Rinnovo ed evoluzioni di un prodotto

Trasformare il codice del passato in soluzioni future

Mirco Vanini – Simone Tolotti

xedotnet.org

Un ringraziamento a...

Chi siamo...

Mirco Vanini
Microsoft MVP Developer Technologies

Consulente con oltre 35 anni di esperienza, specializzato in soluzioni industriali ed embedded, cofondatore della community XeDotNet, relatore e Microsoft MVP dal 2012

Simone Tolotti
Software and Firmware Architect @ Schneider Electric

Sviluppatore Firmware Embedded: C/C++ su piattaforme ARM ed Embedded Linux per prodotti industriali con più di 15 anni di esperienza

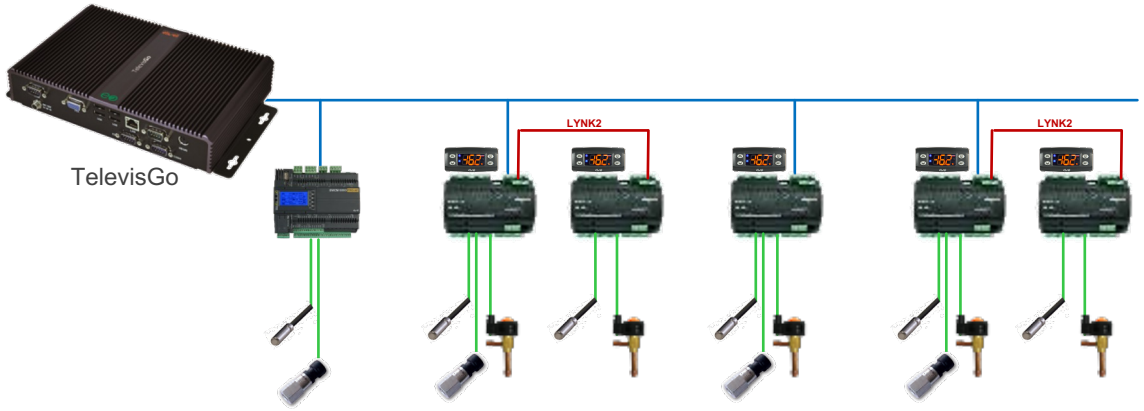
MVP
Microsoft®
Most Valuable Professional

@MircoVanini
www.proxsoft.it
https://www.linkedin.com/in/proxsoft

https://github.com/simonetolotti
simone@tolotti.it
https://www.linkedin.com/in/simone-tolotti-84812224/

Prodotto Televis Go 1/2

- Qual è l'applicazione?
  - Supervisore per Refrigerazione Commerciale
  - Dominio Food Retail / **Supermarket**
  - Tecnologia bus di campo come per l'Industrial Automation



- Cosa fa
  - Acquisizione dati, monitoraggio strumenti e allarmistica
  - Salvataggio dati (DB)
  - Reportistica e HACCP
  - Analisi dati con grafici e logs
  - Manutenzione remota e integrazione con sistemi di Building

Prodotto Televis Go 2/2

- Piattaforma Tecnologica
  - CPU Intel Celeron @ 1.83 GHz (N2930)
  - RAM 4GB
  - HD 200GB+
  - Win 10 Enterprise (64 bit)

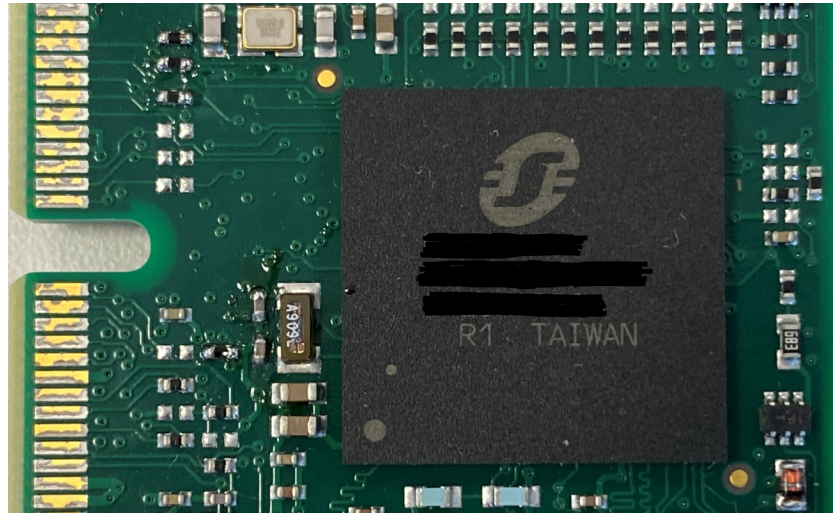
- Software
  - .Net Framework v4.8
  - Database SQL
  - Monolite Data Acquistion + Business Logic + Asp .Net

- Aggiornamenti negli anni
  - HW (start x86…)
  - OS (start Windows CE, Windows XP Embedded…)
  - .NET (Compact Framework 2.0, Net Framework 3.5…)
  - SW (Refresh alcune pagine in React)

Nuove esigenze…

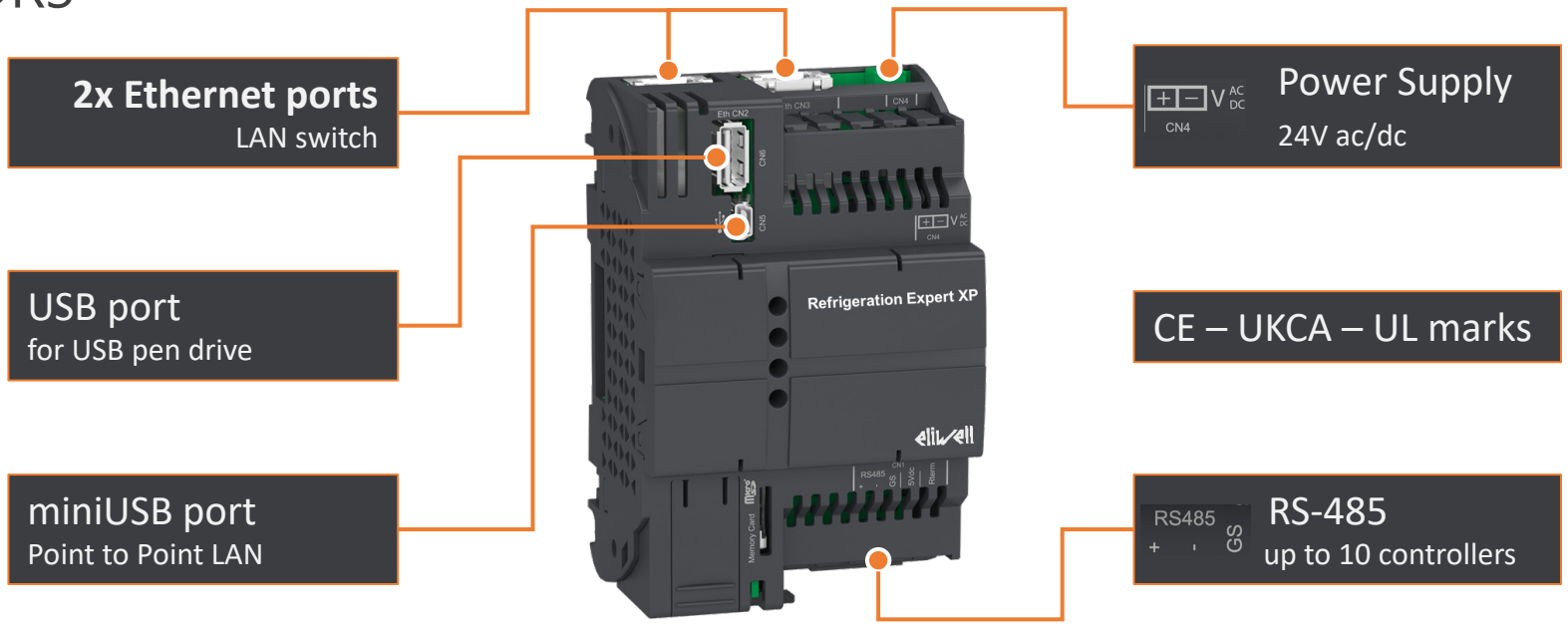
- Dopo 15 anni, i limiti:
  - Architettura PC → HW soggetto ad obsolescenza
  - Soluzione competitiva per supermarket meno food retail
  - Dimensioni e consumi
  - UI da svecchiare
  - Licenza di Windows
  - Gestione macchina Windows
  - Cybersecurity

Nuova Architettura HW/FW di Gruppo



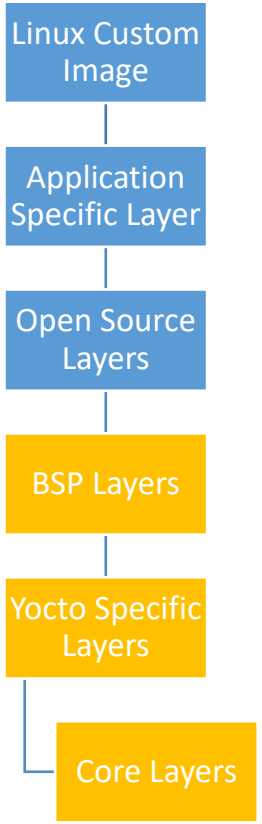
Cambio Tecnologico 1/2

- HW
  - SoC: dual Cortex A7 (ARMv7) @ 500MHz
  - 1024 MB RAM DDR3
  - 1024 MB Nand



Cambio Tecnologico

- OS
  - Yocto Linux



Vantaggi nuova architettura

- HW in «casa»
  - Progettazione e Produzione
- Design in «casa»
  - 4 din
- SoC in «casa»
  - Documentazione non pubblica
  - Architettura Arm
- OS in «casa»
  - Personalizzato per il domino
  - Uso di Yocto

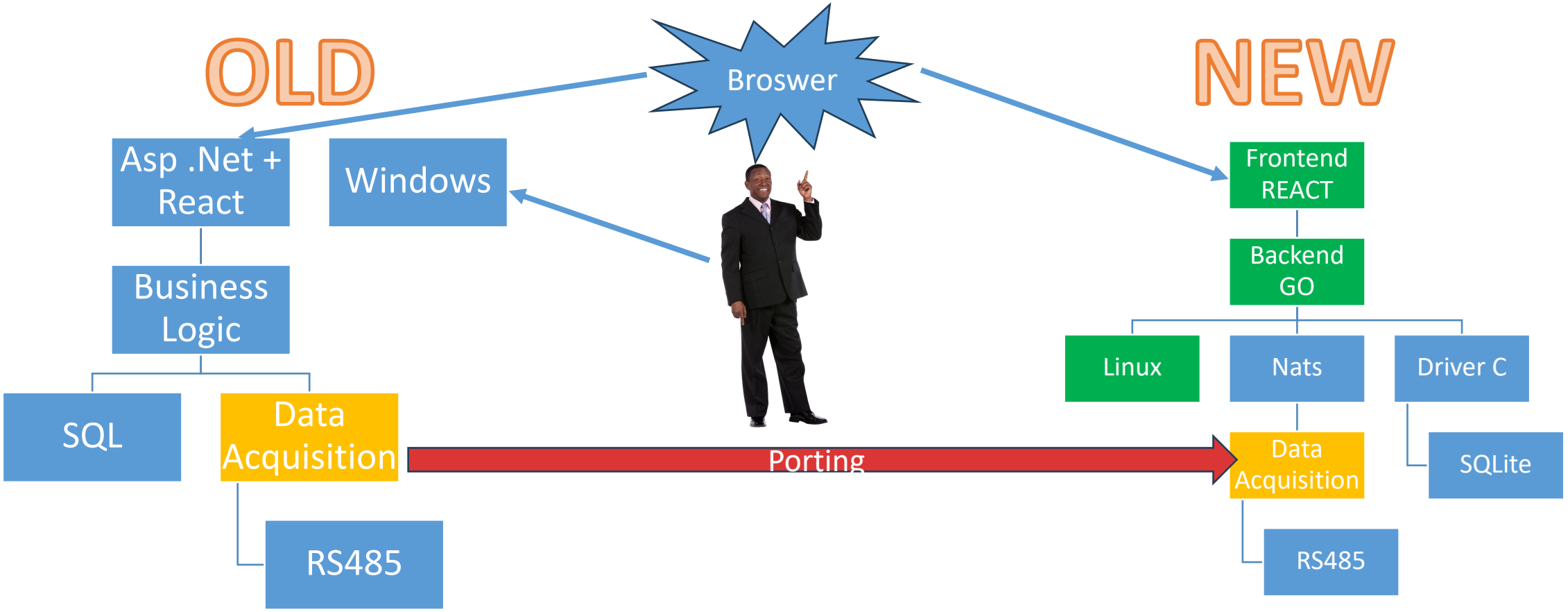
- Maggior controllo produzione
- Prodotto Compatto
- Cybersecurity
- Efficientamento energetico
- No licenza Windows
- Scalabile

+ Efficiente

Miglior Target per «Piccoli»

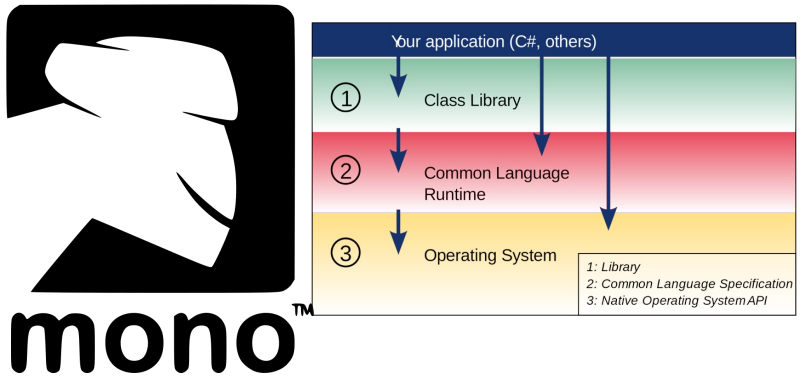
E il Software ??? 3 Obiettivi:

- Riciclare tutti i driver per la comunicazione di centinaia di strumenti
- Riciclare l'Expertise del team Webapp sulla stessa piattaforma Linux
- Utente per alcune opzioni amministrative, non dovrà più accedere al OS ma solo tramite la Webapp



Porting codice Data Acquisition

- Mono Project
  - Esiste dal 2004, ex Mono hanno fondato Xamarin*…
  - Esegue applicazioni .Net Framework multipiattoforma
  - Install Mono on Linux
  - ARM | Mono

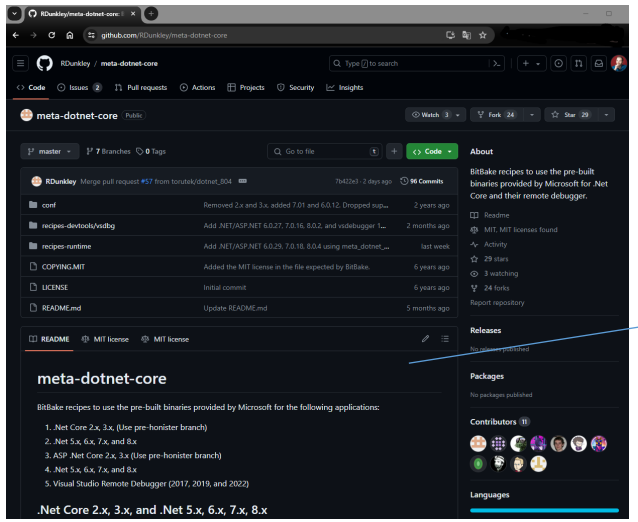


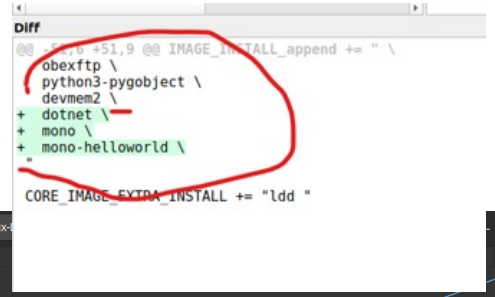
- .NET Core 6.0 LTS
  - 13 December 2022
  - Finalmente Multipiattoforma
    - Windows, Linux, macOS, Android and iOS
    - IA-32, x86-64, s390x and ARM
  - Install .NET on Linux by using an install script or by extracting binaries
  - Dependencies

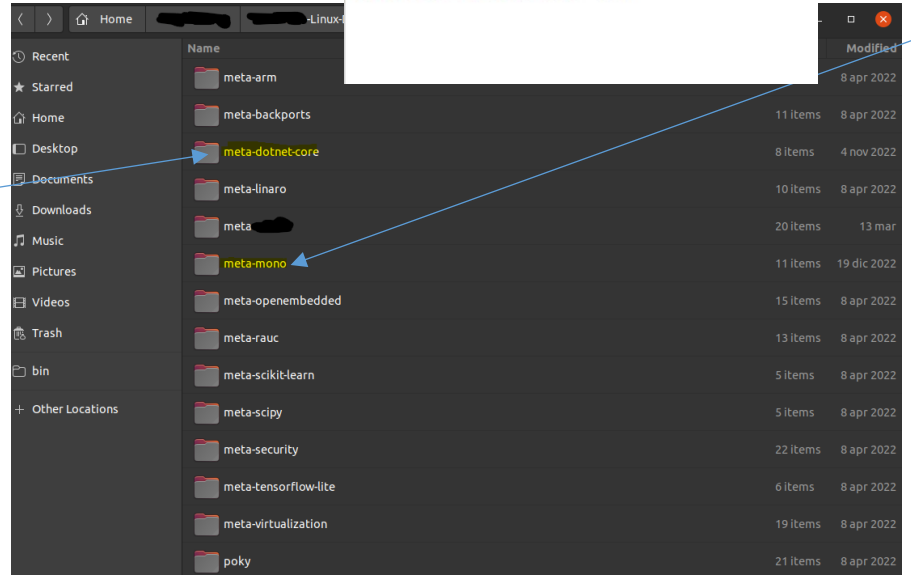
*ma questa è un'altra storia

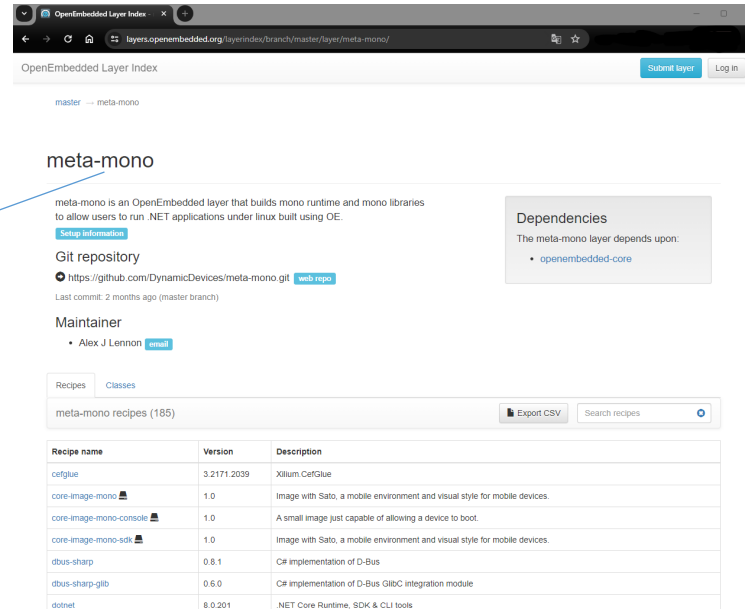
Yocto & .Net

- Aggiunta Meta-layers:
  - Mono: https://layers.openembedded.org/layerindex/branch/master/layer/meta-mono/
  - .net Core: https://github.com/RDunkley/meta-dotnet-core
- Aggiunta Ricette

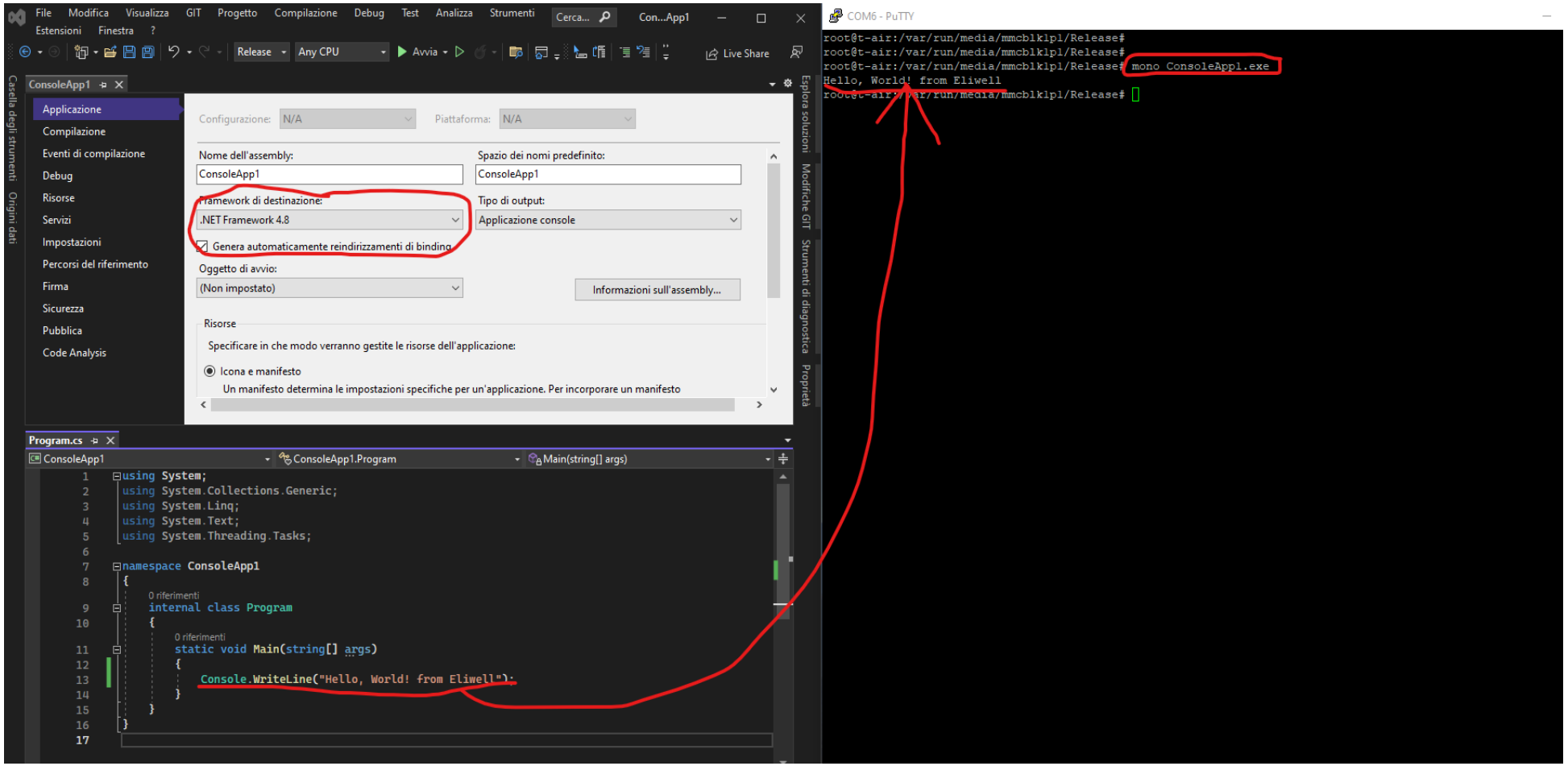






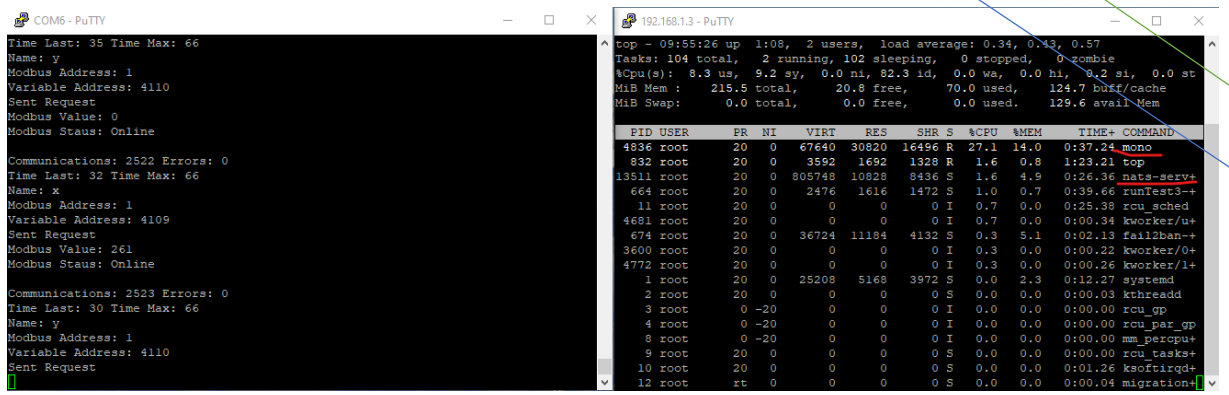


Hello world, «Mono»



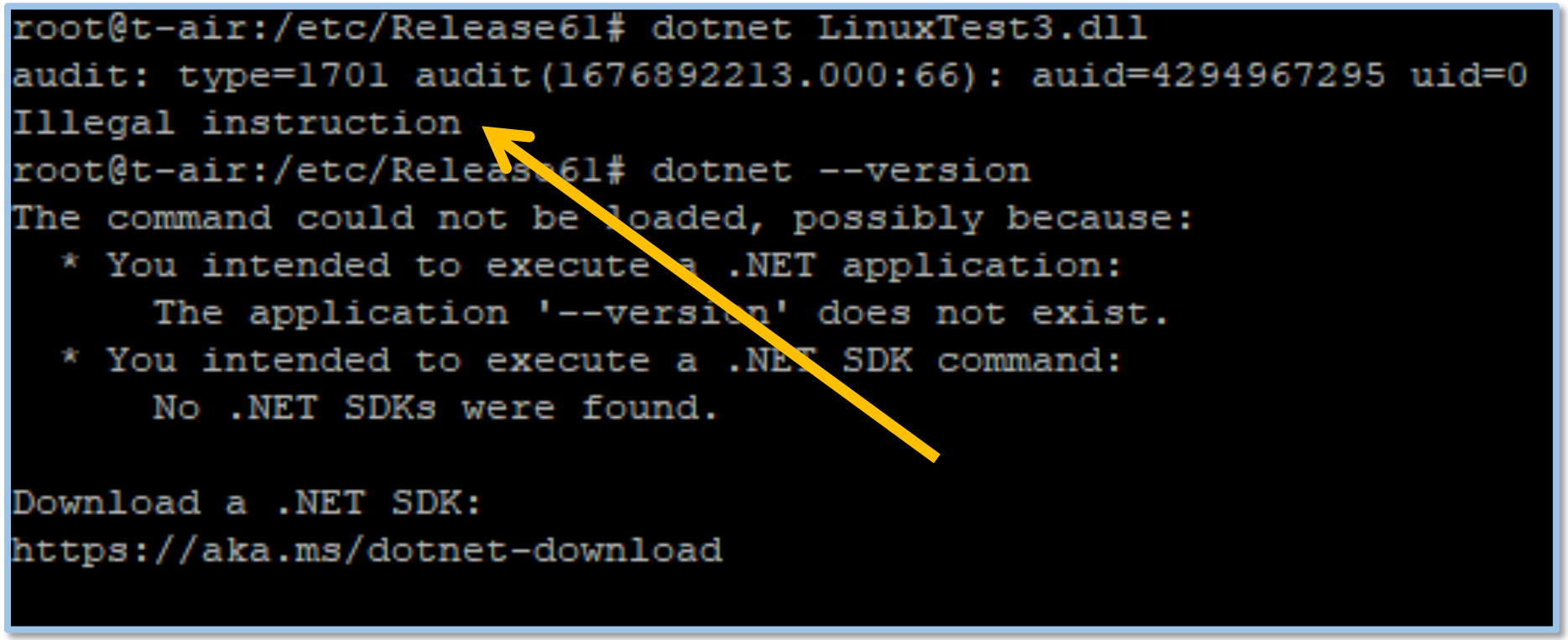
POC Data Acquisition «Mono»

- Deserialization parameters from Json
- Continuos Loop
  - read from Modbus RS485
  - @9600, parity none, 1 stop
  - No delay beetwen packet (maximum bandwith)
  - Approximately 7+3,5*+5+3,5 ms /resource
  - Total cycle: 2 resources (50ms)
- Push data to Nats server
  - Create and push Json
  - Send packet 1 each 10 read from field (push every 250ms)




```
foreach (var device in devices)
{
    Console.WriteLine("Name: " + device.Name);
    Console.WriteLine("Modbus Address: " + device.ModbusInstrumentAddress);
    Console.WriteLine("Variable Address: " + device.ModbusVariableAddress);

    long unixtime;

    device.Status = DeviceStatus.Offline;

    if (modBus.ModbusRequest03((byte)device.ModbusInstrumentAddress, (ushort)device.ModbusVariableAddress, 1))
    {
        Console.WriteLine("Sent Request");

        while (modBus.Ready != true)
        {
            // do nothing !!!
            Thread.Sleep(10);
        }

        unixtime = DateTimeOffset.UtcNow.ToUnixTimeSeconds();

        if (modBus.Valid)
        {
            int bufferValue = ModbusConverter.Bytes_to_Integer(modBus.ReceivedFrame[3], modBus.ReceivedFrame[4]);

            device.Value = bufferValue;
            device.Status = DeviceStatus.Online;
            Console.WriteLine("Modbus Value: " + device.Value);
        }
        else
        {
            Console.WriteLine("Received bytes: " + modBus.ReceivedFrame.Length);

            foreach (byte b in modBus.ReceivedFrame)
            {
                Console.WriteLine(b);
            }
        }

        Console.WriteLine("Modbus Staus: " + device.Status);
        Console.WriteLine();
        Console.WriteLine("Communications: " + modBus.Statistics.NumberOfCommunication.ToString() + " Errors: " + modBus.Statistics.NumberOfError.ToString());
        Console.WriteLine("Time Last: " + modBus.Statistics.TimeLast.ToString() + " Time Max: " + modBus.Statistics.TimeMax.ToString());

        if (device.Cycle++ >= serialSettings.DecimationCycle)
        {
            var meta = device.Status == DeviceStatus.Online ? 0 : 1;
            var realValue = device.Value * 100;

            var jsonToRedis = "{\"name\":\"" + device.Name + "\",\"value\":" + realValue + ",\"timestamp\":" + unixtime + ",\"meta\":" + meta + "}";

            c.Publish("ModbusResource", Encoding.UTF8.GetBytes(jsonToRedis));

            Console.WriteLine(jsonToRedis);
            device.Cycle = 0;
        }
    }
    else
        Console.WriteLine("Error Sent Request");
```

Hello world, «.net Core»



- Illegal instruction on ARM Cortex-A5 #9969
- armv6: Segmentation fault while running dotnet

CPU ARM con vfpv3d16 e .NET Core

```
root@m172-si:/etc/labs# cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 12.50
Features        : half thumb fastmult vfp edsp thumbee vfpv3 vfpv3d16 tls vfpv4 idiva idivt lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5
```

.NET: “Floating-point support in the form of VFPv3-D32 or later must be present in hardware”

Build dotnet runtime

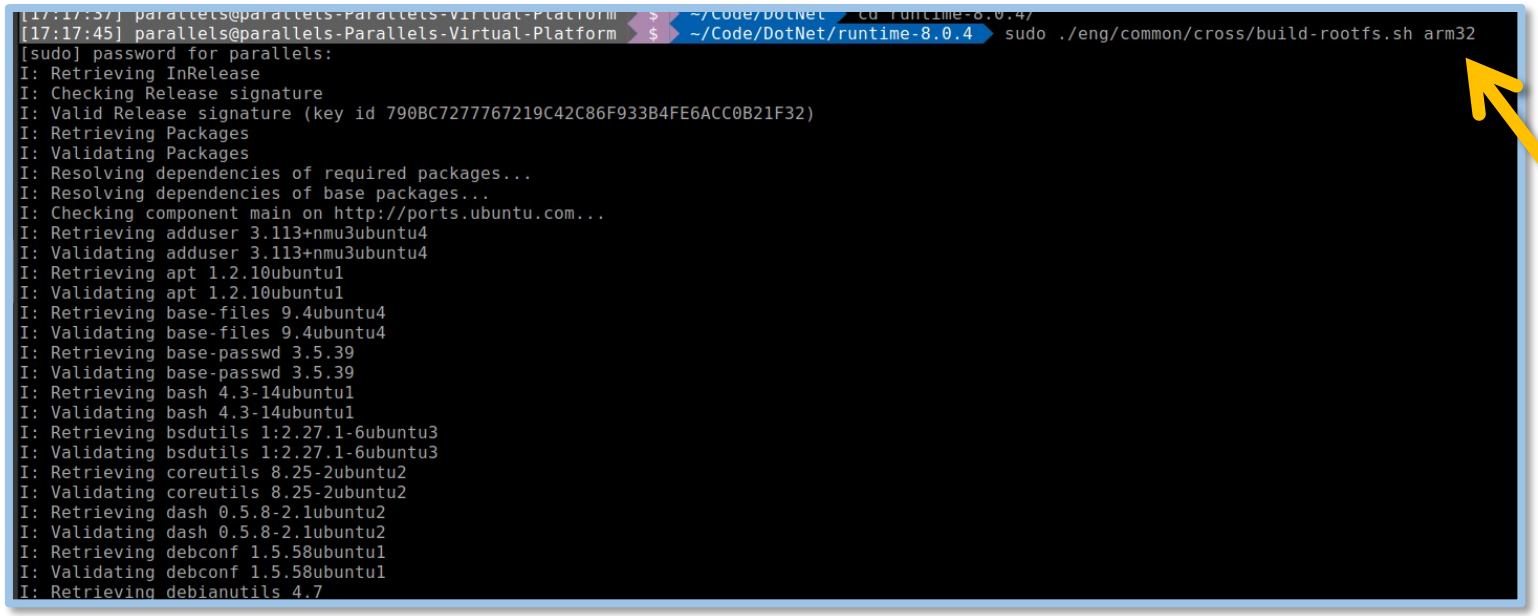
- Building CoreCLR
  - Build CoreCLR on Windows
  - Build CoreCLR on macOS
  - Build CoreCLR on Linux
  - Build CoreCLR on FreeBSD

- Requirements to build dotnet/runtime on Linux
  - CMake 3.20 or newer
  - llvm
  - lld
  - clang
  - build-essential
  - python-is-python3
  - Curl
  - git
  - lldb
  - libicu-dev
  - liblttng-ust-dev
  - libssl-dev
  - libkrb5-dev
  - zlib1g-dev
  - ninja-build (optional, enables building native code with ninja instead of make)

How do I install the latest version of cmake from the command line?

Build dotnet runtime

- dotnet runtime github (.NET 8.0.4)
- Linux Cross-Building
  - Cross-Building for Different Architectures and Operating Systems
    - Generating the ROOTFS (*Before you can attempt to do any Linux cross-building, you will need to generate the ROOTFS corresponding to the platform you want to target.)*



Build dotnet runtime

- Cross-Compiling CoreCLR

```
export ROOTFS_DIR=/home/parallels/Code/DotNet/runtime-8.0.4/.tools/rootfs/arm
```

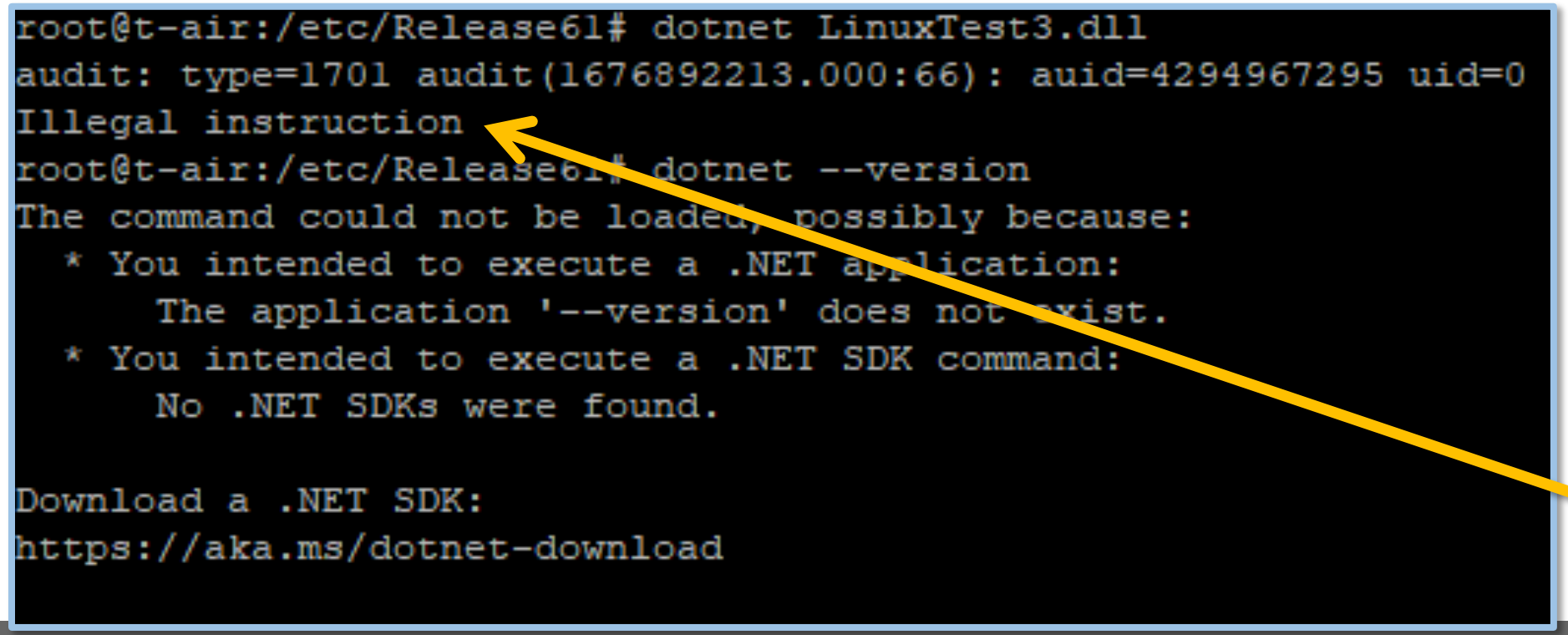
- Cross-Compiling CoreCLR for other VFP Configurations
  - The default ARM compilation configuration for CoreCLR is armv7-a with thumb-2 instruction set, and VFPv3 floating point with 32 64-bit FPU registers.
  - **CoreCLR JIT requires 16 64-bit or 32 32-bit FPU registers.**

  `CLR_ARM_FPU_CAPABILITY` Used by the PAL code to decide which FPU registers should be saved and restored during context switches (the supported options are 0x3 and 0x7):

  Bit 0 unused always set to 1.
  Bit 1 corresponds to 16 64-bit FPU registers.
  Bit 2 corresponds to 32 64-bit FPU registers.

Build dotnet runtime

```
./build.sh --clean --subset clr+libs --configuration Release --cross --arch arm

./build.sh --subset clr+libs --configuration Release --cross --arch arm --cmakeargs
"-DCLR_ARM_FPU_CAPABILITY=0x3" --cmakeargs "-DCLR_ARM_FPU_TYPE=vfpv3-d16" --
framework "net7.0"
```



Build dotnet runtime

- Unable to cross-compile CoreCLR for Linux ARM vfpv4-d16

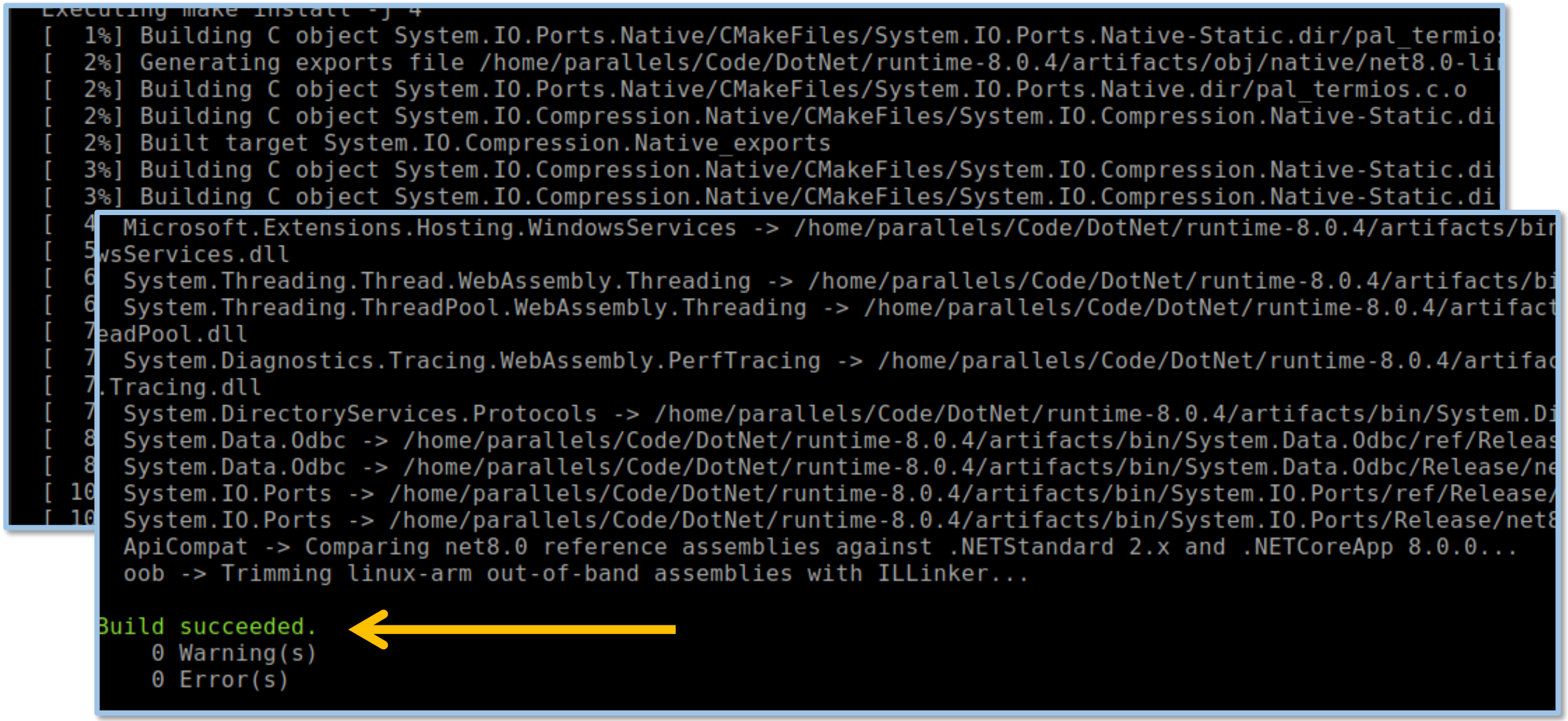
  src/native/corehost/corehost.proj

  src/native/libs/build-native.proj

```
78    <BuildArgs Condition="'$(CrossBuild)' == 'true'">$(BuildArgs) -cross</BuildArgs>
79    <BuildArgs Condition="'$(Compiler)' != ''">$(BuildArgs) $(Compiler)</BuildArgs>
80-   <BuildArgs Condition="'$(CMakeArgs)' != ''">$(BuildArgs) $(CMakeArgs)</BuildArgs>
81    <BuildArgs Condition="'$(Ninja)' == 'true'">$(BuildArgs) -ninja</BuildArgs>
82    <BuildArgs>$(BuildArgs) -runtimeflavor $(RuntimeFlavor)</BuildArgs>
```

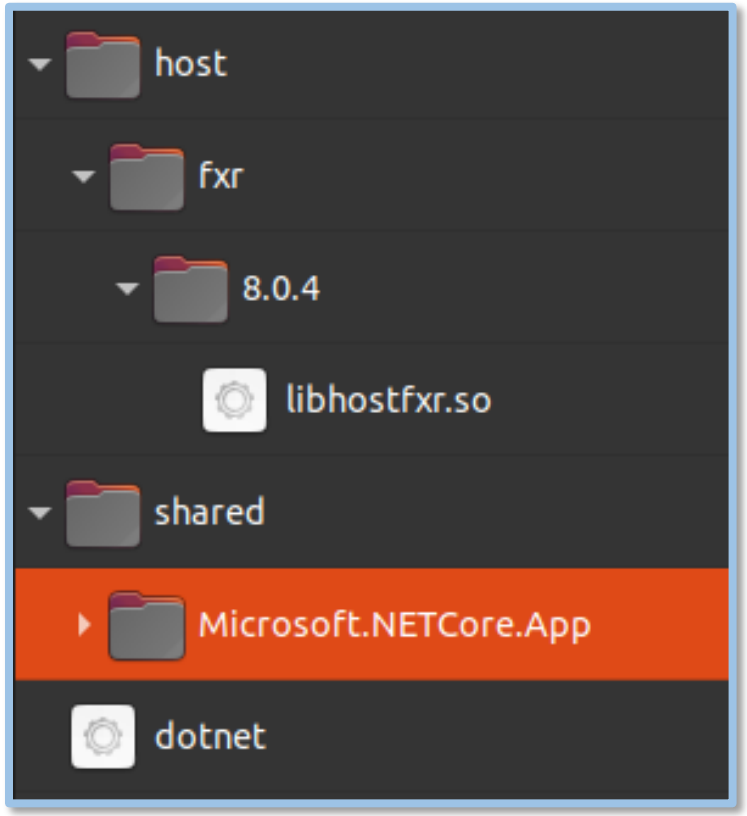
```
78    <BuildArgs Condition="'$(CrossBuild)' == 'true'">$(BuildArgs) -cross</BuildArgs>
79    <BuildArgs Condition="'$(Compiler)' != ''">$(BuildArgs) $(Compiler)</BuildArgs>
80+   <BuildArgs Condition="'$(CMakeArgs)' != ''">$(BuildArgs) -cmakeargs "$(CMakeArgs)"</BuildArgs>
81    <BuildArgs Condition="'$(Ninja)' == 'true'">$(BuildArgs) -ninja</BuildArgs>
```

Build dotnet runtime

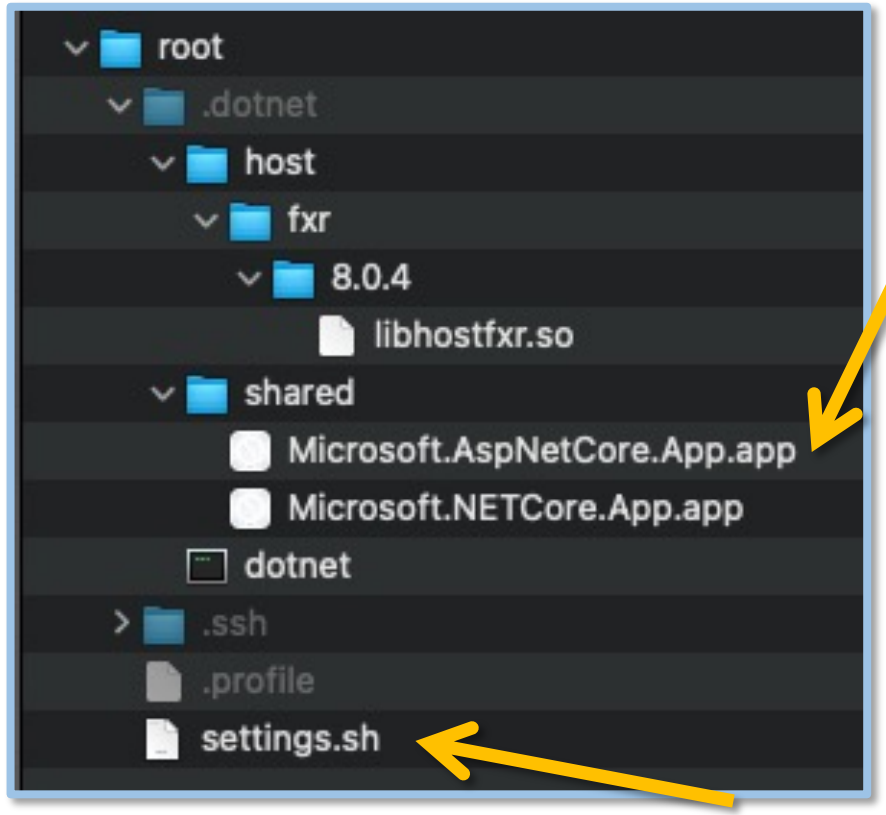


Build dotnet runtime

- /Code/DotNet/runtime-8.0.4/artifacts/bin/testhost



aspnetcore-runtime-8.0.4-linux-arm.tar.gz




```
export DOTNET_ROOT=/home/root/.dotnet
export PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
```

.NET on target

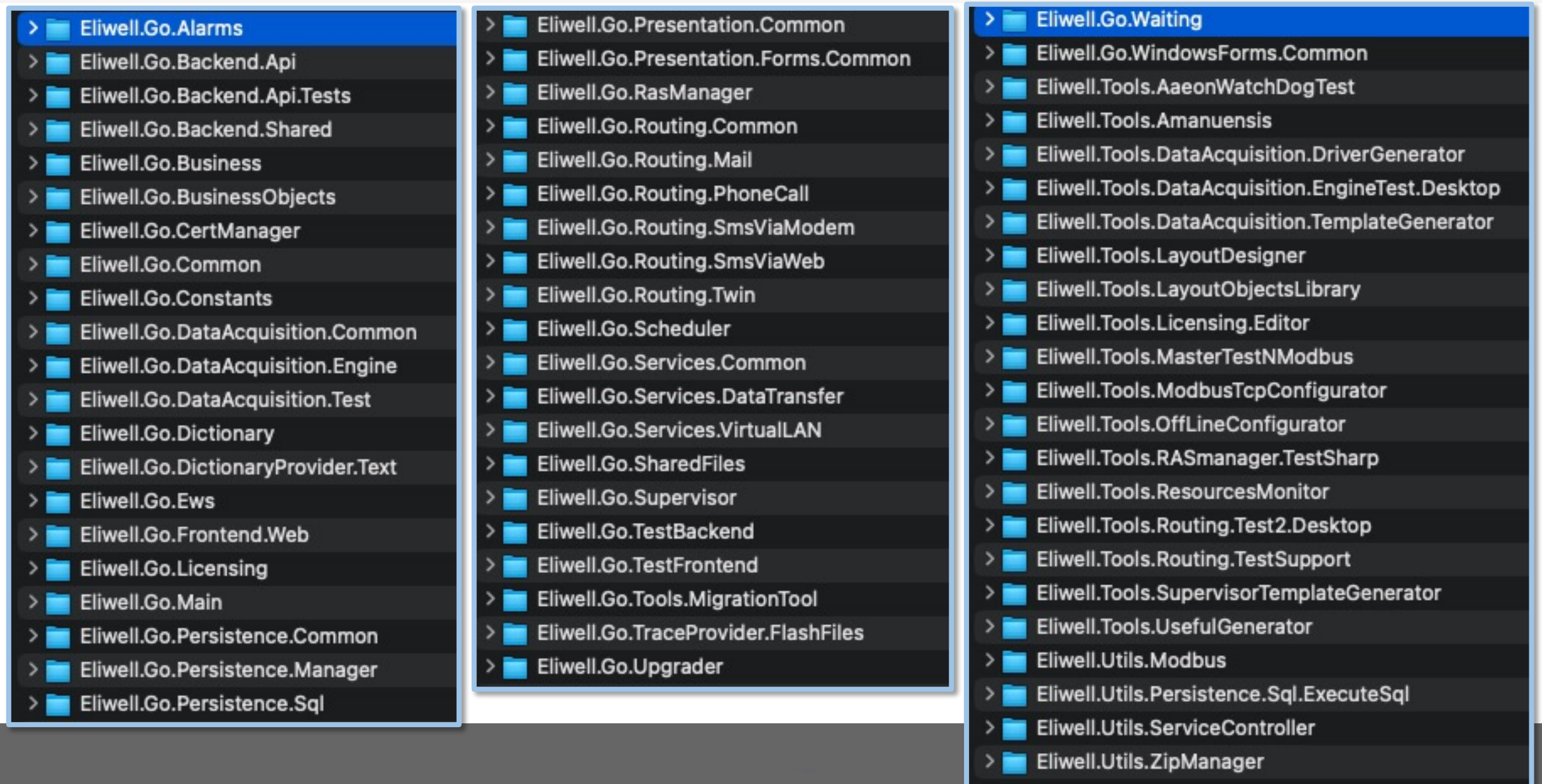
```
root@m172-si:~# dotnet --info

Host:
  Version:      8.0.4
  Architecture: arm
  Commit:       N/A
  RID:          linux-arm

.NET SDKs installed:
  No SDKs were found.

.NET runtimes installed:
  Microsoft.AspNetCore.App 8.0.4 [/home/root/.dotnet/shared/Microsoft.AspNetCore.App]
  Microsoft.NETCore.App 8.0.4 [/home/root/.dotnet/shared/Microsoft.NETCore.App]

Other architectures found:
  None

Environment variables:
  DOTNET_ROOT       [/home/root/.dotnet]
```

It works !

```
root@m172-si:/etc/labs# ./LinuxTest2
05:59:19.053 - Hello, World!
05:59:21.079 - Start task ...
05:59:21.081 - Connection to Modbus ...
05:59:21.129 - Connected to Modbus
05:59:21.130 - Reading from Modbus ...
05:59:21.277 - Read from Modbus ...
05:59:21.414 - Read 4109 - val: 219
05:59:21.415 - Read 4110 - val: 0
05:59:21.420 -
05:59:21.431 - Reading from Modbus ...
05:59:21.504 - Read from Modbus ...
05:59:21.504 - Read 4109 - val: 219
05:59:21.504 - Read 4110 - val: 0
```

```
DumpMessage($"Connection to Modbus ... ");
client.Connect("/dev/ttyS1", ModbusEndianness.BigEndian);
DumpMessage($"Connected to Modbus");

while(true)
{
    DumpMessage($"Reading from Modbus ... ");
    var val1 = client.ReadHoldingRegisters<Int16>(1, 4109, 1)[0];
    var val2 = client.ReadHoldingRegisters<Int16>(1, 4110, 1)[0];
    DumpMessage($"Read from Modbus ... ");

    DumpMessage($"Read 4109 - val: {val1}");
    DumpMessage($"Read 4110 - val: {val2}");
```

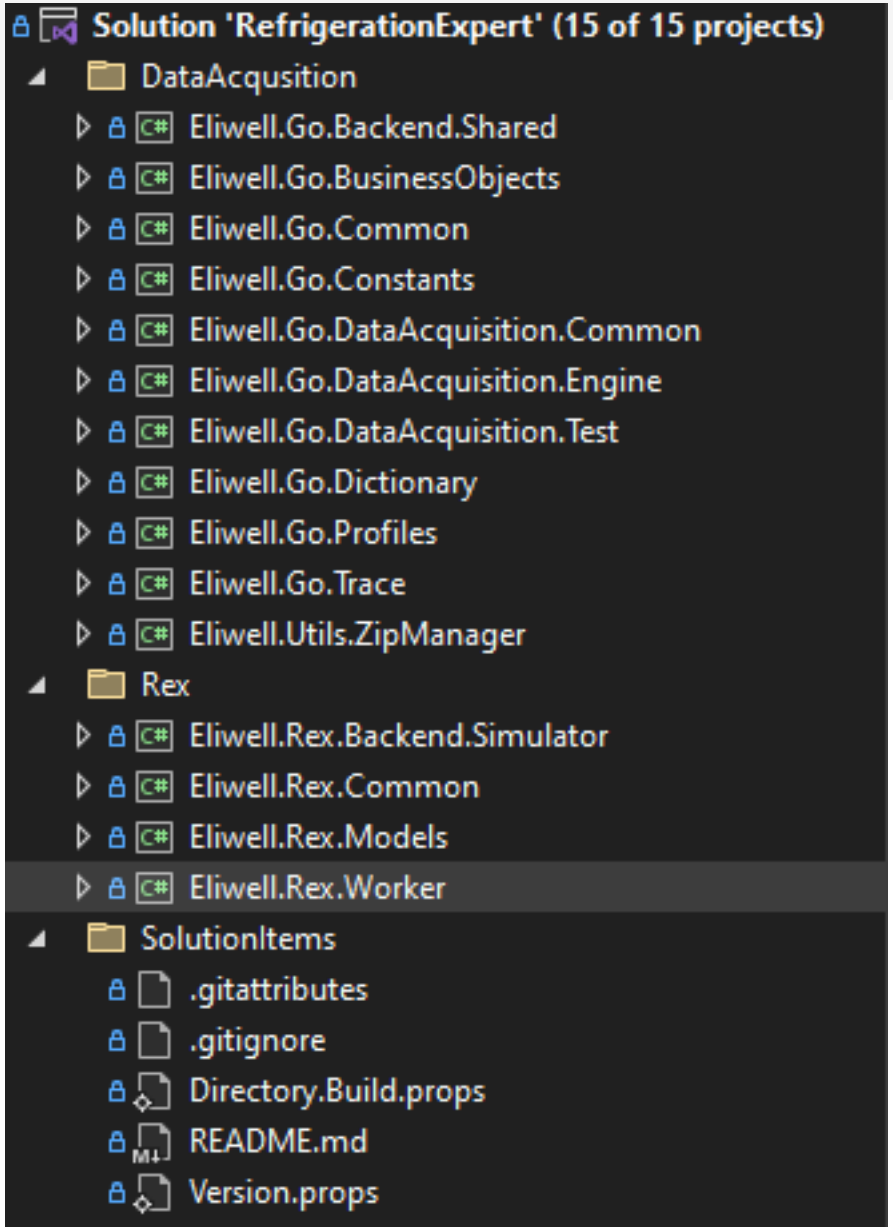
Code Migration



Code Migration

- Disacoppiamento sistemi tramite Nats

- Multipiattaforma Completo
- (Linux ARM32/ Win x86 / Win x64)



Code Migration

```
//Array.Copy(hookFuncBytes, inBuffer, hookFuncBytes.Length);
//Array.Copy(hostPtrBytes, 0, inBuffer, hookFuncBytes.Length, hostPtrBytes.Length);

Buffer.BlockCopy(hookFuncBytes, 0, inBuffer, 0, hookFuncBytes.Length);
Buffer.BlockCopy(hostPtrBytes, 0, inBuffer, hookFuncBytes.Length, hostPtrBytes.Length);
```

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
1 reference | Mirco Vanini, 359 days ago | 1 author, 1 change
private unsafe Int32[] CopyByteToInt32Array(byte [] source, int[] target, int length)
{
    fixed (Byte* pSource = source)
    {
        fixed (Int32* pTarget = target)
        {
            Byte* pSourceTmp = pSource;
            Int32* pTargetTmp = pTarget;

            //for (int i = 0; i < length; i++)
            while (length-- > 0)
            {
                *pTargetTmp = *pSourceTmp;
                pTargetTmp++;
                pSourceTmp++;
            }
        }
    }

    return target;
}
```

.NET on target

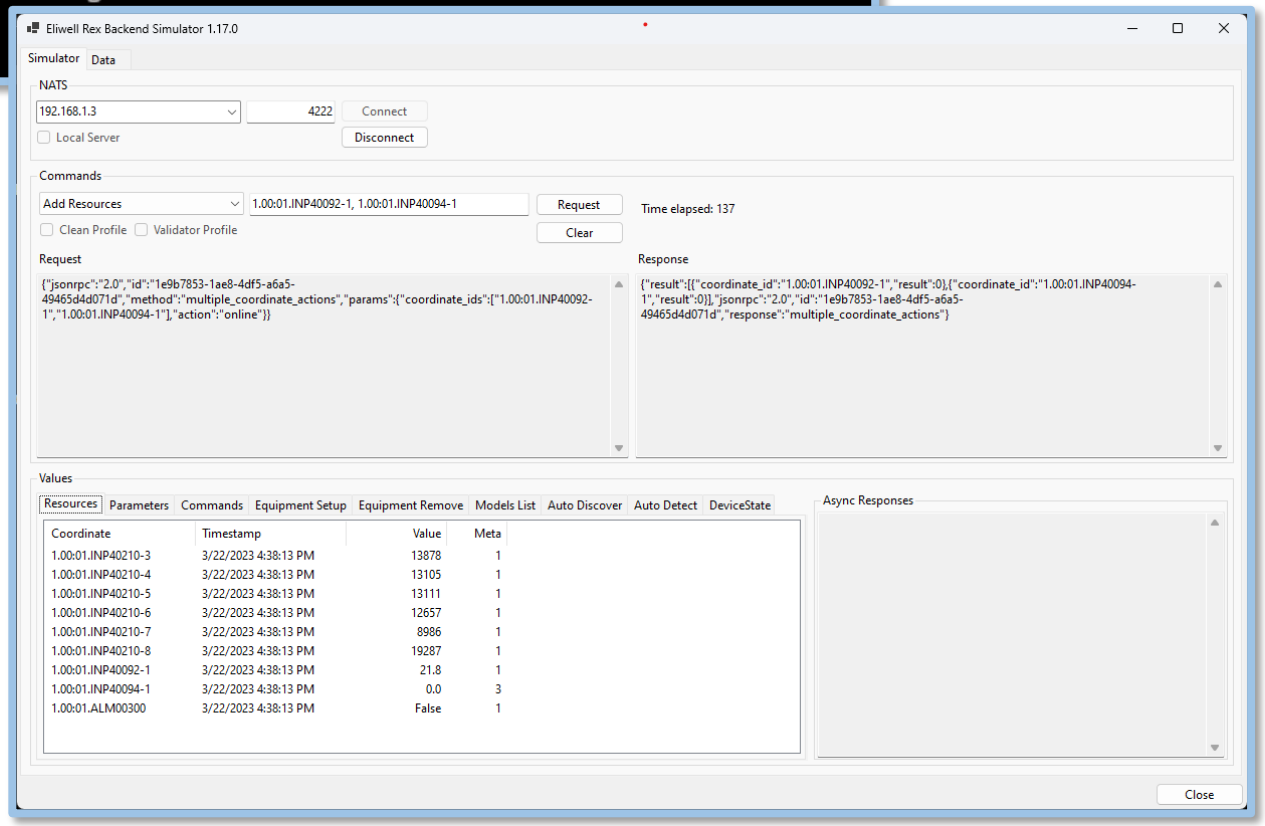
- Globalizzazione .NET e ICU

```
sudo apt install libicu-dev
sudo apt-get update && sudo apt-get install -qqq libicu63
export DOTNET_SYSTEM_GLOBALIZATION_INVARIANT=1
```

.NET Core Globalization Invariant Mode

Stato attuale

```
root@m172-si:/etc/datest# ./Eliwell.Rex.Worker
2023-03-22 17:21:44.614 | INF | Starting up Eliwell.Rex.Worker - 1.17.0 |
2023-03-22 17:21:54.629 | INF | >>> ExecuteAsync | Eliwell.Rex.Worker.Worker
2023-03-22 17:21:55.064 | INF | >>> NatsManager started on 0.0.0.0:4222 | Eliwell.Rex.Worker.NatsManager
2023-03-22 17:22:20.825 | INF | Native (/dev/ttyS1) - MicronetAndModbus - Port settings: [Modbus: 9600 8 E 1], [Micronet: 9600]
2023-03-22 17:22:21.139 | INF | >>> DataAcquisitionManager started! | Eliwell.Rex.Worker.DataAcquisitionManager
2023-03-22 17:22:21.221 | INF | Application started. Press Ctrl+C to shut down. | Microsoft.Hosting.Lifetime
2023-03-22 17:22:21.231 | INF | Hosting environment: Production | Microsoft.Hosting.Lifetime
2023-03-22 17:22:21.240 | INF | Content root path: /etc/datest | Microsoft.Hosting.Lifetime
2023-03-22 17:28:43.611 | INF | Added resources: 6 - commands: 0 @ 1.00:01 |
2023-03-22 17:28:48.506 | INF | Added resources: 6 - commands: 0 @ 1.00:01 |
```

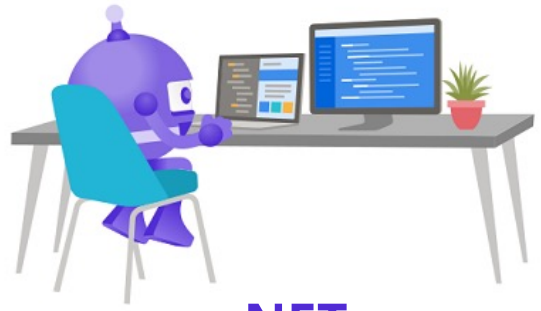


DEMO

Conclusioni

- Porting .NET Core sul target
- Riutilizzo del 100% del codice del Data Acqusition
- Riutilizzo del 100% dei drivers sviluppati per i prodotti precedenti
- Multipiattaforma
  - Linux ARM32
  - Win x86
  - Win x64
- Disaccoppiamento tra i livelli tramite Service Bus
- Piattaforma pronta per le evolutive verso altri HW/SW del gruppo

Grazie !



.NET
your platform for building anything