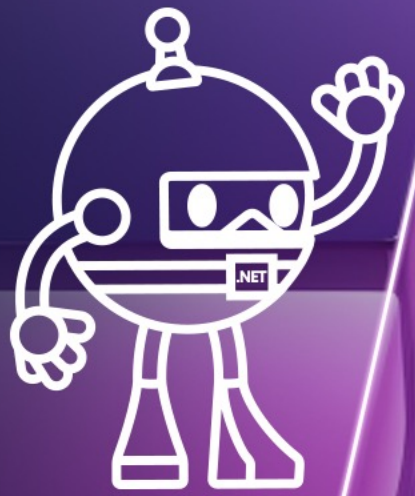
.NET Conf China 2023

2023/12/16
09:30 - 18:00

中国・北京



.NET中文社区

中国 · 北京

.NET Conf China 2023

使用.NET 8 开发IoT网关

黄海鹏

Why .NET?

1.团队背景　技术栈、十年以上的开发经验

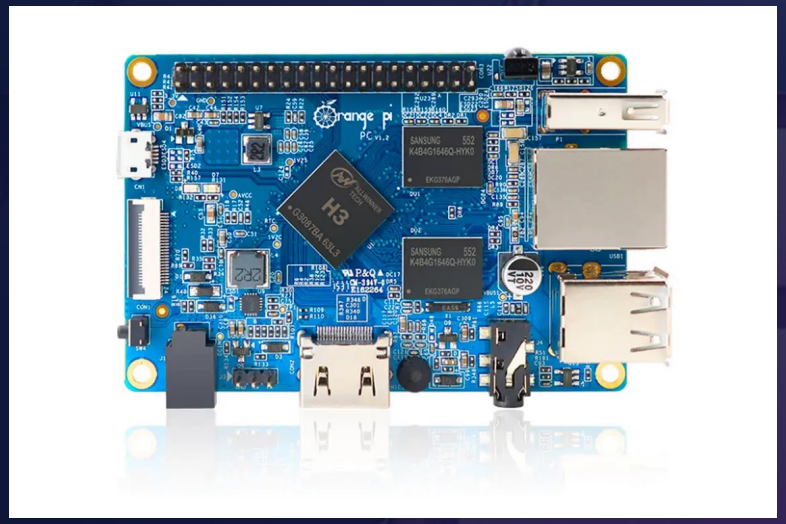
2.应用范围　ARM32/ARM64、Debian10/Ubuntu20.04

3.工作效率　端到端、开发速度、系统运行表现

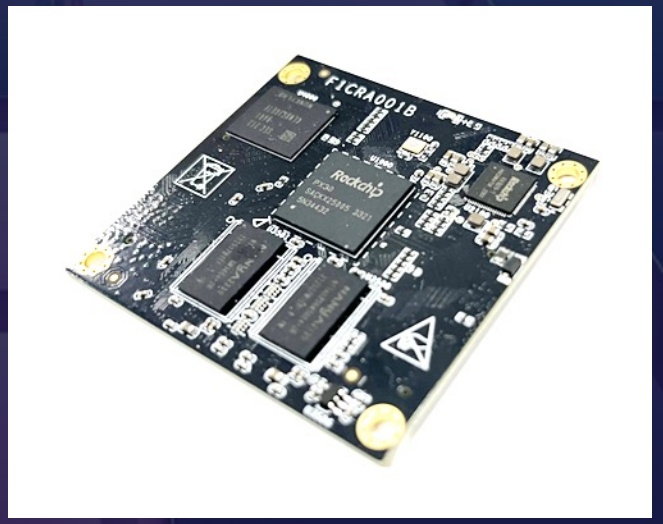
4.前景方向　开源、持续迭代

设备选型历史

BCM2711

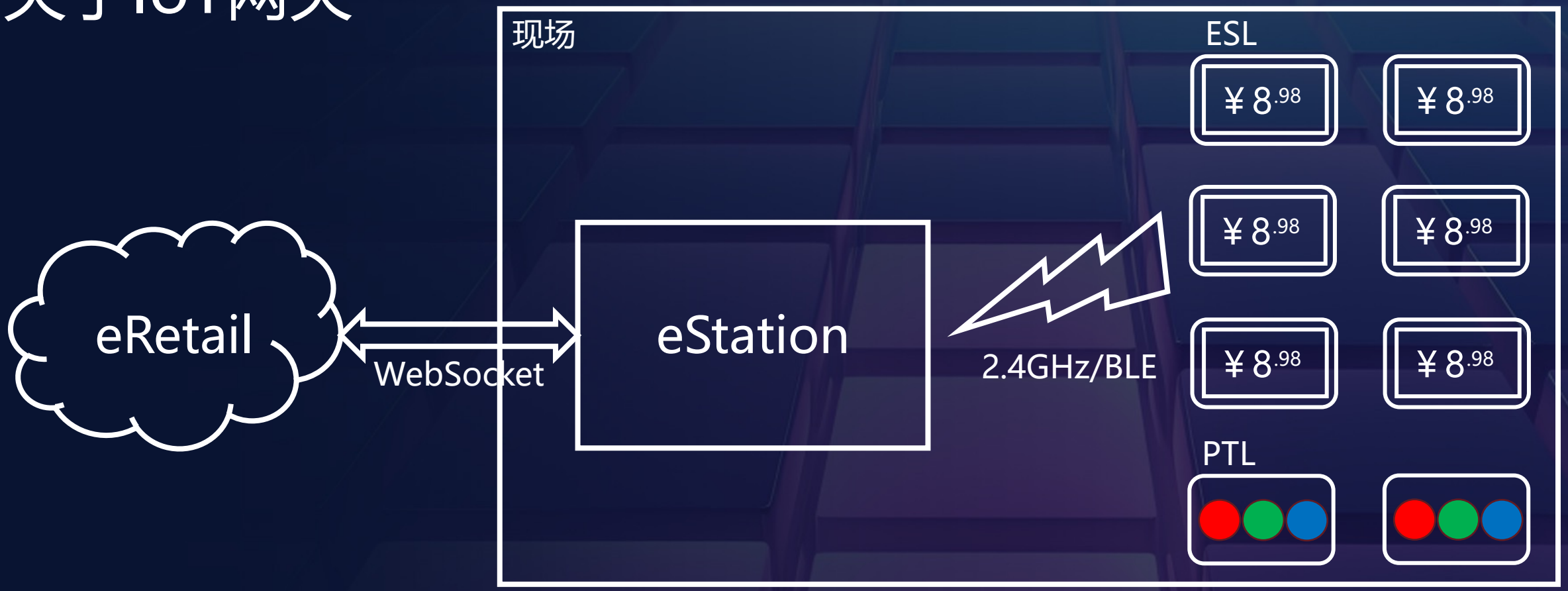


H3



PX30

关于IoT网关


现场
ESL
¥8.98
¥8.98
¥8.98
¥8.98
¥8.98
¥8.98
PTL
eRetail
WebSocket
eStation
2.4GHz/BLE


.NET
8

关于IoT网关：配置


Ethernet
eth0
I2C
Clk & Dio
GPIO
eStation
Serial Port
ttyS0
Serial Port
ttyS1
Serial Port
ttyS2
...


.NET
8
.NET
.NET

关于IoT网关：功能

WebSocket
MQTT
UDP

8888

REST按键
状态灯
Watchdog

Ethernet
eth0

I2C
Clk & Dio

GPIO

eStation

Serial Port
ttyS0

Serial Port
ttyS1

Serial Port
ttyS2

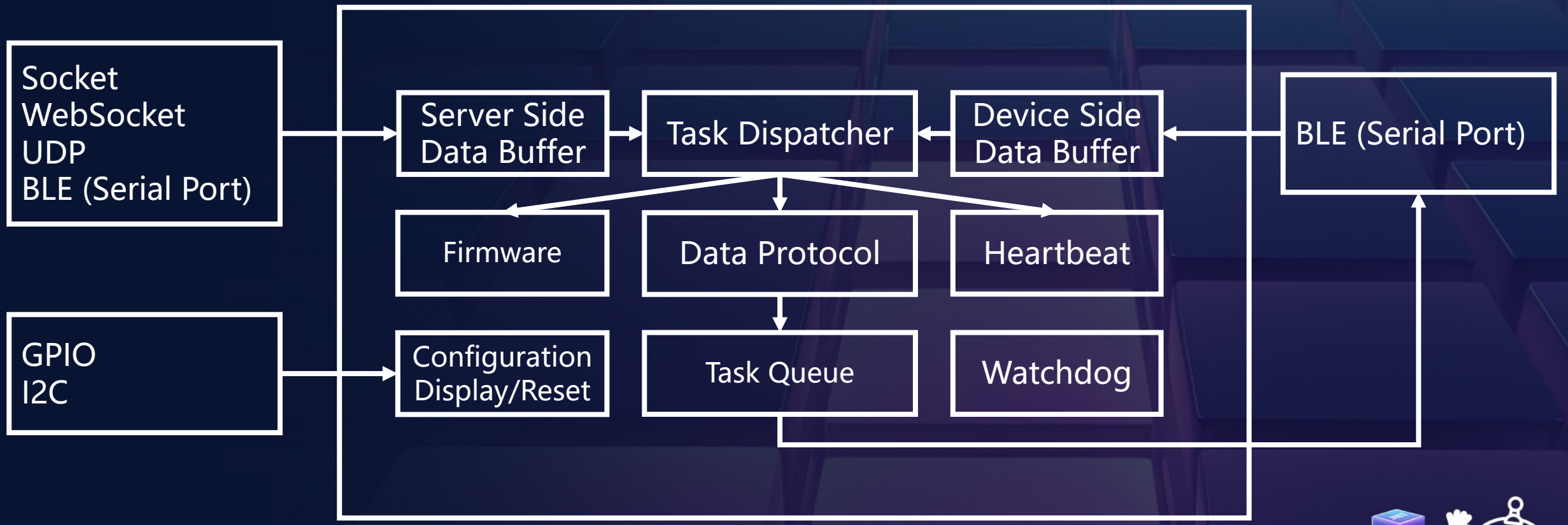
...

BLE
PHY6222

BLE
DS531

BLE
PHY6222

关于IoT网关：结构


Socket
WebSocket
UDP
BLE (Serial Port)
GPIO
I2C
Server Side Data Buffer
Task Dispatcher
Device Side Data Buffer
Firmware
Data Protocol
Heartbeat
Configuration Display/Reset
Task Queue
Watchdog
BLE (Serial Port)

.NET 6 升级到.NET 8

Target framework

Specifies the version of .NET that the application targets. The option can have different values depending on which versions of .NET are installed on your computer.

.NET 8.0

1

```
<PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <Platforms>ARM32</Platforms>
    <AssemblyVersion>1.1.63</AssemblyVersion>
```

Package Manager Console

Package source: All Default project: eStation

2

```
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses

Package Manager Console Host Version 6.8.0.131

Type 'get-help NuGet' to see all available NuGet commands.

PM> Update-Package
Restoring packages for C:\Users\huang\OneDrive\eDummy\eStation.Entity\eStation.Entity.csproj...
  CACHE https://api.nuget.org/v3/vulnerabilities/index.json
  CACHE https://api.nuget.org/v3/vulnerabilities/vulnerability.base.json
  CACHE https://api.nuget.org/v3/vulnerabilities/vulnerability.update.json
Restoring packages for C:\Users\huang\OneDrive\eDummy\eDummy.Common\eDummy.Common.csproj...
  CACHE https://api.nuget.org/v3/vulnerabilities/index.json
  CACHE https://api.nuget.org/v3/vulnerabilities/vulnerability.base.json
  CACHE https://api.nuget.org/v3/vulnerabilities/vulnerability.update.json
Restoring packages for C:\Users\huang\OneDrive\eDummy\eDummy.ServerB\eDummy.ServerB.csproj...
  CACHE https://api.nuget.org/v3/vulnerabilities/index.json
  CACHE https://api.nuget.org/v3/vulnerabilities/vulnerability.base.json
  CACHE https://api.nuget.org/v3/vulnerabilities/vulnerability.update.json
No package updates are available from the current package source for project 'eDummy.Server'.
Restoring packages for C:\Users\huang\OneDrive\eDummy\eDummy.ServerF\eDummy.ServerF.csproj...
  CACHE https://api.nuget.org/v3/vulnerabilities/index.json
```

.NET 6 升级到.NET 8

```
08:52:05.361[I]==========
08:52:06.190[I]Start eStation 1.
08:52:06.227[I]EXEC:insmod /syst
08:52:06.454[I]EXEC:mkdir -p /ro
08:52:08.614[I]F_ID:5484BDEBA165
08:52:08.618[I]M_ID:P202311280000
08:52:09.768[I]D_A_OK:90:A9:F7:3
08:52:10.353[I]#0. Open I2C OK.
08:52:11.029[I]EXEC:stty -F /dev
08:52:11.206[I]Open Result:/dev/
08:52:11.249[I]Open Result:/dev/
08:52:11.2[I]Open Result:/dev/
08:52:11.3[I]Open Result:/dev/
08:52:11.[I]Open Result:/dev/
08:52:11.424[I]Open Result:/dev/
08:52:11.428[I]#1. Open serial p
08:52:11.447[I]#2. Start eDummy
08:52:11.491[I]#3. Start eDummy
08:52:11.510[I]#4. AP05_NO_USB_C
08:52:11.515[I]Start UDP Receive
08:52:11.520[I]#5. Start_Ble_Con
08:52:11.543[I]#6. Start_Heartbe
08:52:11.559[I]#7. Start_Reset_T
08:52:11.567[I]#8. Running main
08:52:11.590[W]Current Status:In
08:52:12.068[W]Client_Disconnect
08:52:12.076[I]Try connect to:ws
```

```
08:32:51.065[I]==========
08:32:51.956[I]Start eStation 1.1.63.0, Type
08:32:51.985[I]EXEC:insmod /system/lib/modu
08:32:52.254[I]EXEC:mkdir -p /root/update/
08:32:54.383[I]F_ID:5484BDEBA165A24EAD100E9
08:32:54.386[I]M_ID:P20231128000001
08:32:55.201[I]D_A_OK:90:A9:F7:30:00:3C
08:32:55.831[I]#0. Open I2C OK.
08:32:56.110[I]EXEC:stty -F /dev/ttyS1 ispe
08:32:56.278[I]Open Result:/dev/ttysWK0:True
08:32:56.320[I]Open Result:/dev/ttysWK1:True
08:32:56.3[I]pe Result:/dev/ttysWK2:True
08:32:56.4[I]pe Result:/dev/ttysWK3:True
08:32:56.4[I]pe Result:/dev/ttyS0:True.
08:32:56.490[I]Open Result:/dev/ttyS4:True.
08:32:56.494[I]#1. Open serial ports OK.
08:32:56.511[I]#2. Start eDummy proxy OK.
08:32:56.540[I]Start UDP Receiver...
08:32:56.549[I]#3. Start eDummy client OK.
08:32:56.567[I]#4. AP05_NO_USB_CONFIG
08:32:56.575[I]#5. Start_Ble_Config_Thread
08:32:56.586[I]#6. Start_Heartbeat_Thread
08:32:56.597[I]#7. Start_Reset_Thread
08:32:56.602[W]Current Status:Init, Socket
08:32:56.612[I]#8. Running main work thread
08:32:57.141[W]Client_Disconnect.
08:32:57.151[I]Try connect to:wss://192.168
```

.NET8.NET6

.NET 6 启动耗时：6.886秒
.NET 8 启动耗时：4.656秒

.NET 8启动比.NET 6快：

约33%

.NET 6 升级到.NET 8

SDK | RT | AOT | PowerShell | PowerShell

```
top - 08:40:48 up 39 min,  2 users,  load average: 0.31, 0.42, 0.47
Tasks: 116 total,   1 running, 115 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.6 us,  6.5 sy,  0.0 ni, 9[illegible] id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :    977.0 total,    412.6 free,    [illegible]2.7 used,    451.7 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    836.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2083 root      20   0 4351268  70472  43628 S  51.8   7.0   3:12.01 dotnet
  344 root      20   0 3402684  40564  29536 S   1.0   4.1   0:27.37 dotnet
 2080 root      20   0       0      0      0 I   1.0   0.0   0:04.12 kworker/0:2-events
 2091 root      20   0       0      0      0 I   1.0   0.0   0:03.10 kworker/1:0-events
```

.NET 6

SDK | RT | AOT | PowerShell | PowerShell

```
top - 08:34:03 up 33 min,  2 users,  load average: 0.49, 0.70, 0.56
Tasks: 121 total,   1 running, 120 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.9 us,  9.0 sy,  0.0 ni, 86.9 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :    977.0 total,    409.8 free,    102.1 used,    465.0 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    833.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2048 root      20   0  261.1g  82352  64600 S  62.4   8.2   0:51.55 dotnet
 2080 root      20   0       0      0      0 I   2.0   0.0   0:00.53 kworker/0:2-events
 1446 root      20   0       0      0      0 I   1.7   0.0   0:02.31 kworker/2:3-events
 1717 root      20   0       0      0      0 I   1.7   0.0   0:03.11 kworker/1:1-events
 1801 root      20   0       0      0      0 I   1.7   0.0   0:02.60 kworker/3:1-events
  344 root      20   0 3403712  40424  29536 S   0.7   4.0   0:23.36 dotnet
 2047 root      20   0    9272   3096   2648 R   0.7   0.3   0:00.57 top
```

.NET 8

同为空闲工况下，
.NET 8比.NET 6:
CPU负载多了10%
内存多了1%（5MB）

说明:
机器运行觉悟

变高了 🤭

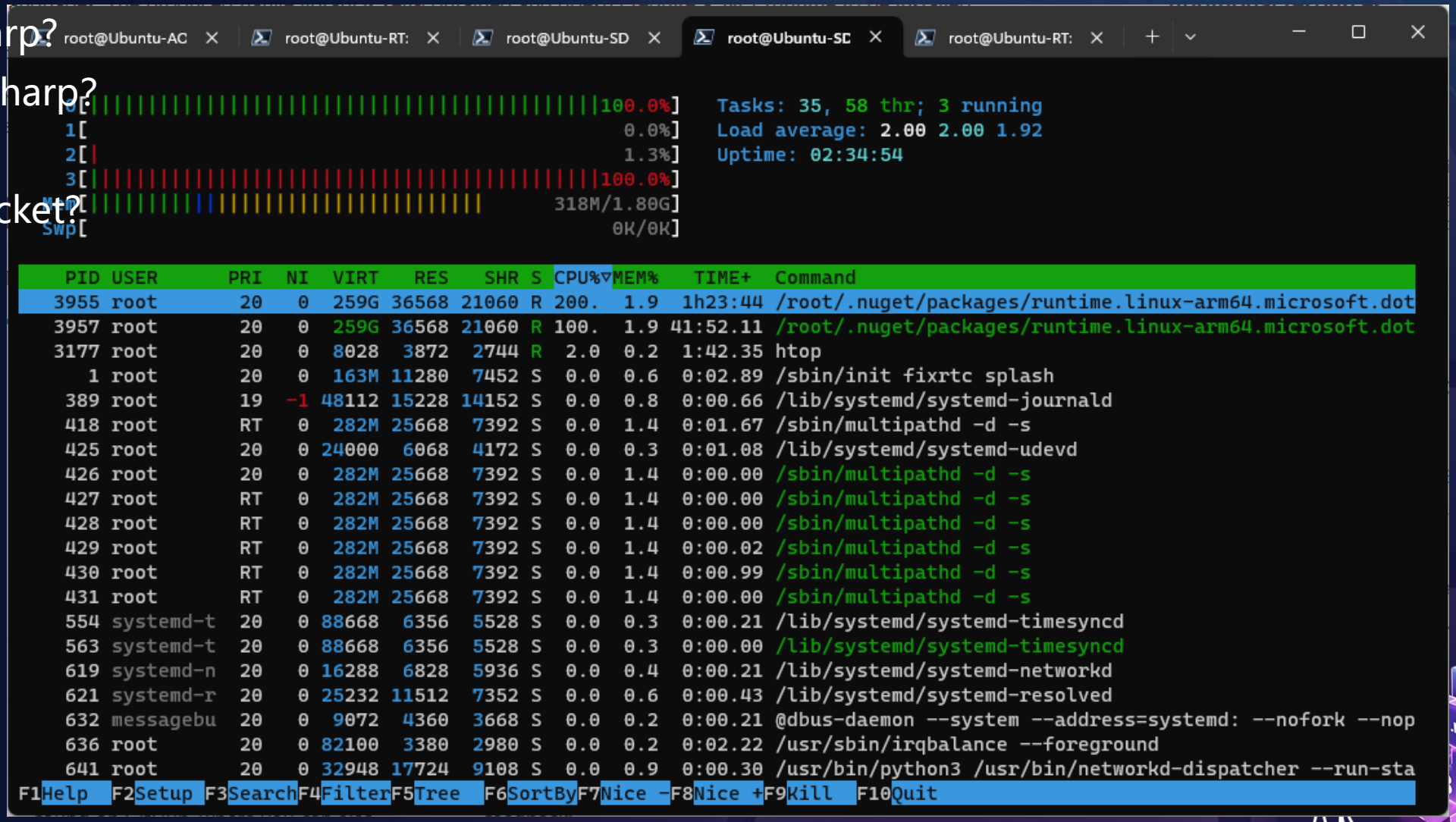
附加题：.NET 8 AOT

SkiaSharp?

ImageSharp?

Linq?

WebSocket?



通信部分 WebSocket

```
<TargetFrameworks>net40;net45;netstandard2.0;netcoreapp2.0</TargetFrameworks>
<TargetFramework>net8.0</TargetFramework>
```

```
# 用于网关与服务端通信，长链接
# 服务端：Fleck
# 客户端：ClientWebSocket

# 线程0：保持连接
Task.Factory.StartNew(async () =>
{
    while (true)
    {
        try
        {
            if (_socket != null && _socket.State == WebSocketState.Open) { ...} // Next Loop

            ReConnect(Certificate, Password);

            await _socket.ConnectAsync(_uri, _cancel);
            await _socket.SendAsync(GetDummyRegisterData(), WebSocketMessageType.Binary, true,
_cancel);
            await Task.Delay(TimeSpan.FromSeconds(15));
        }
        catch (Exception ex) { ...} // Exception handler
    }
});
```

通信部分 WebSocket

```
# 线程1：发送
Task.Factory.StartNew(async () =>
{
    while (true)
    {
        try
        {
            if (_socket is null || _socket.State != WebSocketState.Open || SendDataBuffer.IsEmpty)
            { … } // Next loop

            if (SendDataBuffer.TryPeek(out var data) && data.Count > 0)
            {
                await _socket.SendAsync(data, WebSocketMessageType.Binary, true, _cancel);
                DataBuffer.TryDequeue(out data);
            }
        }
         catch (Exception ex) { …} // Exception handler
    }
});
```

通信部分 WebSocket

```
# 线程2：接受
Task.Factory.StartNew(async () =>
{
    List<ArraySegment<byte>> cache = new();
    while (true)
    {
        try
        {
            if (_socket is null || _socket.State != WebSocketState.Open) { … } // Wait thread #0 to re-connect
            try
            {
                byte[] buffer = new byte[8192]; // Buffer size is 8K
                var result = await _socket.ReceiveAsync(new ArraySegment<byte>(buffer), _cancel);
                if (result.Count > 0)
                {
                    var temp = new byte[result.Count];
                    Array.Copy(buffer, temp, result.Count);
                    cache.Add(temp);
                }
                if (!result.EndOfMessage) continue;
            }
            catch { …} // Eat exception and continue
```

通信部分 WebSocket

```
# 线程2：接受（合并）
        var data = new byte[cache.Sum(x => x.Count)];
        var index = 0;
        foreach (var item in cache)
        {
            Array.Copy(item.ToArray(), 0, data, index, item.Count);
            index += item.Count;
        }

        if (data.Length == 0) continue;

        ... // Data Process
        cache.Clear();
    }
    catch (Exception ex)
    {
        ... // Exception handler
        cache.Clear();
    }
  }
});
```

通常需要寻找帧头，并得到数据包的长度，计算当前收到的数据总长是否达到
也会计算CRC32，或包计数器等。然后再进行数据处理。

不要移到final里，否则每次循环都会清除cache。

通信部分 MQTT

```
# 用于第三方开发者进行设备集成
# 使用MQTTnet
// Mqtt Factory
var factory = new MqttFactory();
var options = new MqttClientOptionsBuilder()
                    .WithClientId(clientID)
                    .WithTcpServer(serverAddress)
                    .WithCredentials(userID, password)
                    .WithKeepAlivePeriod(TimeSpan.FromSeconds(10))
            .Build();
// Mqtt Client
var client = _factory.CreateMqttClient() as MqttClient;
client.ApplicationMessageReceivedAsync += Instance_ApplicationMessageReceivedAsync;
client.ConnectedAsync += Instance_ConnectedAsync;
client.DisconnectedAsync += Instance_DisconnectedAsync;
```

通信部分 MQTT

```
// Subscribe topic
 var subscript = factory
            .CreateSubscribeOptionsBuilder()
            .WithTopicFilter(t => t.WithTopic(TopicNameA))
            .Build();
client.SubscribeAsync(subscript);

// Receive meesage
// MqttApplicationMessageReceivedEventArgs arg
var topic = arg.ApplicationMessage.Topic;
var data = arg.ApplicationMessage.PayloadSegment.ToArray();

// Publish message
client.PublishAsync(new MqttApplicationMessage()
{
            Topic = TopicNameB,
            QualityOfServiceLevel =
MQTTnet.Protocol.MqttQualityOfServiceLevel.AtMostOnce,
            PayloadSegment = MessagePackSerializer.Serialize(MessageContent)
});
```

通信部分 UDP

```
# 用于局域网通信，如发现和配置、内网协同
# 也用于高频数据通信，个人建议避免拆包和双向确认

var client = new UdpClient(port);
var broadcast = new IPEndPoint(IPAddress.Broadcast, port);

// Send
client.Send(data, data.Length, remoteAddress);

// Receive
 var receive = await client.ReceiveAsync();
 var buffer = receive.Buffer;
```

运维部分 SSH.NET

```
// Connection options
var connect = new ConnectionInfo(
            host,
            userName,
            new PasswordAuthenticationMethod(userName, password));
var client = new SshClient(connect);

// Connect
client.Connect();

// Execute command，执行命令
client.RunCommand(command).Execute(); // 或: client.CreateCommand
// Shell stream with communication，交互
var shell = client.CreateShellStream(terminal, columns, rows, width, height, bufferSize);
```

运维部分 SSH.NET

```
// Upload file
var client = new SftpClient(host, port, username, password);
using (FileStream fileStream = File.OpenRead(localPath))
{
        client.Connect();
        client.UploadFile(fileStream, remotePath, null);
        client.Disconnect();
}

// Download file
...
        client.DonwloadFile(remotePath, fileStream); // 或
        var bytes = client.ReadAllBytes(remotePath);
        File.WriteAllBytes(localPath, bytes);
...

// Delete file
...
        client.Delete(remotePath); // 此外还有RenameFile，ListDirectory等
...
```

运维部分 WinSCP

```
# WinSCP提供可视化的界面用于文件同步和命令执行
# 也提供代码级的集成
// Connection options
var option = new SessionOptions
{
        Protocol = Protocol.Sftp,
        HostName = ip,
        UserName =  “YouLoginIDHere",
        Password = "YourPasswordHere",
        SshHostKeyPolicy = SshHostKeyPolicy.GiveUpSecurityAndAcceptAny,
};

// Transfer options
var transferOptions = new TransferOptions()
{
        TransferMode = TransferMode.Binary
};
```

运维部分 WinSCP

```
// Connect
using Session session = new();
session.Open(option);

// Upload files
var transferResult = session.PutFiles(
        "Source_Path",
        "Destination_Path",
        false, // Keep, no remove
        transferOptions);

// Throw on any error
transferResult.Check();

// Print results
foreach (TransferEventArgs transfer in transferResult.Transfers.Cast<TransferEventArgs>())
{
        Console.WriteLine("Upload of {0} succeeded", transfer.FileName);
}
```

运维部分-配置

```
// USB
Task.Run(async () =>
{
    while (++i < byte.MaxValue)
    {
        try
        {
            await Task.Delay(TimeSpan.FromSeconds(5));

            var disks = BashHelper.ExecuteCommandWithReturn("fdisk -l"); // List
            if (string.IsNullOrEmpty(disks)) continue;
            var last = disks.Split('\n').Where(n => !string.IsNullOrEmpty(n.Trim())).LastOrDefault();
            if (string.IsNullOrEmpty(last) || !last.StartsWith("/dev/sd")) continue;

            await BashHelper.ExecuteCommand($"mount {last[..9]} {USBPath}"); // Mount
            if (File.Exists(ConfigFilePath)) { UsbConfig(configPath); break; }
        }
        catch { ... }
    }
}
```

List

Mount

运维部分-配置

```
# 蓝牙，以DS-531为例
# 当存在多个设备时，需要设置Prefix名称，用于过滤
# 并且与数码管搭配，为每个设备分配一个ID，用于匹配

// 0. Rename host name
File.WriteAllText("/etc/hostname", "eStation" + MAC[^2..]);

// 1. Register BLE module
await BashHelper.ExecuteCommand($"stty -F /dev/ttyS1 ispeed 9600 ospeed 9600 -echo &&
echo \"AT+NAMEeStation{MAC[^2..]}\" > /dev/ttyS1 && sync && sync");
```

运维部分-配置

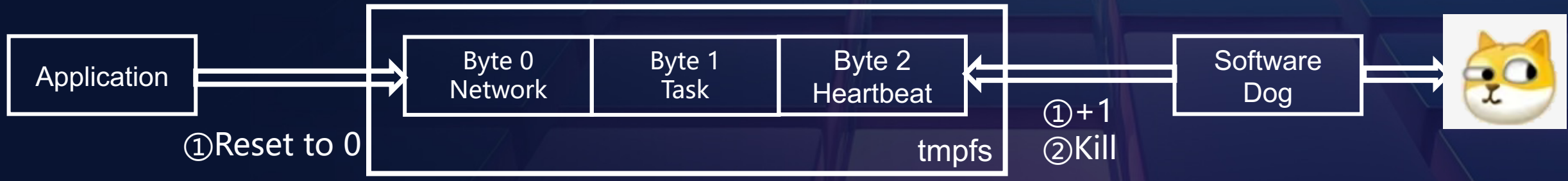
```
// 2. Receive
var buffer = new List<byte[]>();
var serialPort = new SerialPort(HardwareInfor.BLE, 9600, Parity.None, 8, StopBits.One);
serialPort.Open();

// Loop
if (!serialPort.IsOpen || serialPort.ReadBufferSize == 0) { ...} // Wait a moment
byte[] bytes = new byte[1024];
int length = serialPort.Read(bytes, 0, bytes.Length);
if (length == 0) continue;
byte[] temp = new byte[length];
Array.Copy(bytes, temp, length);
buffer.Add(temp);

... // Merge & Process
```

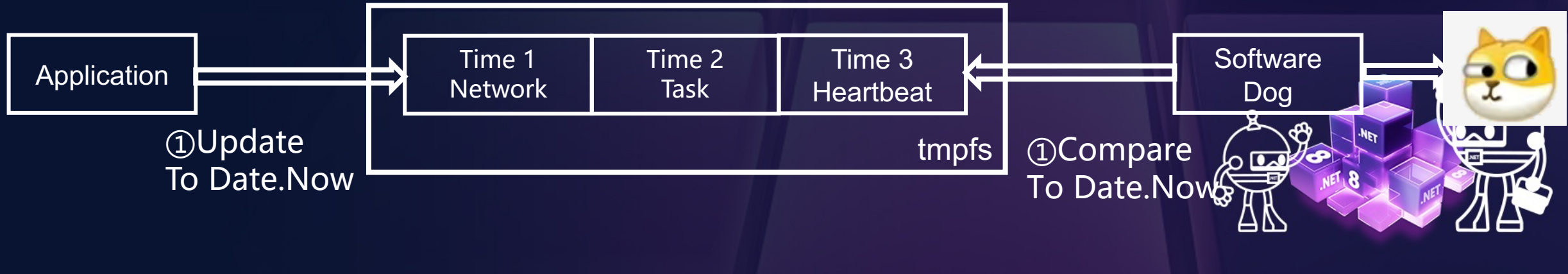
运维部分-Watchdog & OTA

对PX30看门狗的改进：



#PX30看门狗的问题：喂的太频繁，200ms~500ms就要投喂一次：

```
 await Task.Delay(TimeSpan.FromSeconds(0.5));
DogController.Write(DogFeed, PinValue.Low);
await Task.Delay(TimeSpan.FromSeconds(0.1));
DogController.Write(DogFeed, PinValue.High);
```



运维部分-Watchdog & OTA

```
# Application
var files = Directory.GetFiles(FirmwarePath);
if (files is null || files.Length == 0) return;
foreach (var file in files)
{
        ... // Find firmware
}
if (folder.Length == 0) return;
ExecuteCommand($"cp -r {folder}/* {ApplicationPath}");
ExecuteCommand($"rm -rf {FirmwarePath} && sync && sync && sync && reboot");
```

```
# BLE 模块 & ESL Firmware
# 提前缓存，后到者为准
# 一般需要分多帧发送，需要防止变砖
```

```
// Header
var header = new byte[] { 0x40, 0x4F, 0x54, 0x41, ret[0], ret[1], ret[2], pkg[0], pkg[1], 0x02 }
// Fist frame
var first = new byte[2056];
Array.Copy(new byte[] { 0x24, ret[0], ret[1], ret[2], pkg[0], pkg[1] }, 0, first, 0, 6);
Array.Copy(firmware[0..2048], 0, first, 6, 2048);
Array.Copy(CRC16_IBM(firmware[0..2048], 2048), 0, first, 2054, 2);
body.Add(first);
// Next frame
var next = new byte[] { 0x24, 0x00, 0x08, 0x00 };
for (int i = 1; i < count; i++)
{
    var pkgData = firmware[(i * 2048)..((i + 1) * 2048)];
    var frame = new byte[2056];
    var crc = CRC16_IBM(pkgData, pkgData.Length);
    Array.Copy(next, 0, frame, 0, 4);
    Array.Copy(Int2Bytes(i), 0, frame, 4, 2);
    Array.Copy(pkgData, 0, frame, 6, pkgData.Length);
    frame[^2] = crc[0]; frame[^1] = crc[1];
    body.Add(frame);
}
```

运维部分-REST

当用户配置参数错误时，需要提供重置的功能。

```
_gpio.OpenPin(HardwareInfor.PinRst);
_gpio.SetPinMode(HardwareInfor.PinRst, PinMode.Input);

Task.Run(async () =>
{
    while (true)
    {
        try
        {
            while (_gpio.Read(HardwareInfor.PinRst) == HardwareInfor.PinRstDefault)
             await Task.Delay(2000);
            var date = DateTime.Now;
            while (_gpio.Read(HardwareInfor.PinRst) == HardwareInfor.PinRstHold)
            {
                await Task.Delay(20);
                var j = (DateTime.Now - date).Seconds;
                if (j >= 5) goto Reset;
```

运维部分-REST

```
            LED.Display("RST" + (5 - j).ToString(), false, false); // RST5,4,3,2,1
        }
        DisplayCurrentStatus(CurrentStatus); // Restore
    }
    catch { }
}
Reset:
    LED.Display("boot");
    Log.Warning("Detect reset event, start reset...");
    File.Copy(F("appsettings.Default.json"), F("appsettings.json"), true);
    File.Copy(@"/etc/network/interfaces.default", @"/etc/network/interfaces", true);
    await BashHelper.ExecuteCommand("sync && sync && sync && reboot");
});
```

运维部分-OTA


② Kill
Server
①推送固件
IoT Device
Firmware
Watchdog
③ Reboot & Check


Server
①推送固件
IoT Device
Firmware
BEL MOD


.NET
8
.NET
.NET

一些开发心得

- 首要的编程风格是：韧性。
- 与容器类似，网关避免对数据进行持久化，而由服务端/设备端提供数据。
- 网关处理最多的场景是数据传输速率的诧异。如：
  - 对于高频次数据收发操作时，避免在Socket/SerialPort数据接收处进行任何处理，直接进入队列。
  - 蓝牙模块与串口存在速率差：蓝牙模块的速率很高，如BLE 5.0理论上可以达到2Mbps，而串口很难达到这一速率，如波特率在115200时，传输速率大约为11.25KB/秒。
  - 串口的默认BufferSize为4KB，如果需要扩容，需要修改内核和/或应用层的设置。

感谢聆听！