



# Rinnovo ed evoluzioni di un prodotto

Trasformare il codice del passato in soluzioni future

Mirco Vanini – Simone Tolotti

Un ringraziamento a...



**<packt>**

# 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



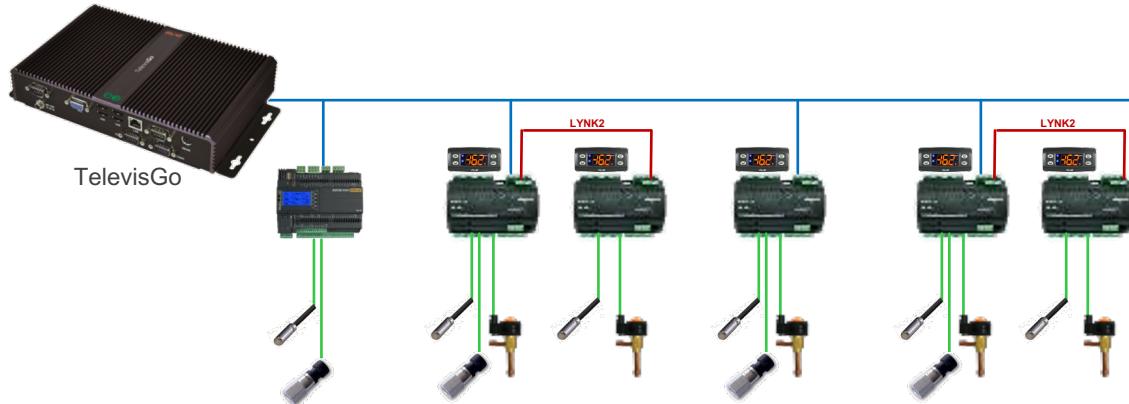
@MircoVanini  
[www.proxsoft.it](http://www.proxsoft.it)  
<https://www.linkedin.com/in/proxsoft>

<https://github.com/simonetolotti>  
[simone@tolotti.it](mailto: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...



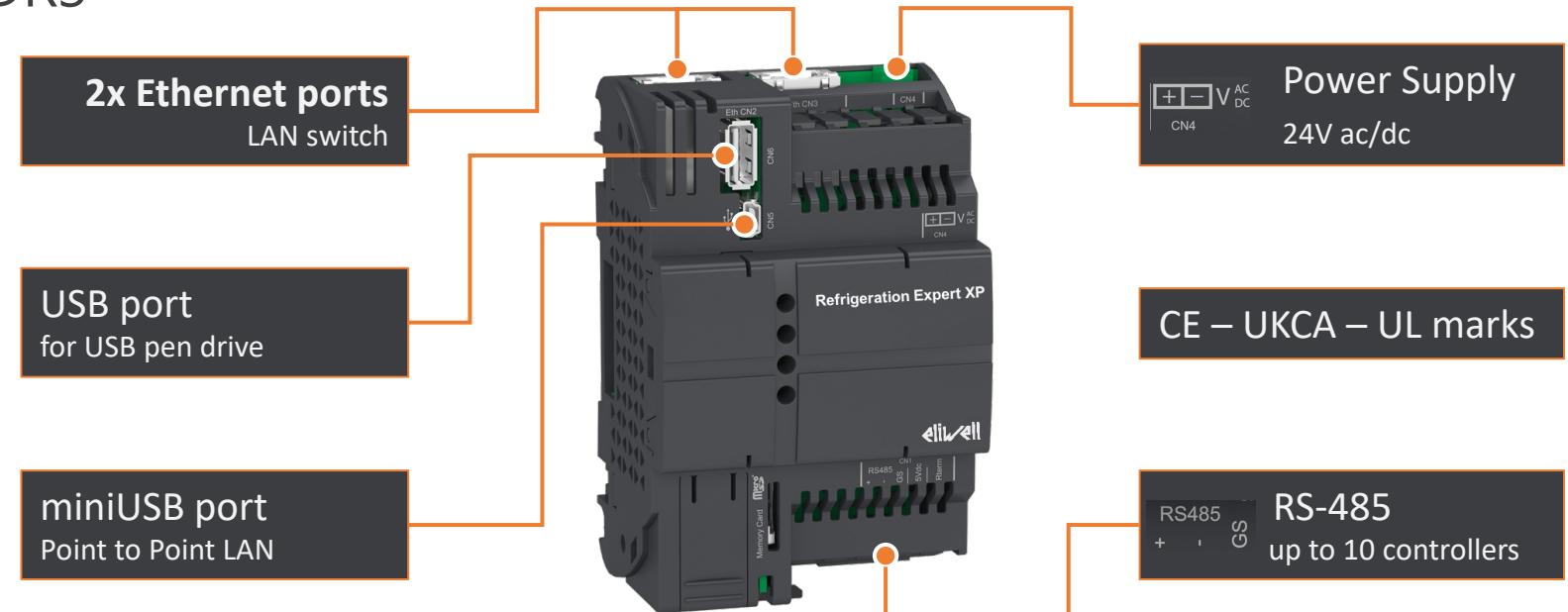
- Dopo 15 anni, i limiti:
  - Architettura PC → HW soggetto ad obsolescenza
  - Soluzione competitiva per supermarket meno food retail
  - Dimensioni e consumi
  - UI da svecchiare
  - Licenza di Windows
  - Gestione macchina Windows
  - Cybersecurity



# Cambio Tecnologico 1/2



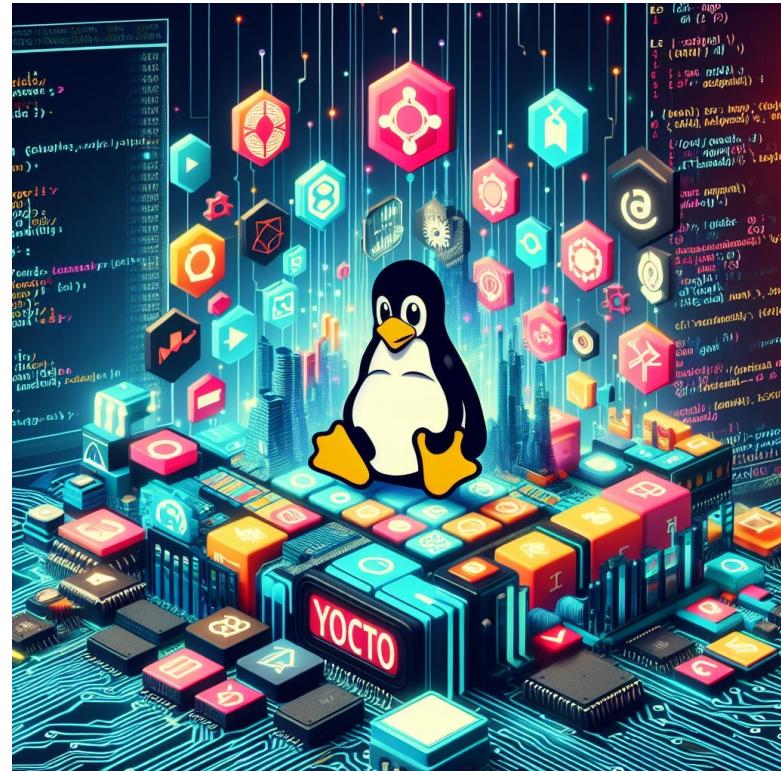
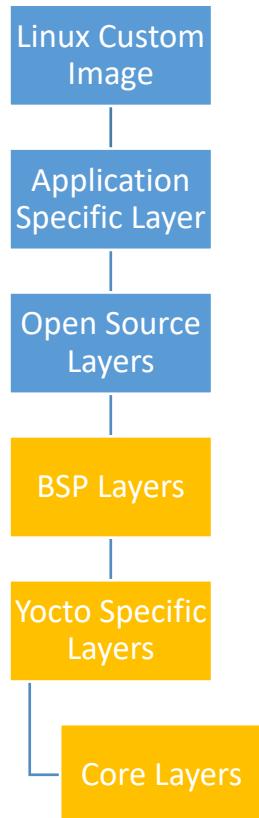
- HW
  - SoC: dual Cortex A7 (ARMv7) @ 500MHz
  - 1024 MB RAM DDR3
  - 1024 MB Nand



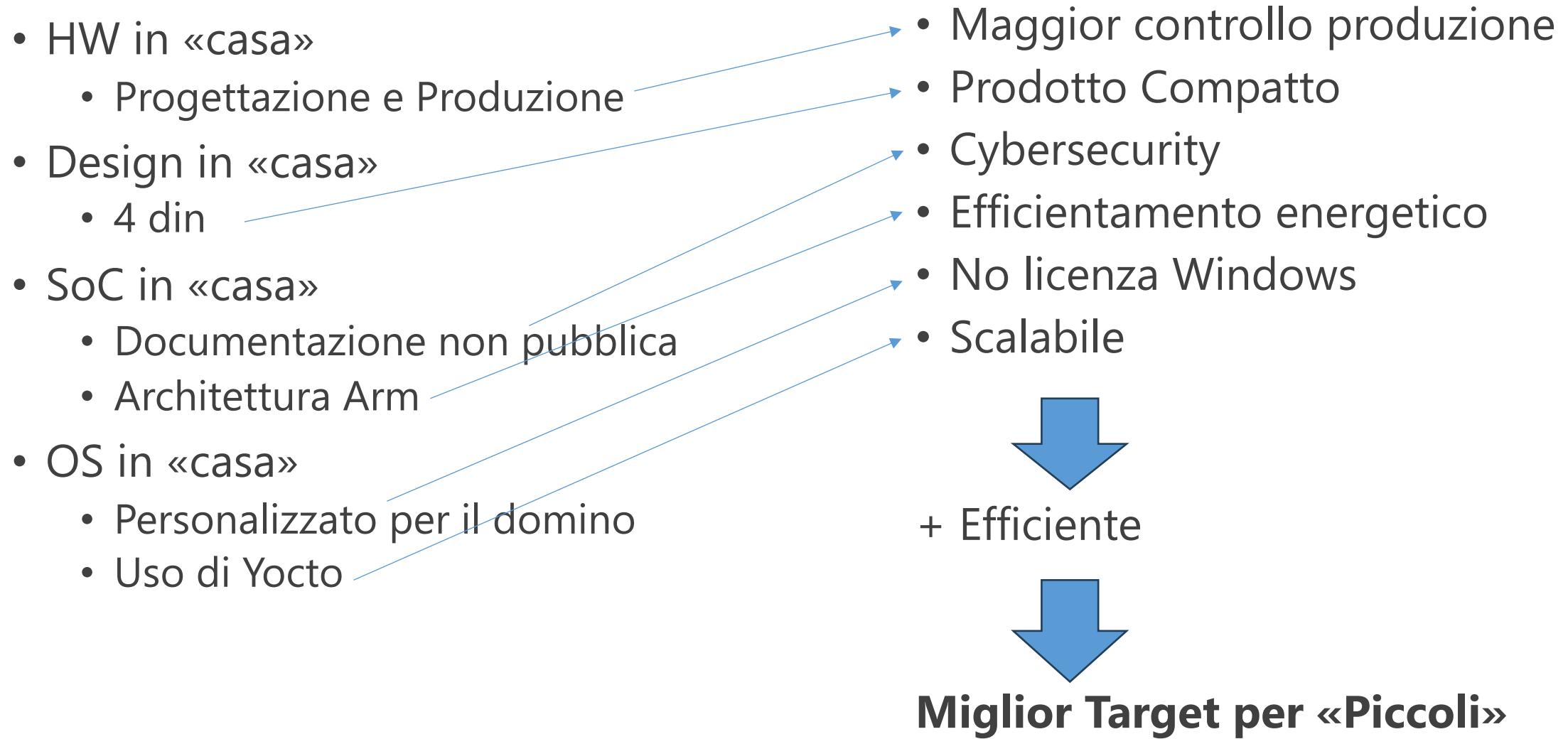
# Cambio Tecnologico



- OS
  - Yocto Linux



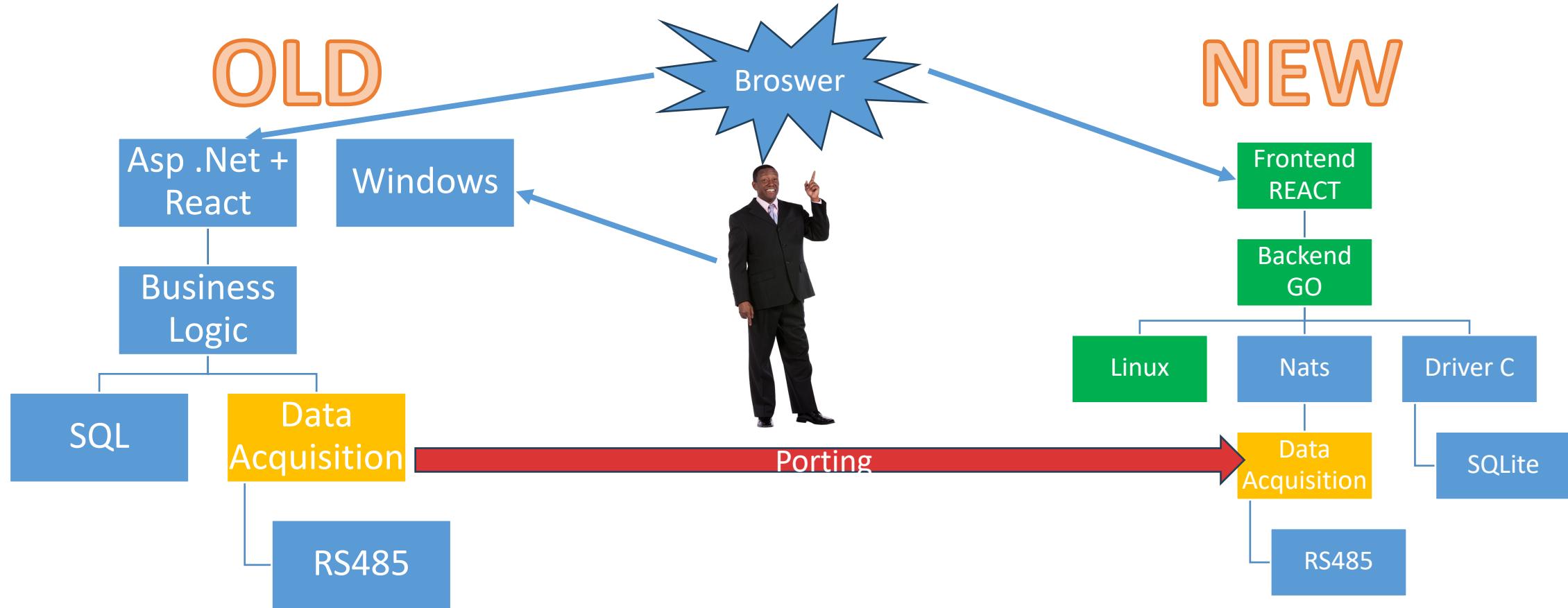
# Vantaggi nuova architettura



# E il Software ??? 3 Obiettivi:



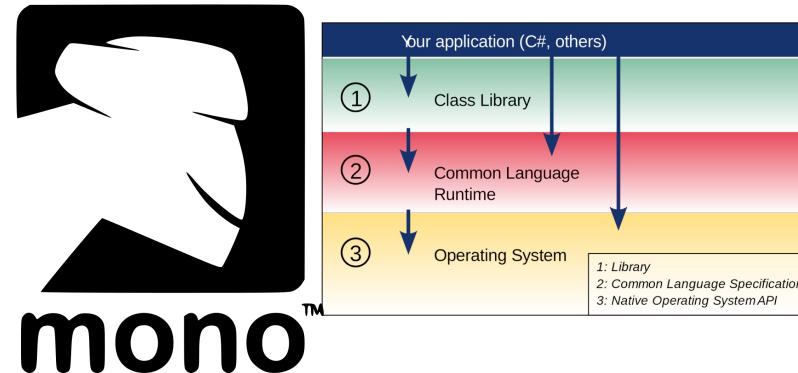
- Riciclare tutti i driver per la comunicazione di centinaia di strumenti
- Riciclare l'Expertise del team Webapp sulla stessa piattaforma Linux
- Utente per alcune opzioni amministrative, non dovrà più accedere al OS ma solo tramite la Webapp



# Porting codice Data Acquisition



- Mono Project
  - Esiste dal 2004, ex Mono hanno fondato Xamarin\*
  - Esegue applicazioni .Net Framework multipiattaforma
  - [Install Mono on Linux](#)
  - [ARM | Mono](#)
- .NET Core 6.0 LTS
  - 13 December 2022
  - Finalmente Multipiattaforma
    - 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

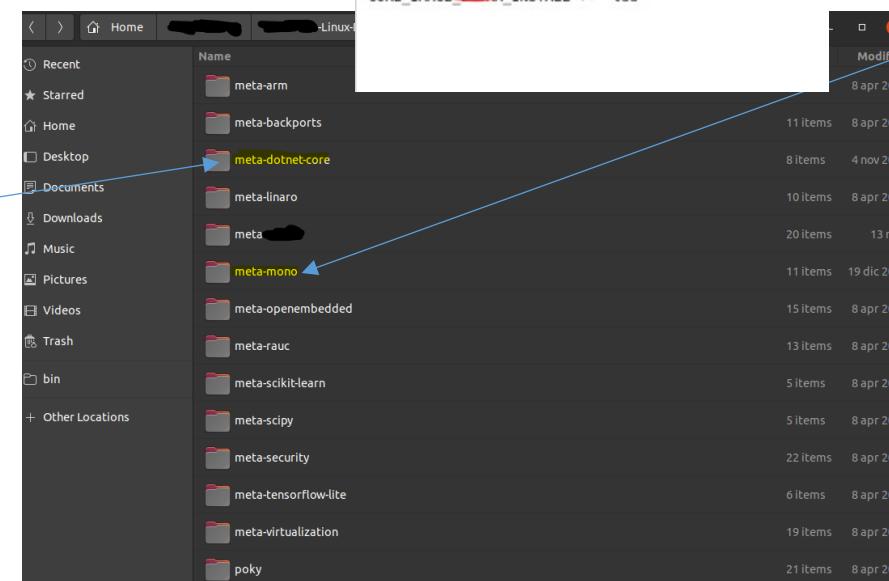


- Aggiunta Meta-layers:

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

- Aggiunta Ricette

The screenshot shows the GitHub repository page for RDunkley/meta-dotnet-core. It displays the README file which includes instructions for BitBake recipes to use pre-built binaries provided by Microsoft for .Net Core and its remote debugger. A pull request is visible in the list.



A screenshot of a diff viewer showing a patch file. The changes include adding 'dotnet' and 'mono' to the 'IMAGE\_INSTALL\_append' variable and adding 'mono-helloworld' to the 'CORE\_IMAGE\_EXTRA\_INSTALL' variable. A red circle highlights the first two additions.

The screenshot shows the OpenEmbedded Layer Index page for the 'meta-mono' layer. It provides details about the layer, including its dependencies (openembedded-core) and a list of recipes. A blue arrow points from the 'meta-mono' section in the middle of the page to the 'meta-mono' entry in the sidebar.

# Hello world, «Mono»



The screenshot shows the Visual Studio IDE interface. On the left, the 'Properties' window is open for 'ConsoleApp1'. The 'Application' tab is selected, showing the assembly name 'ConsoleApp1', the target framework set to '.NET Framework 4.8' (highlighted with a red circle), and the output type as 'Applicazione console'. The code editor window below shows the 'Program.cs' file with the following code:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp1
8 {
9     internal class Program
10    {
11        static void Main(string[] args)
12        {
13            Console.WriteLine("Hello, World! from Eliwell");
14        }
15    }
16
17 }
```

A red arrow points from the highlighted line in the properties window to the corresponding line in the code editor. Another red arrow points from the highlighted line in the code editor to the terminal window on the right.

The terminal window titled 'COM6 - PuTTY' shows the output of running the application: 'Hello, World! from Eliwell'.

# 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)

```
COM6 - PuTTY
Time Last: 35 Time Max: 66
Name: y
Modbus Address: 1
Variable Address: 4110
Sent Request
Modbus Value: 0
Modbus Status: Online

Communications: 2522 Errors: 0
Time Last: 32 Time Max: 66
Name: x
Modbus Address: 1
Variable Address: 4109
Sent Request
Modbus Value: 261
Modbus Status: Online

Communications: 2523 Errors: 0
Time Last: 30 Time Max: 66
Name: y
Modbus Address: 1
Variable Address: 4110
Sent Request
```

```
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 Status: " + 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));

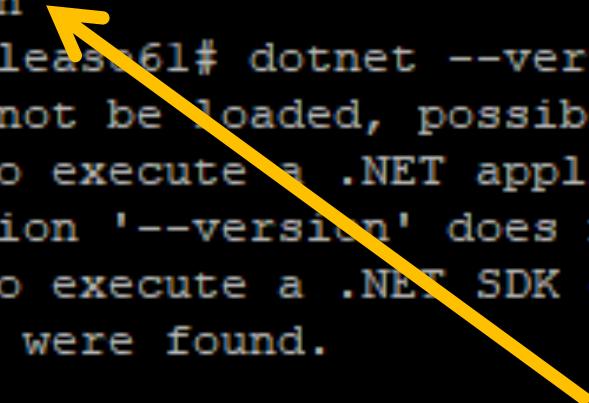
    Console.WriteLine(jsonToRedis);
    device.Cycle = 0;
}
else
    Console.WriteLine("Error Sent Request");
```

# Hello world, <.net Core>



```
root@t-air:/etc/Release61# dotnet LinuxTest3.dll
audit: type=1701 audit(1676892213.000:66): auid=4294967295 uid=0
Illegal instruction
root@t-air:/etc/Release61# dotnet --version
The command could not be loaded, possibly because:
 * You intended to execute a .NET application:
   The application '--version' does not exist.
 * You intended to execute a .NET SDK command:
   No .NET SDKs were found.

Download a .NET SDK:
https://aka.ms/dotnet-download
```



- 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 | • llDb   |
| • llvm                | • libicu-dev   |
| • lld                 | • liblttng-ust-dev   |
| • clang               | • libssl-dev   |
| • build-essential     | • libkrb5-dev  |
| • python-is-python3   | • zlib1g-dev   |
| • Curl                | • ninja-build (optional, enables building native code<br>with ninja instead of make) |
| • git                 |  |



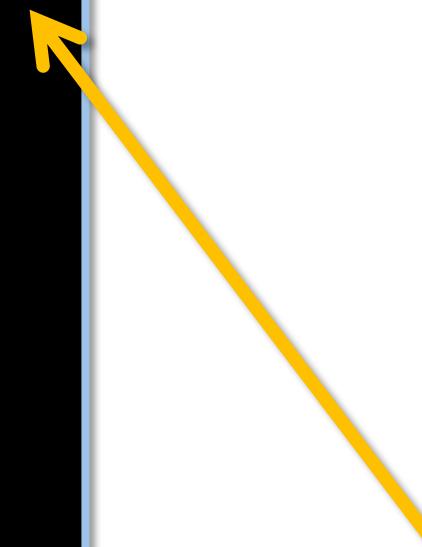
[How do I install the latest version of cmake from the command line?](#)

# 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.*)

```
[17:17:57] parallels@parallels-Parallels-Virtual-Platform ~ $ cd ~/Code/DotNet/runtime-8.0.4/
[17:17:45] parallels@parallels-Parallels-Virtual-Platform ~ $ ./eng/common/cross/build-rootfs.sh arm32
[sudo] password for parallels:
I: Retrieving InRelease
I: Checking Release signature
I: Valid Release signature (key id 790BC7277767219C42C86F933B4FE6ACC0B21F32)
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
I: Checking component main on http://ports.ubuntu.com...
I: Retrieving adduser 3.113+nmu3ubuntu4
I: Validating adduser 3.113+nmu3ubuntu4
I: Retrieving apt 1.2.10ubuntu1
I: Validating apt 1.2.10ubuntu1
I: Retrieving base-files 9.4ubuntu4
I: Validating base-files 9.4ubuntu4
I: Retrieving base-passwd 3.5.39
I: Validating base-passwd 3.5.39
I: Retrieving bash 4.3-14ubuntu1
I: Validating bash 4.3-14ubuntu1
I: Retrieving bsdutils 1:2.27.1-6ubuntu3
I: Validating bsdutils 1:2.27.1-6ubuntu3
I: Retrieving coreutils 8.25-2ubuntu2
I: Validating coreutils 8.25-2ubuntu2
I: Retrieving dash 0.5.8-2.1ubuntu2
I: Validating dash 0.5.8-2.1ubuntu2
I: Retrieving debconf 1.5.58ubuntu1
I: Validating debconf 1.5.58ubuntu1
I: Retrieving debianutils 4.7
```



# 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.

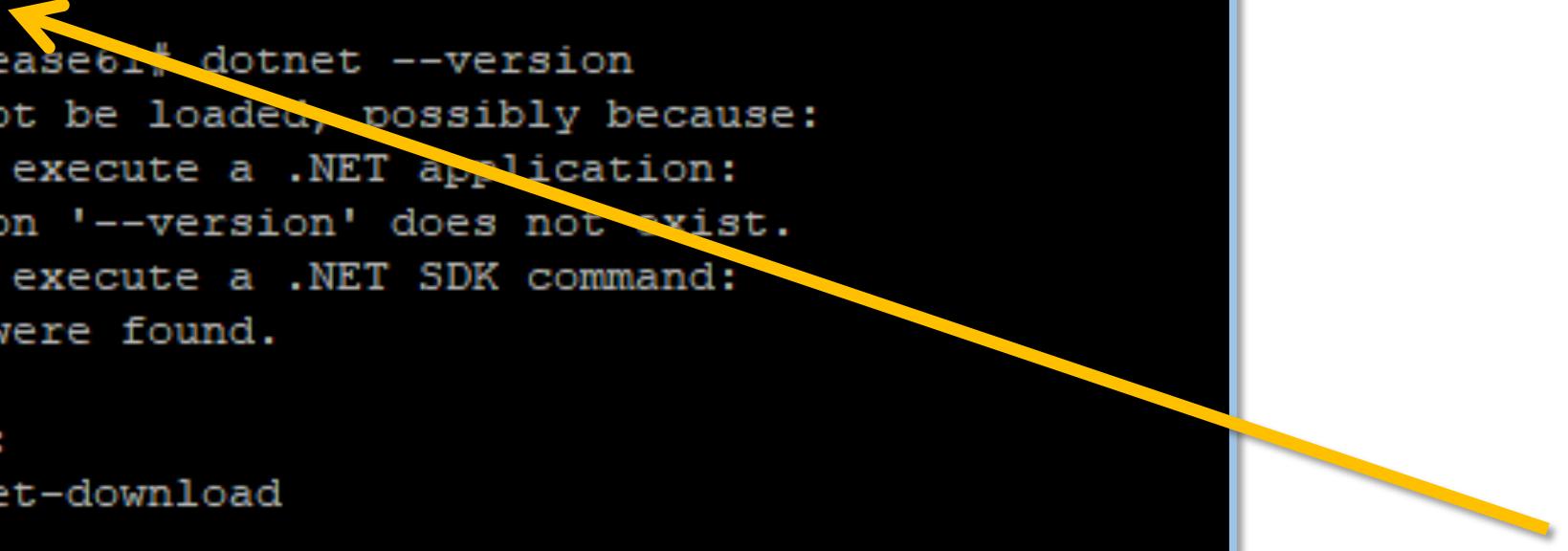
Bit 1 corresponds to 16 64-bit FPU registers.

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"
```

```
root@t-air:/etc/Release61# dotnet LinuxTest3.dll  
audit: type=1701 audit(1676892213.000:66): auid=4294967295 uid=0  
Illegal instruction   
root@t-air:/etc/Release61# dotnet --version  
The command could not be loaded, possibly because:  
* You intended to execute a .NET application:  
    The application '--version' does not exist.  
* You intended to execute a .NET SDK command:  
    No .NET SDKs were found.  
  
Download a .NET SDK:  
https://aka.ms/dotnet-download
```

# Build dotnet runtime



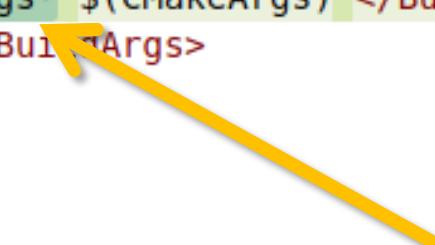
- Unable to cross-compile CoreCLR for Linux ARM vfpv4-d16

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



```
EXECUTING make install -j 4
[ 1%] Building C object System.IO.Ports.Native/CMakeFiles/System.IO.Ports.Native-Static.dir/pal_termio.o
[ 2%] Generating exports file /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/obj/native/net8.0/libSystem.IO.Ports.Native-Static.dll
[ 2%] Building C object System.IO.Ports.Native/CMakeFiles/System.IO.Ports.Native.dir/pal_termios.c.o
[ 2%] Building C object System.IO.Compression.Native/CMakeFiles/System.IO.Compression.Native-Static.dir/pal_compression.o
[ 2%] Built target System.IO.Compression.Native_exports
[ 3%] Building C object System.IO.Compression.Native/CMakeFiles/System.IO.Compression.Native-Static.dir/pal_compression.o
[ 3%] Building C object System.IO.Compression.Native/CMakeFiles/System.IO.Compression.Native-Static.dir/pal_compression.o
[ 4%] Microsoft.Extensions.Hosting.WindowsServices -> /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/bin/WindowsServices.dll
[ 5%] System.Threading.Thread.WebAssembly.Threading -> /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/bin/System.Threading.Thread.WebAssembly.Threading.dll
[ 6%] System.Threading.ThreadPool.WebAssembly.Threading -> /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/bin/System.Threading.ThreadPool.WebAssembly.Threading.dll
[ 7%] System.Diagnostics.Tracing.WebAssembly.PerfTracing -> /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/bin/System.Diagnostics.Tracing.WebAssembly.PerfTracing.dll
[ 7%] System.DirectoryServices.Protocols -> /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/bin/System.DirectoryServices.Protocols.dll
[ 8%] System.Data.Odbc -> /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/bin/System.Data.Odbc/ref/Release/net8.0/System.Data.Odbc.dll
[ 8%] System.Data.Odbc -> /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/bin/System.Data.Odbc/Release/net8.0/System.Data.Odbc.dll
[ 10%] System.IO.Ports -> /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/bin/System.IO.Ports/ref/Release/net8.0/System.IO.Ports.dll
[ 10%] System.IO.Ports -> /home/parallels/Code/DotNet/runtime-8.0.4/artifacts/bin/System.IO.Ports/Release/net8.0/System.IO.Ports.dll
[ 10%] ApiCompat -> Comparing net8.0 reference assemblies against .NETStandard 2.x and .NETCoreApp 8.0.0...
[ 10%] oob -> Trimming linux-arm out-of-band assemblies with ILLinker...
```

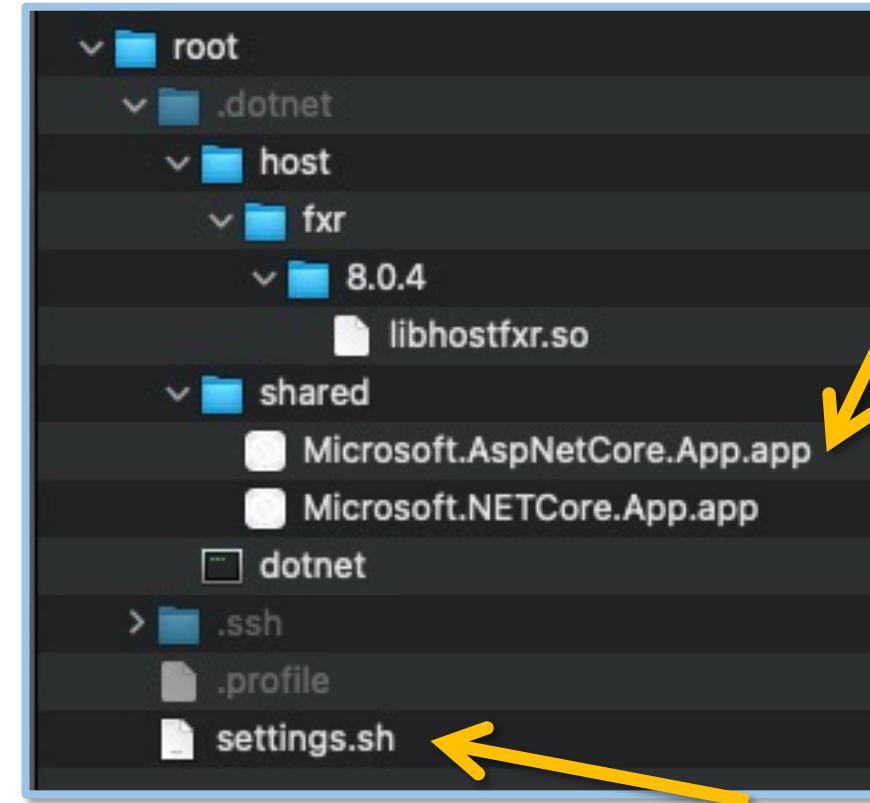
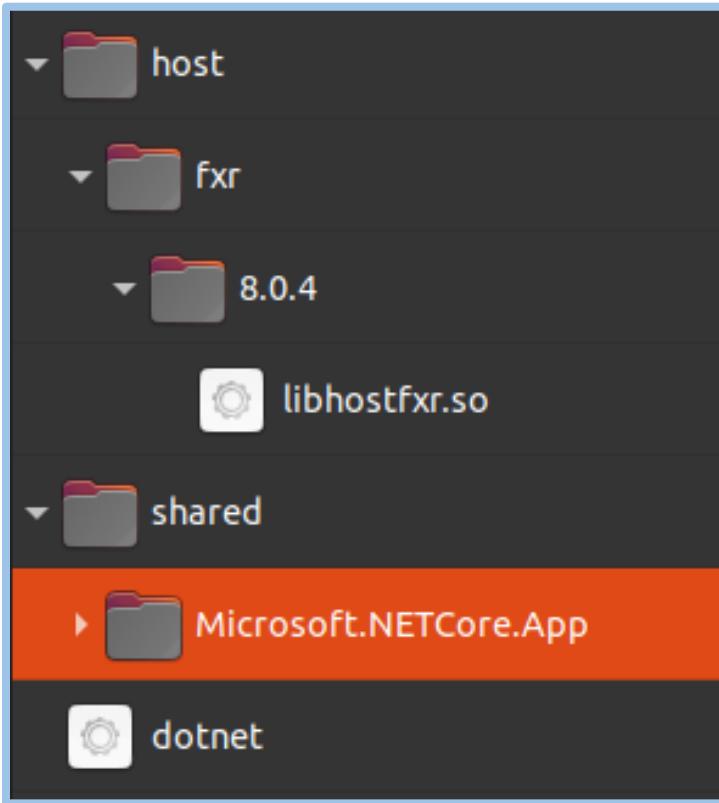
Build succeeded.  
0 Warning(s)  
0 Error(s)



# 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



- > **Eliwell.Go.Altarms**
- > Eliwell.Go.Backend.Api
- > Eliwell.Go.Backend.Api.Tests
- > Eliwell.Go.Backend.Shared
- > Eliwell.Go.Business
- > Eliwell.Go.BusinessObjects
- > Eliwell.Go.CertManager
- > Eliwell.Go.Common
- > Eliwell.Go.Constants
- > Eliwell.Go.DataAcquisition.Common
- > Eliwell.Go.DataAcquisition.Engine
- > Eliwell.Go.DataAcquisition.Test
- > Eliwell.Go.Dictionary
- > Eliwell.Go.DictionaryProvider.Text
- > Eliwell.Go.Ews
- > Eliwell.Go.Frontend.Web
- > Eliwell.Go.Licensing
- > Eliwell.Go.Main
- > Eliwell.Go.Persistence.Common
- > Eliwell.Go.Persistence.Manager
- > Eliwell.Go.Persistence.Sql

- > Eliwell.Go.Presentation.Common
- > Eliwell.Go.Presentation.Forms.Common
- > Eliwell.Go.RasManager
- > Eliwell.Go.Routing.Common
- > Eliwell.Go.Routing.Mail
- > Eliwell.Go.Routing.PhoneCall
- > Eliwell.Go.Routing.SmsViaModem
- > Eliwell.Go.Routing.SmsViaWeb
- > Eliwell.Go.Routing.Twin
- > Eliwell.Go.Scheduler
- > Eliwell.Go.Services.Common
- > Eliwell.Go.Services.DataTransfer
- > Eliwell.Go.Services.VirtualLAN
- > Eliwell.Go.SharedFiles
- > Eliwell.Go.Supervisor
- > Eliwell.Go.TestBackend
- > Eliwell.Go.TestFrontend
- > Eliwell.Go.Tools.MigrationTool
- > Eliwell.Go.TraceProvider.FlashFiles
- > Eliwell.Go.Upgrader

- > **Eliwell.Go.Waiting**
- > Eliwell.Go.WindowsForms.Common
- > Eliwell.Tools.AaeonWatchDogTest
- > Eliwell.Tools.Amanuensis
- > Eliwell.Tools.DataAcquisition.DriverGenerator
- > Eliwell.Tools.DataAcquisition.EngineTest.Desktop
- > Eliwell.Tools.DataAcquisition.TemplateGenerator
- > Eliwell.Tools.LayoutDesigner
- > Eliwell.Tools.LayoutObjectsLibrary
- > Eliwell.Tools.Licensing.Editor
- > Eliwell.Tools.MasterTestNModbus
- > Eliwell.Tools.ModbusTcpConfigurator
- > Eliwell.Tools.OffLineConfigurator
- > Eliwell.Tools.RASmanager.TestSharp
- > Eliwell.Tools.ResourcesMonitor
- > Eliwell.Tools.Routing.Test2.Desktop
- > Eliwell.Tools.Routing.TestSupport
- > Eliwell.Tools.SupervisorTemplateGenerator
- > Eliwell.Tools.UsefulGenerator
- > Eliwell.Utils.Modbus
- > Eliwell.Utils.Persistence.Sql.ExecuteSql
- > Eliwell.Utils.ServiceController
- > Eliwell.Utils.ZipManager

# Code Migration



- Disacoppiamento sistemi tramite [Nats](#)
- Multipiattaforma Completo
- (Linux ARM32/ Win x86 / Win x64)

The screenshot shows the Visual Studio Solution Explorer with the following details:

- Solution:** 'RefrigerationExpert' (15 of 15 projects)
- Projects:**
  - DataAcquisition
  - Eliwell.Go.Backend.Shared
  - Eliwell.Go.BusinessObjects
  - Eliwell.Go.Common
  - Eliwell.Go.Constants
  - Eliwell.Go.DataAcquisition.Common
  - Eliwell.Go.DataAcquisition.Engine
  - Eliwell.Go.DataAcquisition.Test
  - Eliwell.Go.Dictionary
  - Eliwell.Go.Profiles
  - Eliwell.Go.Trace
  - Eliwell.Utils.ZipManager
- Folder:** Rex
  - Eliwell.Rex.Backend.Simulator
  - Eliwell.Rex.Common
  - Eliwell.Rex.Models
  - Eliwell.Rex.Worker** (selected)
- SolutionItems:**
  - .gitattributes
  - .gitignore
  - Directory.Build.props
  - README.md
  - Version.props

# 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;
}
```



- 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
--
```

Eliwell Rex Backend Simulator 1.17.0

Simulator Data

NATS

192.168.1.3 4222 Connect Disconnect Local Server

Commands

Add Resources 1.00:01.INP40092-1, 1.00:01.INP40094-1 Request Time elapsed: 137

Clean Profile Validator Profile Clear

Request

{"jsonrpc": "2.0", "id": "1e9b7853-1ae8-4df5-a6a5-49465d4d071d", "method": "multiple\_coordinate\_actions", "params": {"coordinate\_ids": ["1.00:01.INP40092-1", "1.00:01.INP40094-1"], "action": "online"}}

Response

{"result": [{"coordinate\_id": "1.00:01.INP40092-1", "result": 0}, {"coordinate\_id": "1.00:01.INP40094-1", "result": 0}], "jsonrpc": "2.0", "id": "1e9b7853-1ae8-4df5-a6a5-49465d4d071d", "response": "multiple\_coordinate\_actions"}

Values

Resources	Parameters	Commands	Equipment Setup	Equipment Remove	Models List	Auto Discover	Auto Detect	DeviceState
Coordinate	Timestamp		Value	Meta				
1.00:01.INP40210-3	3/22/2023 4:38:13 PM		13878	1				
1.00:01.INP40210-4	3/22/2023 4:38:13 PM		13105	1				
1.00:01.INP40210-5	3/22/2023 4:38:13 PM		13111	1				
1.00:01.INP40210-6	3/22/2023 4:38:13 PM		12657	1				
1.00:01.INP40210-7	3/22/2023 4:38:13 PM		8986	1				
1.00:01.INP40210-8	3/22/2023 4:38:13 PM		19287	1				
1.00:01.INP40092-1	3/22/2023 4:38:13 PM		21.8	1				
1.00:01.INP40094-1	3/22/2023 4:38:13 PM		0.0	3				
1.00:01.ALM00300	3/22/2023 4:38:13 PM		False	1				

Async Responses

Close



DEMO

# Conclusioni



- Porting .NET Core sul target
- Riutilizzo del 100% del codice del Data Acquisition
- Riutilizzo del 100% dei drivers sviluppati per i prodotti precedenti
- Multiplataforma
  - 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