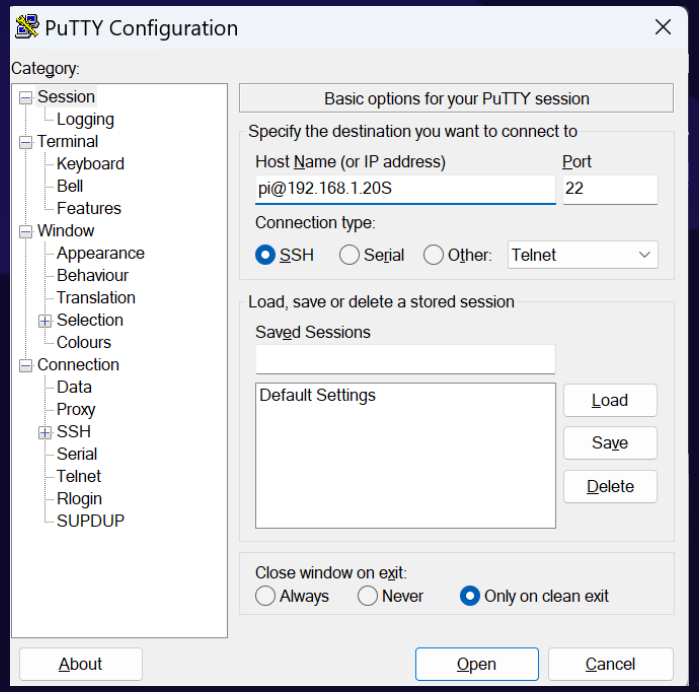
Putty

Download PuTTY - a free SSH and telnet client for Windows



Development Tools

SSH



Try it out!!

Switch your Raspberry PI on

Run Putty

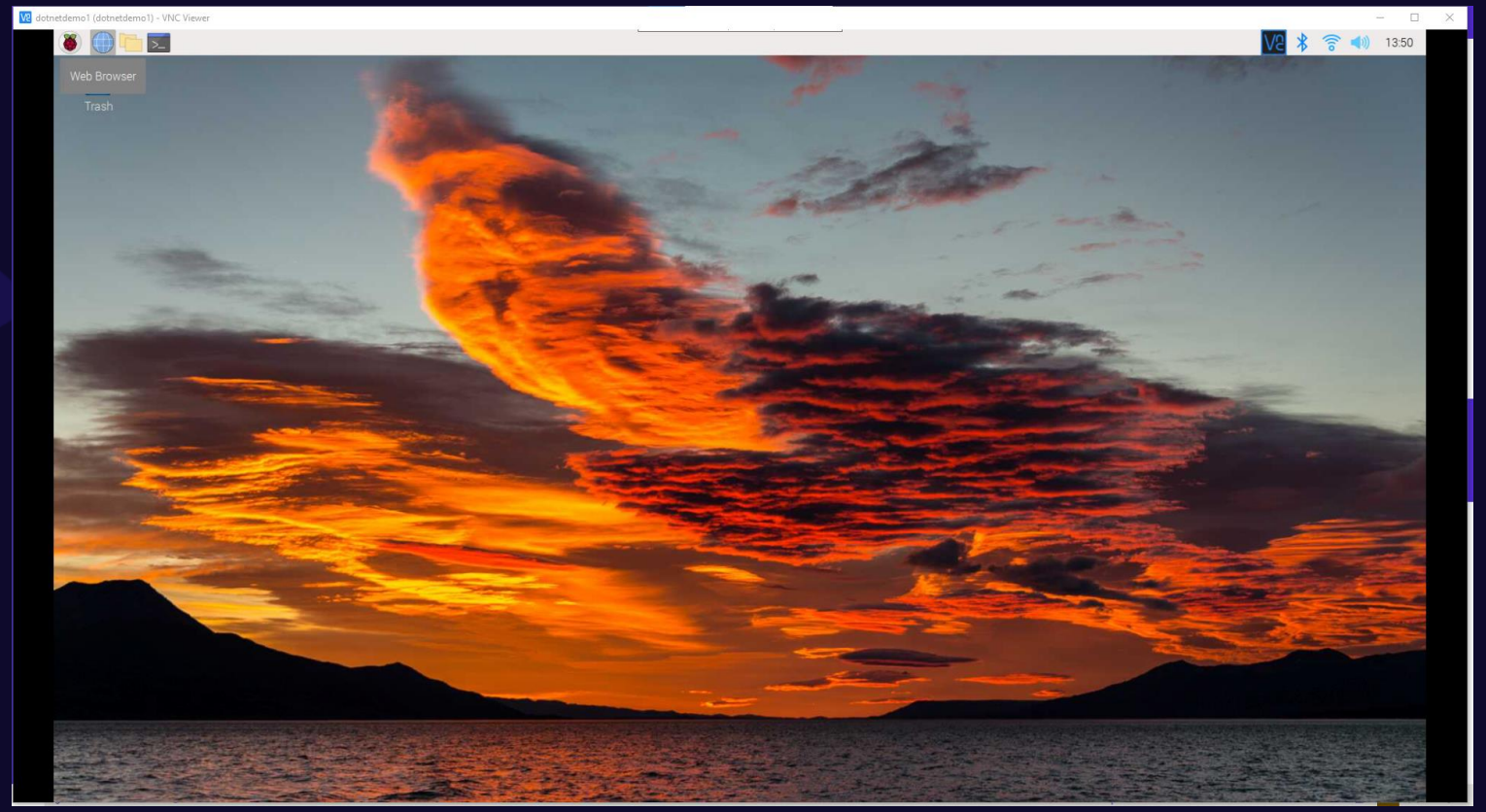
Connect to your PI with: pi@ipaddress

Password: mxchip12345

Development Tools

VNC

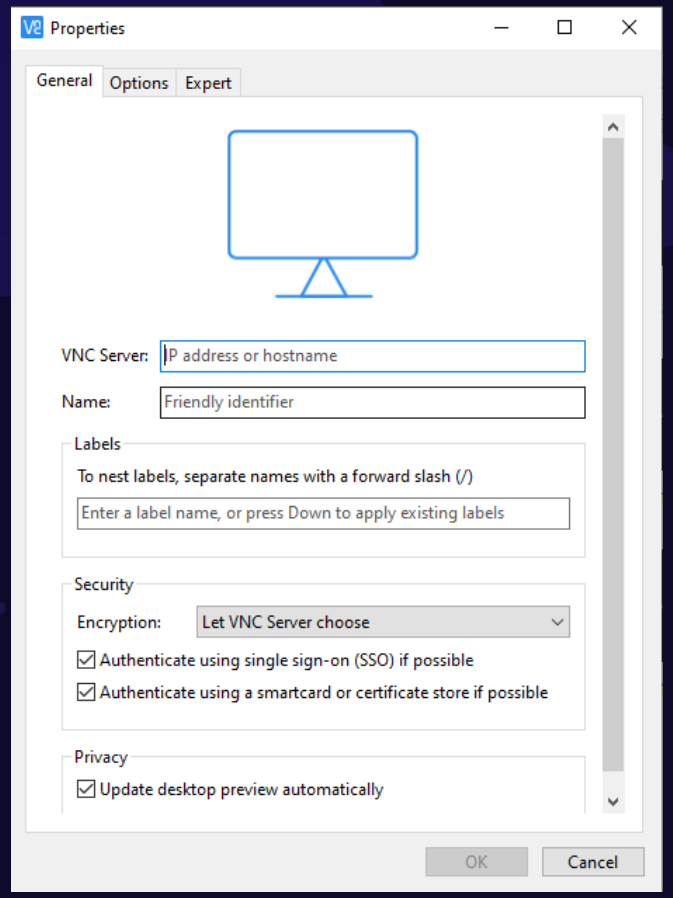
Virtual network computing (VNC) is a type of remote-control software that makes it possible to control another computer over a network connection.



https://www.realvnc.com/en/connect/download/viewer/

Development Tools

VNC



Try it out!!

Switch your Raspberry PI on

Run VNC
Connect to your PI with the IP Address

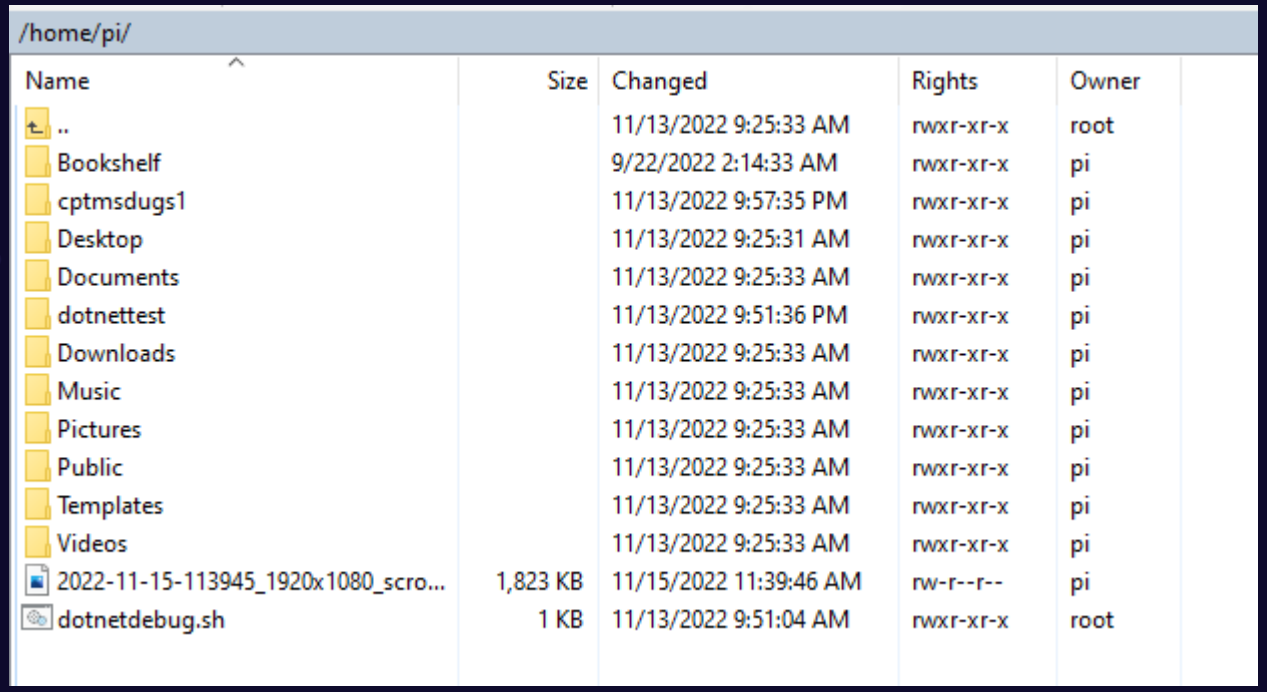
User Name: pi
Password: mxchip12345

https://www.realvnc.com/en/connect/download/viewer/

Development Tools

WinSCP

WinSCP is a popular SFTP client and FTP client for Microsoft Windows! Copy file between a local computer and remote servers using FTP, FTPS, SCP, SFTP, WebDAV or S3 file transfer protocols.



https://winscp.net/eng/index.php

Development Tools

WinSCP

Try it out!!

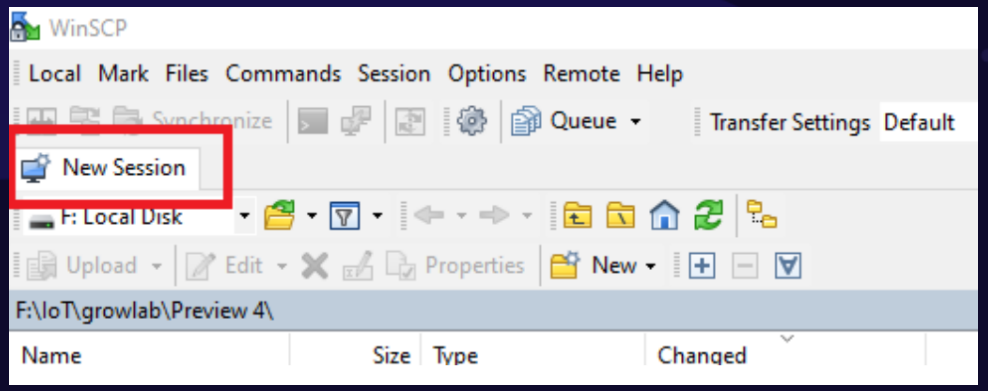
Switch your Raspberry PI on

Run WinSCP
Connect to your PI with the IP Address

User Name: pi
Password: mxchip12345

Development Tools

WinSCP



Try it out!!

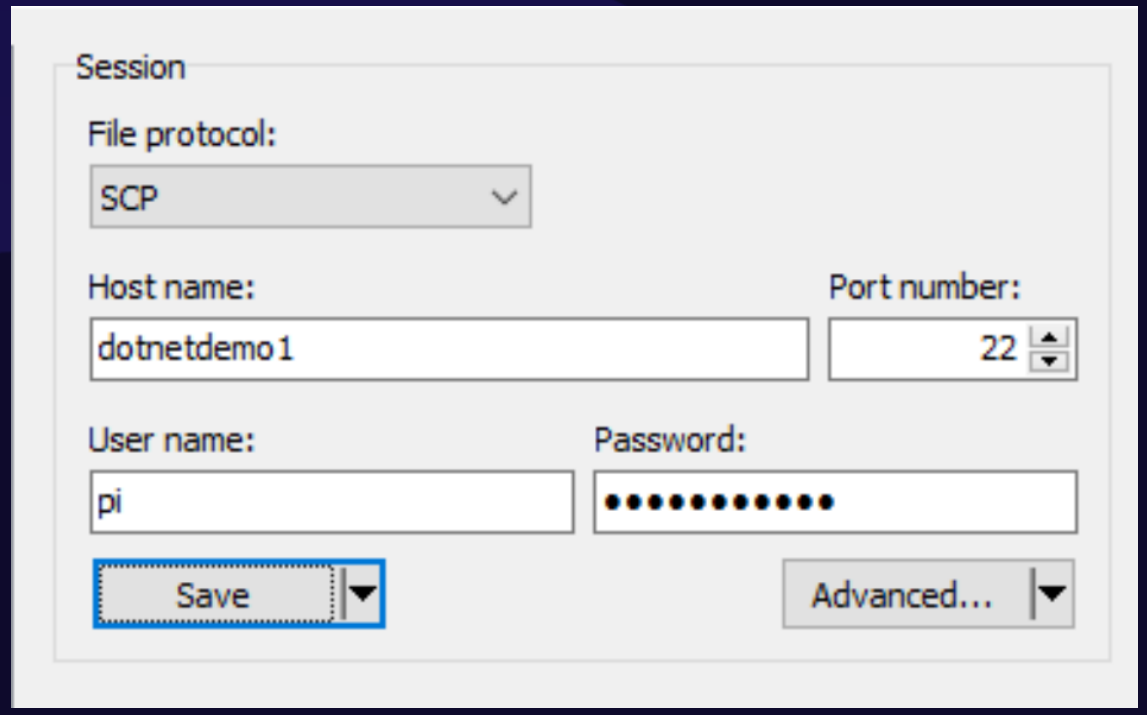
Switch your Raspberry PI on

Run WinSCP

Create New Session

Development Tools

WinSCP



Protocol SCP

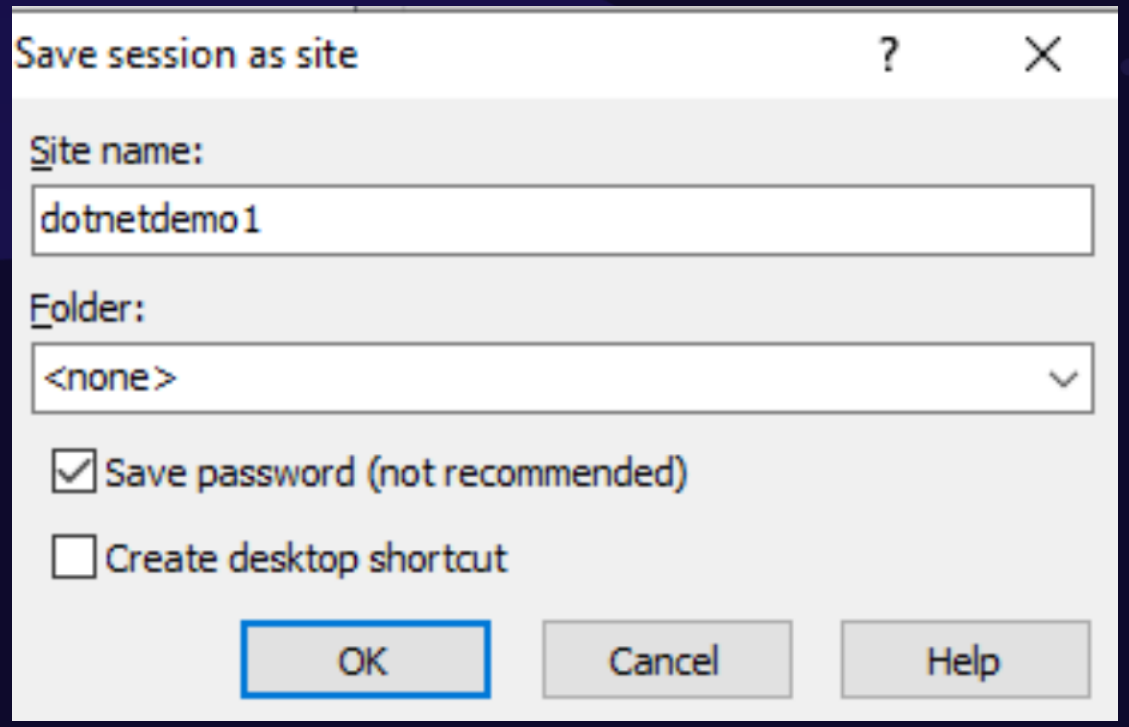
Host name: your ipaddress

Username: pi

Password: mxchip12345

Development Tools

WinSCP



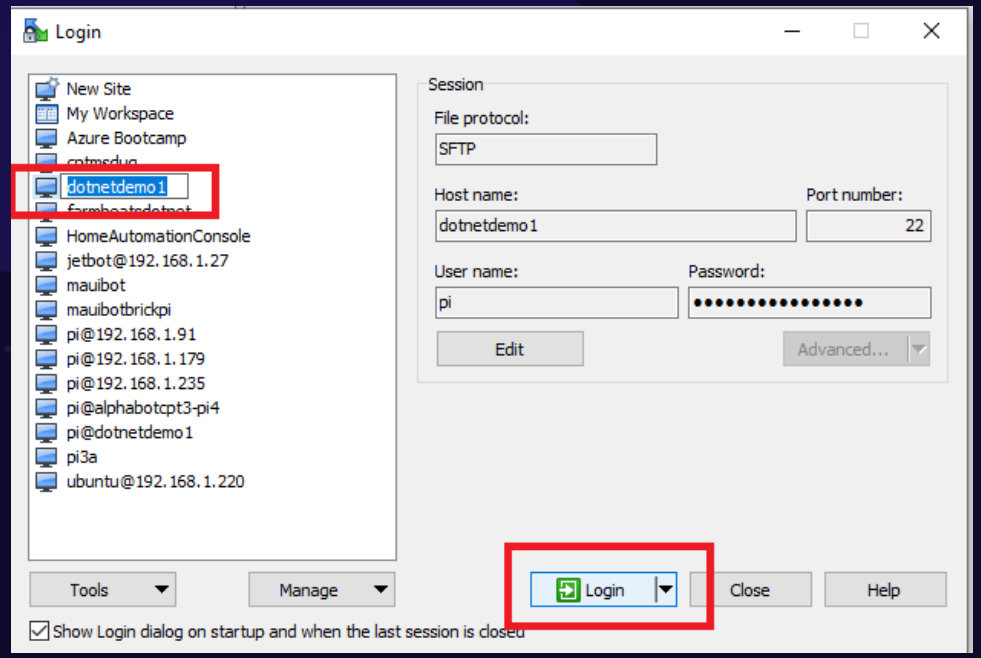
Save

Site name: The name of your pi. Use the sticker on the pi, eg: E1163323

Save Password

Development Tools

WinSCP



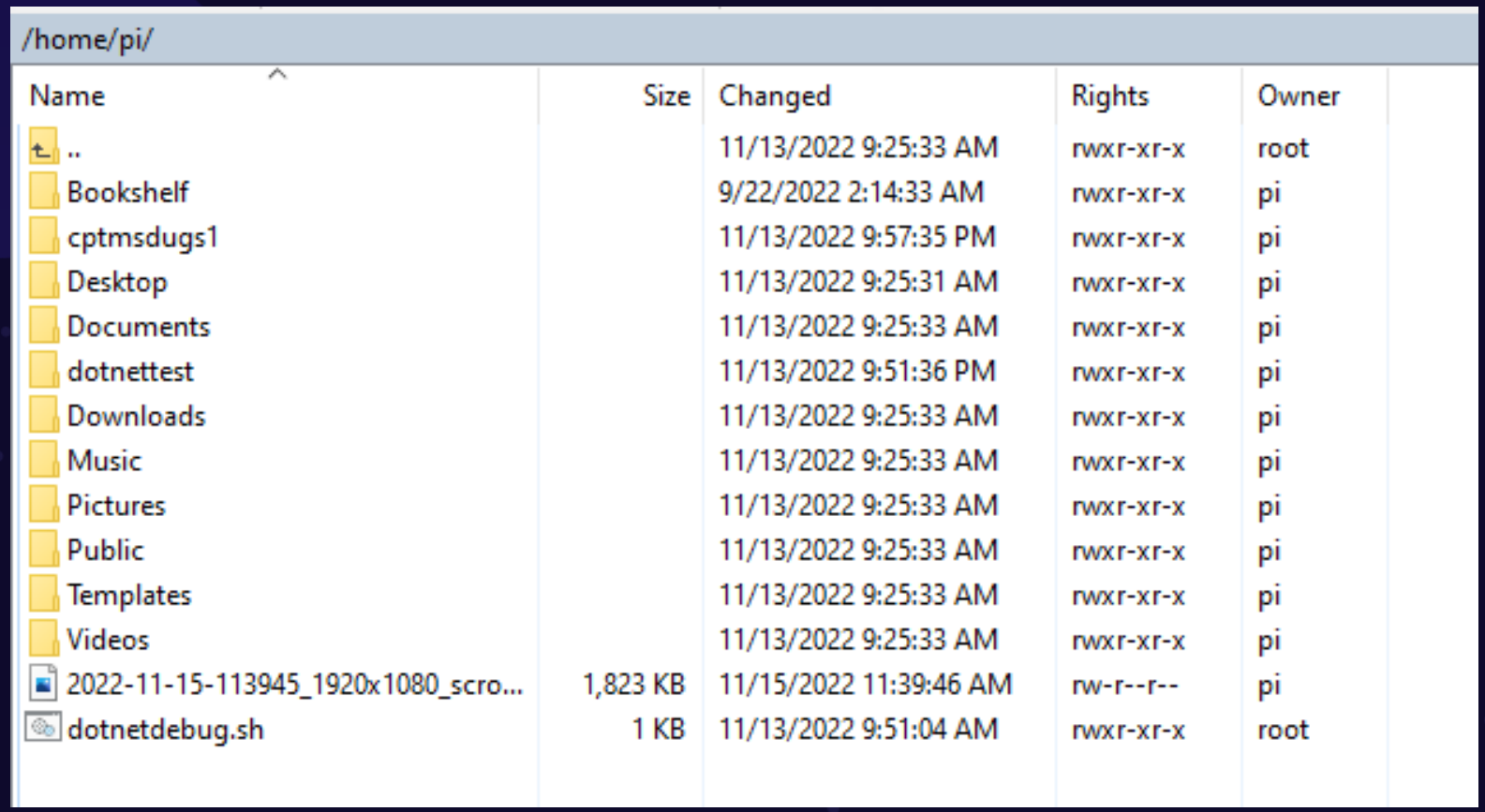
Login

Choose the name of your pi from the sites.

Click login

Development Tools

WinSCP



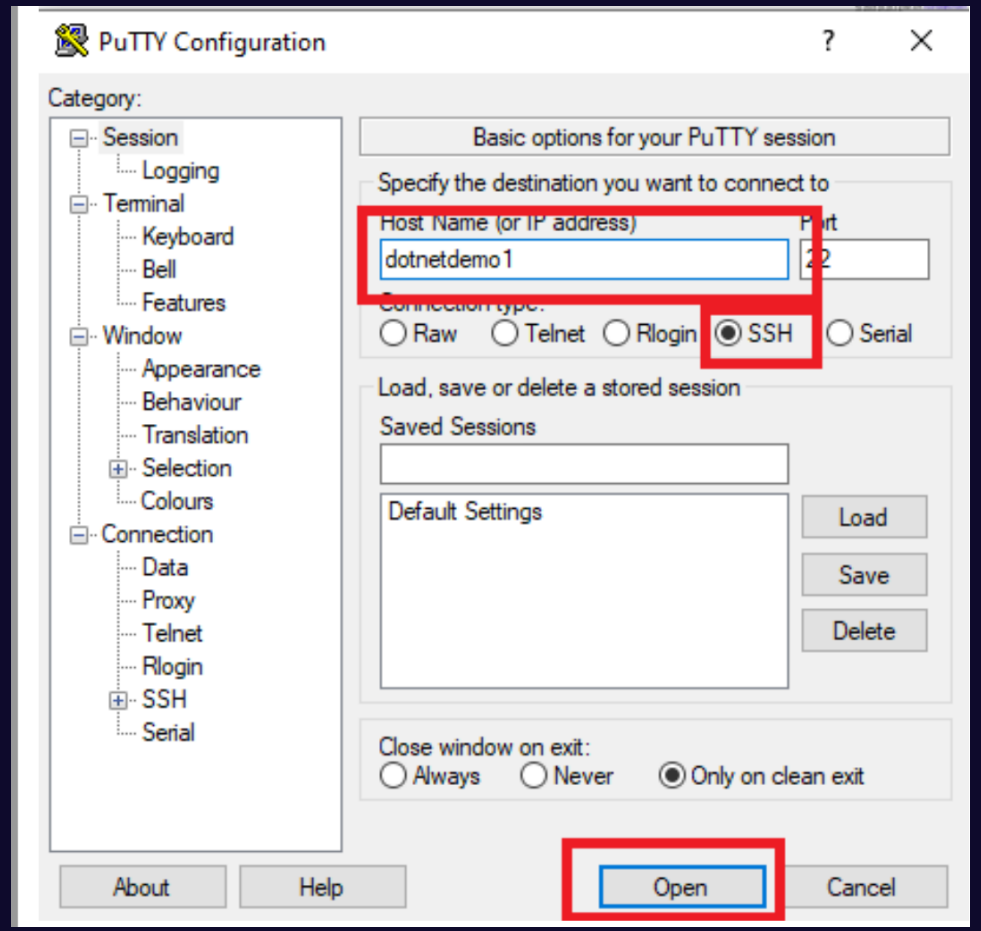
Running your First .NET 7 Application

Run Putty

Host Name: your pi's IP Address

Choose SSH

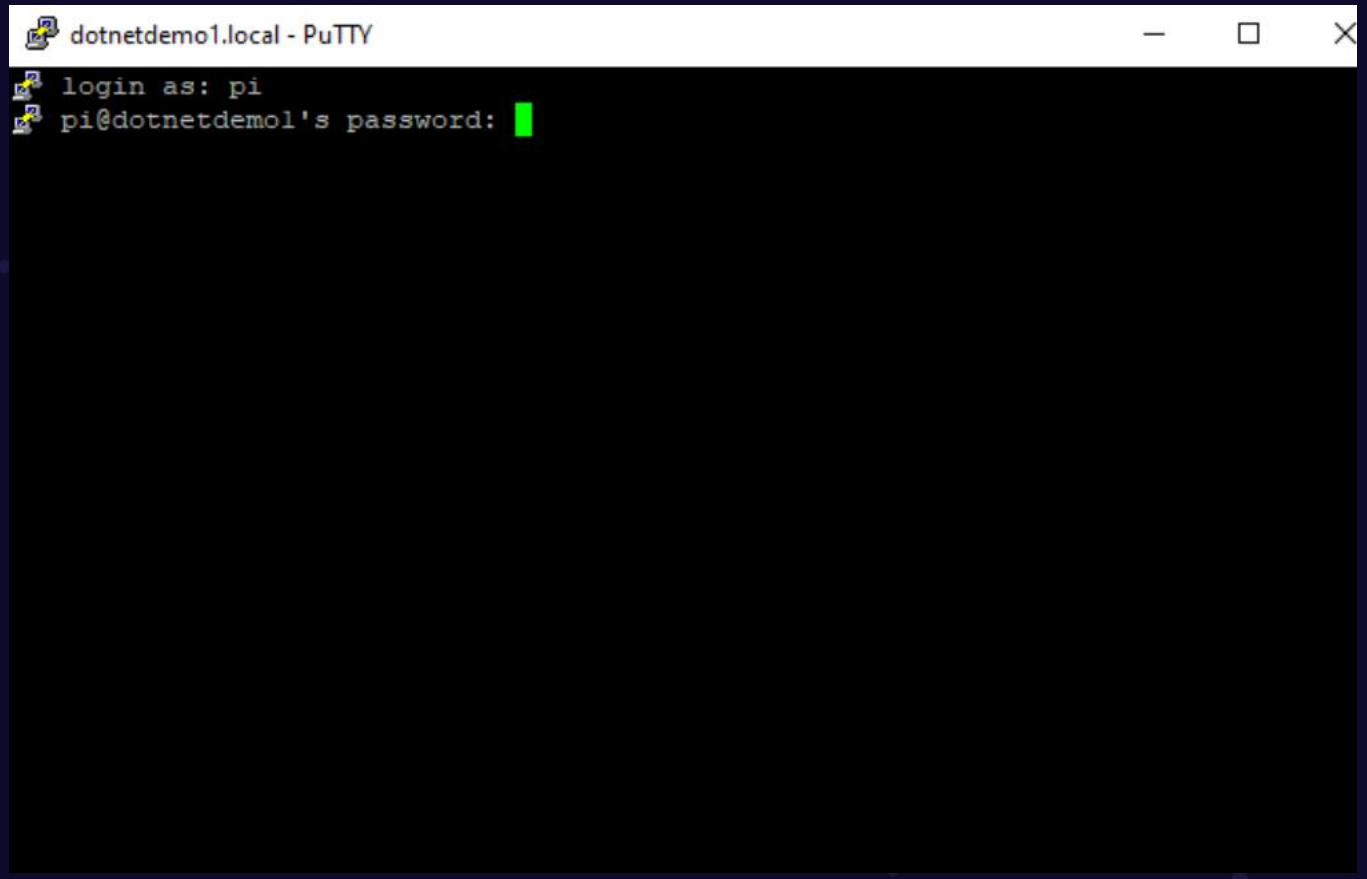
Open



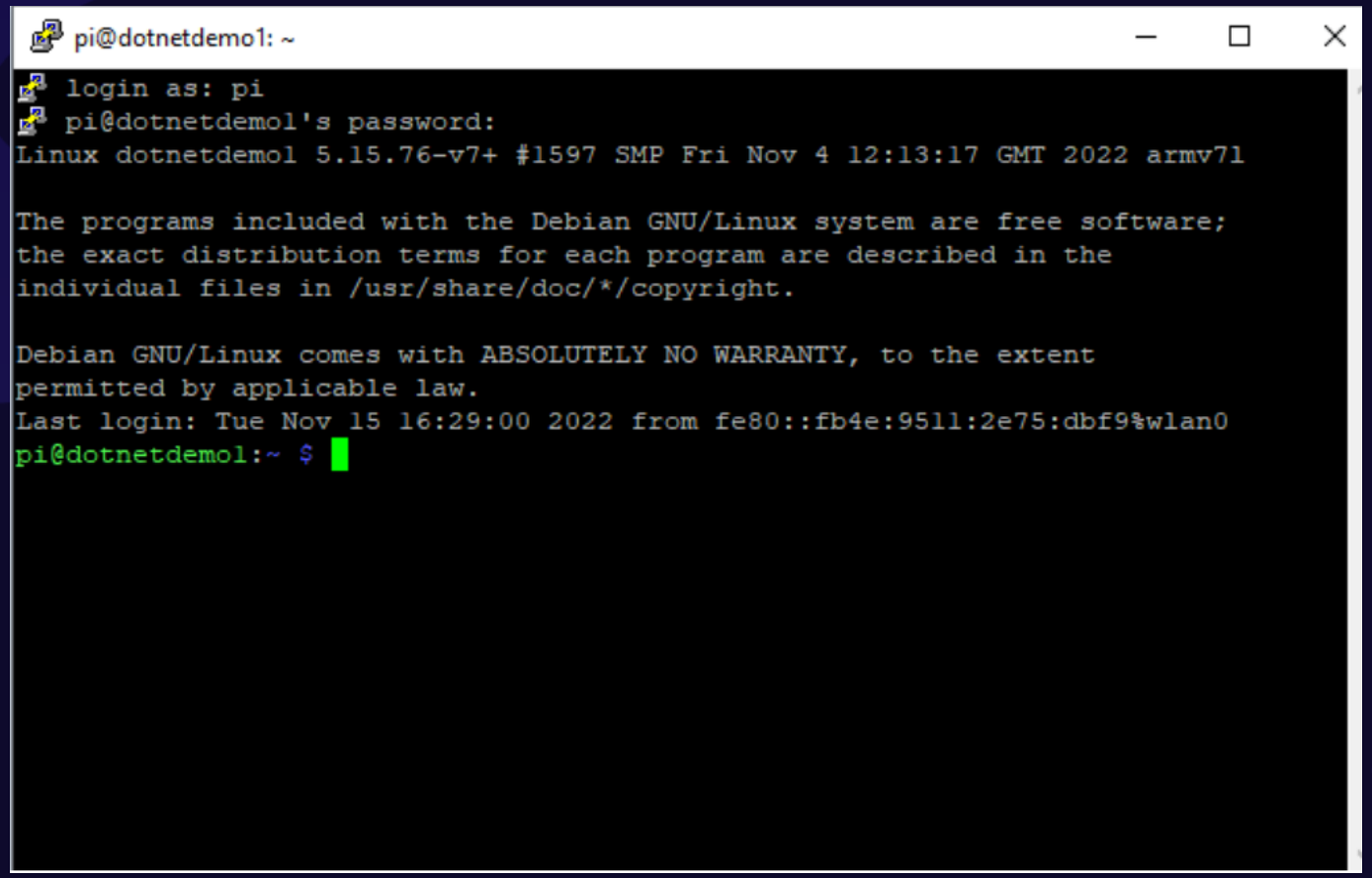
Running your First .NET 7 Application

User Name: pi

Password: mxchip12345



Running your First .NET 7 Application



Running your First .NET 7 Application

Create a new directory / folder on the Pi

mkdir myfirstapp

```
pi@dotnetdemo1:~ $ mkdir myfirstapp
pi@dotnetdemo1:~ $ cd myfirstapp
pi@dotnetdemo1:~/myfirstapp $
```

Running your First .NET 7 Application

Create a new directory / folder on the Pi

mkdir myfirstapp

Change to new directory

cd myfirstapp

```
pi@dotnetdemo1:~ $ mkdir myfirstapp
pi@dotnetdemo1:~ $ cd myfirstapp
pi@dotnetdemo1:~/myfirstapp $
```

Running your First .NET 7 Application

Create a new .NET 7 console app.

dotnet new console

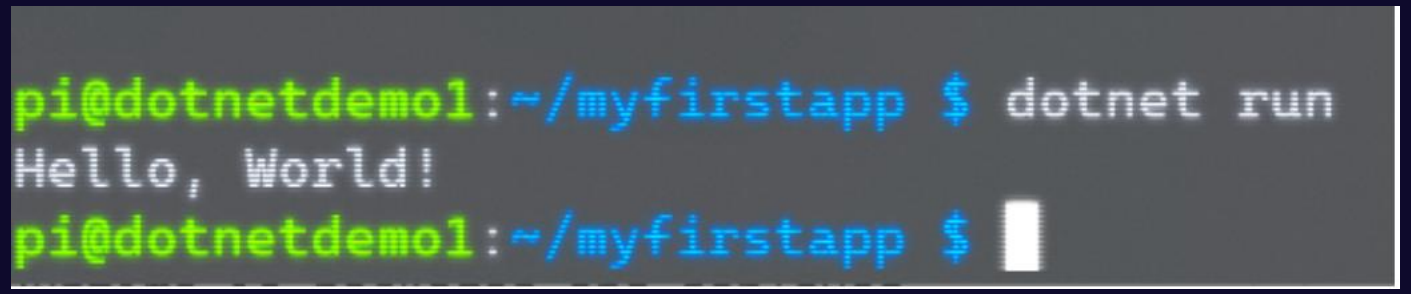
```
pi@dotnetdemo1:~/myfirstapp $ dotnet new console
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring /home/pi/myfirstapp/myfirstapp.csproj:
  Determining projects to restore...
  Restored /home/pi/myfirstapp/myfirstapp.csproj (in 660 ms).
Restore succeeded.
```

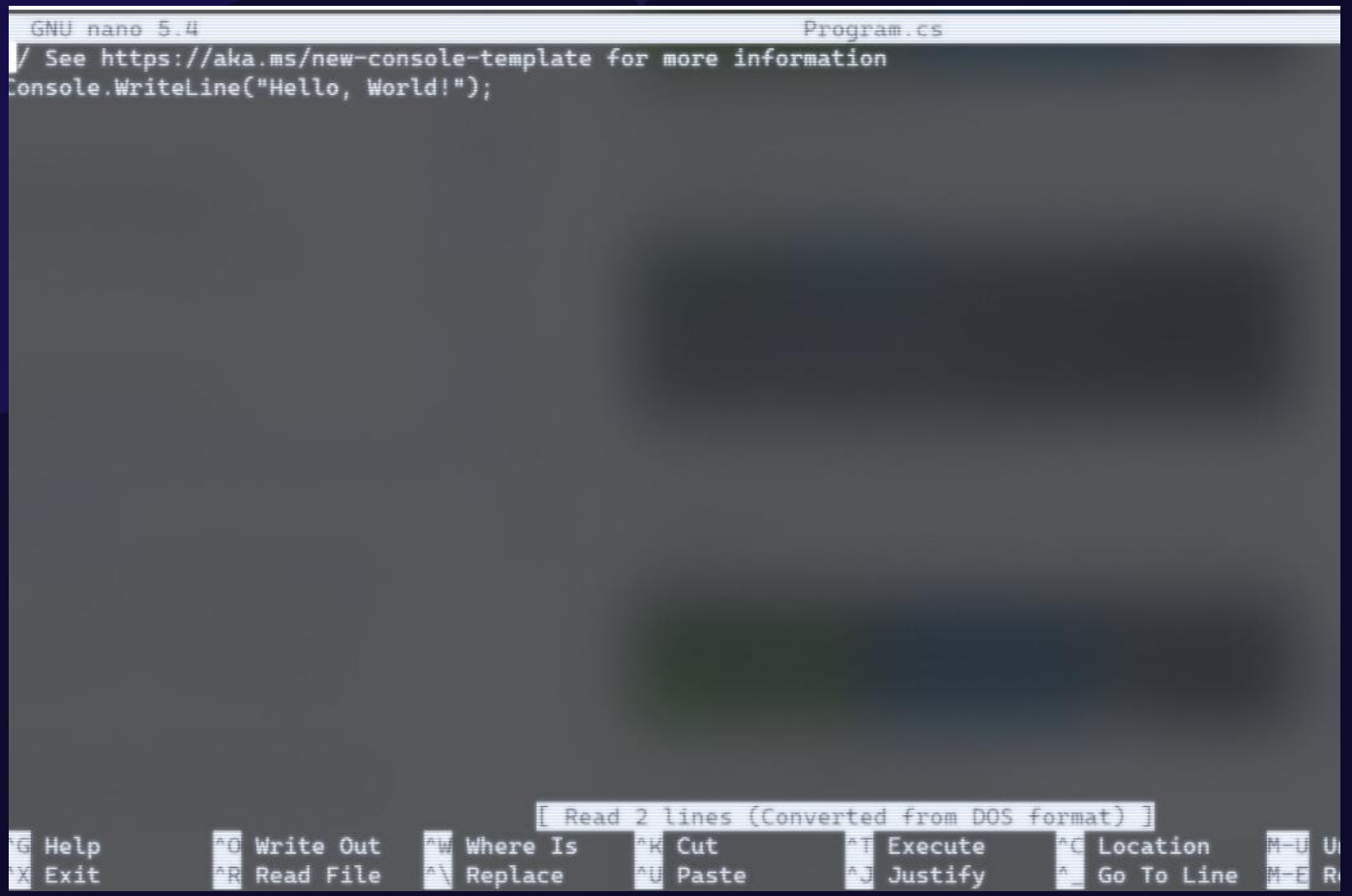
Running your First .NET 7 Application

Run your first .NET 7 app on a Raspberry Pi!

dotnet run



Running your First .NET 7 Application



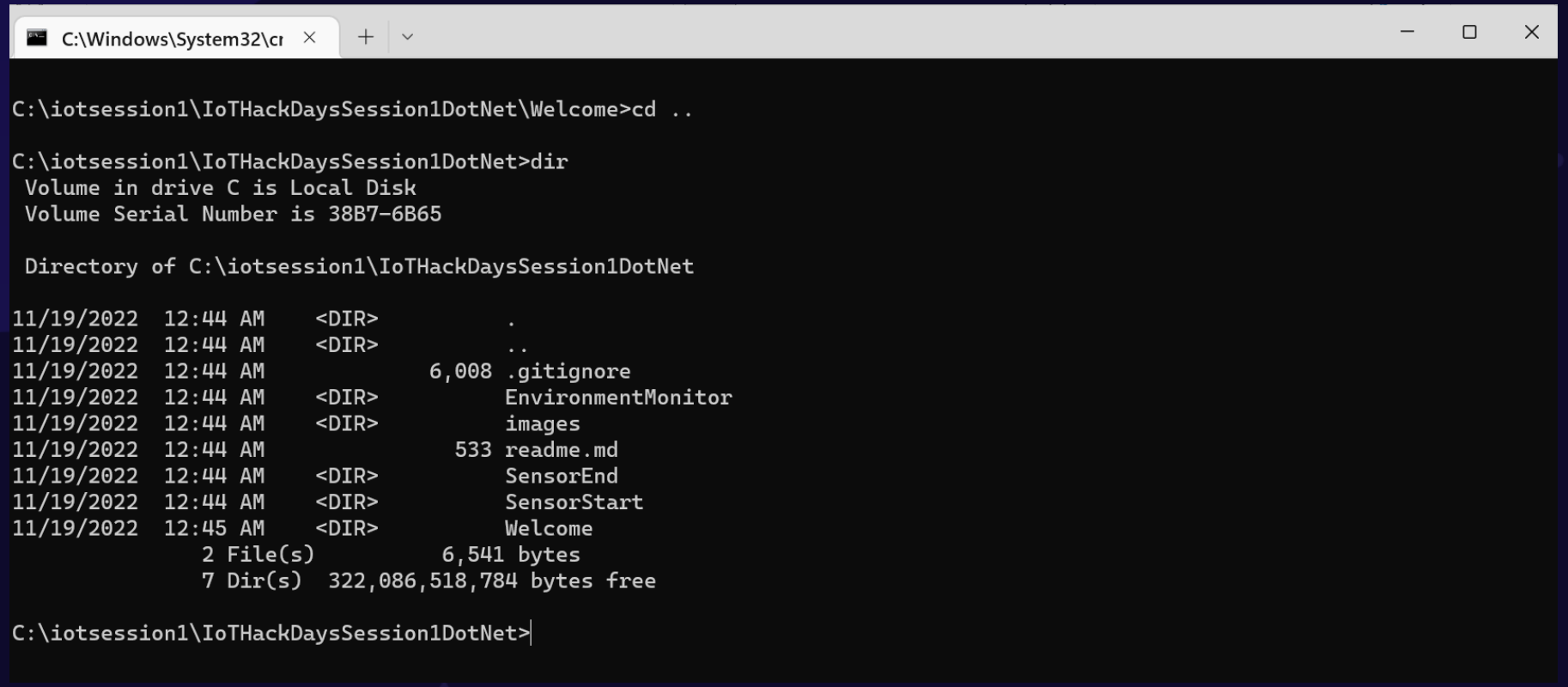
Running and creating your first .NET app on a REMOTE Raspberry pi

Remote development using a PC is much more efficient as the resources available (CPU, RAM, Disk space and Disk Speed) is so much more than an IoT device.

- Make a folder: c:\iotsession1
- Using command line: Clone the IoT Hack Session 1 Git Repository into the folder above
- **git clone** https://github.com/CPTMSDUG/IoTHackDaysSession1DotNet
-

https://github.com/CPTMSDUG/IoTHackDaysSession1DotNet

Running and creating your first .NET app on a REMOTE Raspberry pi



https://github.com/CPTMSDUG/IoTHackDaysSession1DotNet

Running and creating your first .NET app on a REMOTE Raspberry pi

Welcome Project

Change to the Welcome folder

cd Welcome

```
C:\iotsession1\IoTHackDaysSession1DotNet>cd welcome

C:\iotsession1\IoTHackDaysSession1DotNet\Welcome>dir
 Volume in drive C is Local Disk
 Volume Serial Number is 38B7-6B65

 Directory of C:\iotsession1\IoTHackDaysSession1DotNet\Welcome

11/19/2022  12:45 AM    <DIR>          .
11/19/2022  12:44 AM    <DIR>          ..
11/19/2022  12:44 AM                39 .env
11/19/2022  12:44 AM    <DIR>          .vscode
11/19/2022  12:45 AM    <DIR>          bin
11/19/2022  12:44 AM                47 global.json
11/19/2022  12:44 AM               444 netpublish.bat
11/19/2022  12:45 AM    <DIR>          obj
11/19/2022  12:44 AM               301 Program.cs
11/19/2022  12:44 AM               392 Welcome.csproj
11/19/2022  12:44 AM               522 welcometext.txt
               6 File(s)          1,745 bytes
               5 Dir(s)  322,083,233,792 bytes free
```

https://github.com/CPTMSDUG/IoTHackDaysSession1DotNet

Running and creating your first .NET app on a REMOTE Raspberry pi

Welcome Project

Run the project

dotnet run

This is running on your laptop!

https://github.com/CPTMSDUG/IoTHackDaysSession1DotNet

Running and creating your first .NET app on a REMOTE Raspberry pi

Welcome Project

Cross compile for an ARM processor

Raspberry Pi isn't an Intel x86, it's an ARM architecture device

dotnet publish -r linux-arm

https://github.com/CPTMSDUG/IoTHackDaysSession1DotNet

Running and creating your first .NET app on a REMOTE Raspberry pi

Welcome Project

Cross compile for an ARM processor

Not self contained. Relies on installed runtime

dotnet publish -r linux-arm --no-self-contained

Self contained. Runtime is packaged and deployed with the application

dotnet publish -r linux-arm --self-contained

https://github.com/CPTMSDUG/IoTHackDaysSession1DotNet

Running and creating your first .NET app on a REMOTE Raspberry pi

Welcome Project

Cross compile for an ARM processor

Try it out!

dotnet publish -r linux-arm --self-contained

```
C:\iotsession1\IoTHackDaysSession1DotNet\Welcome>dotnet publish -r linux-arm --self-contained
MSBuild version 17.3.2+561848881 for .NET
  Determining projects to restore...
  Restored C:\iotsession1\IoTHackDaysSession1DotNet\Welcome\Welcome.csproj (in 185 ms).
  Welcome -> C:\iotsession1\IoTHackDaysSession1DotNet\Welcome\bin\Debug\net6.0\linux-arm\Welcome.dll
  Welcome -> C:\iotsession1\IoTHackDaysSession1DotNet\Welcome\bin\Debug\net6.0\linux-arm\publish\
```

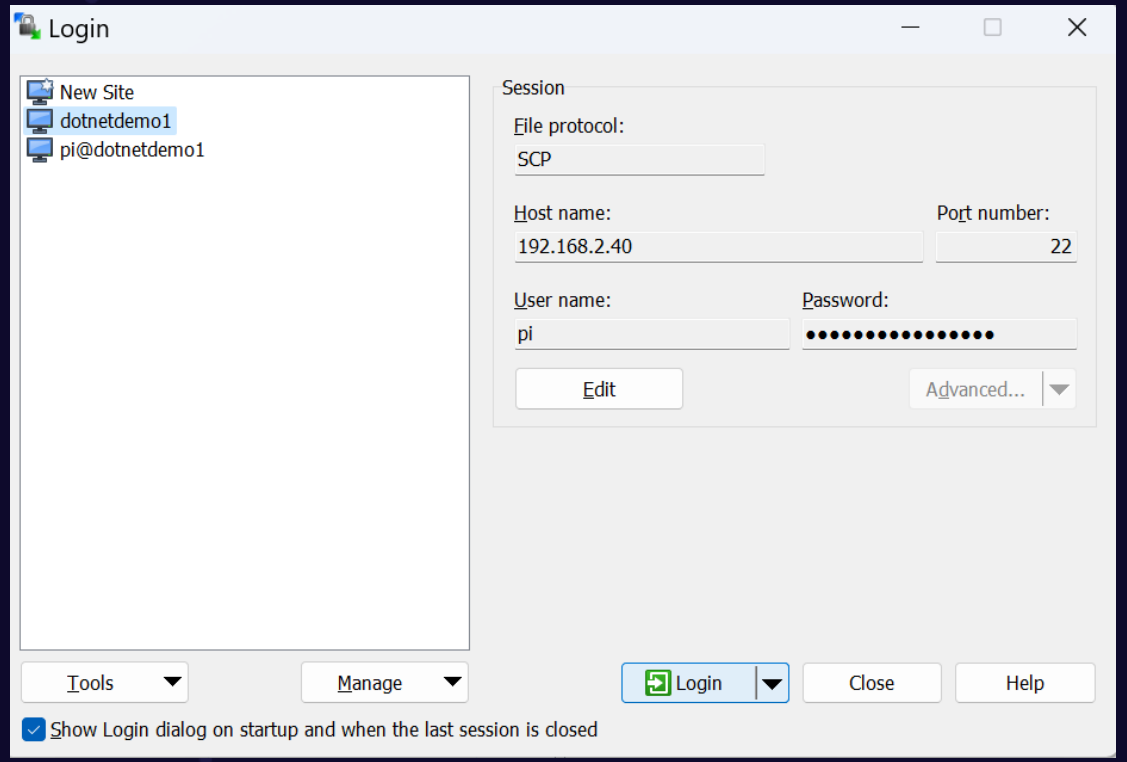
https://github.com/CPTMSDUG/IoTHackDaysSession1DotNet

Running and creating your first .NET app on a REMOTE Raspberry pi

Welcome Project

Cross compile for an ARM processor

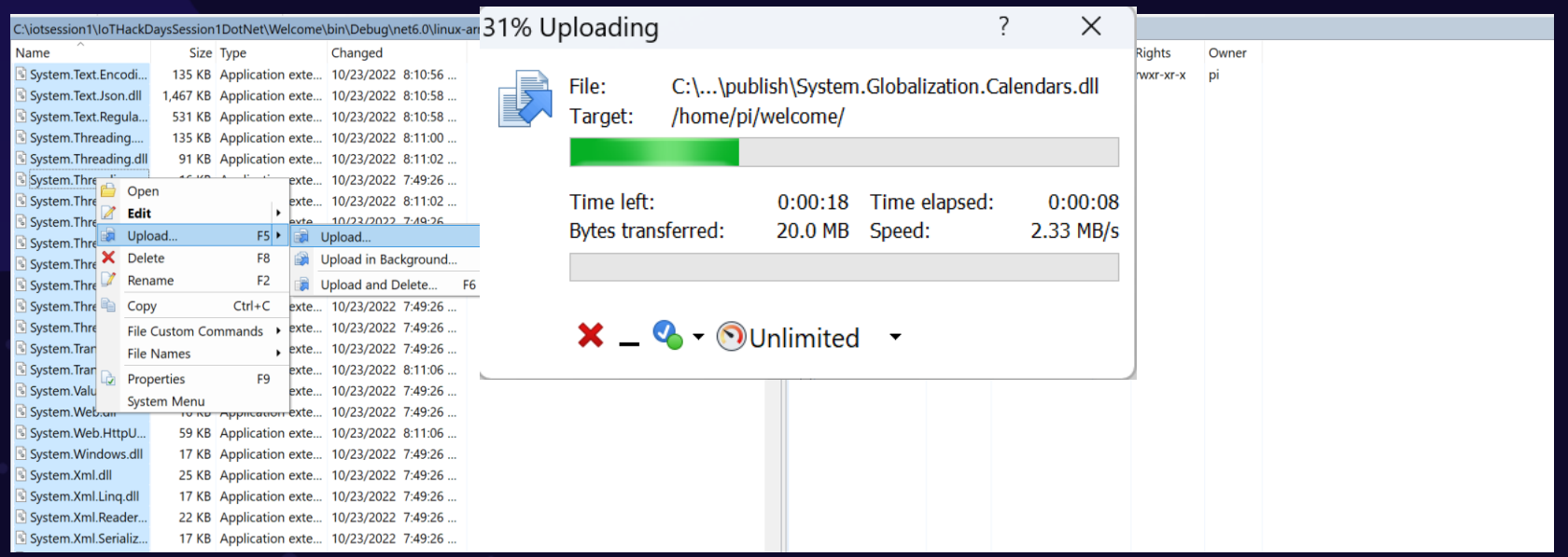
Deploy to the Raspberry Pi with WinSCP



Running and creating your first .NET app on a REMOTE Raspberry pi

Local publish folder

Remote Pi folder

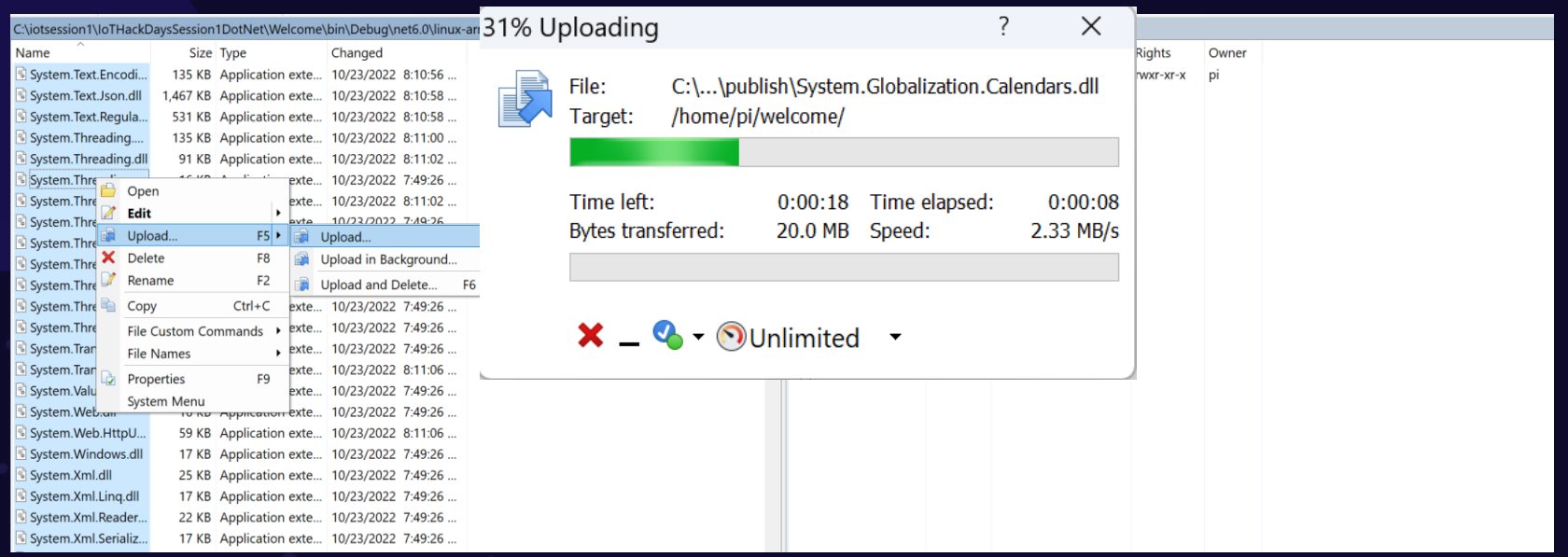


Select all files from the publish folder.
Select the Welcome folder on the right on the PI
Right click: Upload

Running and creating your first .NET app on a REMOTE Raspberry pi

Local publish folder

Remote Pi folder



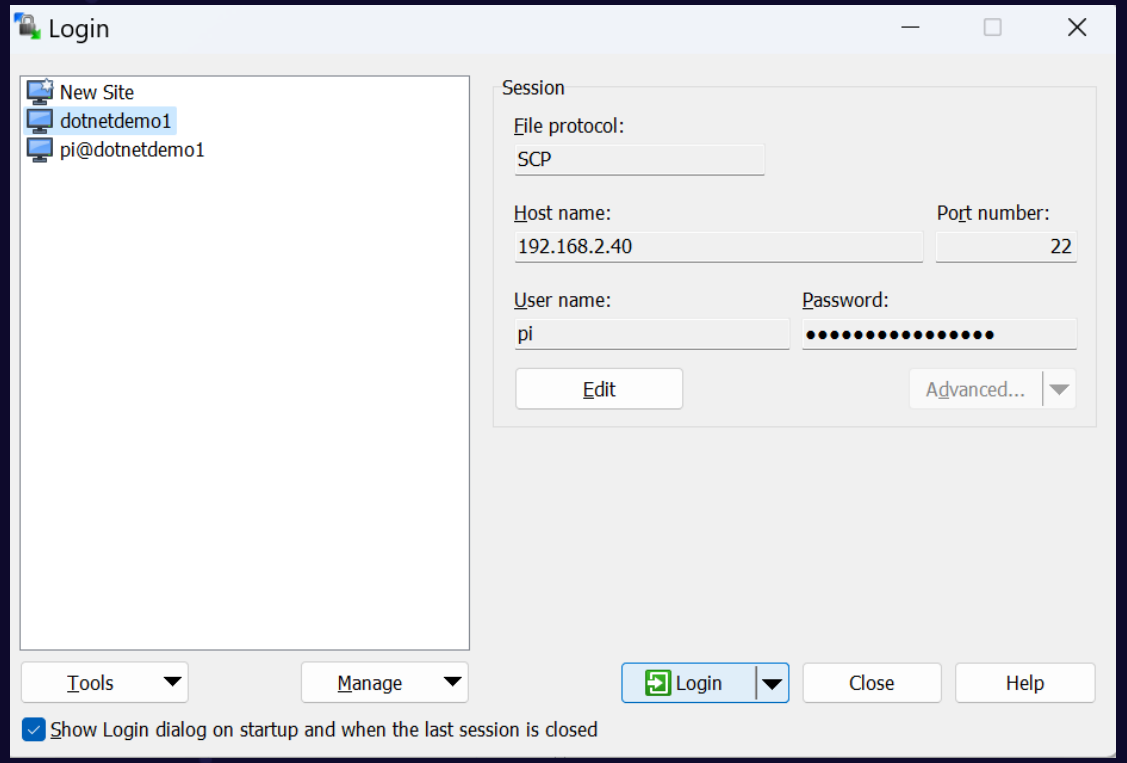
Select all files from the publish folder.
Select the Welcome folder on the right on the PI
Right click: Upload

Running and creating your first .NET app on a REMOTE Raspberry pi

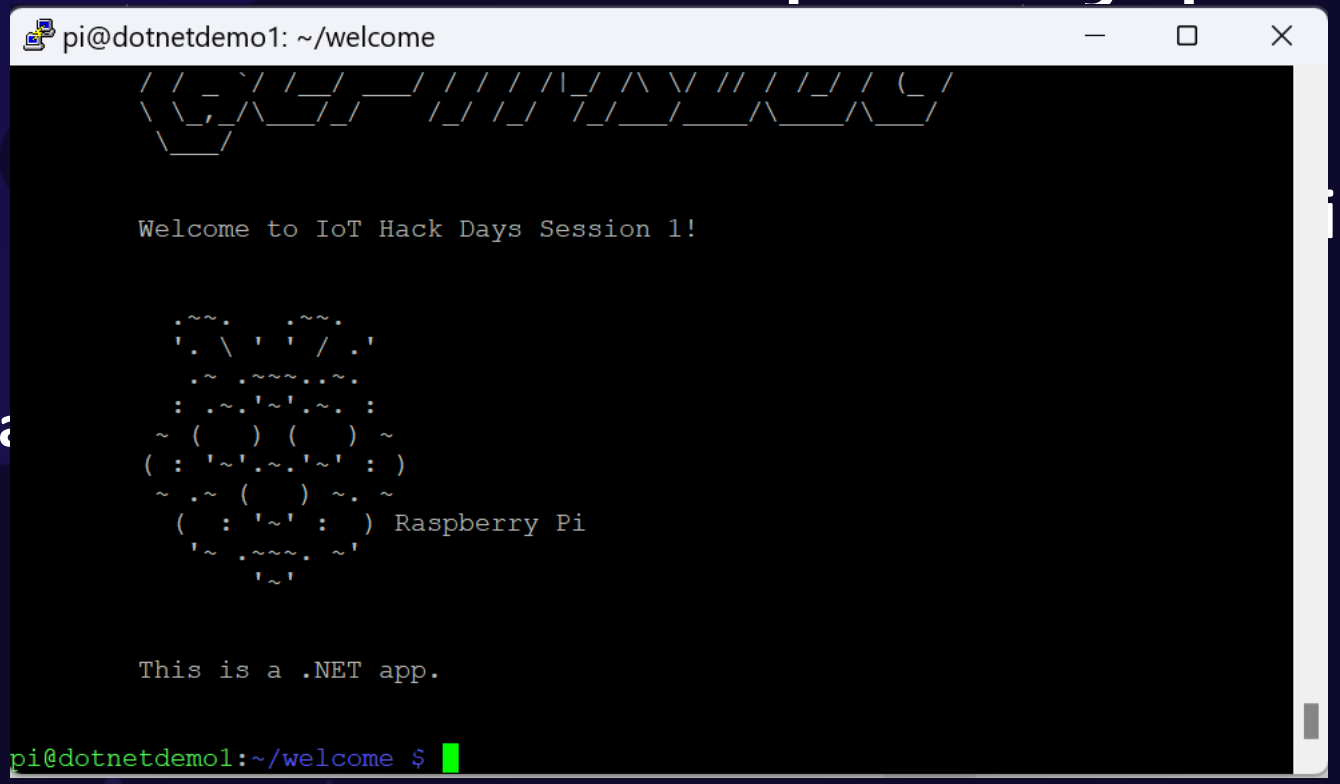
Welcome Project

Cross compile for an ARM processor

Run the application on the pi via SSH



Running and creating your first .NET app on a REMOTE Raspberry pi



ia SSH

Welcome Project

Cross compile for a processor

This is running on your Remote Raspberry Pi!

Visual Studio Code

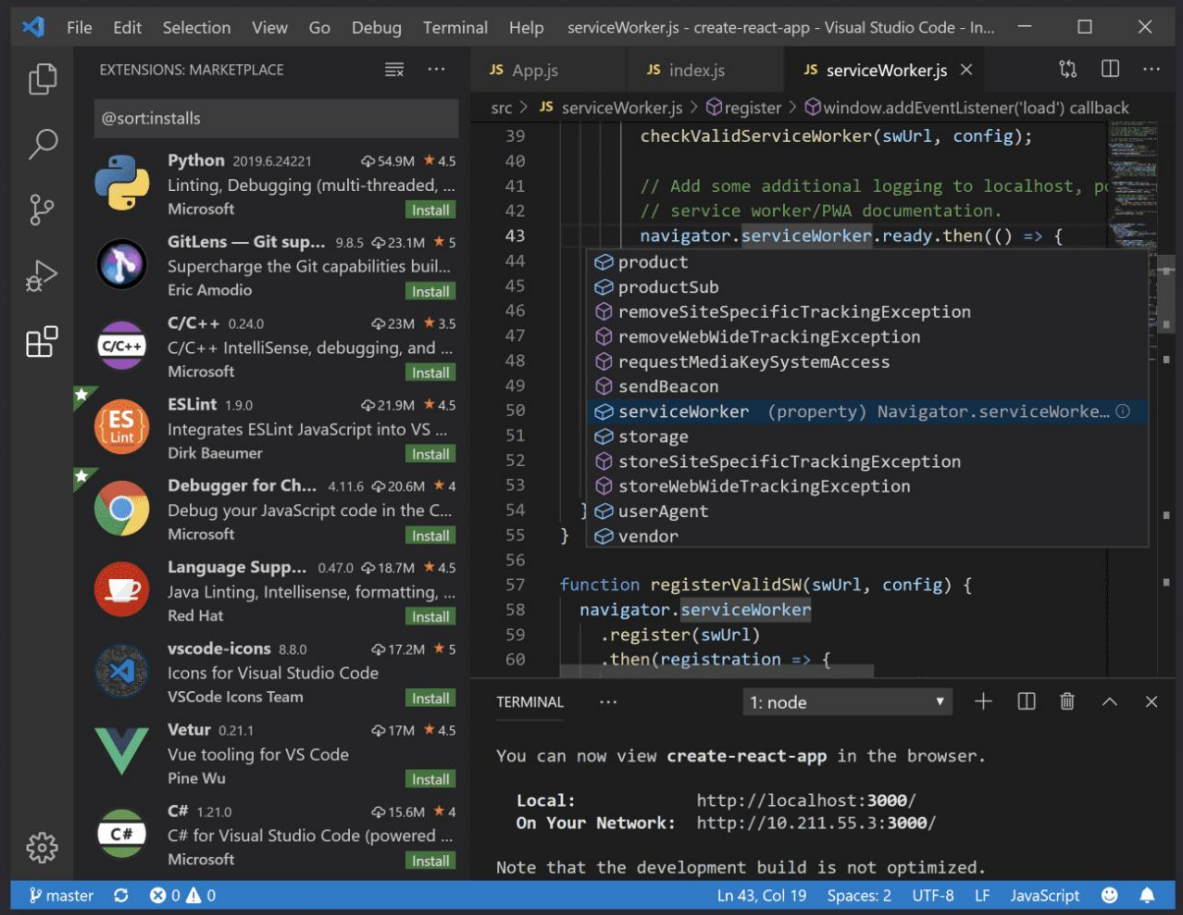
Code editing. Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows
Stable Build

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.



https://code.visualstudio.com/

Visual Studio Code

Visual Studio Remote Debugger

Write code in
deploy, run a

```
pi@dotnetdemo1:~ $ curl -sSL https://aka.ms/getvsdbgsh | /bin/sh /dev/stdin -v l
atest -l ~/vsdbg
Info: Previous installation at '/home/pi/vsdbg' not found
Info: Using vsdbg version '17.4.11017.1'
Info: Creating install directory
Using arguments
    Version                 : 'latest'
    Location                : '/home/pi/vsdbg'
    SkipDownloads           : 'false'
    LaunchVsDbgAfter        : 'false'
    RemoveExistingOnUpgrade : 'false'
Info: Using Runtime ID 'linux-arm'
Downloading https://vsdebugger.azureedge.net/vsdbg-17-4-11017-1/vsdbg-linux-arm
tar.gz
Info: Successfully installed vsdbg at '/home/pi/vsdbg'
```

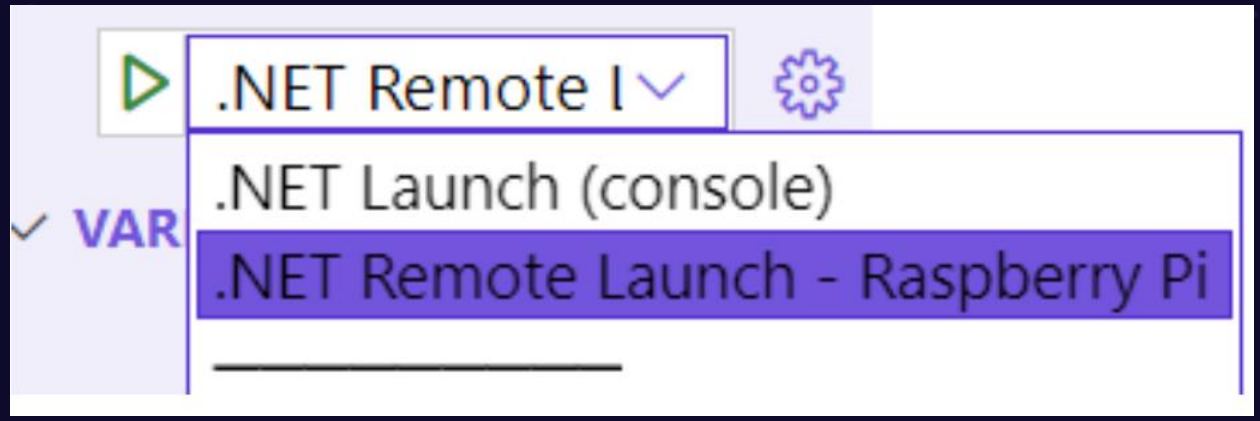
To install the Visual Studio Remote Debugger on your Raspberry pi, run the following command:

curl -sSL https://aka.ms/getvsdbgsh | /bin/sh /dev/stdin -v latest -l ~/vsdbg

Visual Studio Code

Automate Cross Compile and Deploy with Visual Studio code

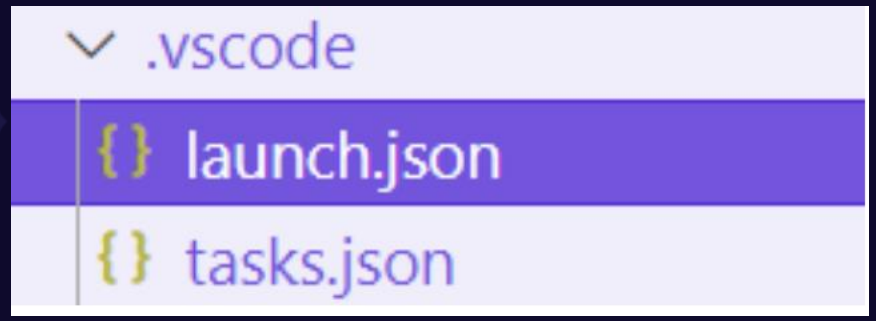
Launch Profiles

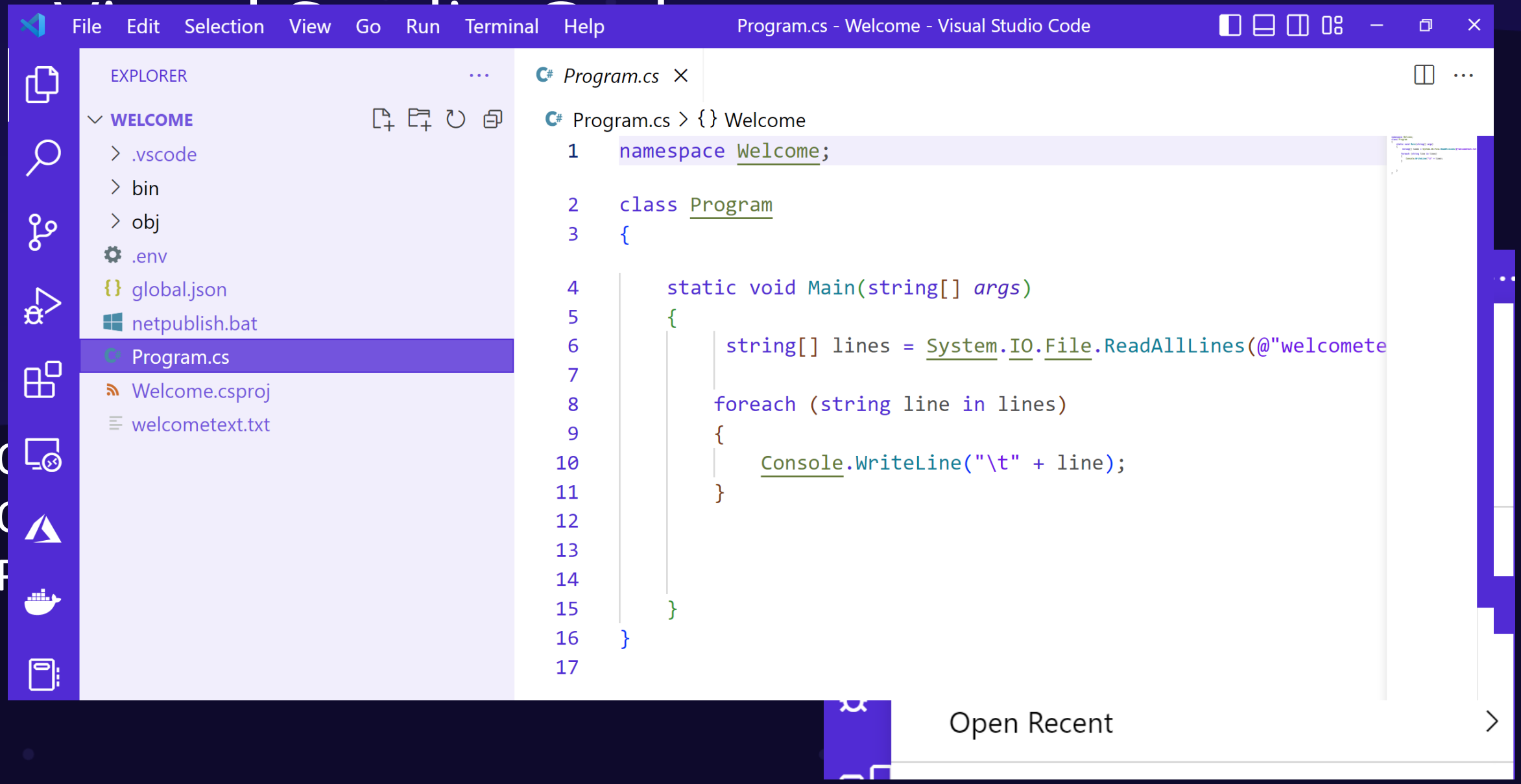


Visual Studio Code

Automate Cross Compile and Deploy with Visual Studio code

To create a launch configuration create a "launch.json" in the .vscode folder.

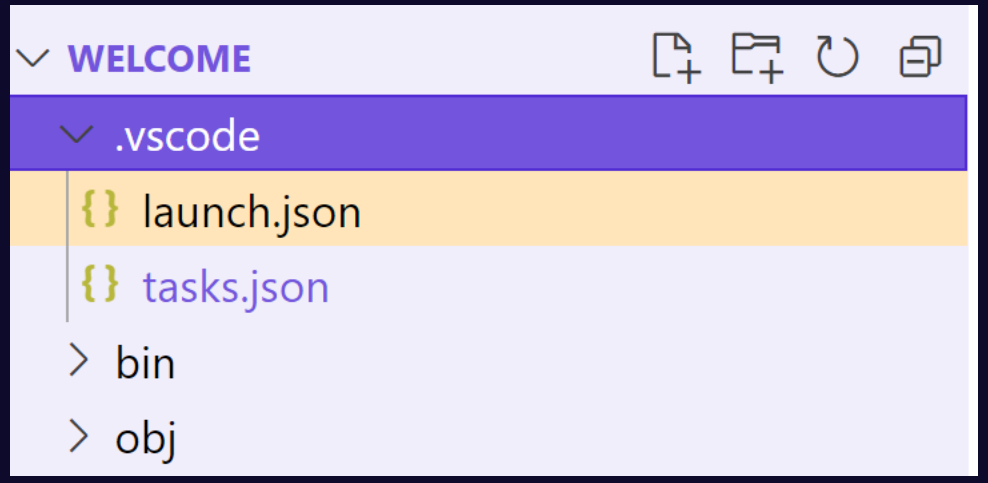



File Edit Selection View Go Run Terminal Help
Program.cs - Welcome - Visual Studio Code
EXPLORER
WELCOME
.vscode
bin
obj
.env
global.json
netpublish.bat
Program.cs
Welcome.csproj
welcometext.txt
Program.cs
Program.cs > { } Welcome
1 namespace Welcome;
2 class Program
3 {
4 static void Main(string[] args)
5 {
6 string[] lines = System.IO.File.ReadAllLines(@"welcomete
7
8 foreach (string line in lines)
9 {
10 Console.WriteLine("\t" + line);
11 }
12
13
14
15 }
16 }
17
Open Recent

Visual Studio Code

Automate Cross Compile and Deploy with Visual Studio code

Open the launch.json



Visual Studio Code

Automate Cross Compile and Deploy with Visual Studio code

Launch.json allows the setting up in one or many launch configurations.

It allows the execution of programs and also different tasks to be executed.

```
{
    "name": ".NET Remote Launch - Raspberry Pi",
    "type": "coreclr",
    "request": "launch",
    "preLaunchTask": "publish",
    "program": "dotnet",
    "args": ["./Welcome.dll"],
    "cwd": "/home/pi/welcome",
    "stopAtEntry": false,
    "console": "internalConsole",
    "pipeTransport": {
        "pipeCwd": "${workspaceRoot}",
        "pipeProgram": "c:\\Program Files\\PuTTY\\plink.exe",
        "pipeArgs": [
            "-pw",
            "${input:PiPassword}",
            "${input:PiHostName}"
        ],
        "debuggerPath": "sudo ~/vsdbg/vsdbg",


    }
}
```

Visual Studio Code

Automate Cross Compile and Deploy with Visual Studio code

Execute the dotnet command
Args parameter to run Welcome.dll
Cwd is the location on the pi
Pipetransport executes via SSH using Putty and plink
DebuggerPath is path to VS Remote Debugger

```
{
    "name": ".NET Remote Launch - Raspberry Pi",
    "type": "coreclr",
    "request": "launch",
    "preLaunchTask": "publish",
    "program": "dotnet",
    "args": ["./Welcome.dll"],
    "cwd": "/home/pi/welcome",
    "stopAtEntry": false,
    "console": "internalConsole",
    "pipeTransport": {
        "pipeCwd": "${workspaceRoot}",
        "pipeProgram": "c:\\Program Files\\PuTTY\\plink.exe",
        "pipeArgs": [
            "-pw",
            "${input:PiPassword}",
            "${input:PiHostName}"
        ],
        "debuggerPath": "sudo ~/vsdbg/vsdbg",


    }
}
```

Visual Studio Code

Automate Cross Compile and Deploy with Visual Studio code

To complete the run experience the app would need to be compile.

prelaunchTasks enable this.

```
{
    "name": ".NET Remote Launch - Raspberry Pi",
    "type": "coreclr",
    "request": "launch",
    "preLaunchTask": "publish",
    "program": "dotnet",
    "args": ["./Welcome.dll"],
    "cwd": "/home/pi/welcome",
    "stopAtEntry": false,
    "console": "internalConsole",
    "pipeTransport": {
        "pipeCwd": "${workspaceRoot}",
        "pipeProgram": "c:\\Program Files\\PuTTY\\plink.exe",
        "pipeArgs": [
            "-pw",
            "${input:PiPassword}",
            "${input:PiHostName}"
        ],
        "debuggerPath": "sudo ~/vsdbg/vsdbg",


    }
}
```

Visual Studio Code

Automate Cross Compile and Deploy with Visual Studio code

To complete the run experience the app would need to be compile.

prelaunchTasks enable this

In this case it's "publish"

```
{
    "name": ".NET Remote Launch - Raspberry Pi",
    "type": "coreclr",
    "request": "launch",
    "preLaunchTask": "publish",
    "program": "dotnet",
    "args": ["./Welcome.dll"],
    "cwd": "/home/pi/welcome",
    "stopAtEntry": false,
    "console": "internalConsole",
    "pipeTransport": {
        "pipeCwd": "${workspaceRoot}",
        "pipeProgram": "c:\\Program Files\\PuTTY\\plink.exe",
        "pipeArgs": [
            "-pw",
            "${input:PiPassword}",
            "${input:PiHostName}"
        ],
        "debuggerPath": "sudo ~/vsdbg/vsdbg",


    }
}
```

Visual Studio Code

Tasks.json allows the setting up one or many tasks to executed. These tasks can also be configured with task dependencies.

```
{
    "label": "build",
    "command": "dotnet",
    "type": "shell",
    "args": [
        "build",
        // Ask dotnet build to generate full paths for file names.
        "/property:GenerateFullPaths=true",
        // Do not generate summary otherwise it leads to duplicate errors in Problems panel
        "/consoleloggerparameters:NoSummary"
    ],
    "group": "build",
    "presentation": {
        "reveal": "silent"
    },
    "problemMatcher": "$msCompile"
},
{
    "label": "publish",
    "type": "shell",
    "dependsOn": "build",
    "presentation": {
        "reveal": "always",
        "panel": "new"
    },
    "options": {
        "cwd": "${workspaceFolder}"
    },
    "windows": {
        "command": "${cwd}\\netpublish.bat",
        "args": ["${input:WinscpSite}"]
    },
```

Visual Studio Code

Publish task is configured to call a batch file, "netpublish.bat". This batch file ensures the built app is built and published as an arm binary.
Winscp synchronizes the binaries between the local built code and the remote Raspberry Pi.

netpublish.bat

```
1   SET currentPath=%cd%
2   ECHO %currentPath%
3   SET buildPath=%currentPath%\bin\Debug\net6.0\linux-arm\publish
4   echo %buildPath%
5   SET command=synchronize remote %buildPath% /home/pi/welcome
6   echo command %command%
7
8   dotnet publish -r linux-arm /p:ShowLinkerSizeComparison=true --self-contained
9   winscp %1 /command "%command%" "exit"
10
```

Visual Studio Code

Publish task has a dependency of the **build** task.

```
{
    "label": "build",
    "command": "dotnet",
    "type": "shell",
    "args": [
        "build",
        // Ask dotnet build to generate full paths for file names.
        "/property:GenerateFullPaths=true",
        // Do not generate summary otherwise it leads to duplicate errors in Problems panel
        "/consoleloggerparameters:NoSummary"
    ],
    "group": "build",
    "presentation": {
        "reveal": "silent"
    },
    "problemMatcher": "$msCompile"
},
{
    "label": "publish",
    "type": "shell",
    "dependsOn": "build",
    "presentation": {
        "reveal": "always",
        "panel": "new"
    },
    "options": {
        "cwd": "${workspaceFolder}"
    },
    "windows": {
        "command": "${cwd}\\netpublish.bat",
        "args": ["${input:WinscpSite}"]
    },
```

Visual Studio Code

Launch.json and **tasks.json** supports parameters

Parameters

Pi HostName
Pi Password
WinScp Site Name

Configurable via **.env** file

```
"inputs": [
    {
      "id": "PiHostName",
      "type": "command",
      "command": "extension.commandvariable.file.content",
      "args": {
        "fileName": "${workspaceFolder}/.env",
        "key": "PiHostName",
        "default": ""
      }
    },
    {
      "id": "PiPassword",
      "type": "command",
      "command": "extension.commandvariable.file.content",
      "args": {
        "fileName": "${workspaceFolder}/.env",
        "key": "PiPassword",
        "default": ""
      }
    }
```

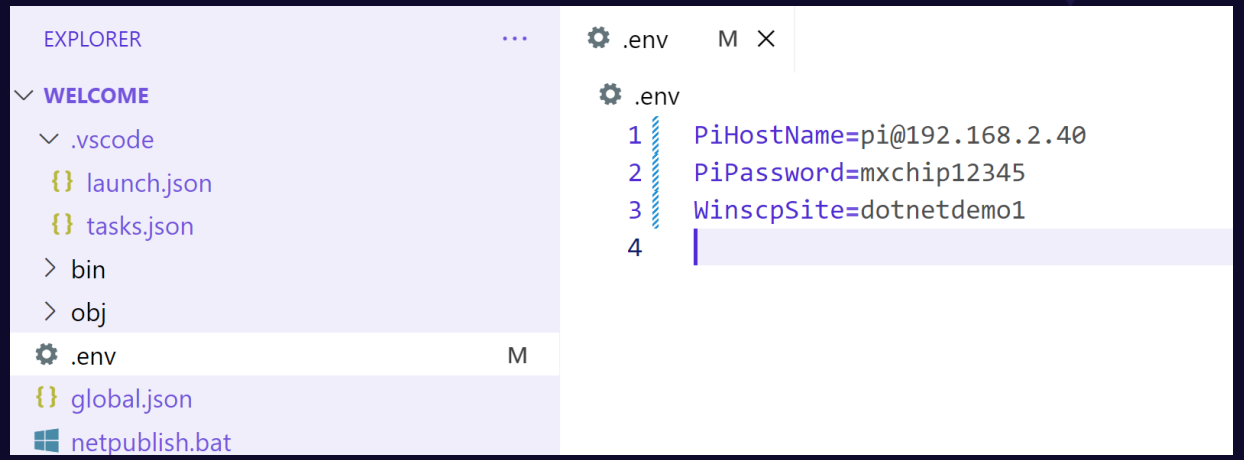
Visual Studio Code

Reuse these preconfigured **Launch.json** and **tasks.json** in all your Raspberry Pi projects to easily automate remote deployment and debugging

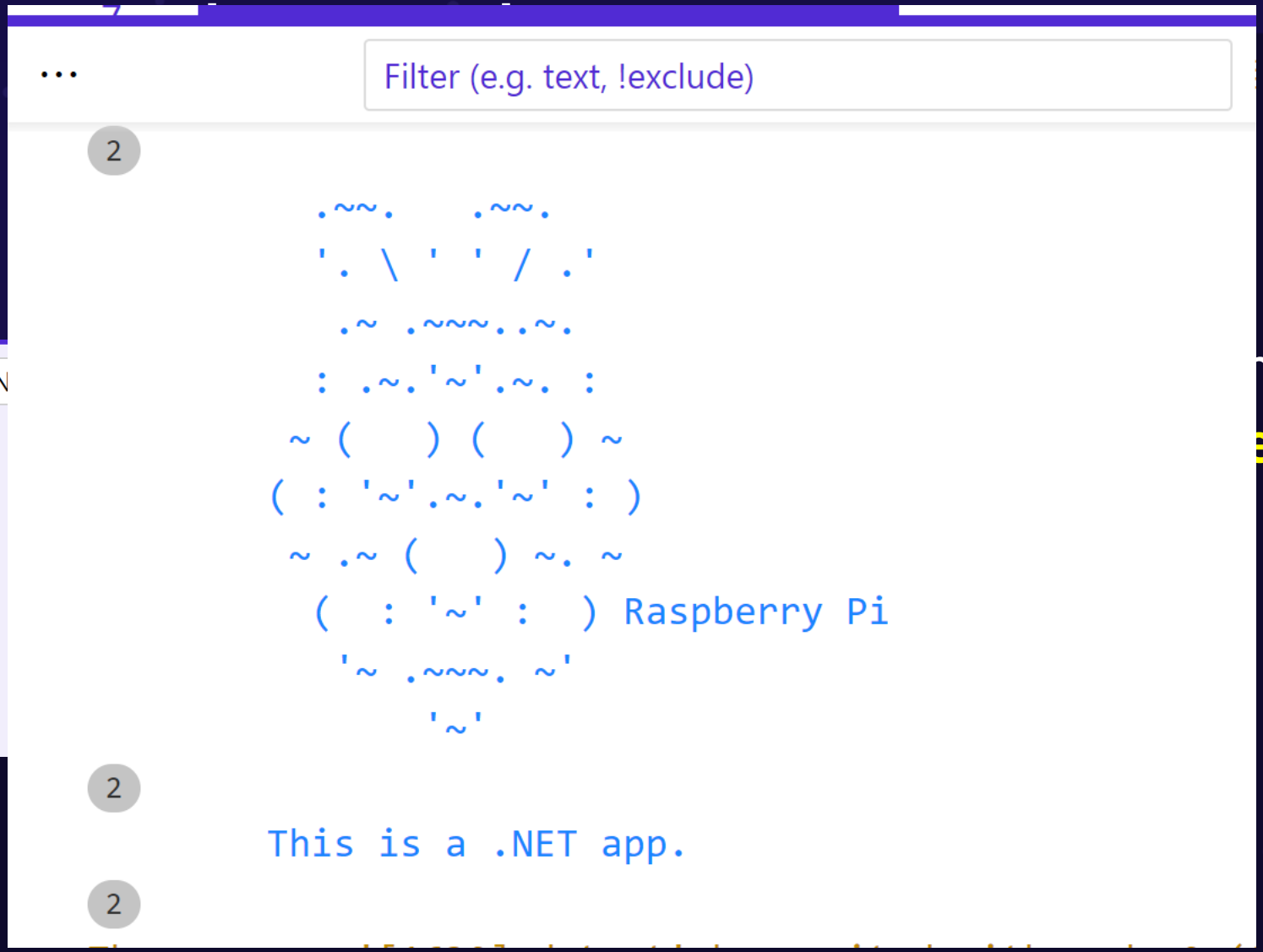
Just edit the **.env** file and code

Try it out!!

Edit the .env file and add your Pi's details.



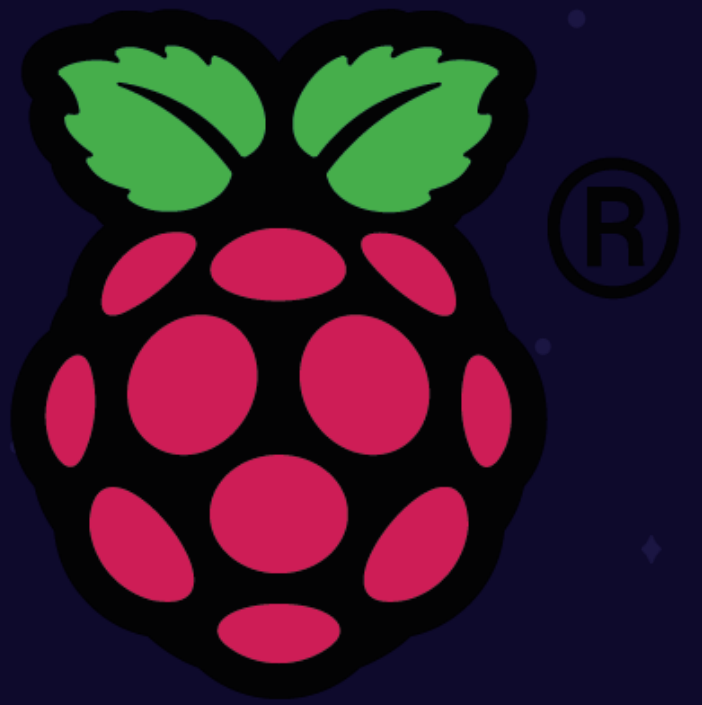
Visual



.NET

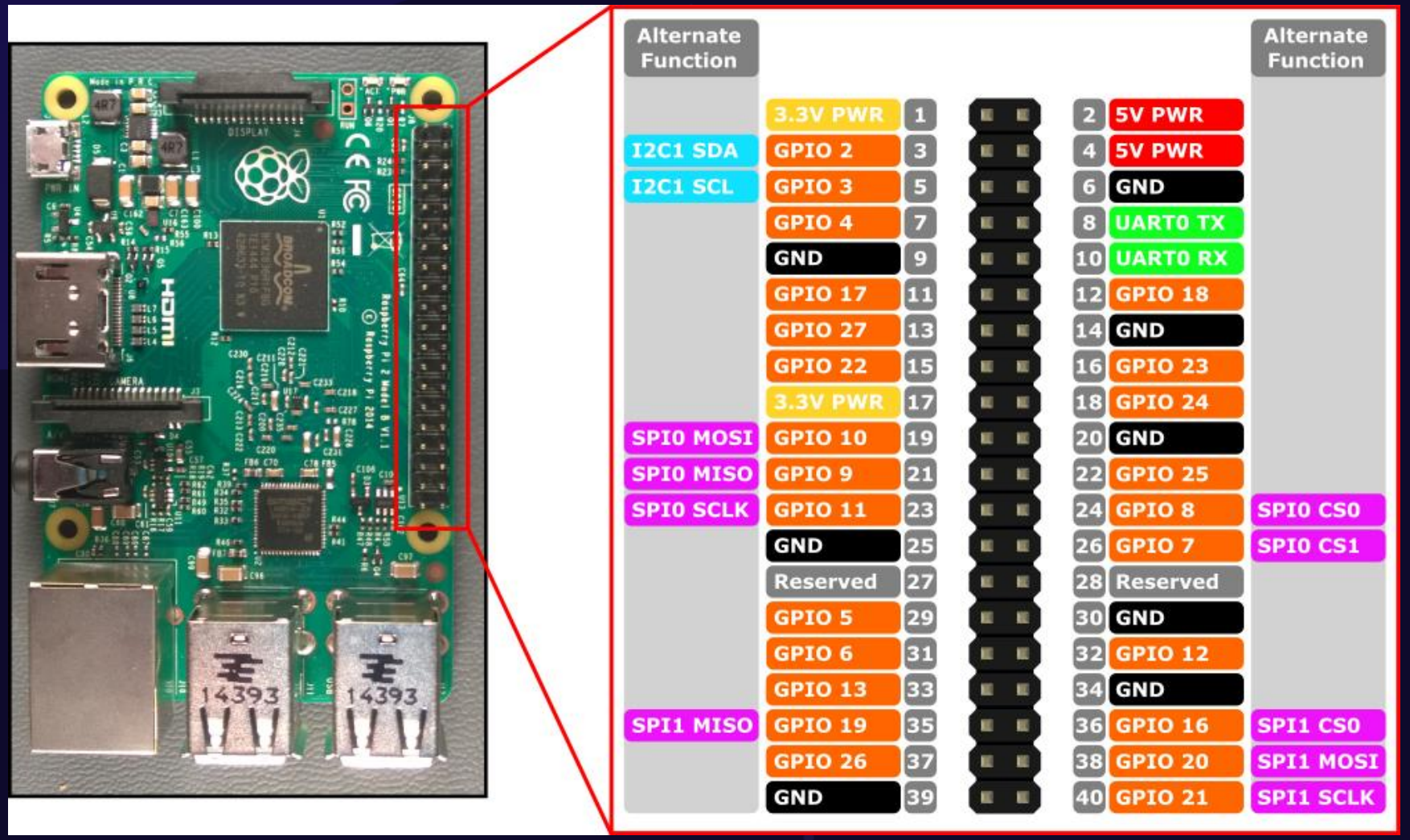
Pi

Module 2



Getting started with Hardware and Sensors connected to a Raspberry Pi with .NET

Interfacing to the Raspberry Pi



Interfacing to the Raspberry Pi

GPIO - General-purpose input/output

Switching things on, switching things off

Analog

No just 1 (on) and 0 (off)
Variable Input and Output voltage.
Raspberry Pi has no Analog pins. Requires an Analog to Digital Converter

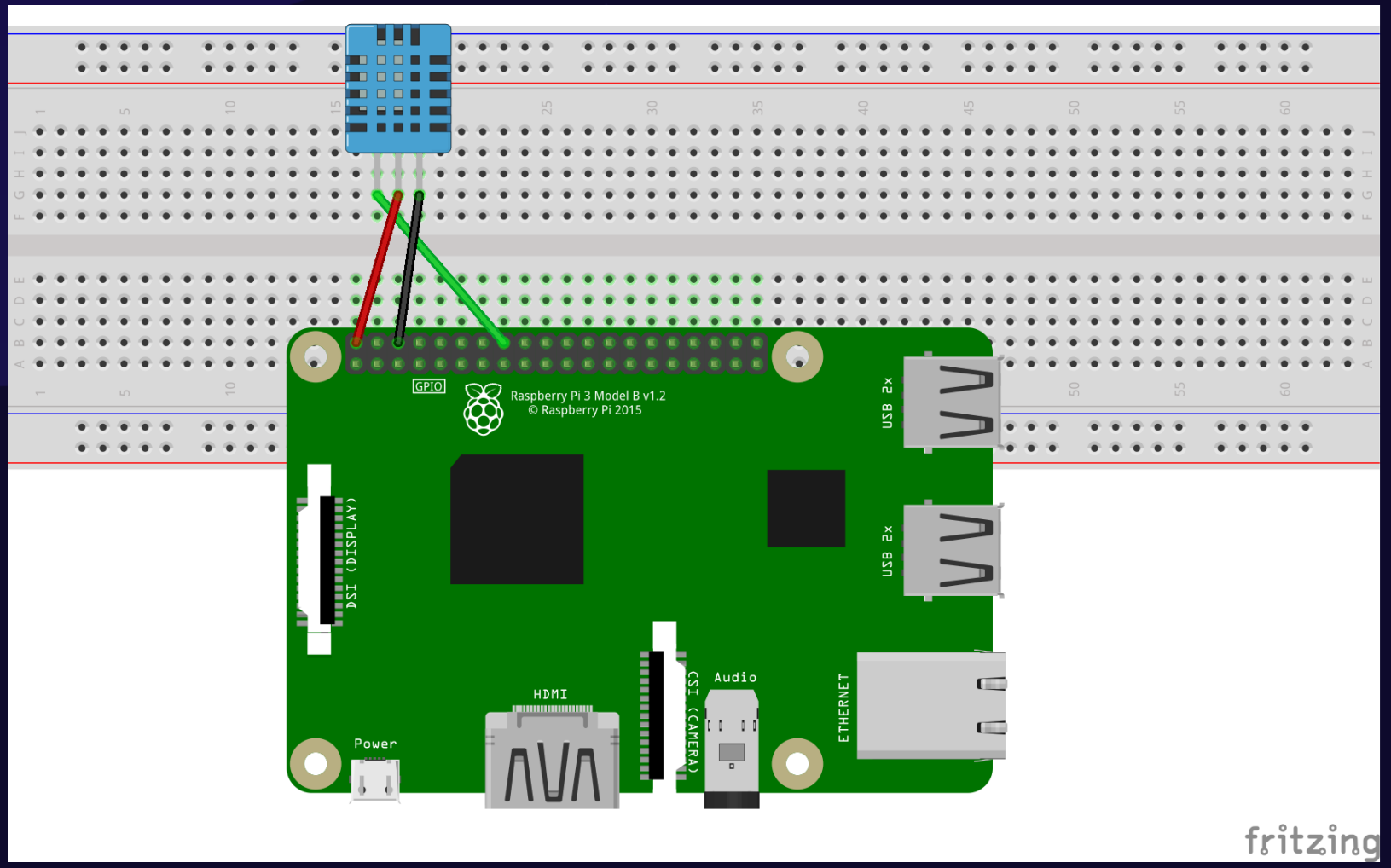
I2C

Serial communication Bus
Handy as it’s usually 2 wires for complex data communication

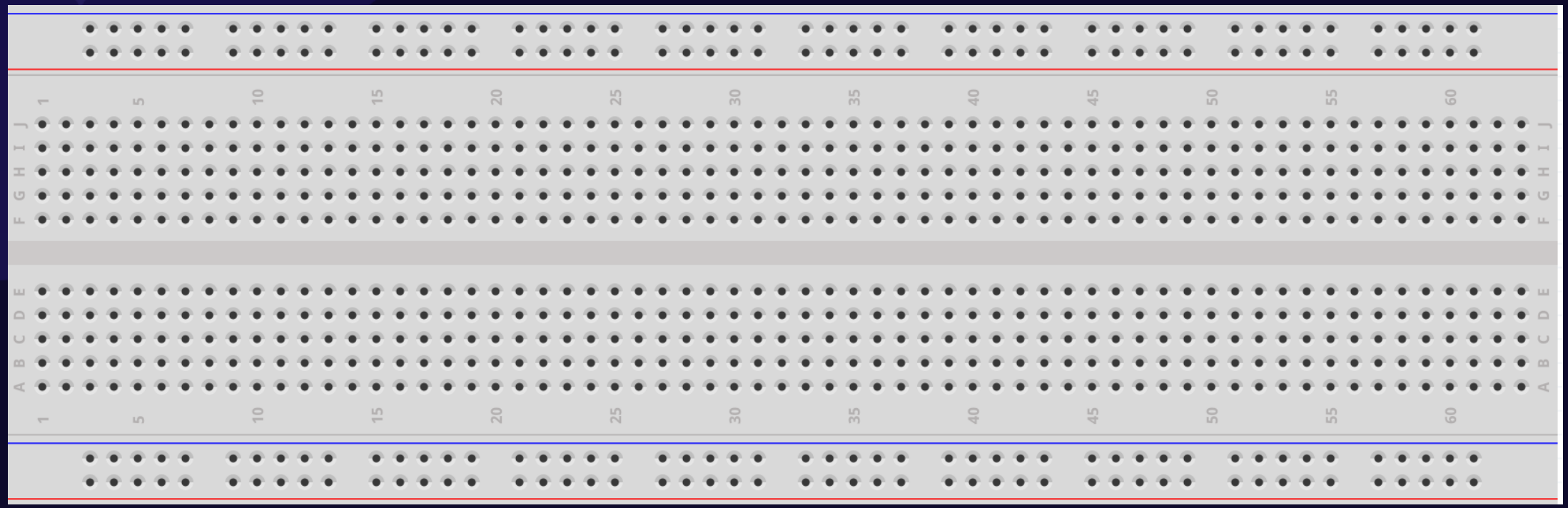
SPI (Serial Peripheral Interface)

Serial communication bus
Usually 3 wire communication

Let’s Build an Environment Monitor



The breadboard



The breadboard

Pins Connected Horizontally

Pins connected vertically

Divider (splits board electrically)

Pins connected vertically

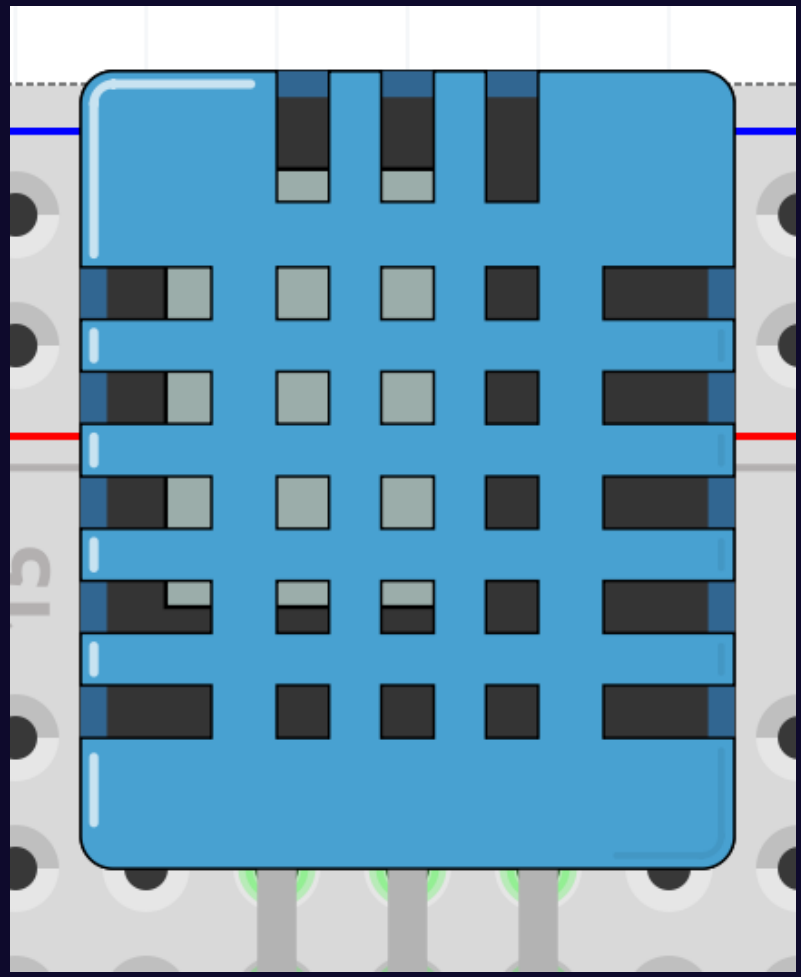
Pins Connected Horizontally

Used only for experimenting and prototyping

DHT 11

Temperature and humidity sensor

Serial Data



DHT 11

Pin 2 - Vcc

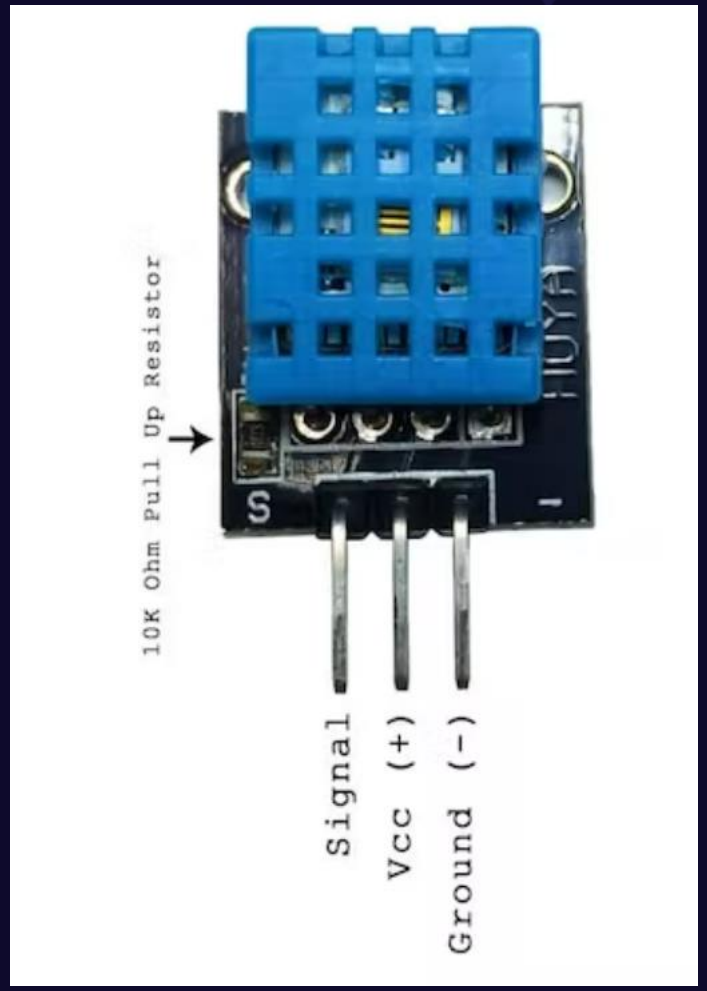
Power supply 3.5V to 5.5V

Pin 1 - Data

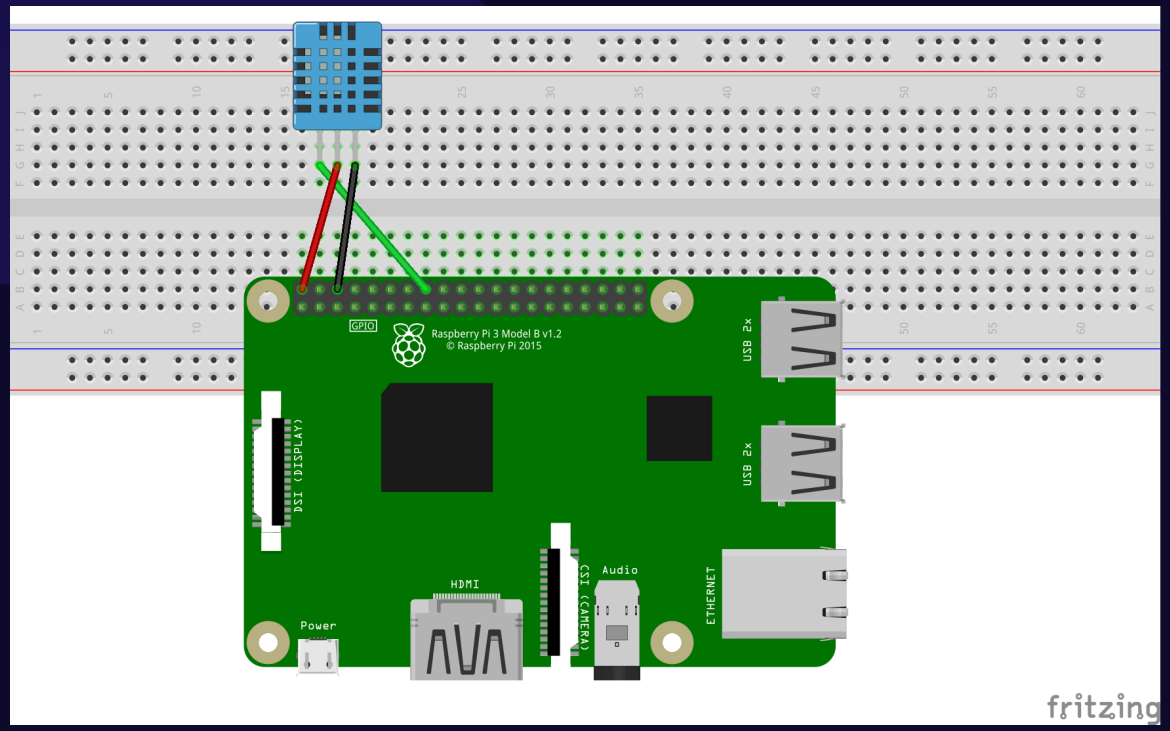
Outputs both Temperature and Humidity through serial Data

Pin 3 - Ground

Connected to the ground of the circuit



Try it out!



Using the component pack, build the circuit using your Raspberry Pi, breadboard, DHT11 and wires.

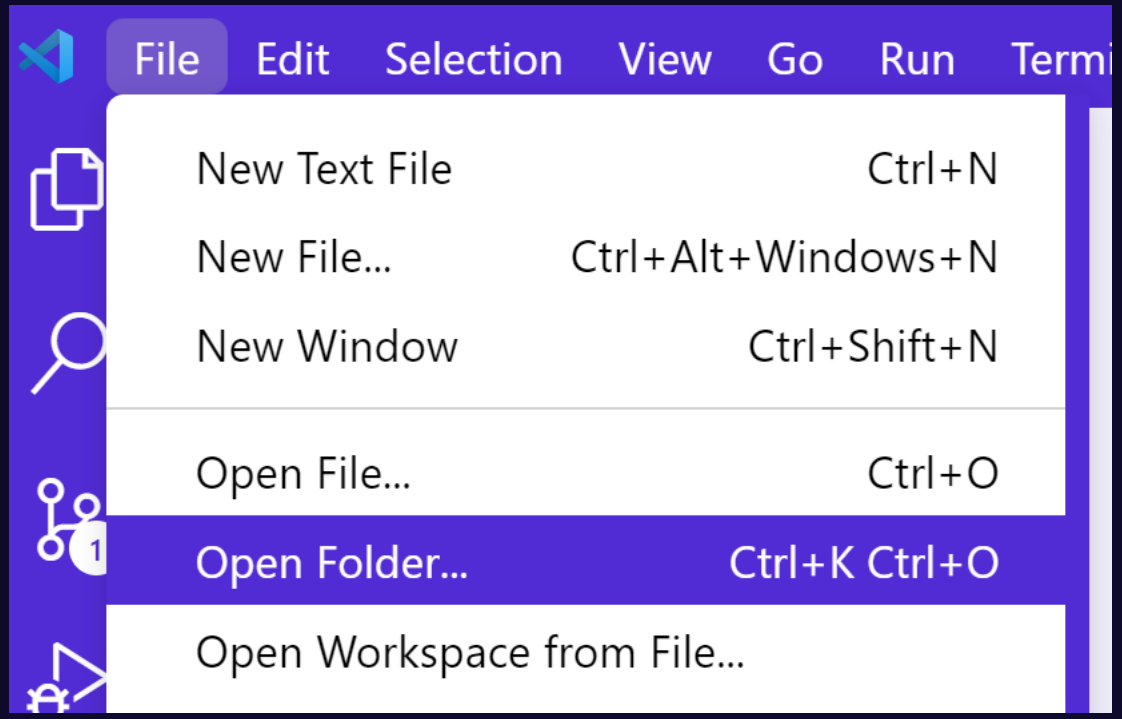
5V from the Raspberry Pi to the middle pin of the DHT 11 Sensor (VCC)

Ground from the Raspberry Pi to the right pin of the DHT Sensor (Ground)

GPIO Pin 23 from the Raspberry Pi to the left pin on the DHT Sensor (Data / Signal)

Try it out!

In VS Code open the folder SensorStart from the code cloned from Github

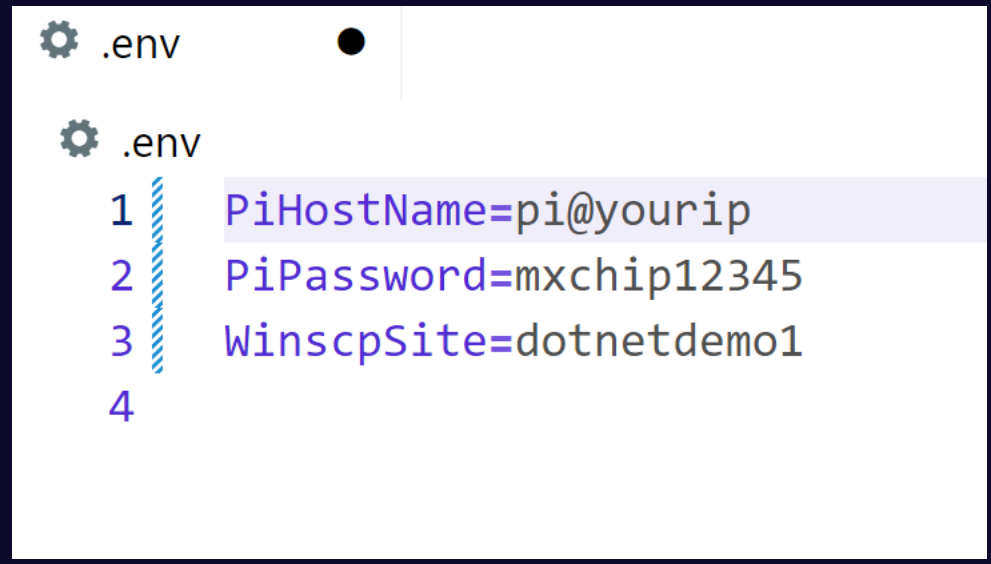


Try it out!

The SensorStart is a starter project to write the code to read the Temperature and Humidity

Don’t forget to edit the .env file with the details of your Raspberry Pi

Test to see if it deploys before you start coding



Try it out!

The **Iot.Device.Bindings** package has already been included for you

Definition

Namespace: Iot.Device.DHTxx

Assembly: Iot.Device.Bindings.dll

Package: Iot.Device.Bindings v2.2.0

Temperature and Humidity Sensor DHT11

https://learn.microsoft.com/en-us/dotnet/api/iot.device.dhtxx.dht11?view=iot-dotnet-latest

```
<PackageReference
Include="Iot.Device.Bindings"
Version="2.2.0" />
```

Try it out!

Coding Challenge

Read the Temperature and Humidity repetitively every 5 seconds and display it on the Console (Console.Writeline).

Hints

To create the Dht11 class

```
Dht11 sensor = new Dht11(23);
```

To attempt to read the Temperature

```
sensor.TryReadTemperature(out
lastTempReading)
```

https://learn.microsoft.com/en-us/dotnet/api/iot.device.dhtxx.dht11?view=iot-dotnet-latest

```
<PackageReference
Include="Iot.Device.Bindings"
Version="2.2.0" />
```

Module 3

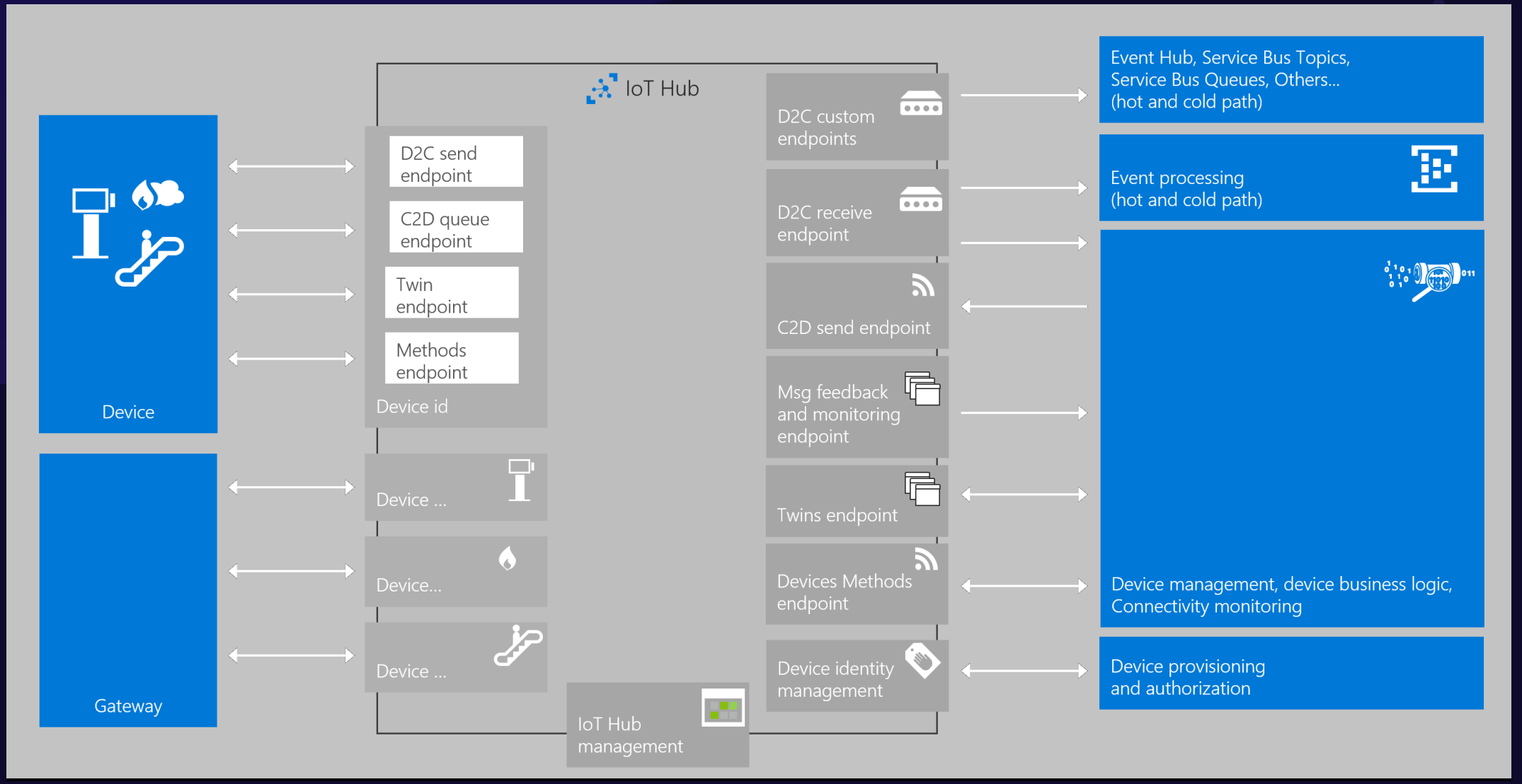
Azure IoT Central

Getting Started with Azure IoT with Azure IoT Central

Azure IoT Hubs

- **Establish bi-directional communication with billions of IoT devices**
- **Work with familiar platforms and protocols (standard and custom protocols, including HTTP, Advanced Message Queuing Protocol (AMQP), and MQ Telemetry Transport (MQTT)**
- **Authenticate per device for security-enhanced IoT solutions**
- **Device Management**

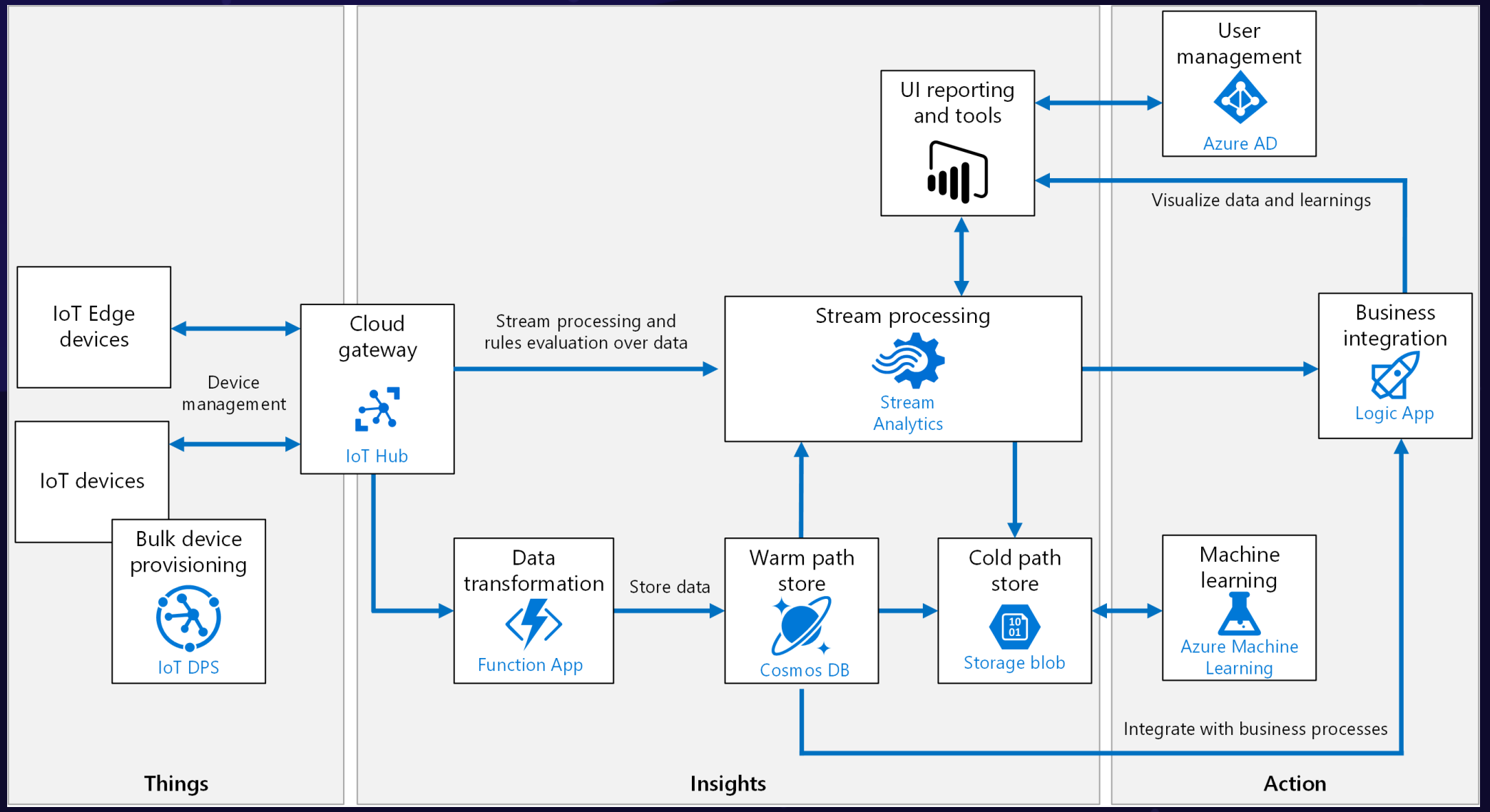
Azure IoT Hubs



Device SDKs platform/OS support

- Android (Java or Xamarin)
- Arduino
- Debian Linux (v 7.5+)
- ESP8266
- Fedora Linux (v 20+)
- FreeRTOS
- iOS (Xamarin)
- mbed OS (v 2.0+)
- OpenWRT
- Raspbian Linux (v 3.18+)
- STM32
- TI RTOS
- Ubilinux (v3.0+)
- Ubuntu Linux (v 14.04+)
- Windows Desktop (7, 8, 10)
- Windows IoT Core (v 10)
- Windows Server (v 2012 R2+)
- Yocto Linux (v 2.1+)
- Azure Sphere
- … and more

Azure IoT Reference Architecture



IoT Solutions – IoT Central

Fast and easy

- Start in minutes and create a finished solution in hours
- Build without any cloud development expertise

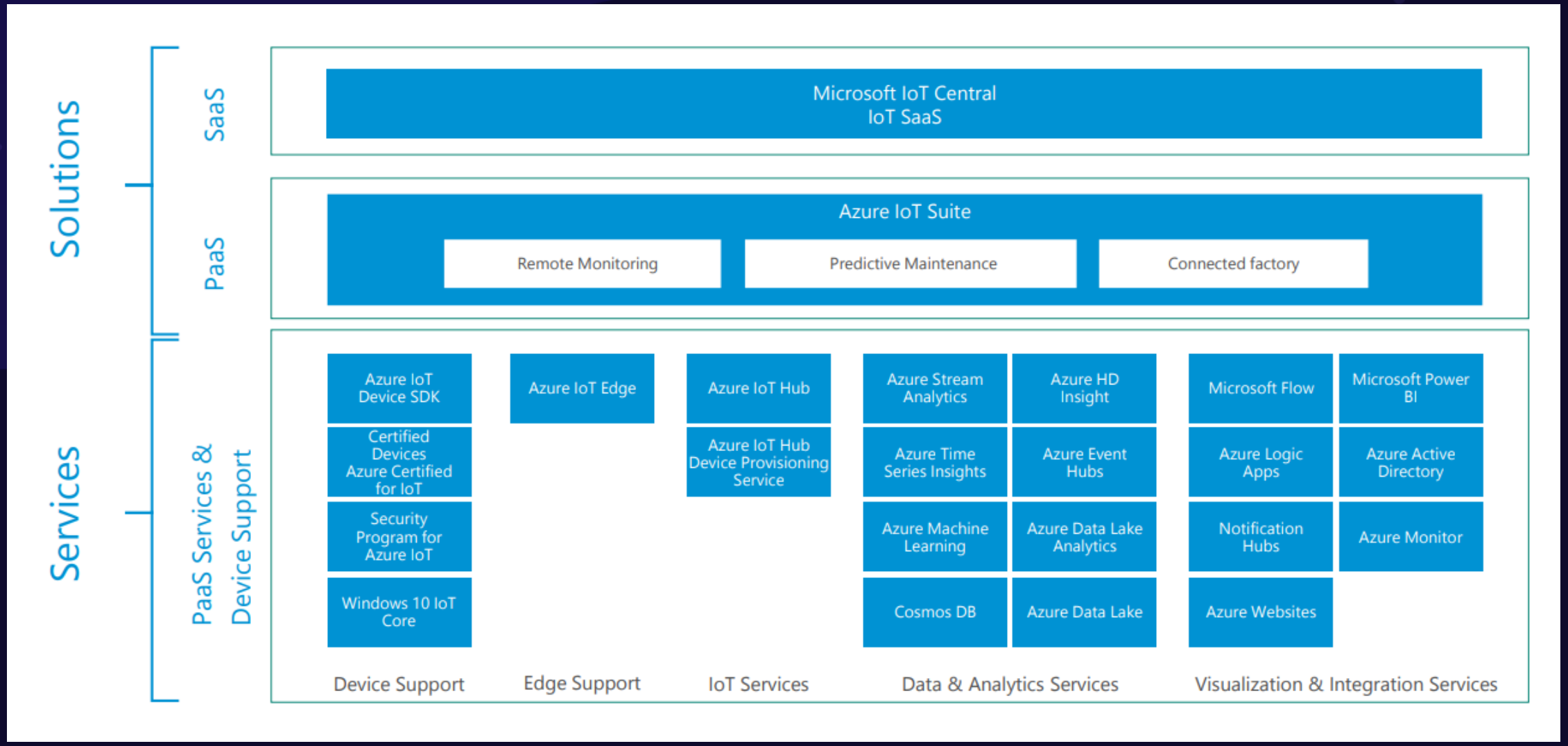
Scalable and secure

- Connect your devices at any scale without worrying about infrastructure
- Get best-in-class security

Enterprise grade

- Built on proven enterprise-grade Azure services
- Full integration into your existing business systems and processes

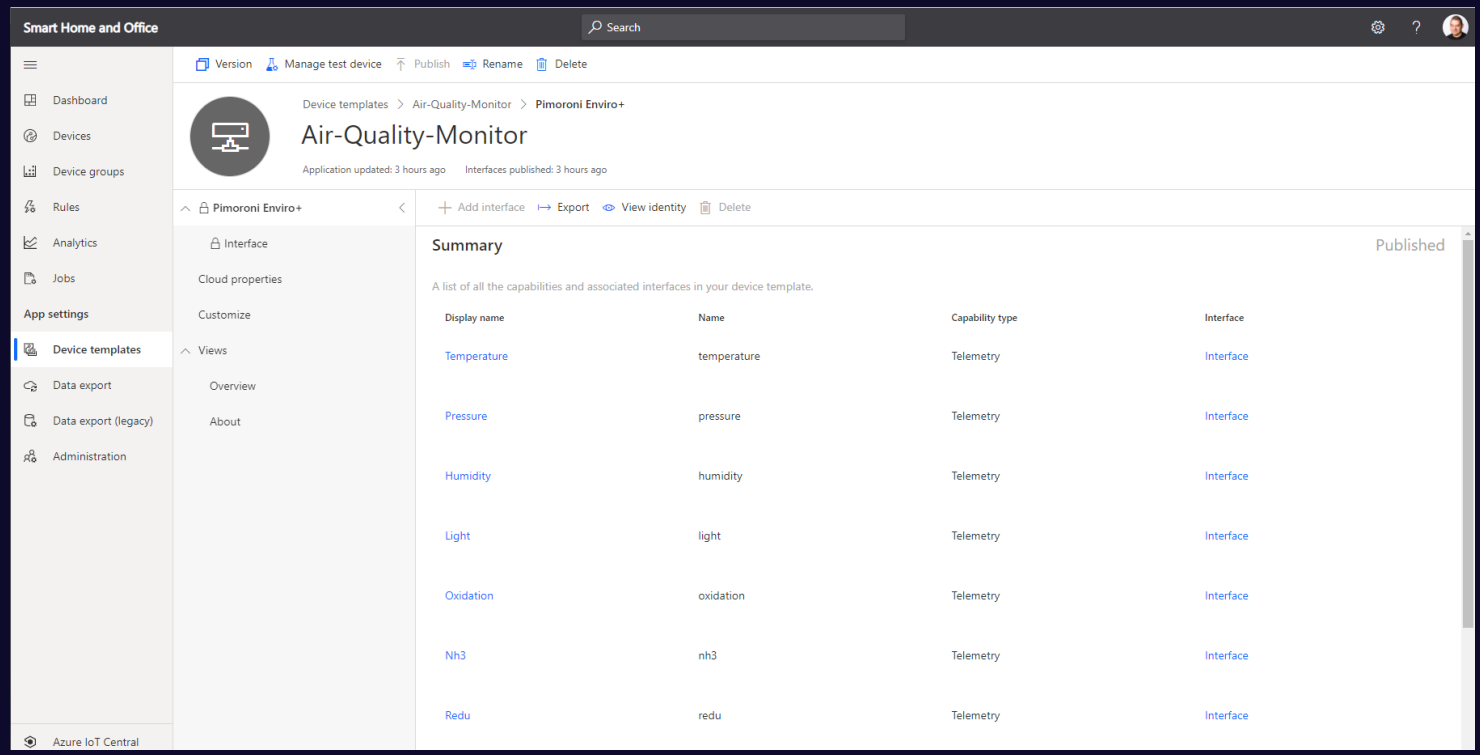
IoT Central


Solutions
SaaS
Microsoft IoT Central
IoT SaaS
PaaS
Azure IoT Suite
Remote Monitoring
Predictive Maintenance
Connected factory
Services
PaaS Services & Device Support
Azure IoT Device SDK
Certified Devices Azure Certified for IoT
Security Program for Azure IoT
Windows 10 IoT Core
Azure IoT Edge
Azure IoT Hub
Azure IoT Hub Device Provisioning Service
Azure Stream Analytics
Azure HD Insight
Azure Time Series Insights
Azure Event Hubs
Azure Machine Learning
Azure Data Lake Analytics
Cosmos DB
Azure Data Lake
Microsoft Flow
Microsoft Power BI
Azure Logic Apps
Azure Active Directory
Notification Hubs
Azure Monitor
Azure Websites
Device Support
Edge Support
IoT Services
Data & Analytics Services
Visualization & Integration Services

IoT Central - Device Templates

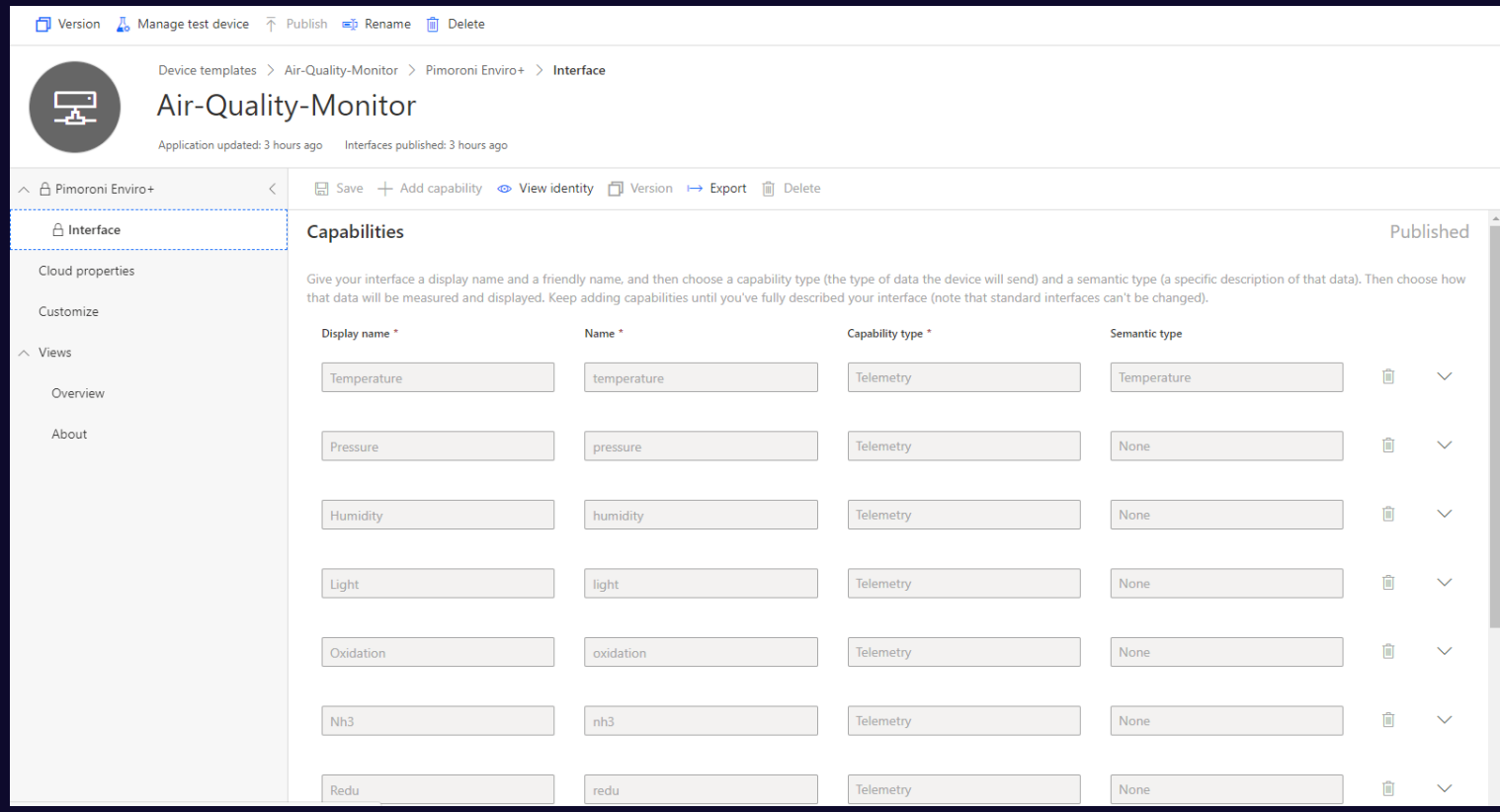
- **Model a Device without writing code**
- **Model Telemetry**
- **Model Device Commands**
- **Model Device Properties**
- **Simulate Devices**
- **Versionable Devices**

- **Supports IoT Plug ‘n Play**



Device Templates - Capabilities

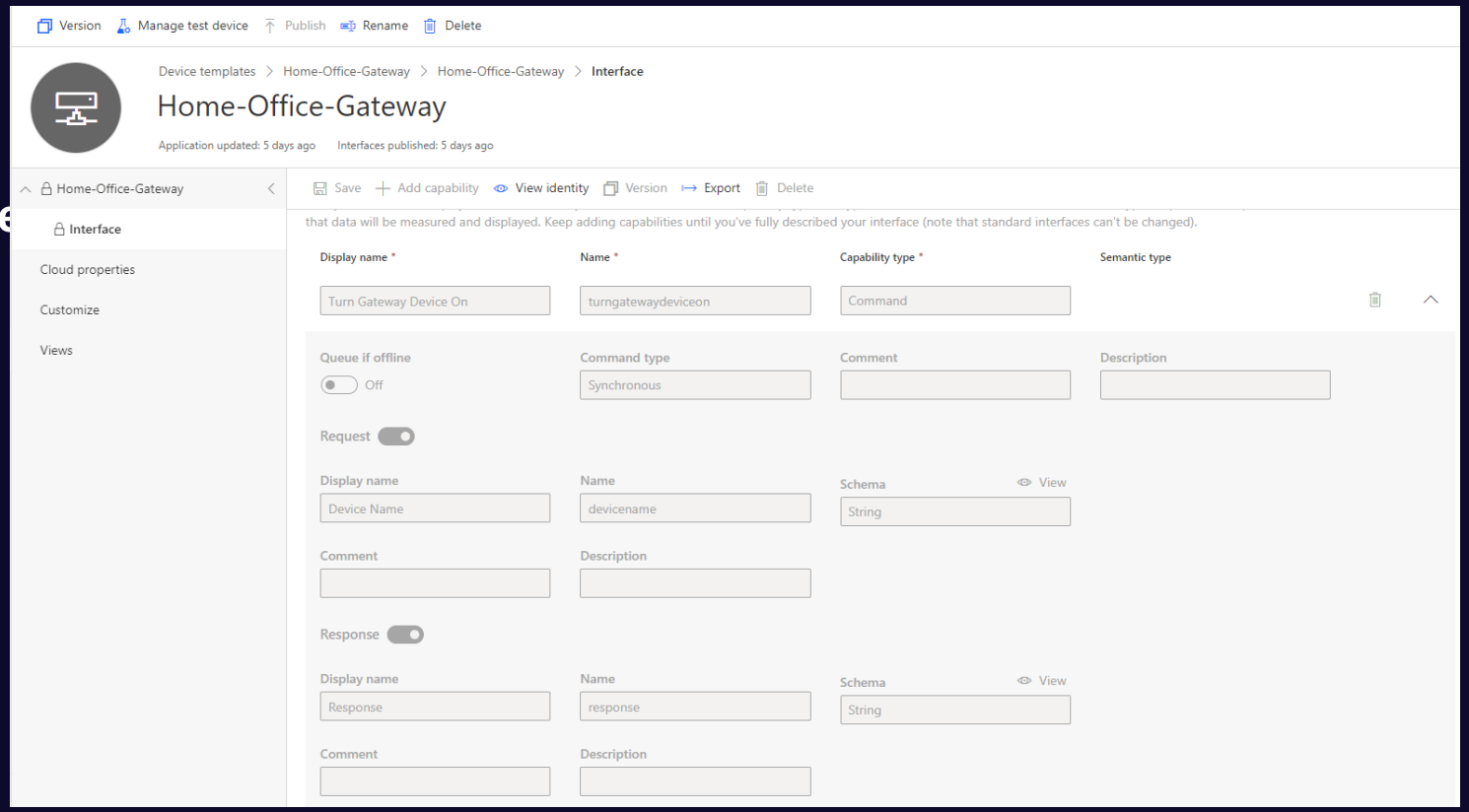
- **Model the device sensors or commands**
- **Model types and units of telemetry item**
- **Versionable Interface**



Commands

- **Model Commands to be invoked on Device**
- **Model Request / input parameters**
- **Model Response / output parameters**

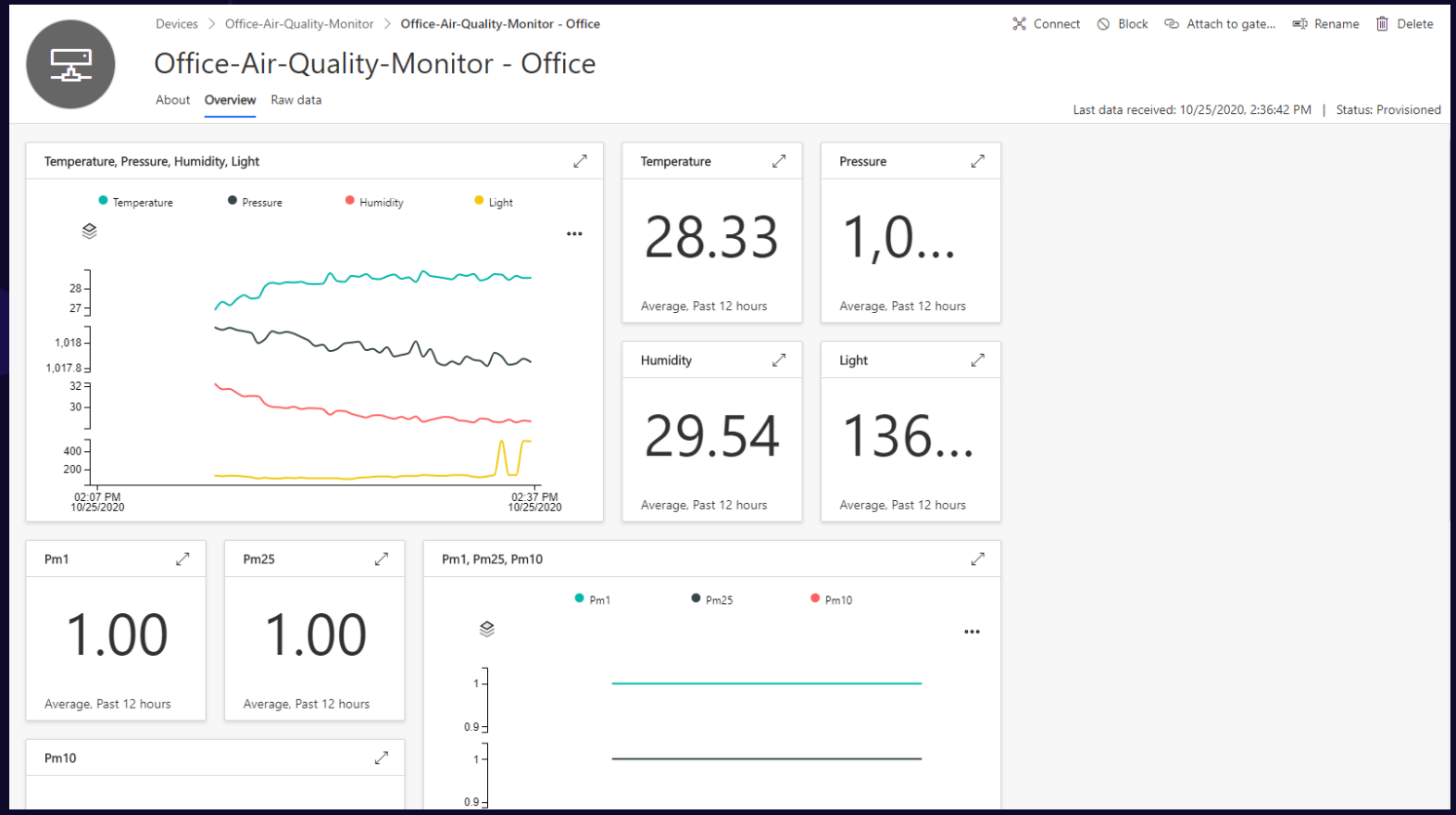
- **Versionable Interface**



Dashboards

- **Device Dashboards**
- **Customize Dashboards**
- **Customize Dashboards per user**

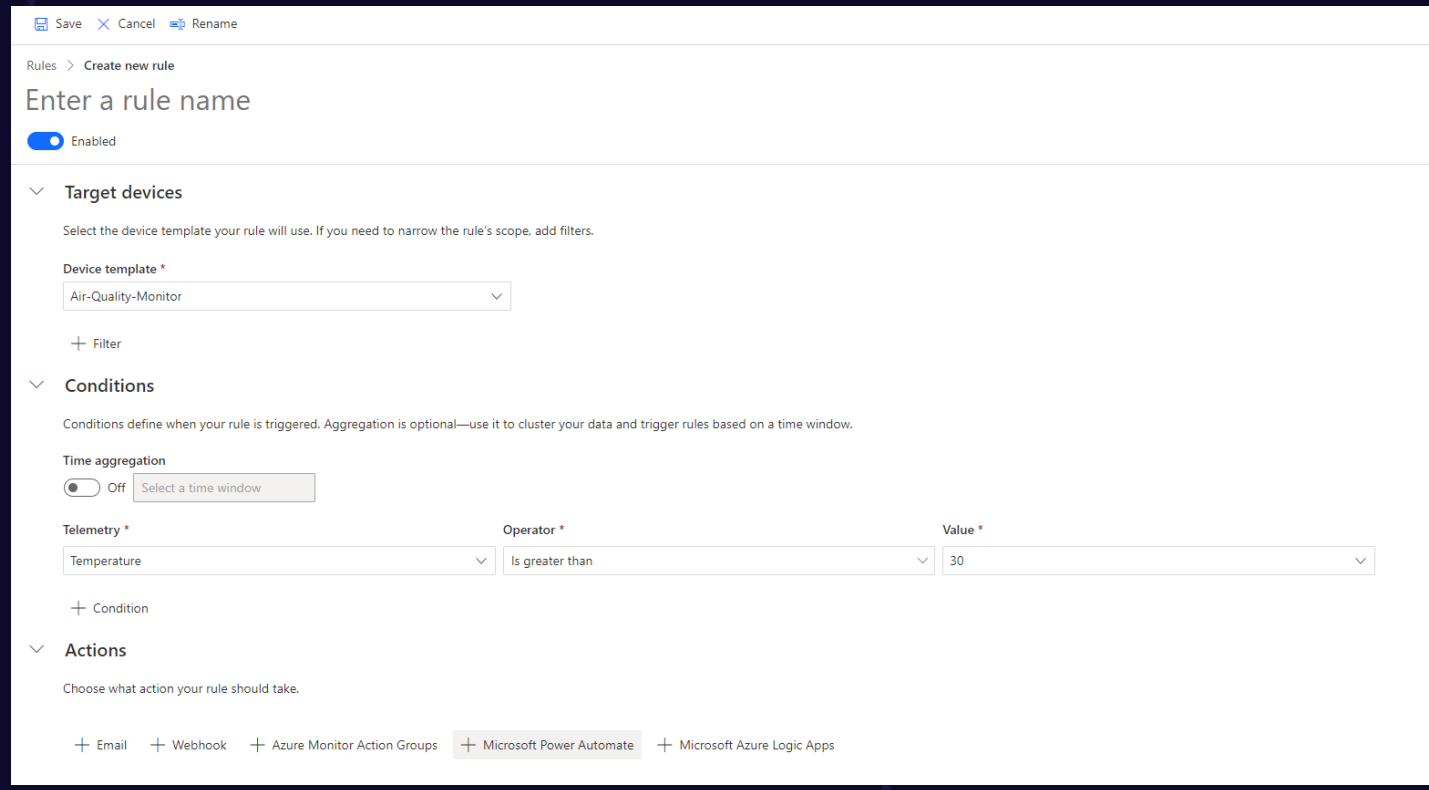
- **Analytics Views**



Rules

- **Create rules based on conditions of modeled capabilities**

- **Performs Actions based on conditions met**
  - **Email**
  - **Webhook**
  - **Power Automate**
  - **Logic Apps**
  - **Azure Monitor Action Groups**

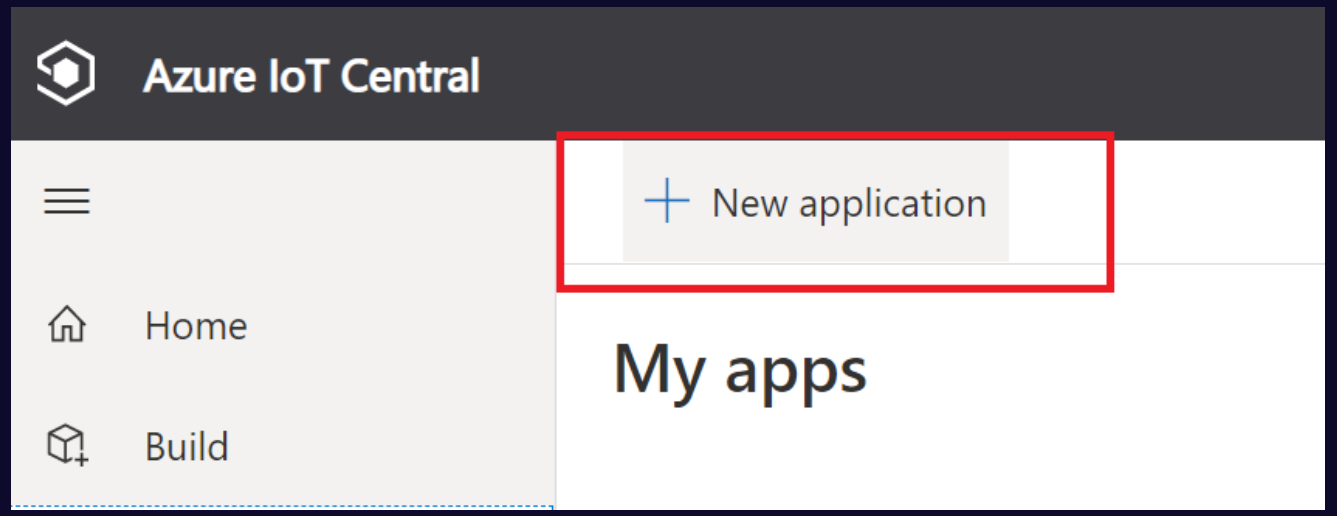


Try it Out!

Log into Azure IoT Central

https://apps.azureiotcentral.com/

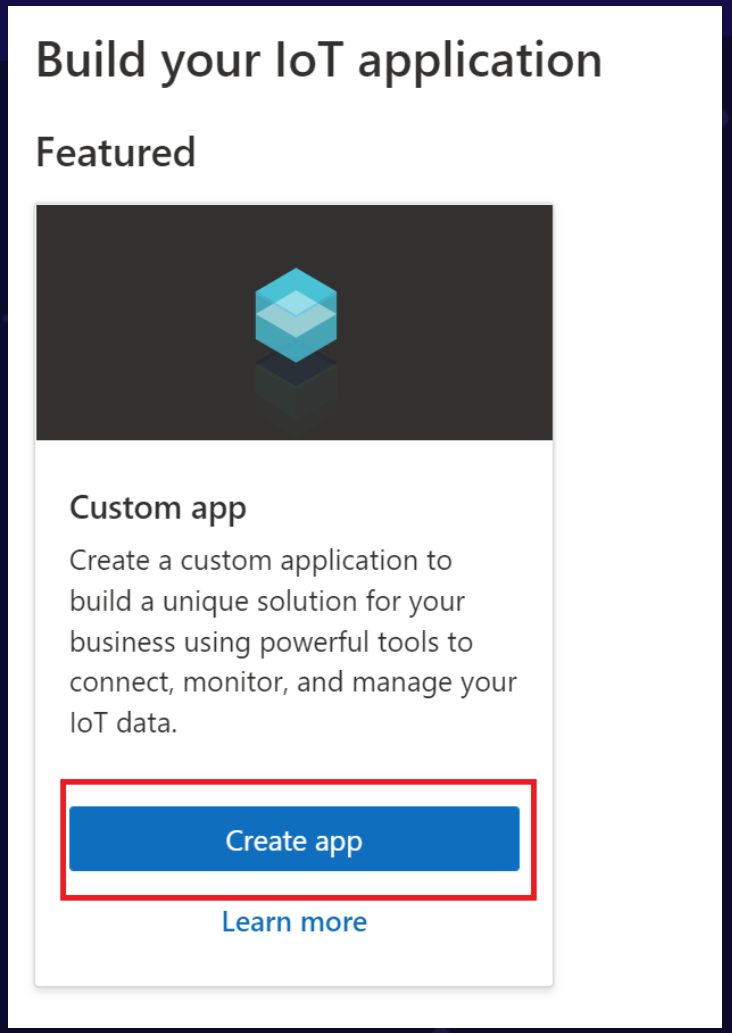
Create a New Application



This Requires an Azure Subscription!

Try it Out!

Create Custom App

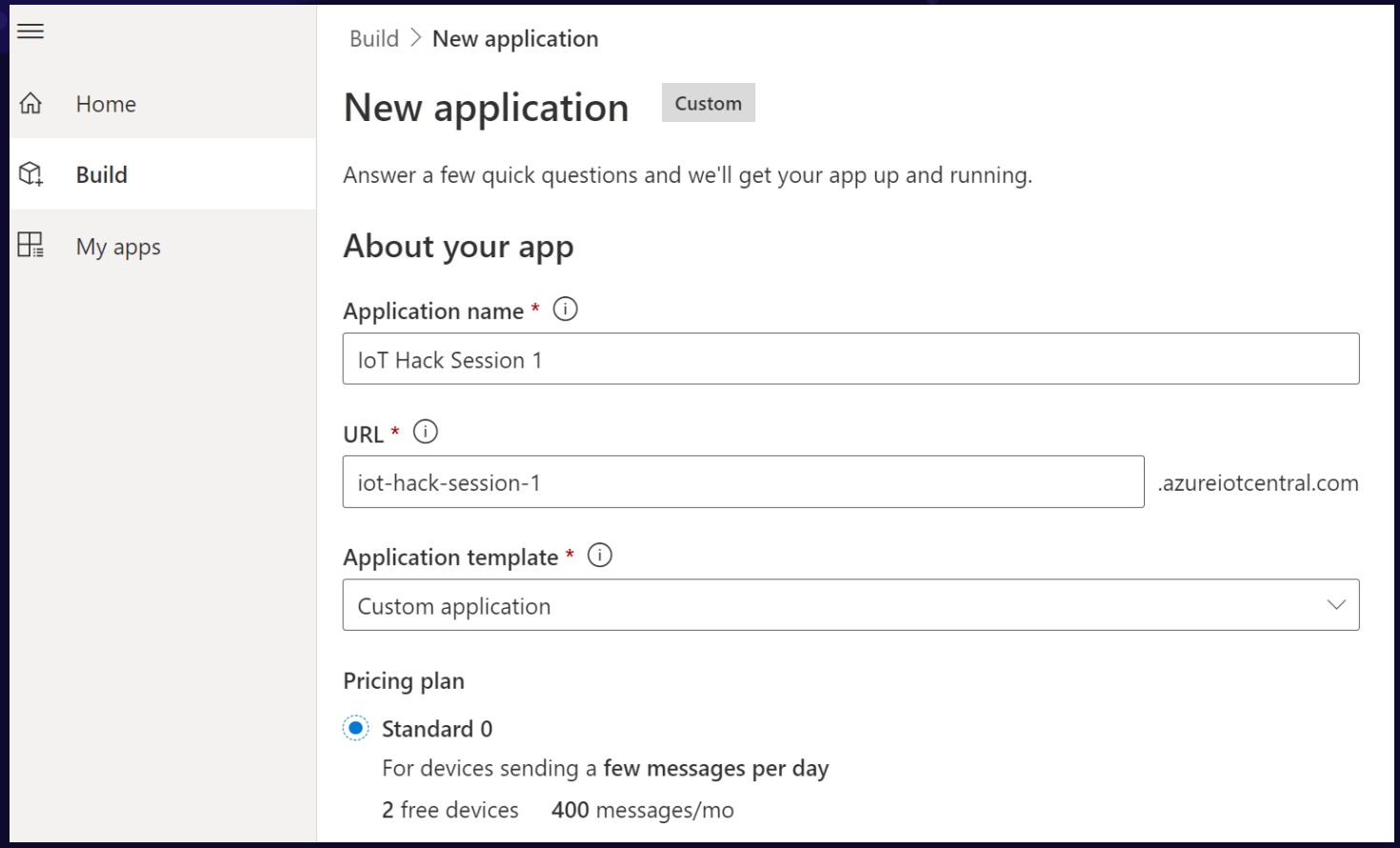


This Requires an Azure Subscription!

Try it Out!

Give your Application a Name

Chose the Standard 0 Price Plan (2 devices free)

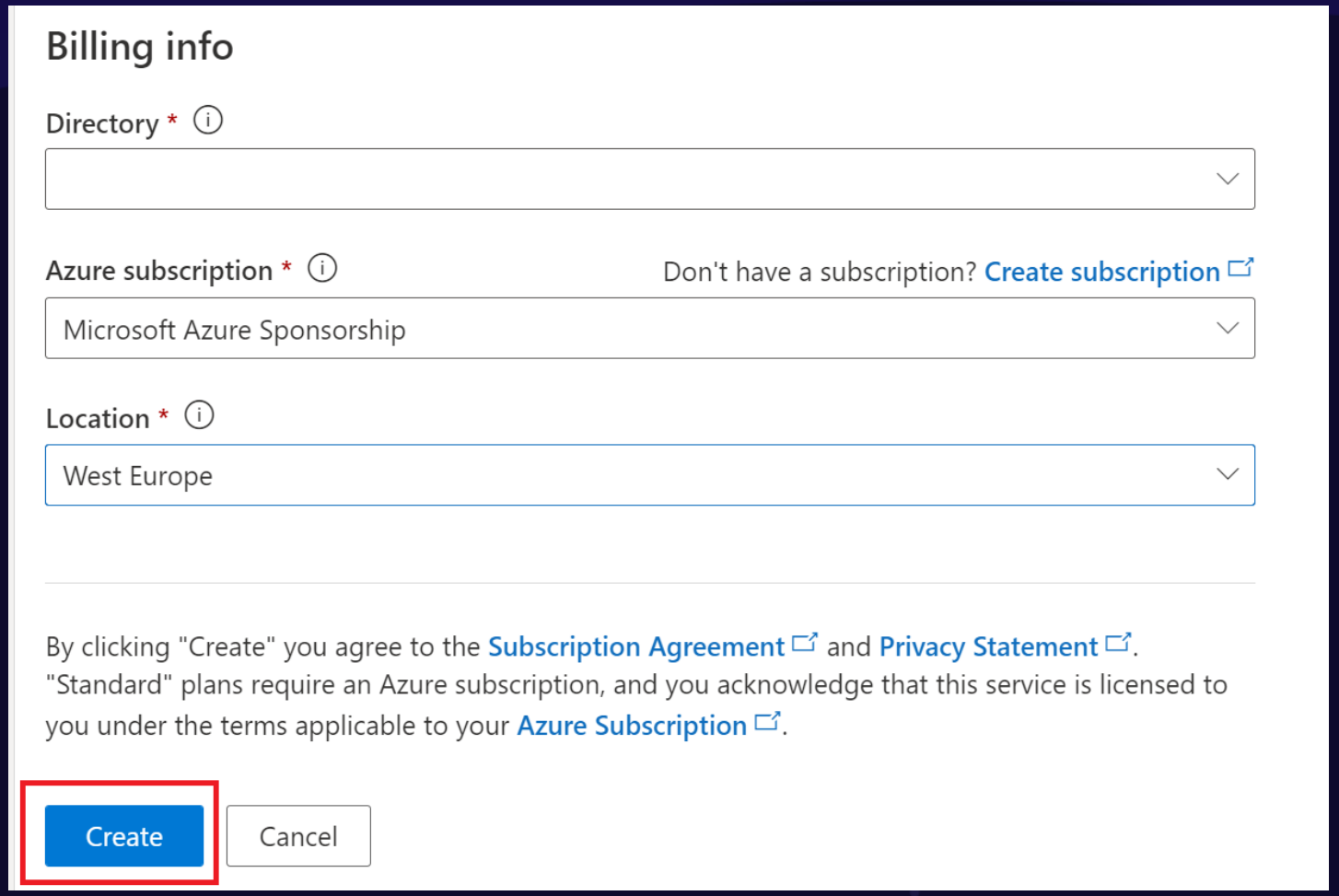


This Requires an Azure Subscription!

Try it Out!

Choose an Active Azure Subscription

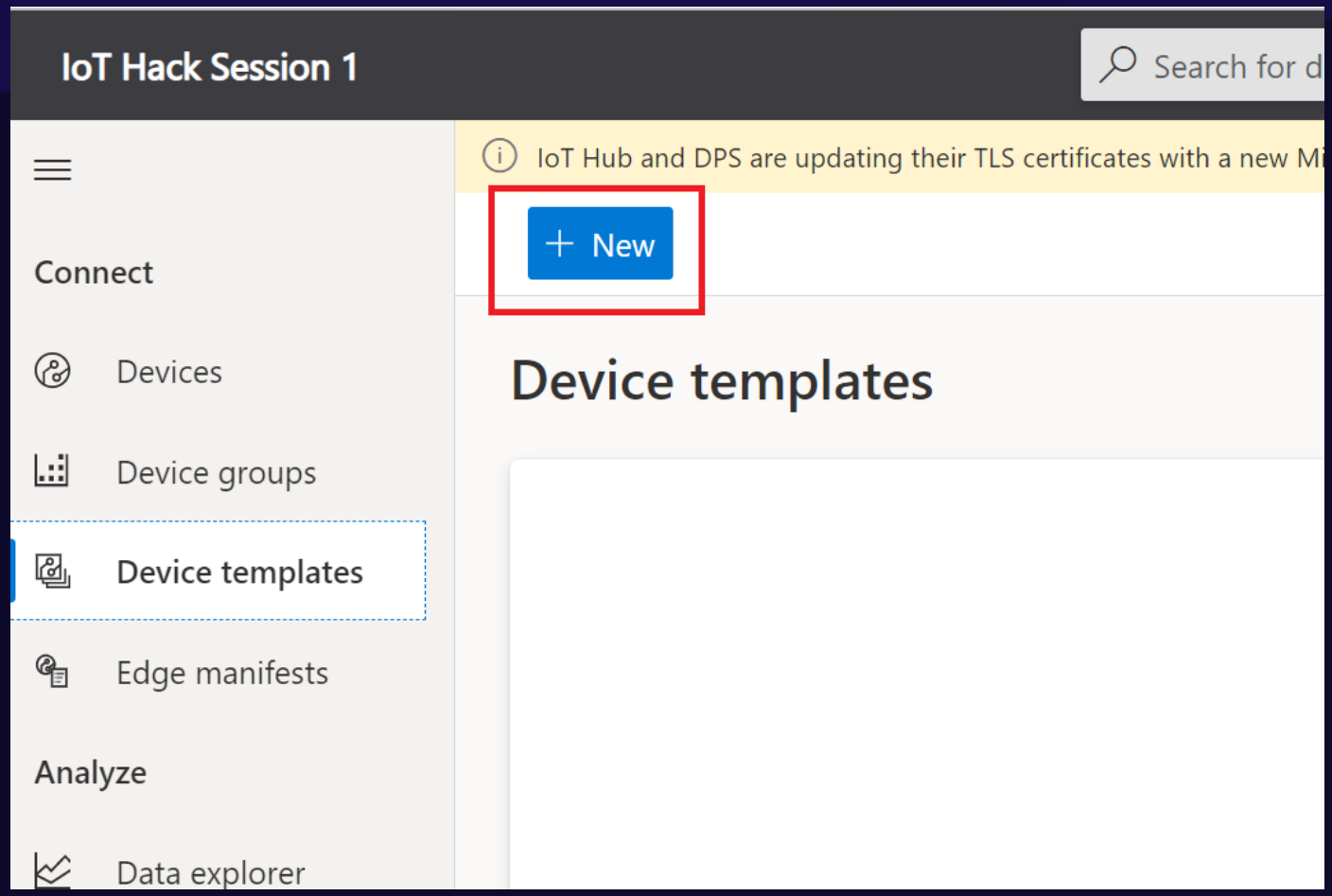
Create the Application



This Requires an Azure Subscription!

Try it Out!

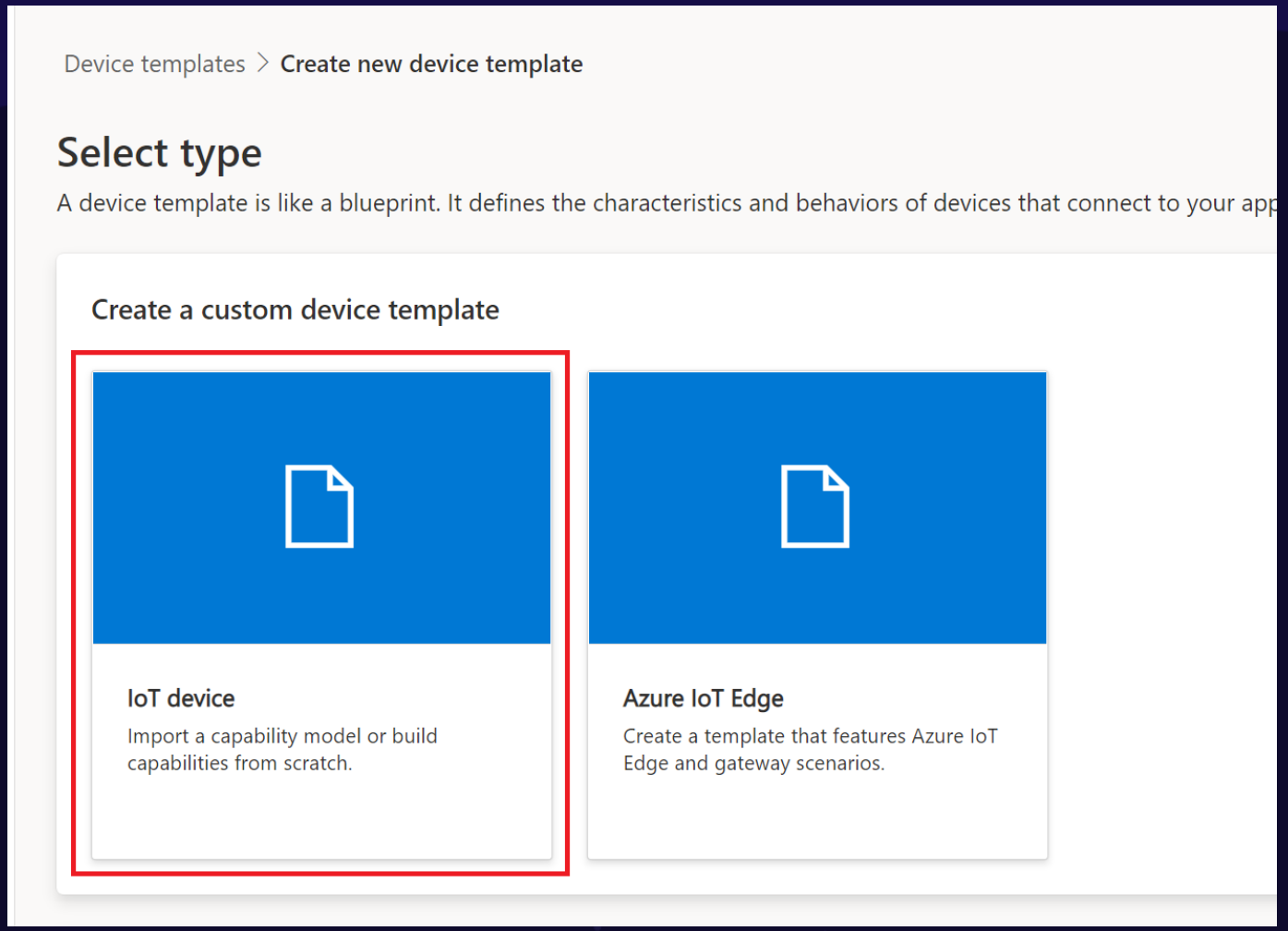
Create a Device Template



This Requires an Azure Subscription!

Try it Out!

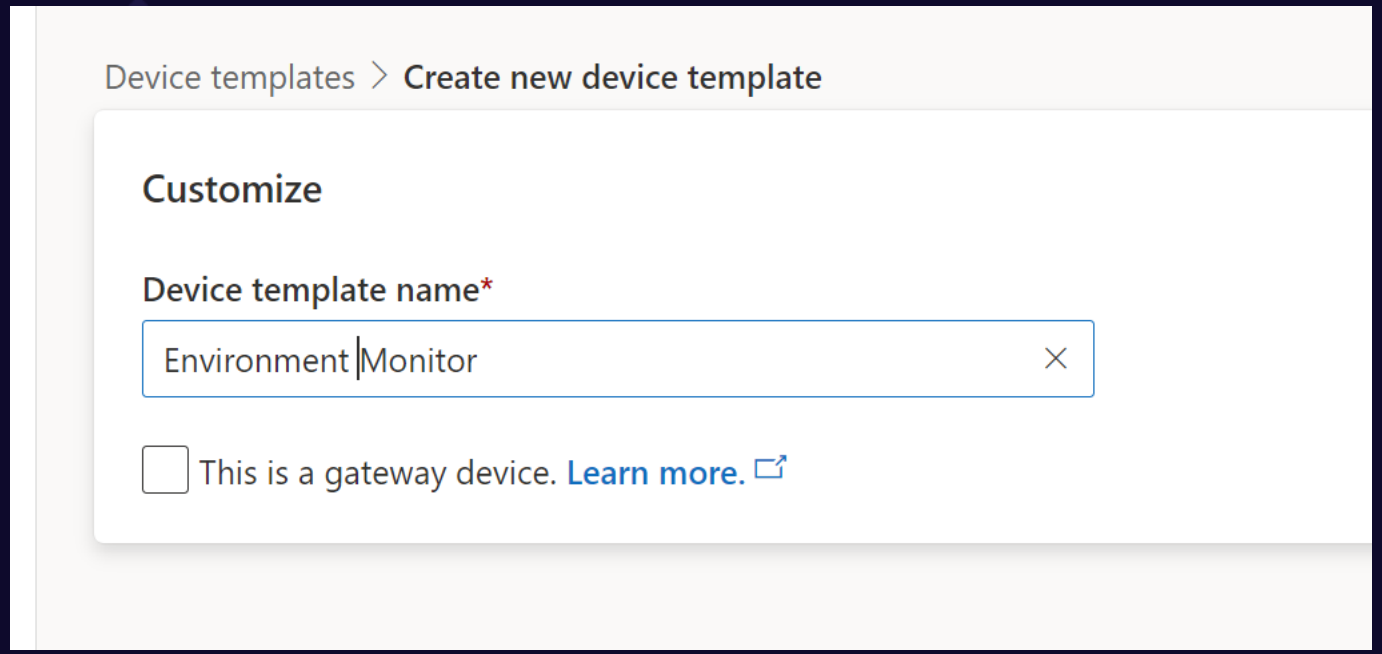
Select IoT Device



This Requires an Azure Subscription!

Try it Out!

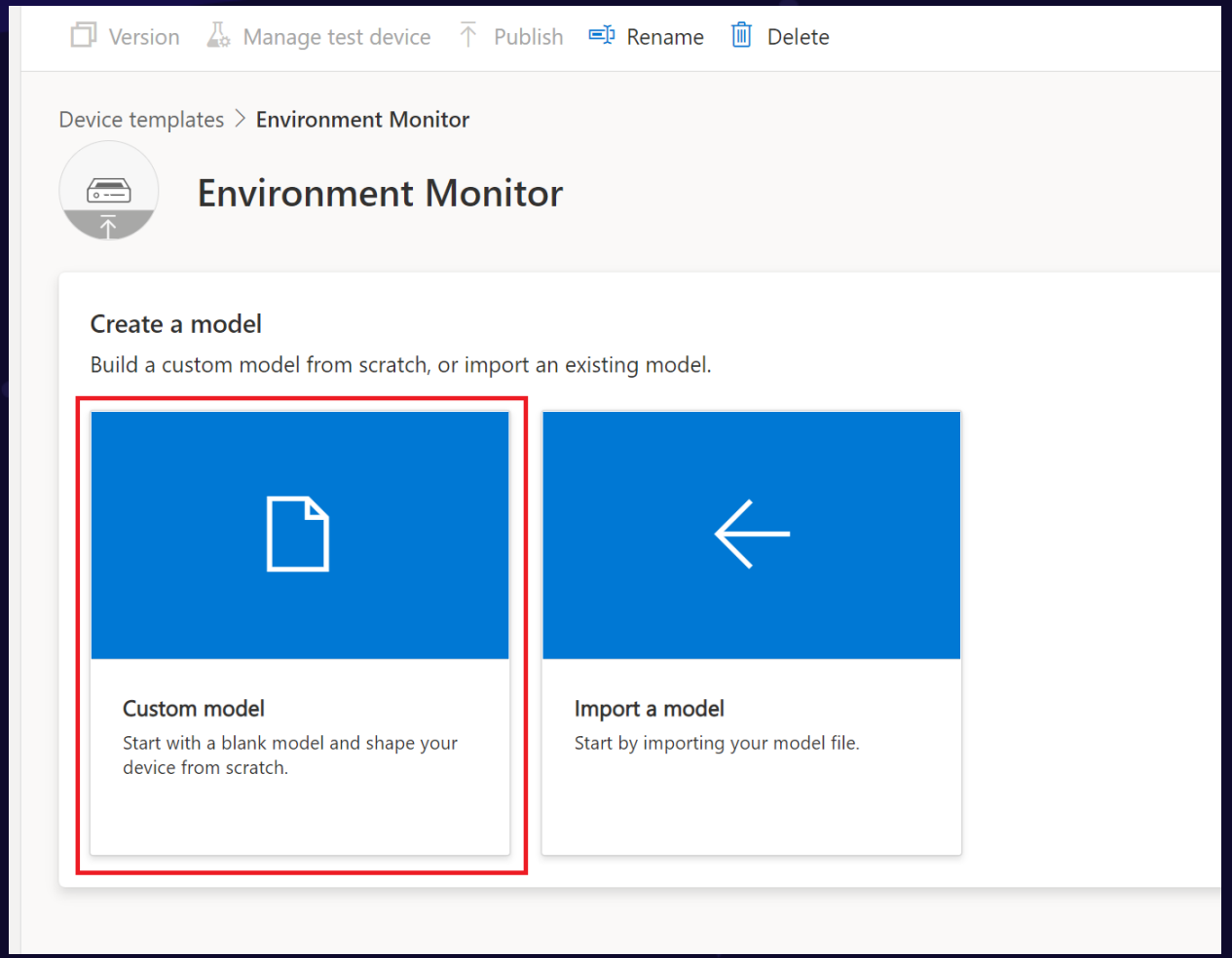
Provide the Template a name



This Requires an Azure Subscription!

Try it Out!

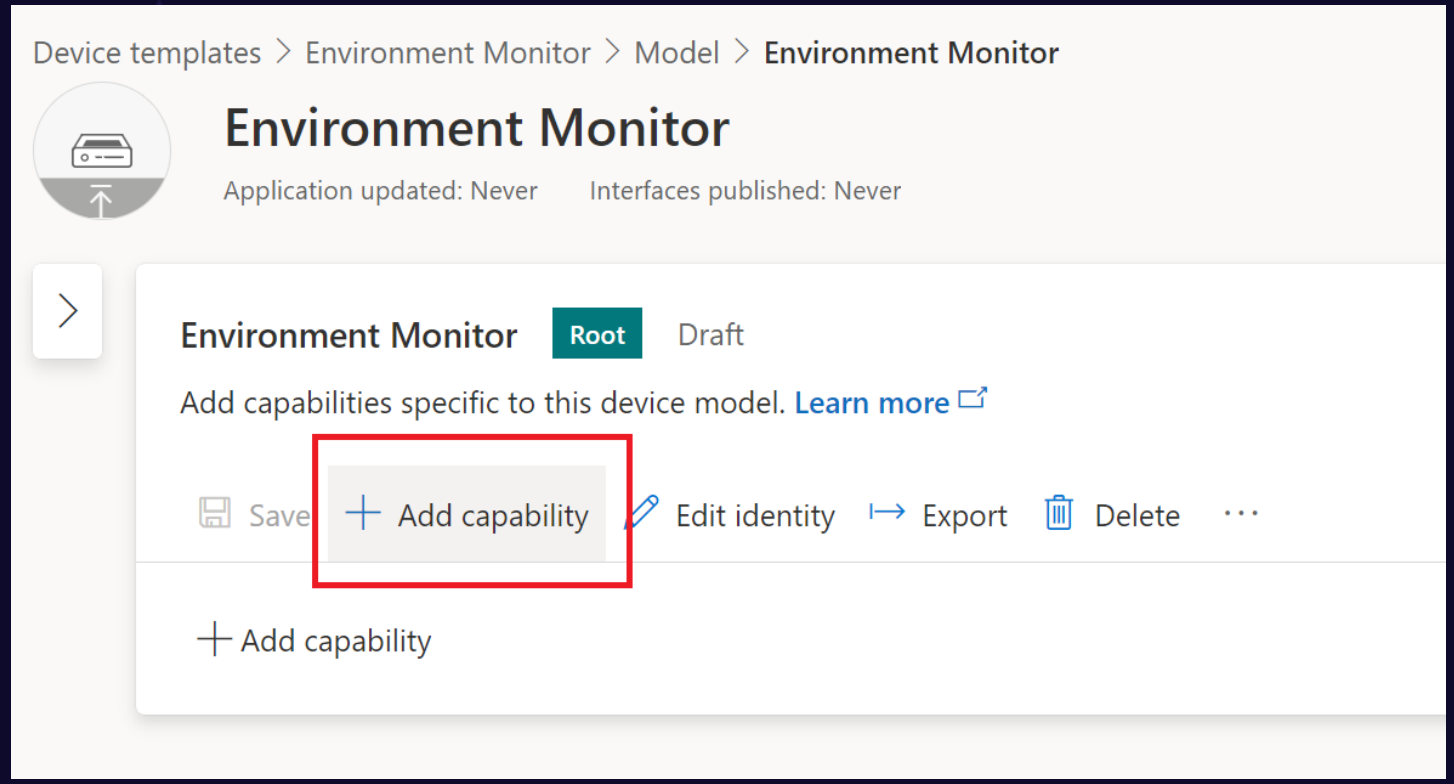
Create a Custom Model for the Device Template



This Requires an Azure Subscription!

Try it Out!

Add a Capability



This Requires an Azure Subscription!

Try it Out!

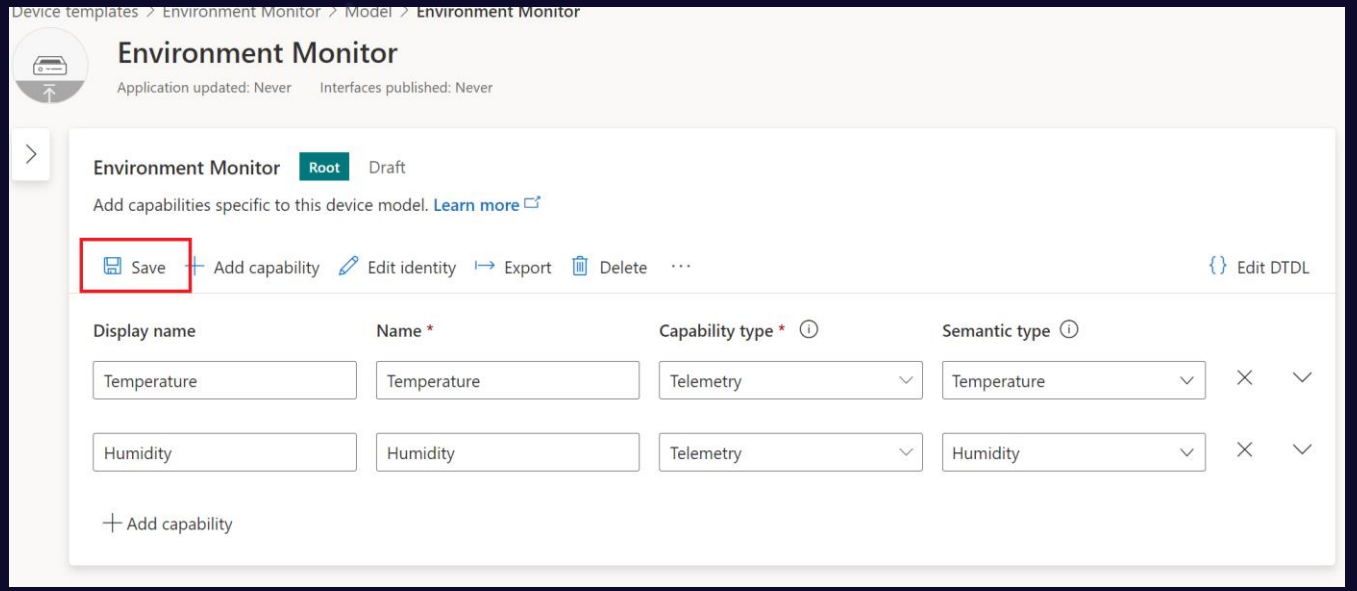
Add a Capability for Temperature

- Semantic Type Temperature

Add a Capability for Humidity

- Semantic Type Humidity

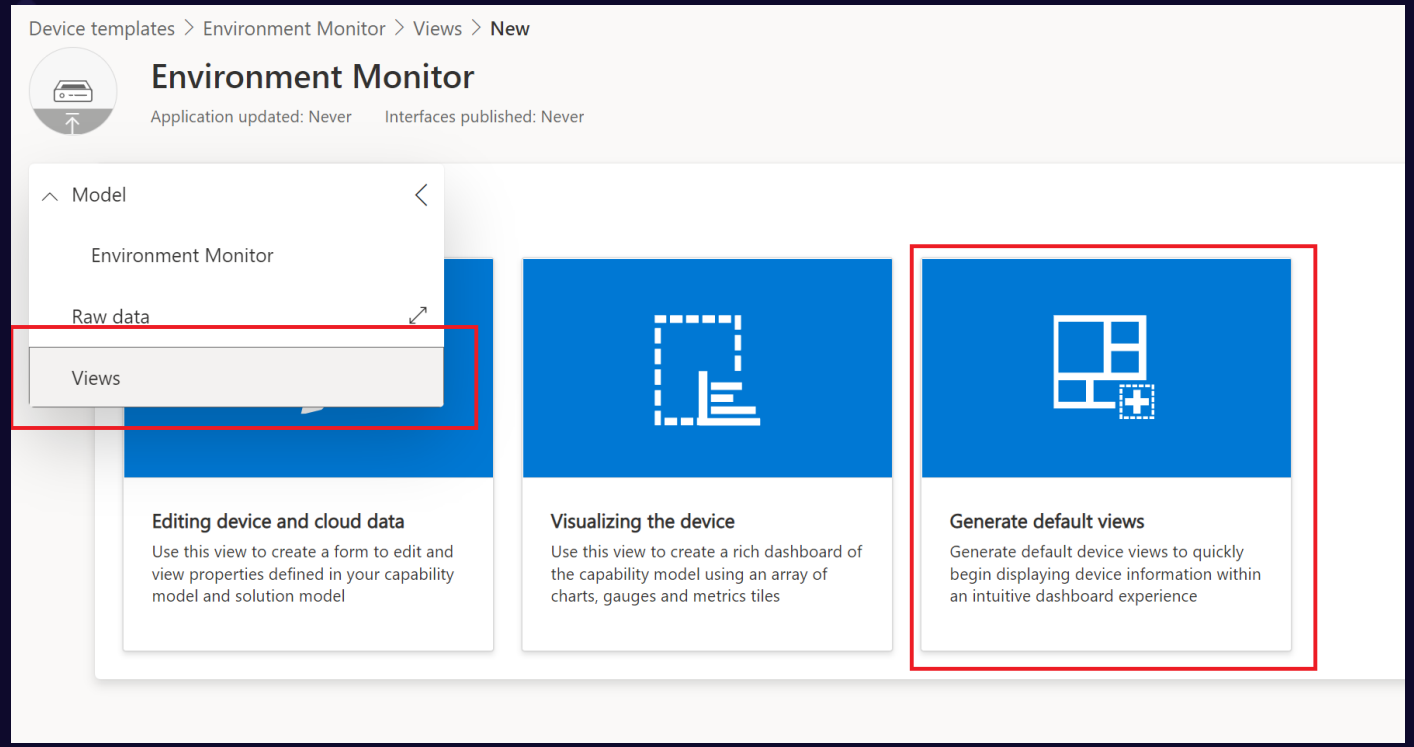
Make sure you Save



This Requires an Azure Subscription!

Try it Out!

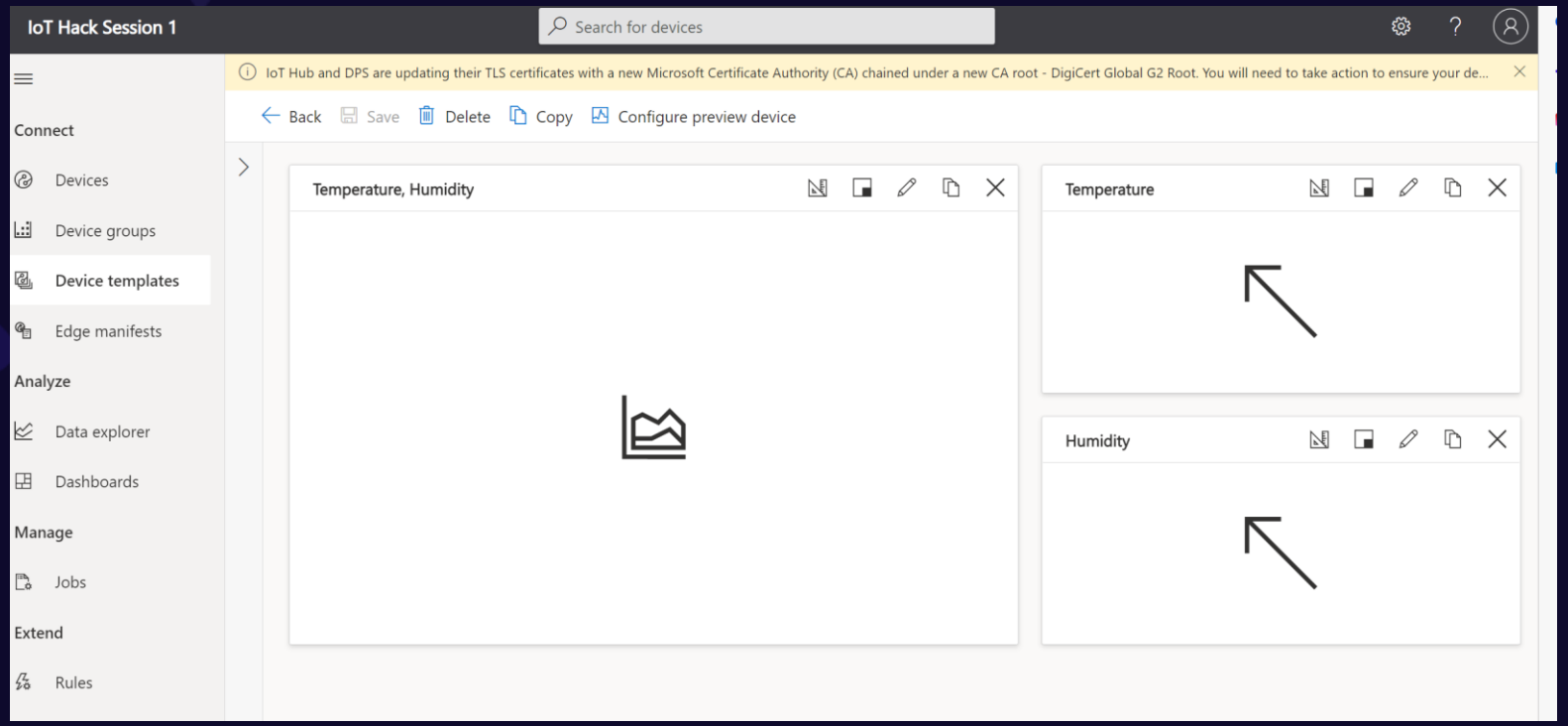
Auto-generate a view / dashboard from the model created.



This Requires an Azure Subscription!

Try it Out!

Generated Views

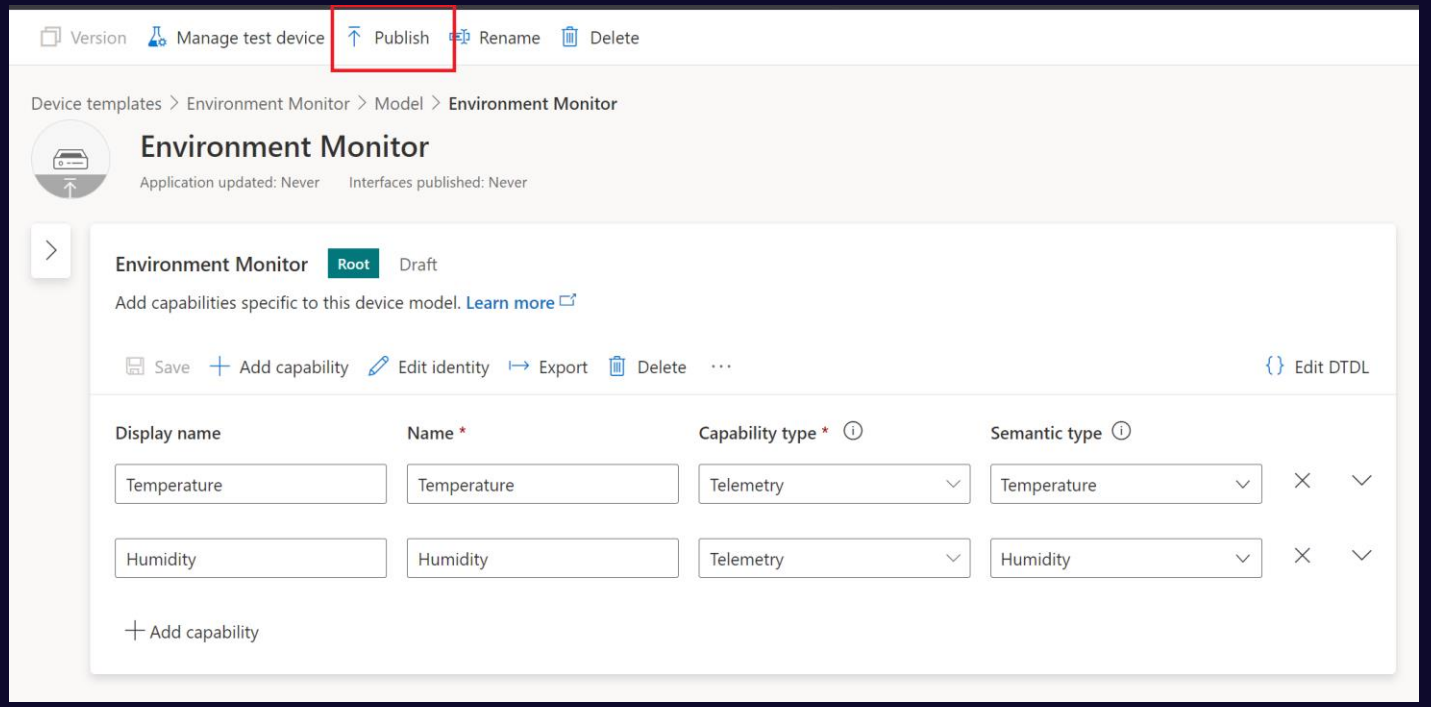


This Requires an Azure Subscription!

Try it Out!

Publish a version of the model

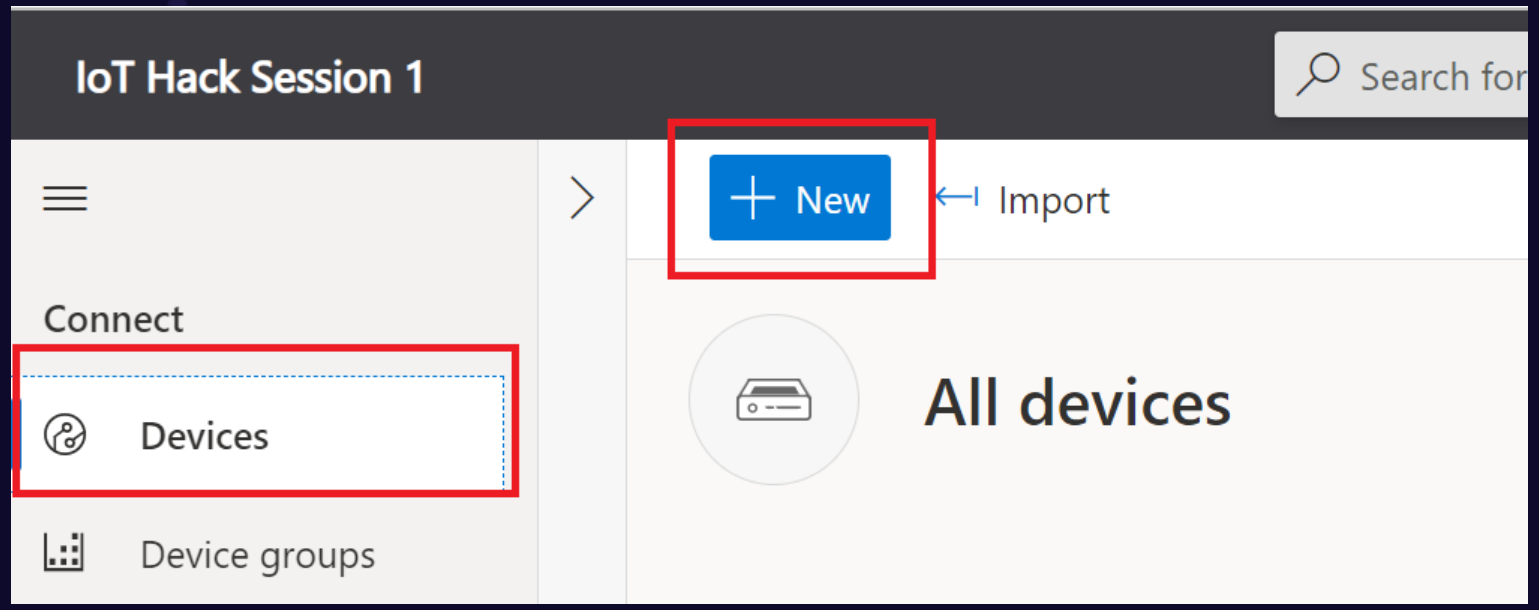
The versioned model can now be used as a device.



This Requires an Azure Subscription!

Try it Out!

Now create an instance of a device based on the model created



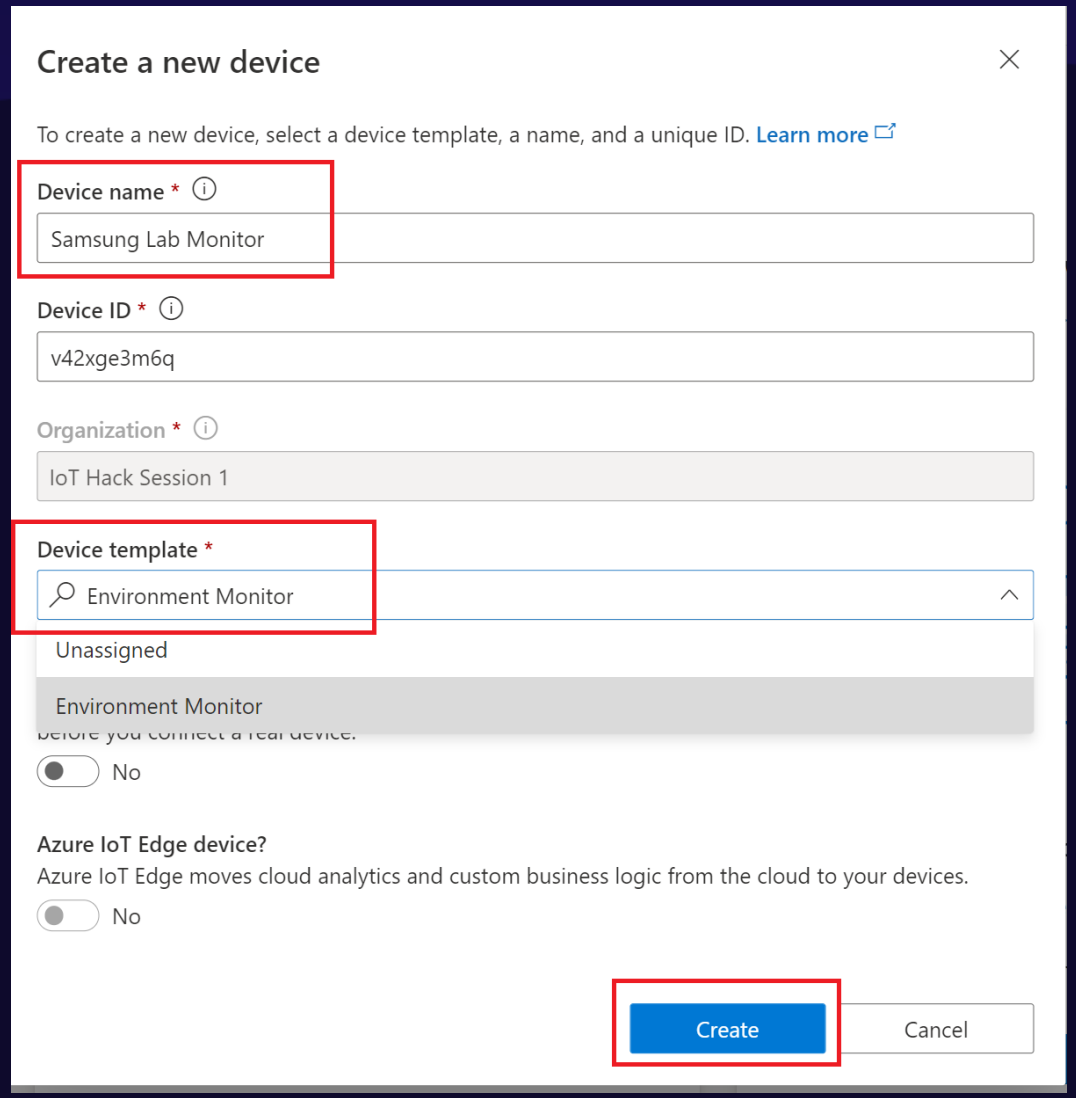
This Requires an Azure Subscription!

Try it Out!

Give the device a name

Assigned the previously created device template to this instance of the device

Create the Device

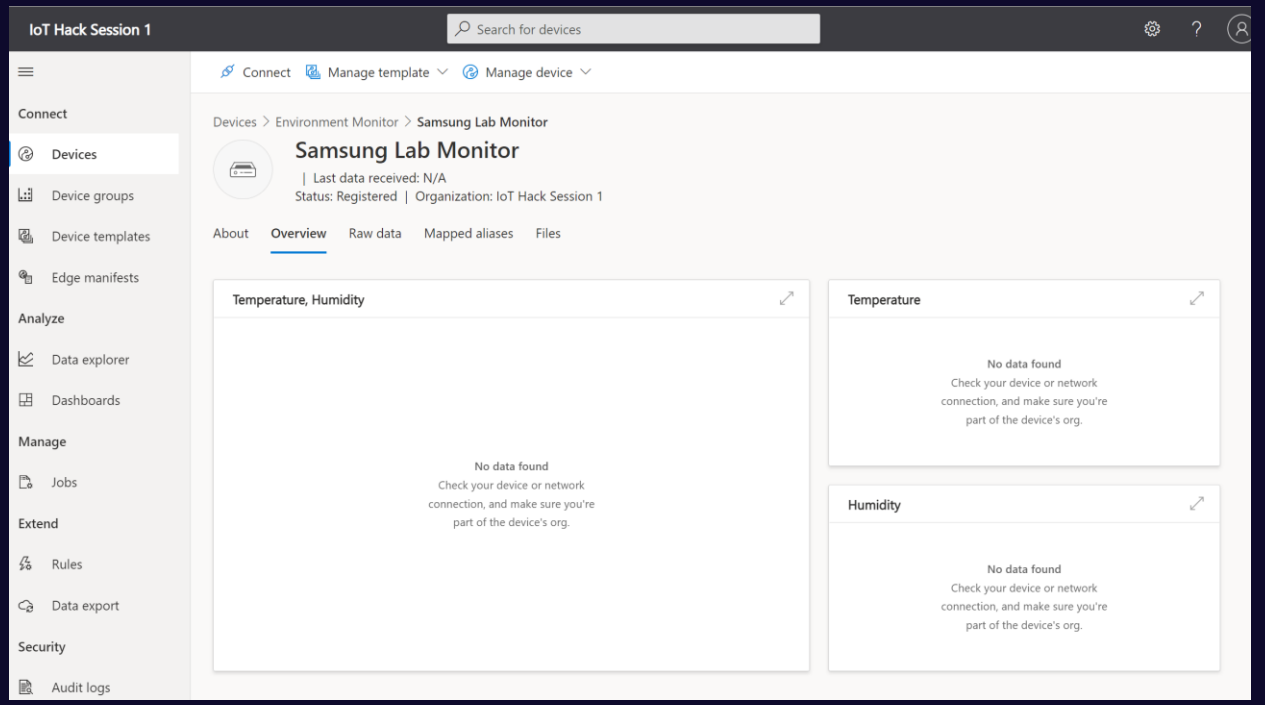


This Requires an Azure Subscription!

Try it Out!

The device has been created in Azure IoT Central.

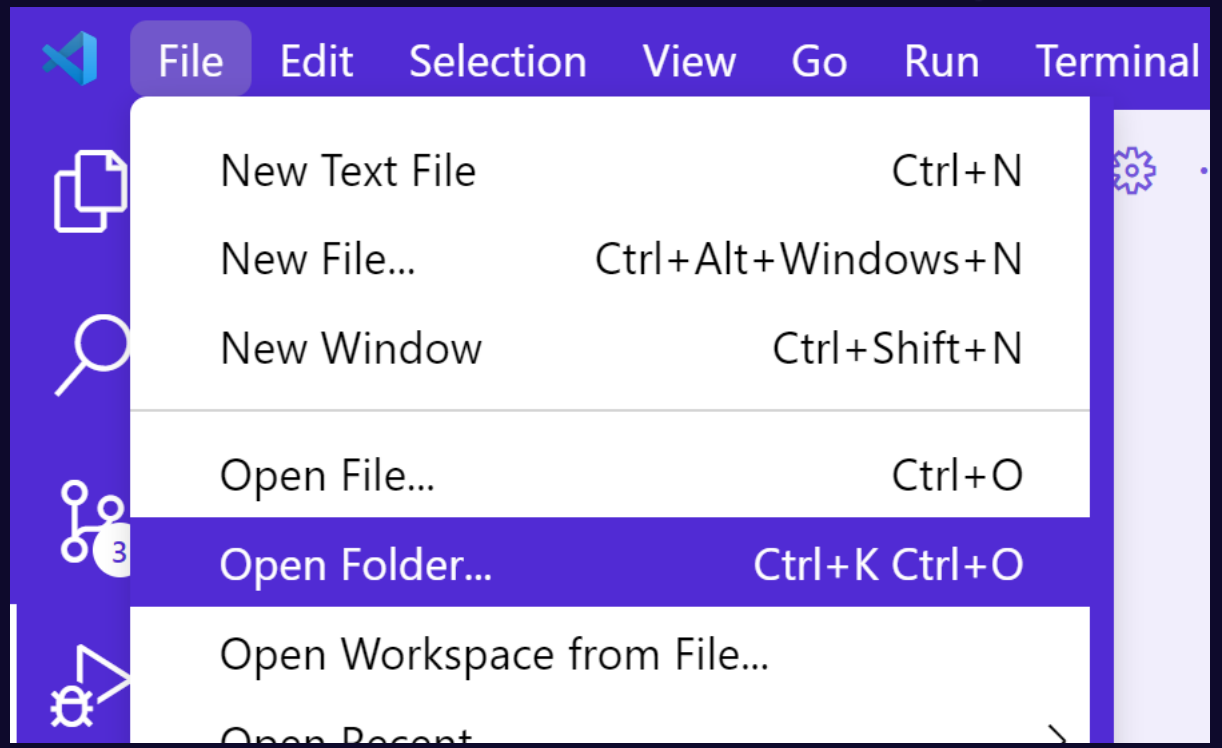
Now lets send it some telemetry from our DHT 11 sensor and Raspberry Pi



This Requires an Azure Subscription!

Try it Out!

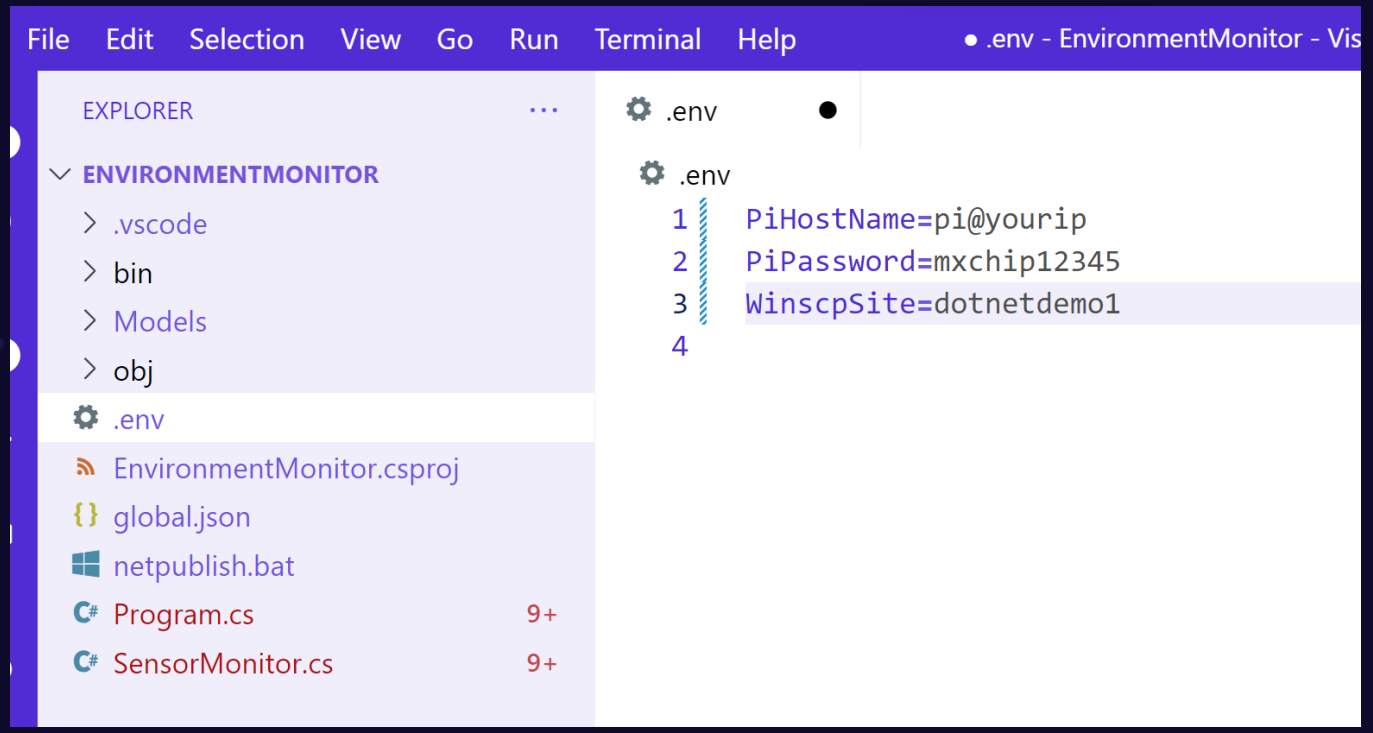
Open the Environment Monitor folder in Visual Studio Code



This Requires an Azure Subscription!

Try it Out!

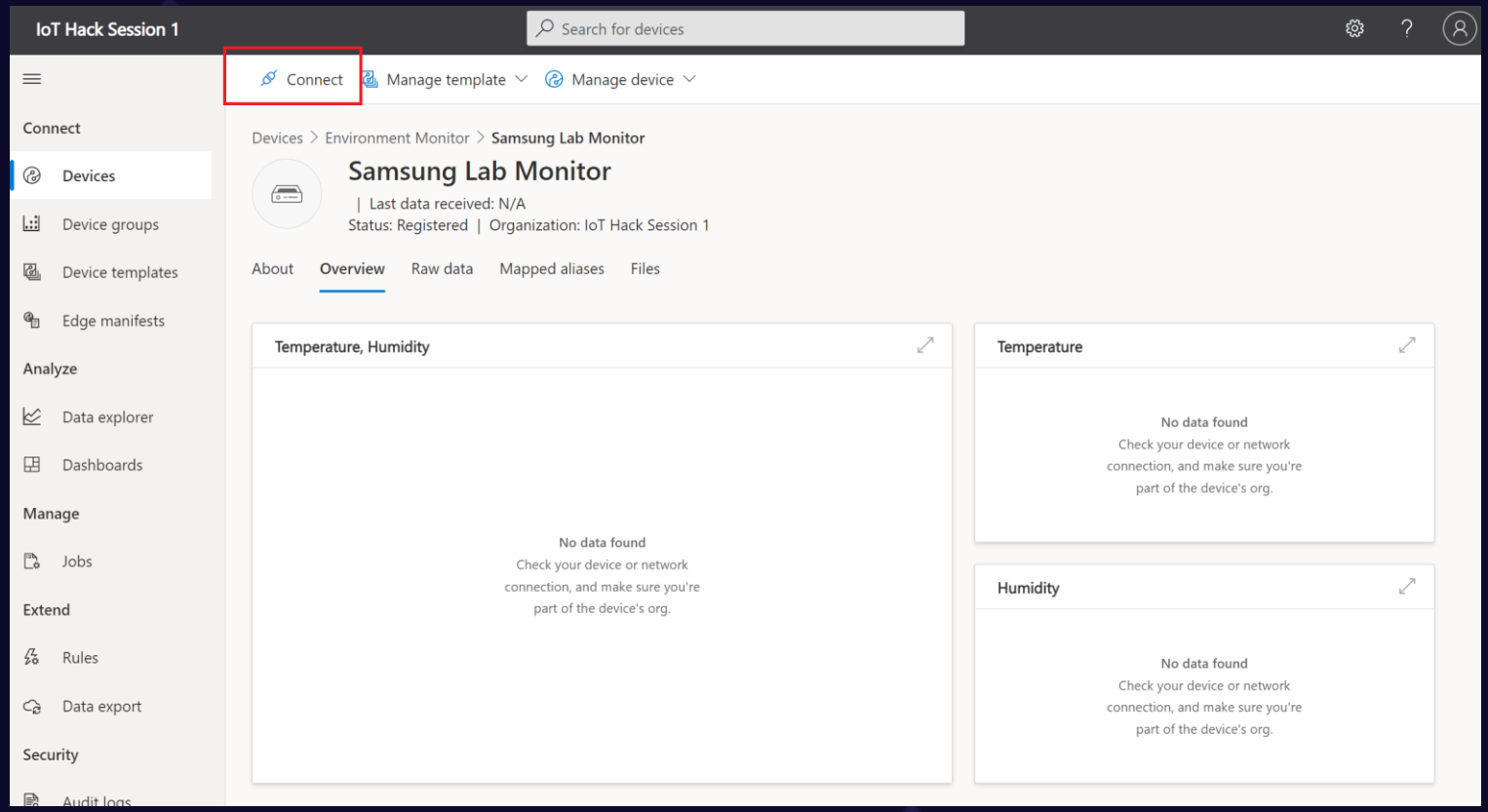
Configure the .env file with your Raspberry Pi's details



This Requires an Azure Subscription!

Try it Out!

Within IoT Central select the device and press the “Connect” button.

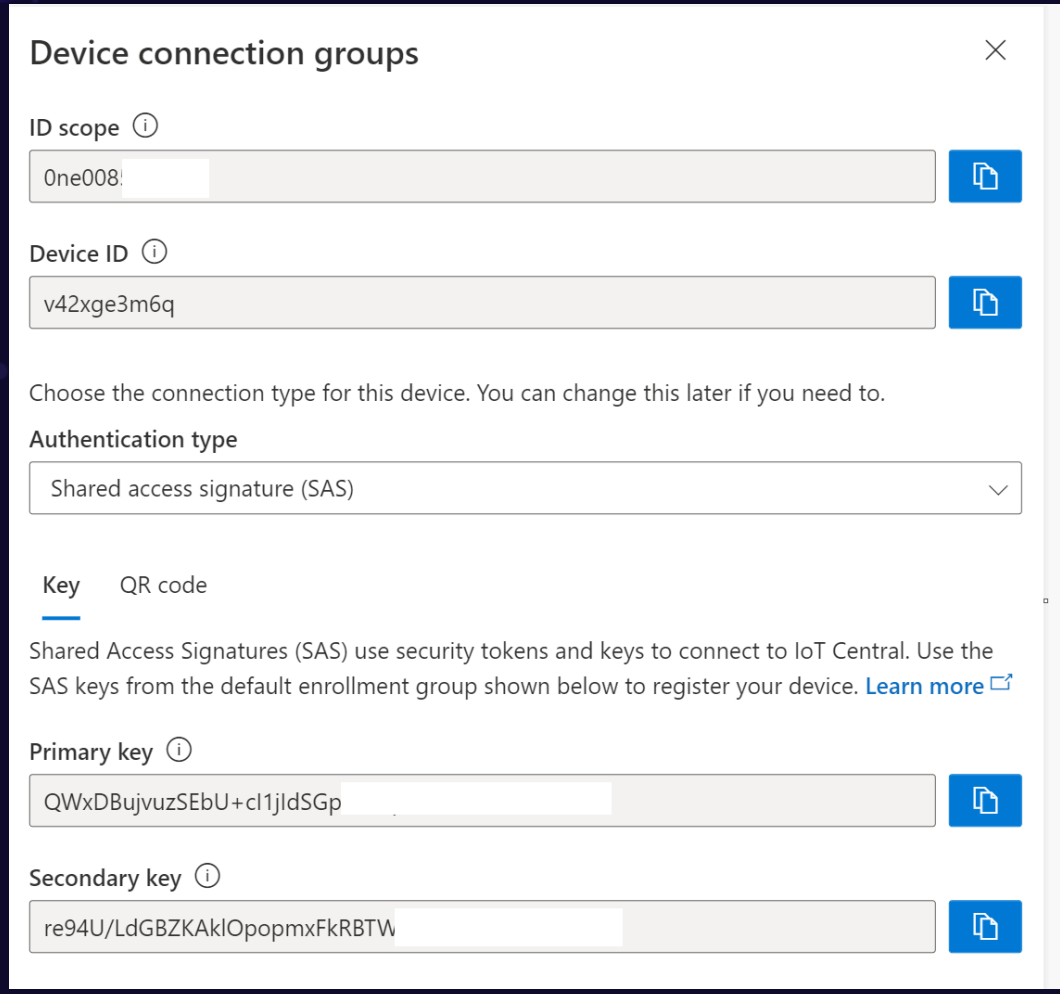


This Requires an Azure Subscription!

Try it Out!

Take note of the:

ID Scope
Device ID
Primary Key

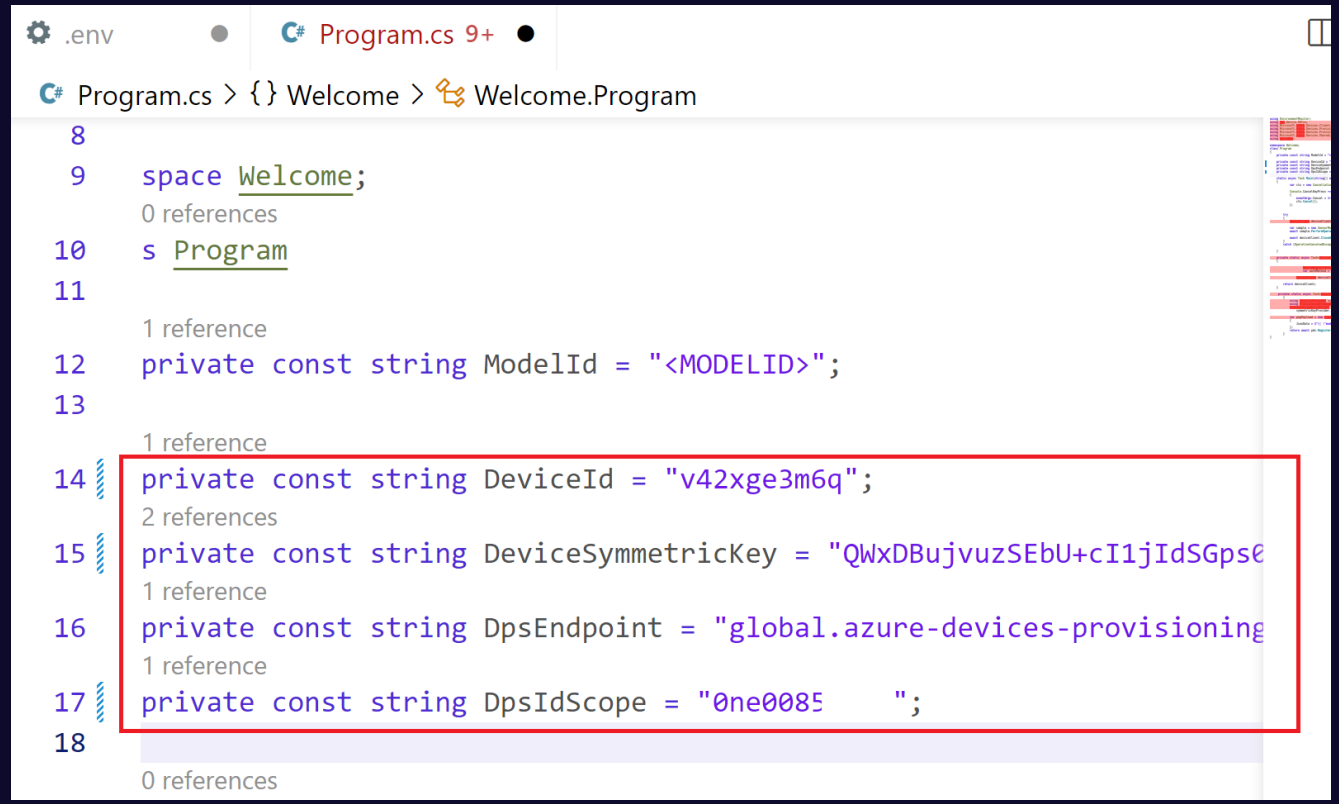


This Requires an Azure Subscription!

Try it Out!

In Visual Studio Code fill in the Programs.cs file

ID Scope -> DpsIdScope
Device ID -> DeviceId
Primary Key -> DeviceSymmetricKey

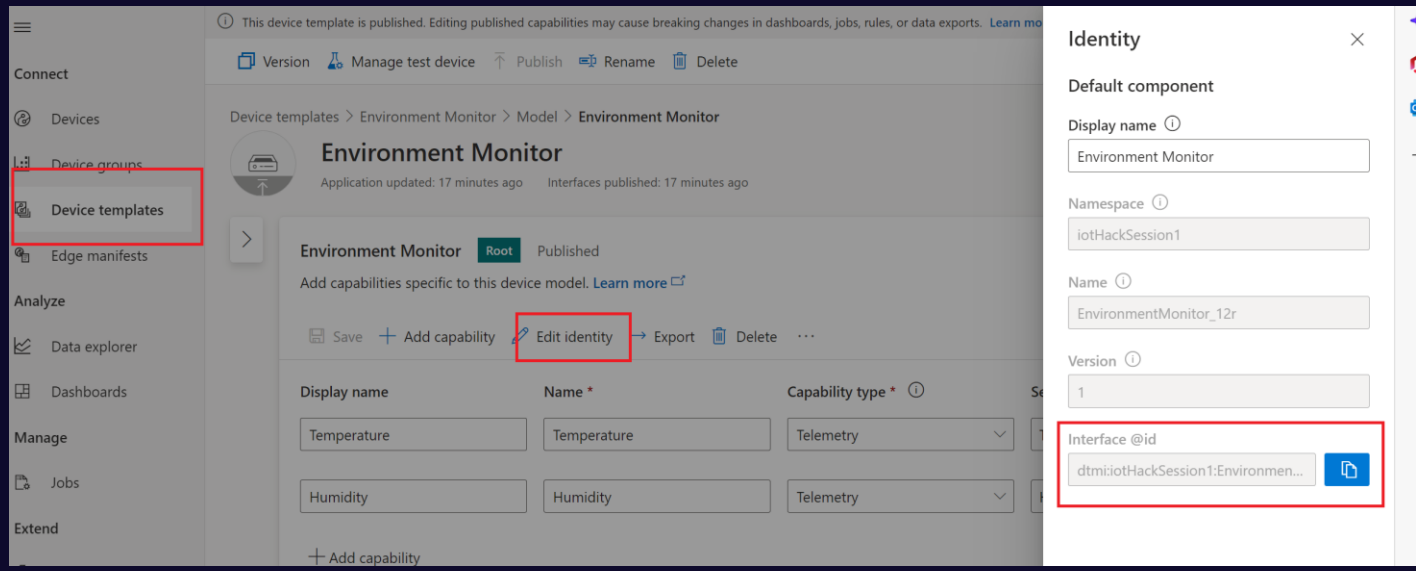


This Requires an Azure Subscription!

Try it Out!

Within IoT Central take note of the

Interface @id



This Requires an Azure Subscription!

Try it Out!

In Visual Studio Code fill in the Programs.cs file

Interface @id -> ModelId

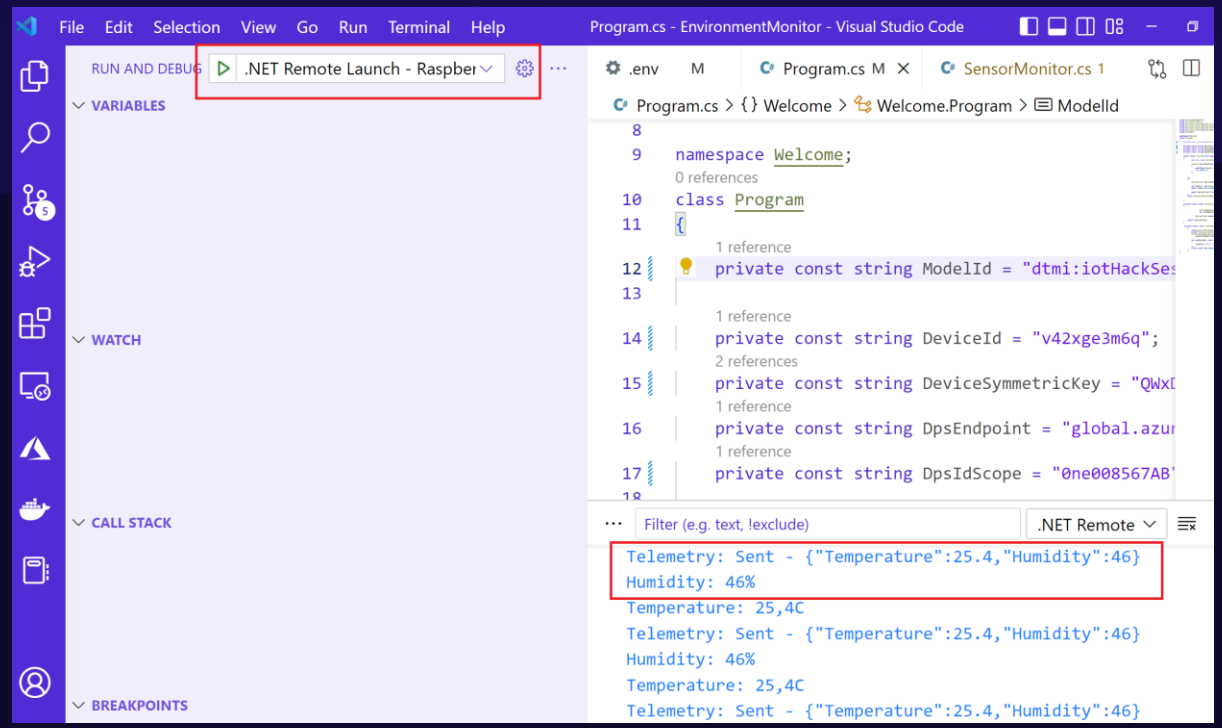
```
8
9   namespace Welcome;
    0 references
10  class Program
11  {
        1 reference
12      private const string ModelId = "dtmi:iotHackSession1:Environment
13
        1 reference
14      private const string DeviceId = "v42xge3m6q";
        2 references
15      private const string DeviceSymmetricKey = "QWxDBujvuzSEbU+cI1jId
        1 reference
16      private const string DpsEndpoint = "global.azure-devices-provisi
        1 reference
```

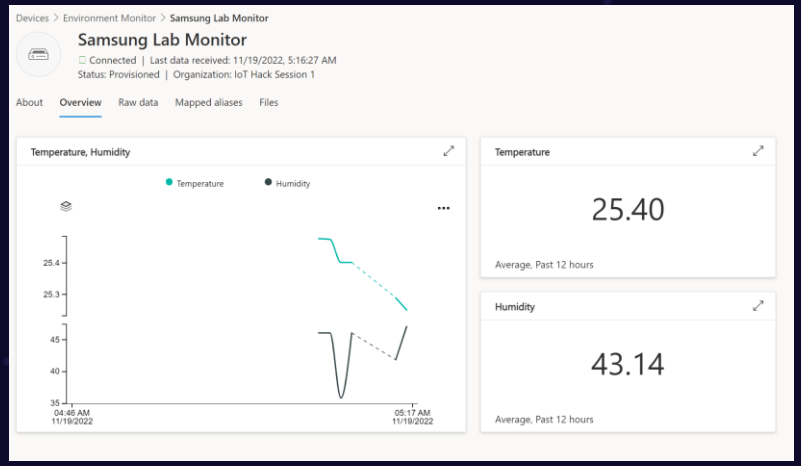
This Requires an Azure Subscription!

Try it Out!

In Visual Studio Code fill in the Programs.cs file

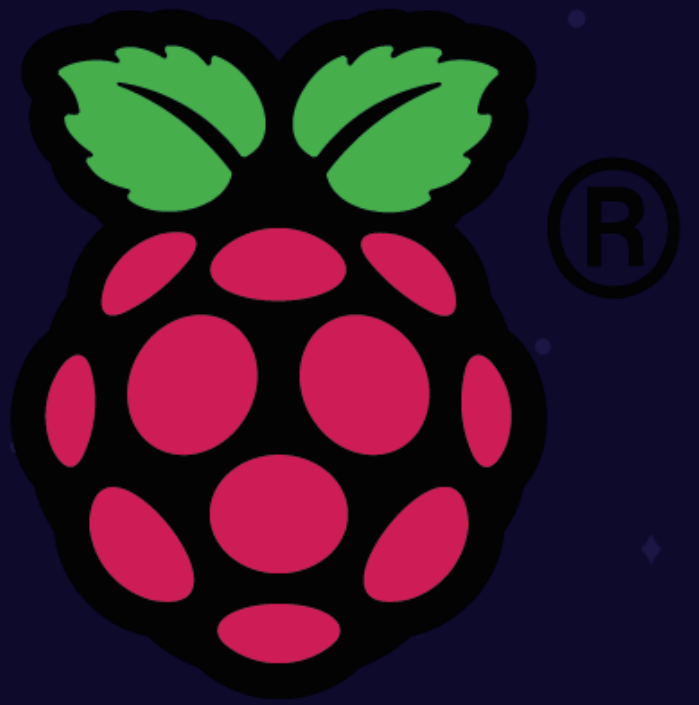
Interface @id -> ModelId





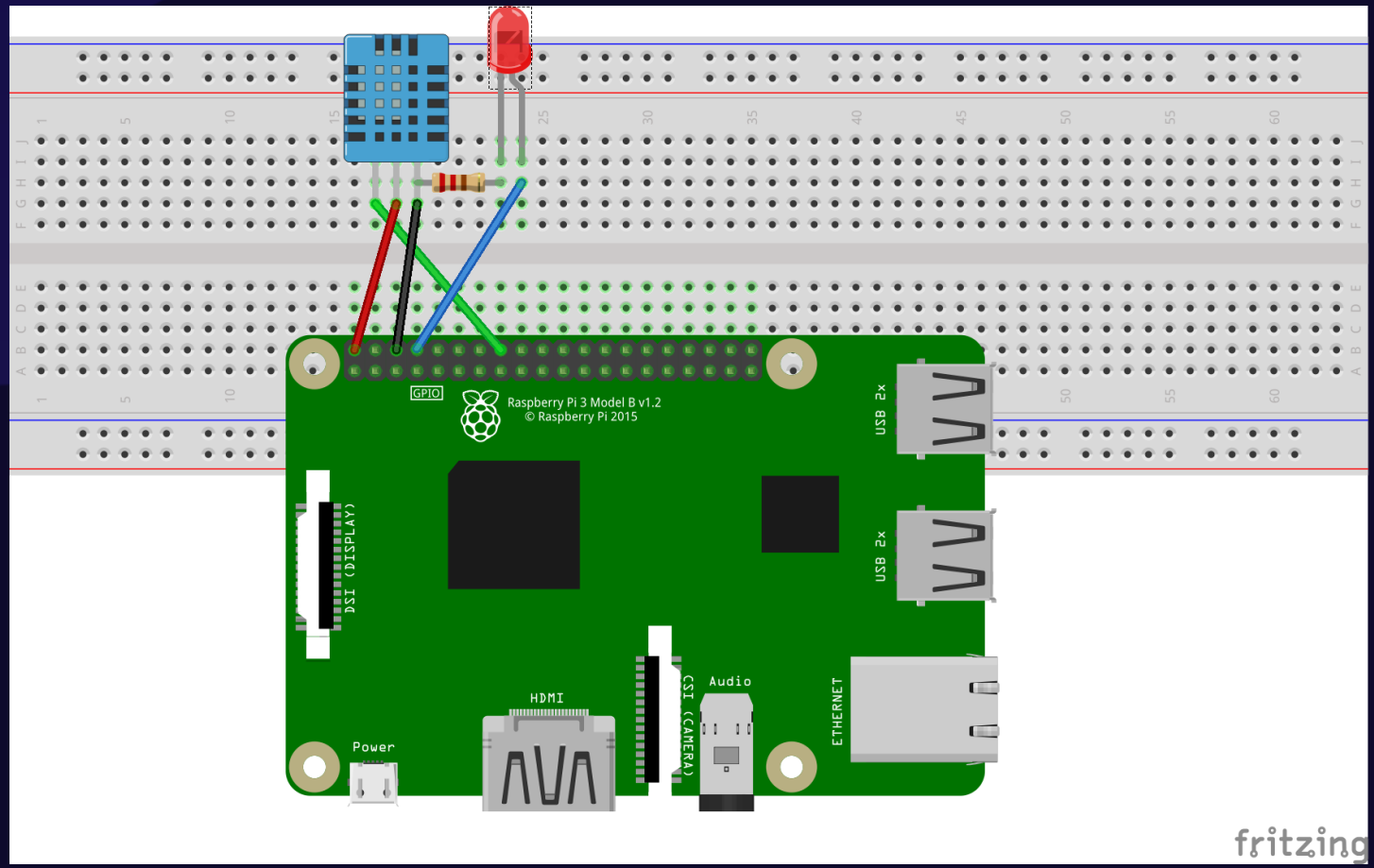
This Requires an Azure Subscription!

Module 4

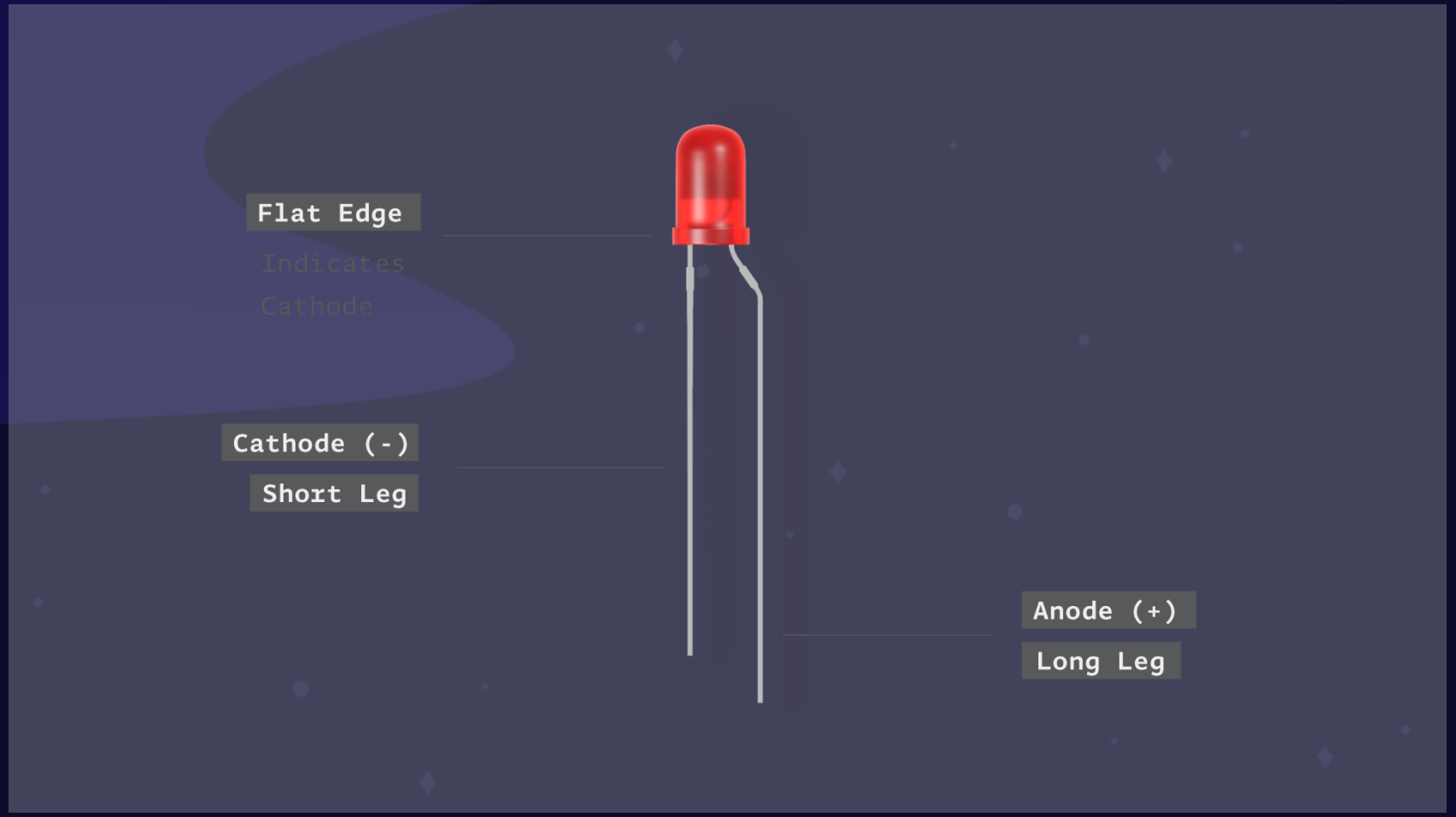


LED Fun

IoT is not IoT without a LED!



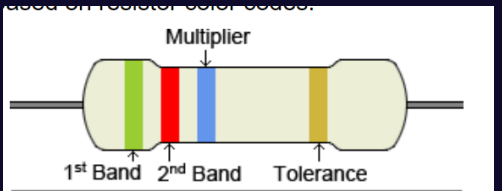
IoT is not IoT without a LED!



IoT is not IoT without a LED!

Why the resistor?

(Vsupply– VF)/ IF= (3.3 -1.7) / 7mA = 220 Ω



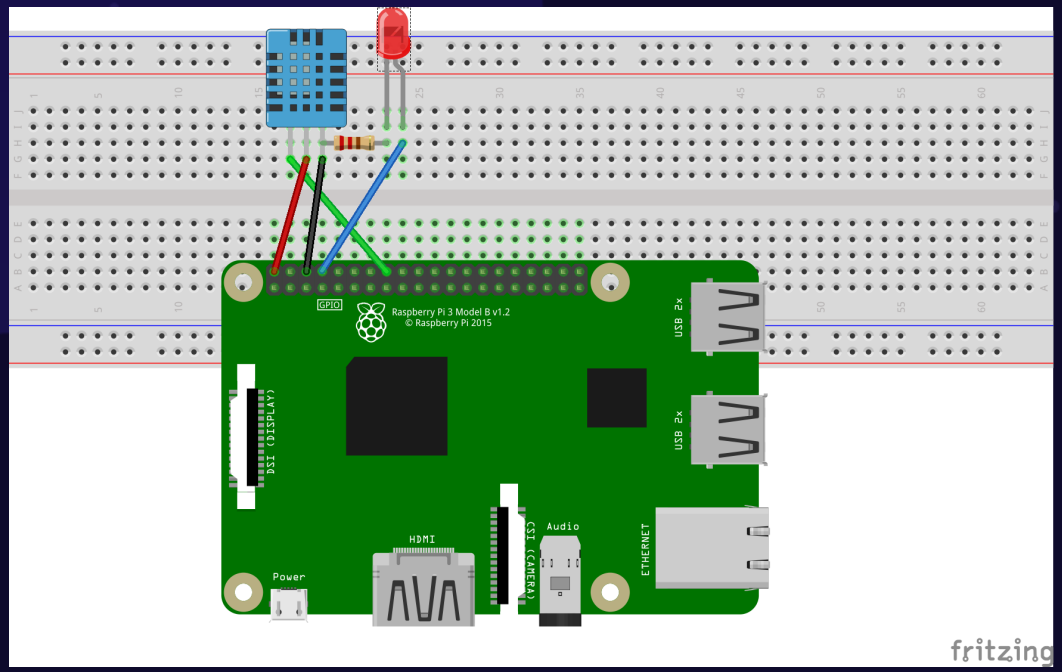
Color	1st, 2nd Band Significant Figures	Multiplier	Tolerance
Black	0	× 1	
Brown	1	× 10	±1% (F)
Red	2	× 100	±2% (G)
Orange	3	× 1K	±0.05% (W)
Yellow	4	× 10K	±0.02% (P)
Green	5	× 100K	±0.5% (D)
Blue	6	× 1M	±0.25% (C)
Violet	7	× 10M	±0.1% (B)
Grey	8	× 100M	±0.01% (L)
White	9	× 1G	
Gold		× 0.1	±5% (J)
Silver		× 0.01	±10% (K)

In this circuit the resistor is used to reduce current flow.

The Raspberry Pi is designed to only provide 23 mA max current. A resistor is important to not damage both the Raspberry Pi and the LED

https://www.calculator.net/resistor-calculator.html

IoT is not IoT without a LED!



Using the component pack, build the circuit using your Raspberry Pi, breadboard, LED, resistor and wires.

Connect resistor from ground (on DHT11) to LED cathode (short leg)

GPIO Pin 14 from the Raspberry Pi to LED anode

Try it out!

Coding Challenge

Turn the LED on when a certain temperature is reached. (check ambient temperature and use that as a starting point)

Hints

```
int ledPin = 14;

var controller = new GpioController();


controller.OpenPin(ledPin,PinMode.Output)

 controller.Write(ledPin,PinValue.High);
```

https://learn.microsoft.com/en-us/dotnet/api/system.device.gpio.gpiocontroller?view=iot-dotnet-latest

System.Device.Gpio

@CPTMSDUG

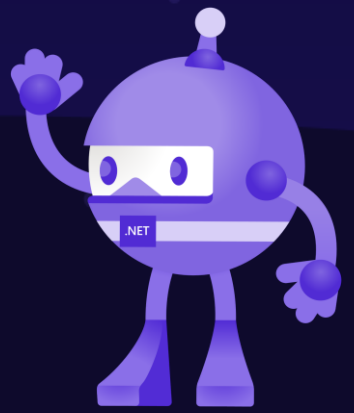
- Twitter: @adpead
- About.me: https://about.me/allan_pead
- Linkedin: https://www.linkedin.com/in/adpead/

- Blog: https://explorationspace.co.za

- Raspberry Pi South Africa
- https://www.facebook.com/groups/1493503984198019

Thank you!!

Resources



Getting Started with .NET

https://learn.microsoft.com/en-us/dotnet/standard/get-started

Visual Studio Remote Debugger

https://www.hanselman.com/blog/remote-debugging-with-vs-code-on-windows-to-a-raspberry-pi-using-net-core-on-arm

Getting Started with .NET Nanoframework

https://docs.nanoframework.net/content/getting-started-guides/getting-started-managed.html

MVP Microsoft® Most Valuable Professional