Jan Kučera (2015):
Managed End-User Development That Scales.
In International Reports on Socio-Informatics (IRSI),
Proceedings of the CHI 2015 - Workshop on End User Development in the Internet of Things Era
(Vol. 12, Iss. 2, pp. 21-26)

Managed End-User Development That Scales

Jan Kučera

Newcastle University, UK
j.kucera1@ncl.ac.uk

Abstract. This paper presents the current state of .NET Micro Framework, the Internet of Things technology developed and deployed before it was cool, and .NET Gadgeteer, a rapid prototyping platform built on top of it, which anyone from K-12 students to industrial researchers can use to design and develop their own things on the edge.

1 Introduction

As a software developer, I am disappointed how user interfaces are becoming less and less configurable and customizable, and I believe that taking this responsibility and capability away from users, we are not only losing their interest in technologies but also making them less credible and aware of how they work, eventually resulting in issues like reduced privacy and/or increased fear. These concerns are only going to increase with Internet of Things, and putting people back in control of devices they are surrounded with and allow them to understand how technology works is an effective way to rebut this trend. To achieve this goal, we need an end-user development environment that is both powerful and reliable for advanced users and easy to learn and fun to use for inexperienced users. I believe that .NET Micro Framework and .NET Gadgeteer frameworks coming from Microsoft Research, which are both open source, meet these criteria and allow for simple yet effective end-user development. I am happy to demonstrate my experience with these platforms to participants of the workshop.

2 .NET Micro Framework

The .NET Micro Framework is a Common Language Runtime running directly on hardware without need for an underlying operating system. It shipped commercially in 2004 in couple of consumer products like smartwatches or home appliances, and in 2006 it was introduced to general public as a development platform.

Currently, it is targeting devices of at least 64kB RAM and 256 kB FLASH memory, and several processors and development kits spanning from small one like FEZ Mini on Figure 1 to very powerful like Blackfin signal processors were introduced on the market by various vendors.

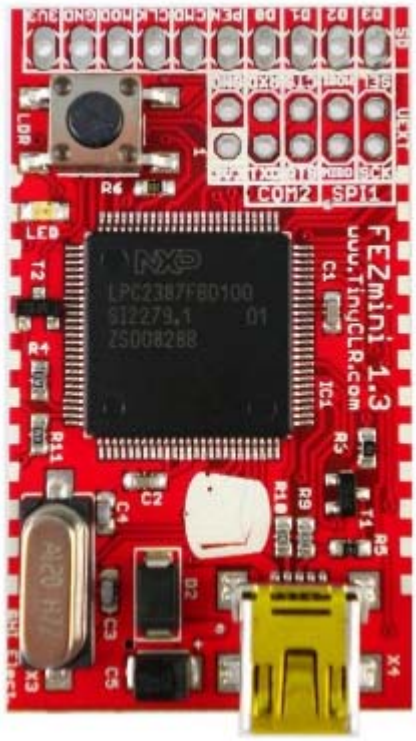

Fig. 1: FEZ Mini by GHI Electronics. One of the smallest development boards with an LPC2387 system on chip. 72 MHz, 96 kB RAM, 512 kB FLASH.

One of the core feature of .NET Micro Framework is the managed runtime environment, which makes development much simpler for end-users, since a garbage collector takes care about the memory and the hardware architecture is abstracted away. Both C# and Visual Basic.NET languages are supported, so that users that have no programming experience can have an easy starting point.

Compared to other hardware development platforms, the main advantage would be the full-featured Visual Studio IDE for live on-device debugging and advanced features like XML parser, Web Services for Devices or reflection, which enables

the unique scenario of devices supporting plug-ins and other code deployable at runtime.

However, the framework has also its weak points, such as suboptimal speed since the user applications are interpreted, inability to fulfill any real-time requirements or lack of USB host support. Remote firmware updates are currently limited. It would be interesting to compare framework's features and issues with other emerging end-user development platforms.

3 .NET Gadgeteer

Prototyping devices costs large amount of money, time and expertise. Microsoft Research solved this issue in 2011 by introducing the .NET Gadgeteer [1].

.NET Gadgeteer is built on top of the .NET Micro Framework and one of its goals was rapid hardware prototyping. This is accomplished by standardizing the hardware form factors of individual components like sensors and actuators and providing managed drivers conforming to common interfaces. For example, users can then grab any temperature sensor module, display module and WiFi module, connect them together without any soldering and develop and deploy a simple monitoring thing with notification capabilities in couple lines of code. One such commercially available kit with several sensors and modules is shown on Figure 2.

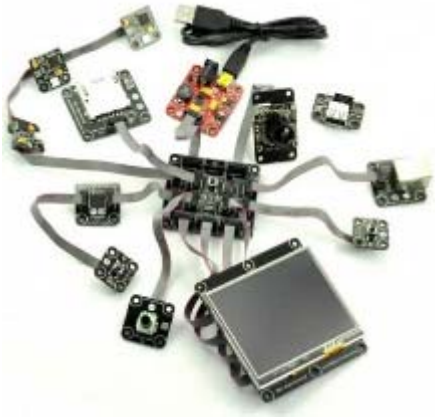

Fig. 2: FEZ Spider Kit by GHI Electronics. An example of modular .NET Gadgeteer hardware.

Apart from rapid prototyping, the .NET Gadgeteer comes with a very appealing and simple to use visual designer. Not only it generates the required initialization code for users, but it can also automatically suggest how to connect modules to the mainboard, so that technical details of the hardware involved does not need to be known (see Figure 3).

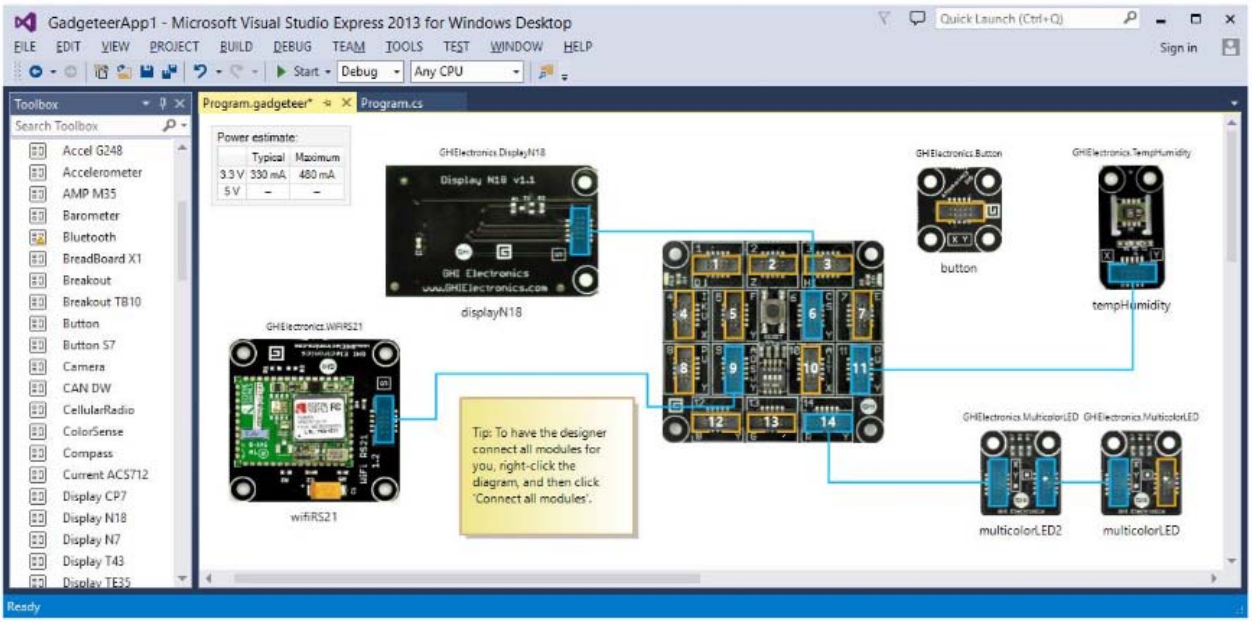

Fig. 3: .NET Gadgeteer designer in Microsoft Visual Studio

The .NET Gadgeteer platform therefore became an interesting way how to learn programming for students on primary and secondary schools as well as universities. A pilot studies had been run in the United States and the United Kingdom, which showed interest and enthusiasm among students [2], and the platform can be used as a part of delivering the UK's national curriculum for computer science [3]. I believe that students without any previous experience in hardware or software development represent an important target group of end-user development systems in home and hobbyists deployments of Internet of Things. I have participated on developing the .NET Gadgeteer core and helped to run several workshops with both students and researchers in various countries, gathering feedback from community of end-users related to the topics of this workshop.

4 Finalizing things

While there exists a considerable amount of platforms for end-user development of hardware and software, the options of designing the final product housing and appearance seem to be still limited. The .NET Gadgeteer as a prototyping platform is often connected with 3D printing or laser cutting [1], but these are still quite expensive and inaccessible technologies to most households and usually require specialized expertise to be used. I would like to have the opportunity to discuss alternative approaches or suggestions for introducing the same amount of modularity, feasibility and ease of use into finalizing the things of Internet of Things.

5 Conclusion

In this paper I have presented free and open source tools that users of various age and experience can use to build their own devices. Although these tools offer a significant step towards user friendliness and straightforward development process, there are still issues and scenarios where they are not a perfect fit. I have also questioned the existence of affordable and accessible options and tools for physical design of devices. I hope that participation in the workshop would give me incentives about how to further improve the platforms I have experience with as well as share their possibilities with others interested in the field.

6 References

[1] N. Villar, J. Scott, S. Hodges, K. Hammil a C. Miller, „.NET Gadgeteer: A Platform for Custom Devices," v Proceedings of Pervasive 2012, Lectures in Computer Science, 2012.

[2] S. Hodges, J. Scott, S. Sentance, C. Miller, N. Villar, S. Schwiderski-Grosche, K. Hammil a S. Johnston, „.NET Gadgeteer: A New Platform for K-12 Computer Science Education," v SIGCSE '13 Proceedings of the 44th ACM Technical Symposium on Computer Science Education, 2013.

[3] S. Sentance, S. Johnston, S. Hodges, J. Kučera, J. Scott a S. Schwiderski-Grosche, Learning to Program with Visual Basic and .NET Gadgeteer, Cambridge: Microsoft Research, 2013.