An Introduction to the Microsoft .NET Micro Framework
PASCAL SPÖRRI – ENJOYS HACKING ON DEVICES CONNECTED TO THE INTERNET.

This article is both intended to give an overview over the .NET Micro Framework or NETMF in short and to give some insight into my internship at Oberon microsystems AG.

NETMF is a platform for resource constrained embedded devices with at least 256 KB of flash storage, 64 KB of RAM and a 32 bit CPU. It is intended for running on small embedded devices with low production costs (<$10 per device in high volumes)... It is ideal for both prototyping and end user devices. Programs are typically executed on top of a bytecode VM supporting both a subset of the .NET common intermediate language and a subset of the .NET standard library. This allows a developer to test programs on Windows using the .NET standard library and Visual Studio and then directly port the code to a NETMF device! An over-engineered hello world program would look like this:

```csharp
using System.Threading;
using Microsoft.SPOT.Hardware;
using Mountaineer.Netmf.Hardware;

namespace Blink
{
    public class Program
    {
        public static void Main()
        {
            while (true)
            {
                Leds.Red.Toggle();
                Thread.Sleep(500); // Sleep for 500ms
            }
        }
    }
    public static class Leds
    {
        public static OutputPort Red { get; private set; }
        public static OutputPort Green { get; private set; }
        public static OutputPort Blue { get; private set; }

        static Leds()
        {
            Red = new OutputPort(OnboardIO.LedRed, false);
            Green = new OutputPort(OnboardIO.LedGreen, false);
            Blue = new OutputPort(OnboardIO.LedBlue, false);
        }

        public static void Toggle(this OutputPort port)
        {
            port.Write(!port.Read());
        }
    }
}
```
NETMF was originally developed for devices like the Microsoft SPOT smart watch introduced in 2004. That’s also one of the reasons why a lot of the namespaces in NETMF contain the prefix “Microsoft.SPOT”. Since 2009, the source code is available under the Apache 2.0 license.

**Architecture**

- The framework is typically run directly on the hardware without any OS layer. To provide hardware independence, the framework is split into:
  - HAL - the hardware abstraction layer, which contains all device specific drivers written in C/C++.
  - PAL - the platform abstraction layer, which abstracts the HAL and provides rudimentary operating system services.
  - CLR - the common language runtime, which executes the CIL (common immediate language) instructions; i.e. runs the deployed C# application on the device. This layer also implements threads and provides debugging support.
  - System libraries - default libraries provided with the hardware – a small subset of the full .NET framework plus some extensions for accessing hardware over I2C, SPI, etc.
  - Managed application - application deployable onto the device.

**Writing code for NETMF**

NETMF is tightly integrated with Visual Studio: Code can be written, deployed and even debugged. Unfortunately, the software development and deployment support for Linux and OSX is unclear. Integration into MonoDevelop should be doable since the deployment tools and the Visual Studio integration are open source as well.

The fascination starts when one wants to customise the internals of NETMF. The complete source code, even including the CLR and the minimal .NET C# SDK for NETMF is open source. Thus, customisations for different hardware are generally also provided as source code.

One example of a hardware specific port is Oberon’s open source port for the widely deployed STM32F4 ARMv7 processor platform.

---

Mountaineer Ethernet Mainboard

[Diagram of NETMF architecture]
Native Interop

NETMF makes it easy to integrate native C/C++ driver level code directly into the framework. Tagging the desired function with a C# attribute will automatically generate C++ code, which when integrated into the firmware will be executed when the managed C# function is called on the device.

Several data types are supported:

<table>
<thead>
<tr>
<th>C# Type</th>
<th>C++ Type</th>
<th>C++ Array Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>System.Byte</td>
<td>UINT8, BYTE</td>
<td>CLR_RT_TypedArray_UINT8</td>
</tr>
<tr>
<td>System.UInt16</td>
<td>UINT16</td>
<td>CLR_RT_TypedArray_UINT16</td>
</tr>
<tr>
<td>System.UInt32</td>
<td>UINT32</td>
<td>CLR_RT_TypedArray_UINT32</td>
</tr>
<tr>
<td>System.UInt64</td>
<td>UINT64</td>
<td>CLR_RT_TypedArray_UINT64</td>
</tr>
<tr>
<td>System.SByte</td>
<td>INT8</td>
<td>CLR_RT_TypedArray_INT8</td>
</tr>
<tr>
<td>System.Int16</td>
<td>INT16</td>
<td>CLR_RT_TypedArray_INT16</td>
</tr>
<tr>
<td>System.Int32</td>
<td>INT32</td>
<td>CLR_RT_TypedArray_INT32</td>
</tr>
<tr>
<td>System.Int64</td>
<td>INT64</td>
<td>CLR_RT_TypedArray_INT64</td>
</tr>
<tr>
<td>System.Single</td>
<td>float</td>
<td>CLR_RT_TypedArray_float</td>
</tr>
<tr>
<td>System.Double</td>
<td>double</td>
<td>CLR_RT_TypedArray_double</td>
</tr>
<tr>
<td>System.String</td>
<td>LPCSTR</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>
Devices

NETMF devices come in various form factors and colours. The two most common kinds are Gadgeteer and Netduino.

- Netduino devices usually have a form factor which is Arduino compatible, allowing the use of Arduino shields. Devices can be ordered from
  2. and various resellers around the world.

- Gadgeteer devices have standardised miniature socket connectors that makes it easier to build systems without soldering. There's also a C# library that allows for an easy integration of the individual modules and a visual designer integrated into Visual Studio. Devices can be ordered from:
  3. GHI Electronics: [https://www.ghi-electronics.com/](https://www.ghi-electronics.com/)

It's also possible to install NETMF on various developer boards. The Mountaineer Group provides NETMF builds for various evaluation boards. The Mountaineer Group is an open source software/hardware cooperation based on NETMF, founded by Oberon microsystems AG in Zurich and CSA Engineering AG in Solothurn.

Internship

During my internship I ported a state-of-the-art crypto library complete with C# bindings to the Oberon microsystems NETMF port. I also helped integrate the NETMF update framework (MFUpdate) into said NETMF port. A new Mountaineer firmware release with my additions is also due for release in April.

What really impressed me is the conciseness and extensibility of the micro framework. Having used Arduino before it wouldn't have come to my mind to even run a webserver on an embedded device, let alone connect it to the Internet. After having ported the MFUpdate to our build we were able to install a firmware update to a remote device over the Internet using HTTP PUT.

It's also interesting to see that these devices can be configured as gateway devices between local device connected via Bluetooth Low Energy. And the gateway being directly connected to the Internet via mobile internet.
Conclusion

Before I started working at Oberon microsystems, I considered the term „Internet of Things“ as just the next fad from someone’s marketing department. Having administered countless servers I just couldn't make sense of the term, especially given the various marketing arguments. And seeing IP (Internet Protocol) problems didn't help. And I couldn't really understand why someone wanted to connect everything to the Internet, especially the fridge.

Over the last couple of months my position slowly started to shift in favour of the “Internet of Things” terminology. But with these low cost devices I see possibilities in the raw data these devices can provide. Now I’m especially interested in connecting our fridge to the Internet because it will give us all kinds of data. Is the temperature low enough for the beer to be chilled correctly? Does the fridge work or does it need to be replaced? I don’t want to necessarily walk to the fridge and find out. I’m more intrigued by getting the passive data from somewhere.

Interestingly, I see more problems relating to IP (intellectual property) than to TCP/IP, because a lot of use cases are already patented.

For a more detailed introduction into NETMF and the Internet of Things I recommend the book „Getting started with the Internet of Things“ by Cuno Pfister¹⁰...

Reference

[1] Link: http://www.mountaineer.org/netmf-for-stm32/

Footnote

Dr. Cuno Pfister is the CEO of Oberon microsystems AG and was my supervisor for the duration of the internship
Take on the challenge:
www.swisscom.ch/project365d

Move into the apartment of the future and take a big step towards your dream job. We offer you a room in a shared apartment with other roommates for 365 days, in addition to an exciting position in the innovation department of Swisscom.

Start date: October 1st, 2014.

Take on the challenge:
www.swisscom.ch/project365d