

# **.NET Gadgeteer Ultimate Guide**

The fun and easy way, the **FEZ** way...



Brought to you by Gus Issa



October 16, 2011

Licensed under Creative Commons Share Alike 3.0



**Copyright © GHI Electronics, LLC**  
[www.ghielectronics.com](http://www.ghielectronics.com)

## Table of Contents

1.Audience.....	3	11.1..NET Gadgeteer Sockets.....	18
2.Open Source Book.....	4	11.2..NET Gadgeteer Designer.....	18
3.Introduction.....	5	11.3.Writing Code.....	21
4.open platform helps!.....	6	12.Dasylink.....	25
5.Using Arduino, RTOS, Linux.....	7	13.Expanding the Hardware.....	26
6.Prerequisite.....	8	14.Graphics.....	27
7.GHI Electronics Gadgeteer Offers.....	9	15.Designing a Module.....	28
7.1.FEZ Spider.....	9	15.1.The Hardware.....	28
7.2.FEZ Next!.....	11	15.2.The Software.....	28
8.Other Gadgeteer offers.....	12	16.Moving to Production.....	29
8.1.Seeed Studio, Inc.....	12	16.1.Hardware Design Background.....	29
8.2.Systech Designs, Ltd.....	12	16.2.Case Study.....	31
9.Community Offers.....	13	Non-Gadgeteer Design.....	31
10.System Setup.....	14	Gadgeteer-Based Design.....	32
10.1.Ping the Device.....	14	17.Profitting from Gadgeteer.....	34
11.Blink an LED.....	18	18.What is next?.....	35



# 1. Audience

---

This ebook is provided by GHI Electronics for FREE as a “thank you” and a way to give back to the great NETMF community. The book will cover .NET Gadgeteer in general but will also list more info on GHI's offers for .NET Gadgeteer.

GHI also provides 2 other free ebook surrounding NETMF (.NET Micro Framework)

- Beginners Guide to NETMF (in the process of updating it) found at

<http://www.ghielectronics.com/downloads/FEZ/Beginners%20guide%20to%20NETMF.pdf>

- FEZ Internet of Things found at

[http://www.ghielectronics.com/downloads/FEZ/FEZ\\_Internet\\_of\\_Things\\_Book.pdf](http://www.ghielectronics.com/downloads/FEZ/FEZ_Internet_of_Things_Book.pdf)

If you have feedback on this book please provide it here <http://tinyclr.com/forum>

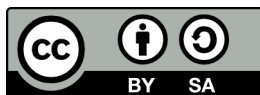


## 2. Open Source Book

---

This book is written using open-source Open Office, and provided as open source with the Creative Commons Share Alike license 3.0

<http://creativecommons.org/licenses/by-sa/3.0/>



This wiki page is open for anyone to add notes and for all to be in sync of what is changing in this open book. [http://wiki.tinyclr.com/index.php?title=Gadgeteer\\_Guide\\_Book](http://wiki.tinyclr.com/index.php?title=Gadgeteer_Guide_Book)

Please help by pointing out any errors in the book or by adding enhancements. Or by translating the book to your own language.

If you need to contact us directly then use any of the GHI Electronics contacts <http://www.ghielectronics.com/contact/>. If you need the writer pacifically, just ask fro Gus!



## 3. Introduction

---

.NET Gadgeteer is open source hardware and software, even better, it's an open source platform. How does this work? Gadgeteer defines a "connection point" between one "mainboard" and one or more "modules". The "connection point" is a 1.27mm 10pin socket which can hold one or more types. For example, socket type A is analog and S is SPI so when we see AS on a socket then this socket can work with modules type A or S.

In terms of software, there are a set of core drivers (called GadgeteerCore) that do things like check that modules are plugged into valid sockets, help manage cross-thread calls, provide advanced networking functionality and additional graphics libraries. The GadgeteerCore is released as open source on the codeplex website, and is independent from any mainboard, module or vendor. When a vendor produces a mainboard, they will need to add extensions to the Gadgeteer core and when a vendor produces a module they also need to provide extensions.

.NET Gadgeteer software is written in C# and made to run on top of .NET Micro Framework, which is another open source project by Microsoft.

The Gadgeteer project is found at <http://gadgeteer.codeplex.com/> (includes GHI's code)

The .NET Micro Framework project is found at <http://netmf.codeplex.com/>

.NET Micro Framework website <http://www.netmf.com>

.NET Gadgeteer website <http://www.netmf.com/gadgeteer/>

GHI Electronics' website <http://www.ghielectronics.com/>

**And most importantly**, GHI's community website <http://www.tinyclr.com/> with forum, chat, wiki, code-share, tutorial, free ebooks and a dedicated support from GHI Electronics engineers.



## 4. Open Platform Helps!

---

Open source projects are helpful because they allow users to dig into sources and design files to understand the system better to to enhance it. This is good but, in most cases, the user is still working with one company. What makes .NET Gadgeteer better than typical open sources projects is that it is an open platform and the people who invented it “Microsoft” do not even make hardware for it! They even let companies use .NET Gadgeteer to create open source and closed source devices.

What does this mean? You can take about any module from any vendor (open source or not), connect it to any mainboard (open source or not), and you should be able to use that module with a couple lines of code; thanks to .NET Gdgeteer, in most cases, it is under five lines of code. It is up to the vendor if they want to open their hardware/software or not.

Allowing for open source and closed source is very beneficial. For example, the WiFi module by GHI Electronics has agreements and NDAs behind it. Which means GHI will not be able to release it as open source. However, having WiFi is important for many users. The way Gadgeteer is designed, we can easily have complete open source modules/mainboards and still make use of closed source modules when necessary. Of course mainboards may not have every socket type or drivers for every single module. For example, a mainboard may not have socket A (analog) so you will not be able to use module type A. Overall, we believe that 99% of Gadgeteer modules will be generic and will run on any mainboard with a decent array of socket types.

GHI Electronics is working very close, with the .NET Gadgeteer team and other .NET Gadgeteer vendors, on verifying the compatibility and keeping the open platform concept alive.

**I may refer to Microsoft .NET Gadgeteer as just Gadgeteer for short. Also, .NET Micro Framework is shortened to NETMF.**



## 5. Using Arduino, RTOS, Linux

---

Not a .NET C# developer and wanting to use your own software? No problem, you can still take advantage of the Gadgeteer modules and mainboards with your preferred language and preferred environment. For example, a shield can be added to arduino which includes Gadgeteer compatible sockets. This allows you to take advantage of the growing list of available modules from multiple vendors. You can also use the open source C# module drivers as a base to write the c++ (sketch) drivers for arduino. This can be taken even further, why not run linux or your favorite RTOS preference right on the mainboard?

Basically, it is all possible but I highly suggest you first try the Gadgeteer software. The Gadgeteer team has really put a lot of work into the included open source code.

A member of GHI's community ported eLua to run on FEZ Panda, so running non-NETMF firmware is not new to GHI.



## 6. Prerequisite

---

This book expects you to have basic knowledge of C# and visual studio. Not to worry, if you don't, there is another beginner guide to get you through it all.

This page is a compilation of key resources and tutorials <http://www.tinyclr.com/support>

Not to forget about the very friendly and active community available at

<http://www.tinyclr.com/forum/> and the hundreds of code examples found at <http://code.tinyclr.com>

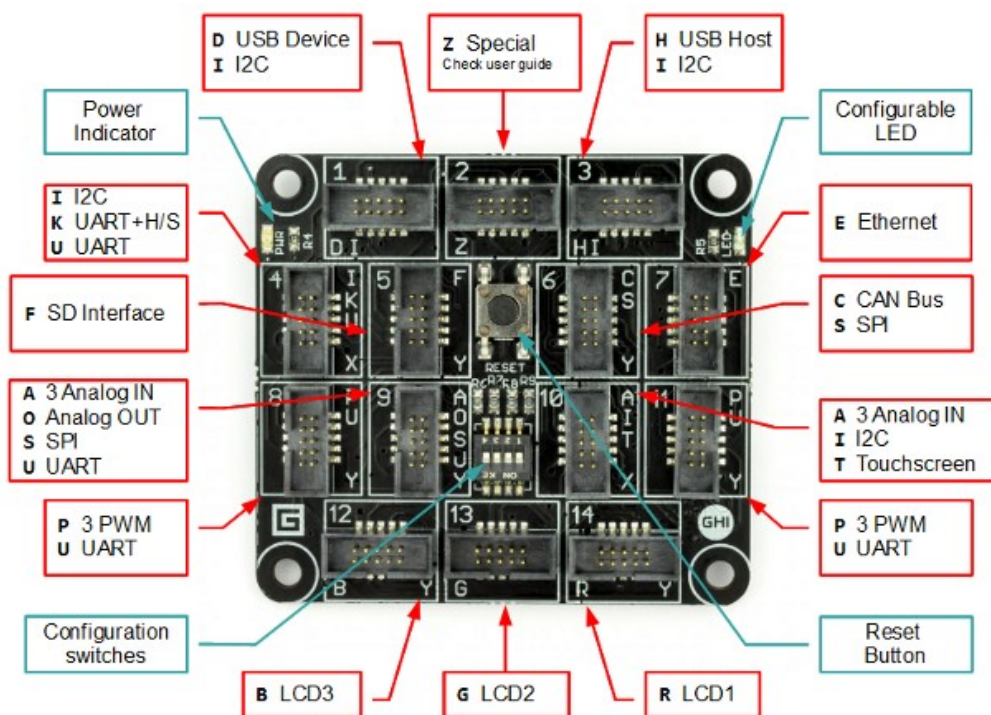
**Important note:** To make hardware understanding more friendly, Gadgeteer wraps all hardware dependent code in an easy to use and understand libraries. If you look around the GHI tutorials you will see things like OutputPort for example. This is wrapped into simpler calls in Gadgeteer. Normal Gadgeteer users will not need to learn the low level details but taking a quick tour through the tutorials will give you better understanding of the system internals. The tutorials are found here <http://www.tinyclr.com/support/>



## 7. GHI Electronics Gadgeteer Offers

### 7.1. FEZ Spider

FEZ Spider Mainboard <http://www.ghielectronics.com/catalog/product/269> is a .NET Gadgeteer-compatible mainboard based on GHI Electronics' EMX module. This makes FEZ Spider Mainboard the most feature-full .NET Gadgeteer compatible device in the market. It contains all of .NET Micro Framework core features and adds many exclusive features, such as USB host, WiFi and RLP (loading native code).



FEZ Spider also comes in a Starter Kit <http://www.ghielectronics.com/catalog/product/297> which is the first commercially available .NET Gadgeteer-compatible kit. It includes





## 7.2. FEZ Next!

Next FEZ mainboard will be added here! To be continued!



## 8. Other Gadgeteer Offers

---

There are many companies beside GHI working on .NET Gadgeteer and many new ones will come in near future. Unfortunately, most are still in development.

The book points out GHI's kits in most examples but the same code should apply very closely to any other mainboard or module. After all, they are all .NET Gadgeteer, no matter who is the maker. GHI Electronics is trying its best to communicate with all companies interested in .NET Gadgeteer and with Microsoft to ensure solid compatibility.

I will try to give a summary of companies offering .NET Gadgeteer compatible mainboards and modules. There aren't many as Gadgeteer is still very new. I will try to update this list when new offers become available. Companies are listed alphabetically.

### 8.1. Seeed Studio, Inc

Found at <http://www.seeedstudio.com>, located in China, Seeed offers many Gadgeteer modules. At this time, Seeed is still finalizing these modules and so they are not available yet. I have seen some prototypes myself. One of the samples I have seen was for a GPS Module which I think is a very important addition to a lot of users.

### 8.2. Systech Designs, Ltd

Found at <http://www.sytechdesigns.com>, located in United Kingdom, Systech Designs offer a mainboard called NANO which is also available as a kit. I will try to add more info once I get a hold of one and try it out.



## 9. Community Offers

The open source platform of .NET Gadgeteer allows for amazing community contributions. The platform is still very new and we already have few modules by Valentin Ivanov, known as Architect in the community. His first module is a Power over Ethernet (PoE) module. Here is an image of FEZ Spider showing an RSS feed and it is powered using Ethernet cable.



Have an idea? Share it now <http://tinyclr.com/forum/21/>



## 10. System Setup

Although using GHI's (and other vendors') gadgeteer boards is user friendly, there are few software components that need to be installed first in order to be able to use .NET Gadgeteer.

The components are:

- Visual C# Express 2010 or Visual Studio Professional 2010
- Microsoft .NET Micro Framework 4.1 SDK
- GHI .NET Micro Framework 4.1 SDK
- Microsoft .NET Gadgeteer Core SDK
- GHI .NET Gadgeteer SDK
- USB Drivers
- GHI Firmware Updater Application

To simplify the process, GHI packs many of the components into one download. A user will **only need** 3 downloads:

- Visual C# express
- Microsoft NETMF 4.1 SDK found at
- GHI NETMF & Gadgeteer Package found at

The above downloads are listed here <http://www.tinyclr.com/support/>

If you are using a non-GHI-offered mainboard or modules, you may also need to install any other SDKs provided by the other companies.

### 10.1. Ping the Device

Once you have all software components installed, we need to verify that the mainboard is



detected by windows and we can communicate with it.

The first step is to find the main board in your kit and then add a power module to it. The power module is the only Red board in the kit. Looking at the Red board's socket type, you see it is D. This is the USB Device (client) type. This module has 2 goals. First power up the mainboard (and other modules to be added) and also to provide access to the USB Device (client) functionality.

If you are using FEZ Spider, your setup should look like

(image)

Note that GHI's *USB Client DP* module is more advanced than typical "red modules". It can safely work with USB, with power pack or even power from both simultaneously. It also accepts input voltages from 7V to 30V, providing up to 800mA of current. DP stands for Dual Power.

You are now ready to plug the USB cable to a PC. Windows should detect it and automatically install the needed drivers. This is automatic because GHI's software package already included the USB drivers. I highly recommend you use a "powered USB hub". Most PCs, especially laptops, may not provide enough power. A powered hub is a hub that uses a power pack (usually included) to power the devices that are plugged in instead of relying on the power coming from the PC. This is also important as it keeps your PC safe from the devices that are plugged in. If you do not own a powered hub then plug your cable directly to the PC, not through a non-powered hub.

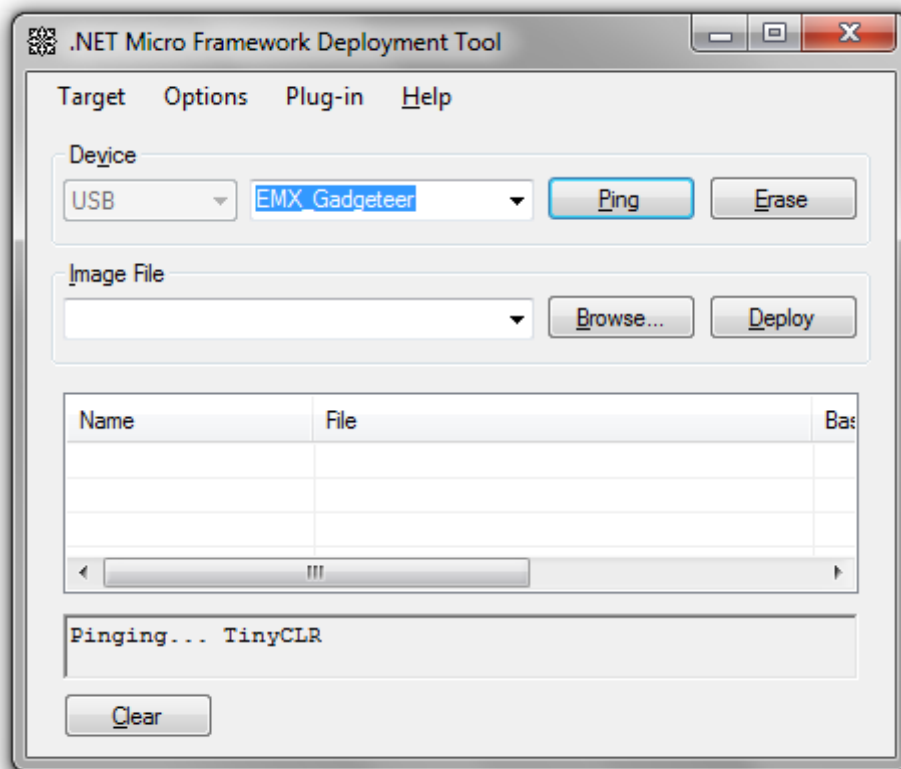
In some very rare cases, if the PC failed to detect the device or detected "unknown device", try to power up the mainboard using a power pack. Any voltage 7V to 30V is safe.

Once the USB is loaded, we want to "ping" the mainboard. "Ping" is a way for the software to ask the device "hello" and the device responds with "hi!". NETMF SDK ships with an application called MFDeploy, found at

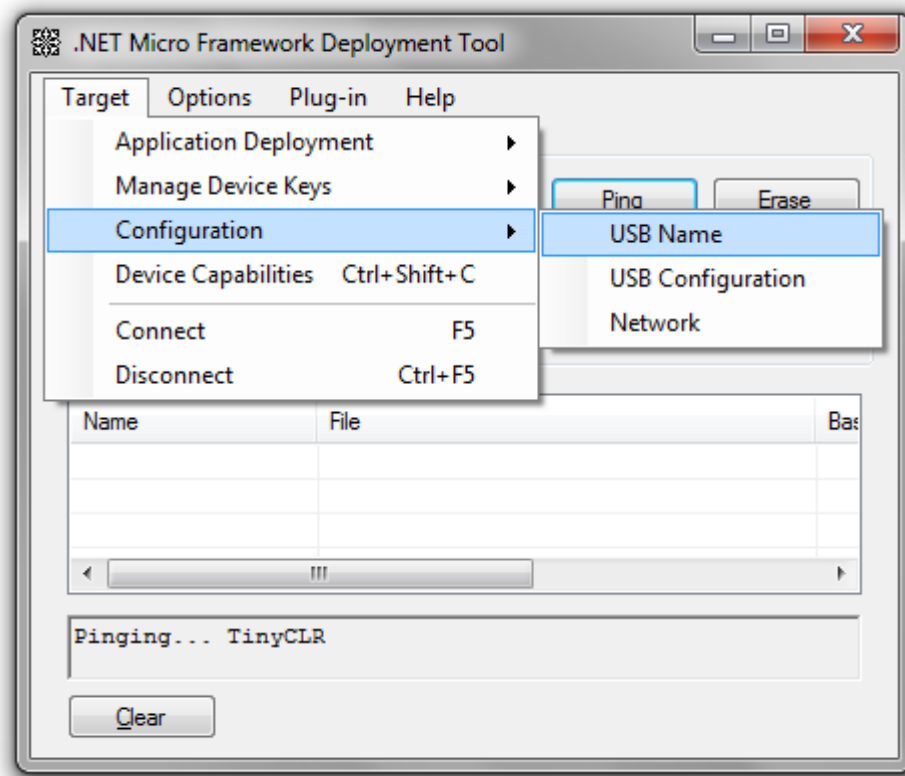
C:\Program Files (x86)\Microsoft .NET Micro Framework\v4.1\Tools\MFDeploy.exe

Open MFDeploy and select *USB* under *Device*. This will show *EMX\_Gadgeteer* in the list. Click the Ping button and the device will respond with "TinyCLR" as shown below.





If your device showed as EMX\_EMX instead of EMX\_Gadgeteer, change the name buy going to Target → Configuration → USB Name



**Important:** Set the name to Gadgeteer not EMX\_Gadgeteer. The “EMX\_” part is fixed so if you use EMX\_Gadgeteer you will wind up with EMX\_EMX\_Gadgeteer and this is not what we need.

GHI provide an application to update the firmware and it automatically updates the USB Name as required. It is found under C:\Program Files (x86)\GHI Electronics\GHI NETMF v4.1 SDK\Firmware Update\FEZSpiderMainboardUpdater.exe

(add image)



# 11. Blink An LED

In software, the first program you learn to write is how to print “Hello World” on the screen. In electronics, the “hello world app” is translated into a blinking LED (Light Emitting Diode). Why is it so important? Because this assures that the system is functional, your PC is setup with needed software, the device has the correct firmware on it. And most importantly, blinking LEDs is a lot of fun!

## 11.1. .NET Gadgeteer Sockets

The Gadgeteer mainboard has multiple sockets. Each socket can have one or more type. For example, Socket type D is for USB Device (Client), which what we need to plug the mainboard to a PC. Other modules maybe type S for SPI or A for Analog. What is important here is that all are designed to utilize the processors signals best and at the same time, be miss-use friendly. Connecting the wrong socket type will not work but it will also not harm the mainboard nor will harm the module.

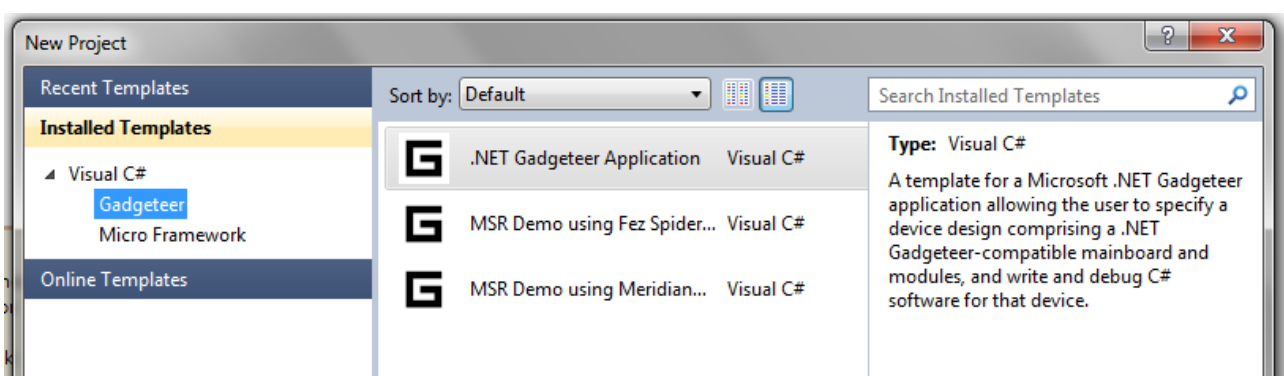
As for power, all sockets on mainboard share the same 5V, 3.3V and Ground signals. This means that all sockets can supply power to modules and the mainboard can take power from any socket. For this reason, the module that supply power is always colored Red and only one “Red module” can be connected to a mainboard.

Better details are highlighted in the Gadgeteer design guide provided by Microsoft.

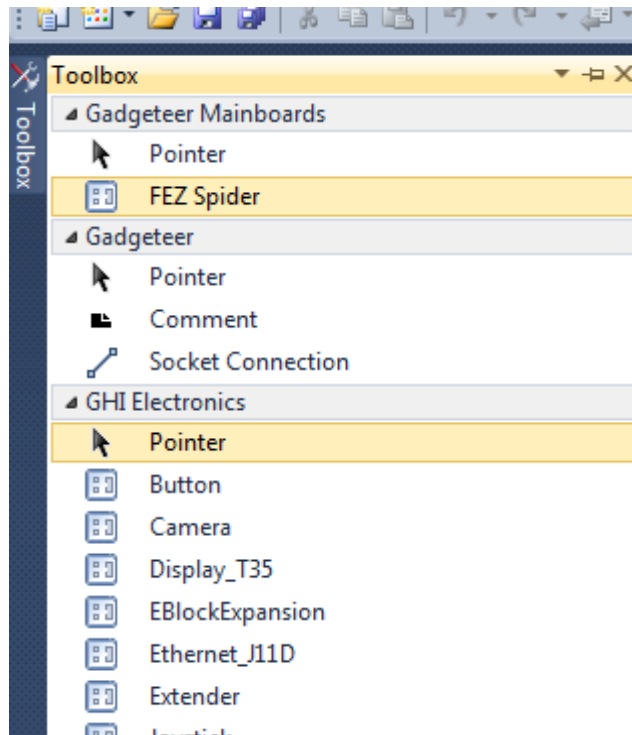
## 11.2. .NET Gadgeteer Designer

.NET Gadgeteer includes a designer that makes creating projects even easier. It allows you to configure your system graphically.

- Open Visual C# express (or professional) and create a new project. This is done by clicking File → New Project... in the top menu.
- From the templates, select Gadgeteer and then select “.NET Gadgeteer Application”

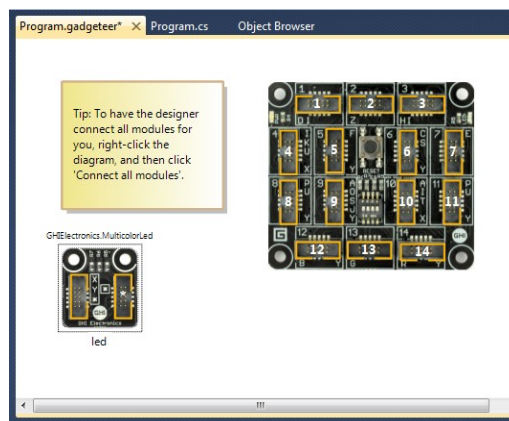


Give your project a good name, like “MyLED blinker” and click “OK”. This should create a project from yu and the designer will open by default. Move the mouse over “toolbar” and a menu will show up with available mainboards and modules.

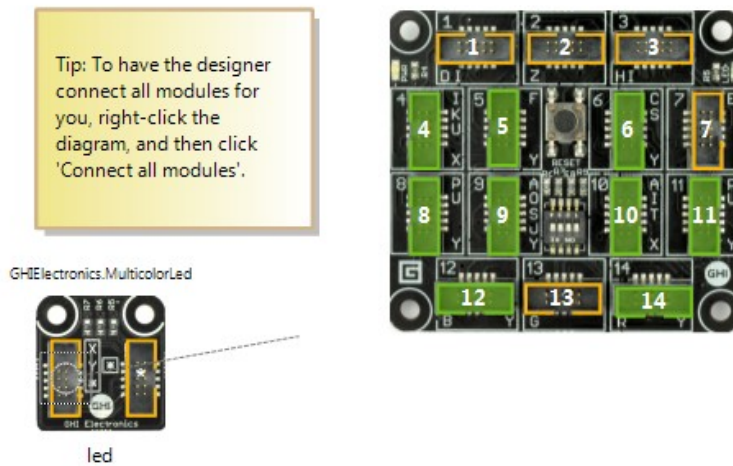


If FEZ Spider mainboard was already showing on the designer then skip this step, if not showing, double-click “FEZ Spider” under “Gadgeteer Mainboards”

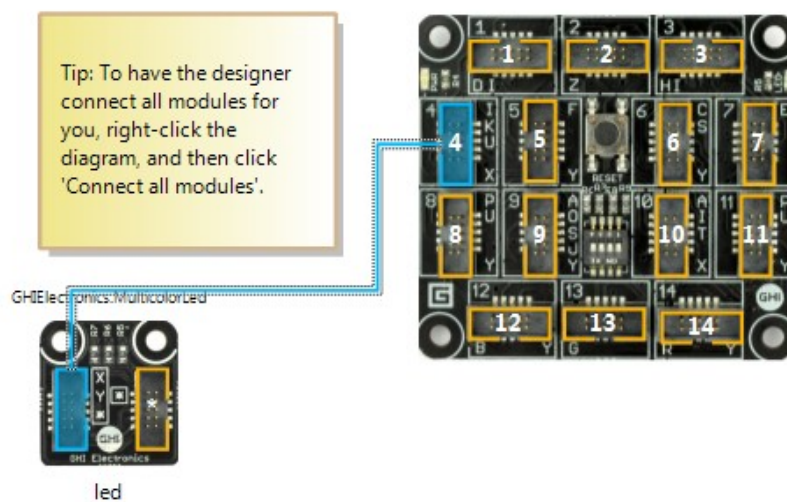
Now add the Milticolorled modules by double-clicking on it. The screen should something like this.



We now need to connect the LED module, on software on physically. But what socket to use? The LED module shows X and Y so you can basically use most sockets on the mainboard as most have type X or Y. I will pick socket 4, so move the mouse pointer over the socket on the LED model, till the pointer changes. Now, click on the socket once, remember it is the XY socket on the LED. This will case the mainboard sockets to change color as showing below



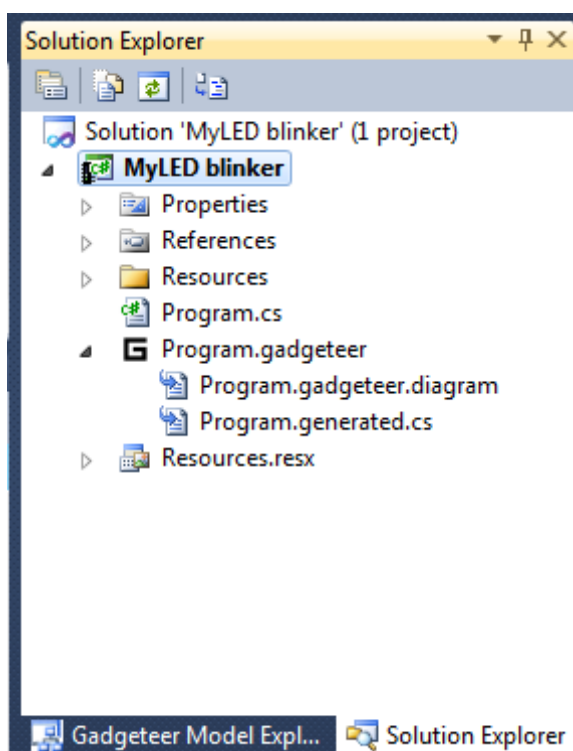
The sockets highlighted with green are the ones compatible with this LED module. Let us click on socket 4 on main board now. This will create a connection point.



Time to save your project. Click on File → Save All on the top menu. Give your project a good location and click the save button.

## 11.3. Writing Code

The saved project from last step has generated all the needed code to initialize your setup. Time to make the LED do something. Start by finding the “Solution Explorer” and double-click the file “Program.cs”



The default file will look something like this

```
using System;
using Microsoft.SPOT;
using Microsoft.SPOT.Presentation;
using Microsoft.SPOT.Presentation.Controls;
using Microsoft.SPOT.Presentation.Media;

using GT = Gadgeteer;
using GTM = Gadgeteer.Modules;
using GHI = Gadgeteer.Modules.GHIElectronics;
```



```

namespace MyLED_blinker
{
    public partial class Program
    {
        void ProgramStarted()
        {
            /*****
            Access modules defined in the designer by typing their name:

            e.g.  button
                camera1

            Initialize event handlers here.
            e.g.  button.ButtonPressed += new
            GTM.MSR.Button.ButtonEventHandler(button_ButtonPressed);
            *****/

            // Do one-time tasks here
            Debug.Print("Program Started");
        }
    }
}

```

As you can see, there is a note on where to start adding handlers and then the program start after the debug message “Program Started”. Gadgeteer is a modern system with events, threads, dynamic allocation. More on all this come later.

Blinking an LED involves an infinite loop that turns the LED on, waits, turns the led off, wait again then loop back. Processors are fast, if you do not add a delay, you will not see the LED blinking. Back in our designer, we had Multicolored module added. Its name was simply “led”. Now how easy it is to make this LED work? I can't wait to show you!

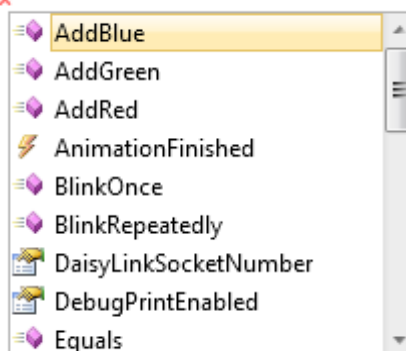
Go under “Program Started” and type led then hit the “dot” key. For those who are not familiar with intellisense, VisualStudi is already aware of what the “led” object can do and it lists them for you. What does this mean? You do not need to memorize the available methods and you do not need to look up long documentation.

```

// Do one-time tasks here
Debug.Print("Program Started");

```

led.



Join our community at [www.Tiny.com](http://www.Tiny.com)

The Multicolored module has an Red/Green/Blue LED (Light Emitting Diode). This allows the LED to be set to any color you want, through controlling what color is active and at what intensity. Lets start by adding the blue color to our LED. Then we need some delay, remove blue, another delay...and now we have a blinking LED. Since we are going to need to use delays, we need to “use” the threading library. To do so, add “using System.Threading” at the very top of your code.

Here is the code and do not forget to add “using System.Threading” at the very top of your code.

```
Debug.Print("Program Started");

while (true)
{
    led.AddBlue();
    Thread.Sleep(300);

    led.RemoveBlue();
    Thread.Sleep(300);
}
```

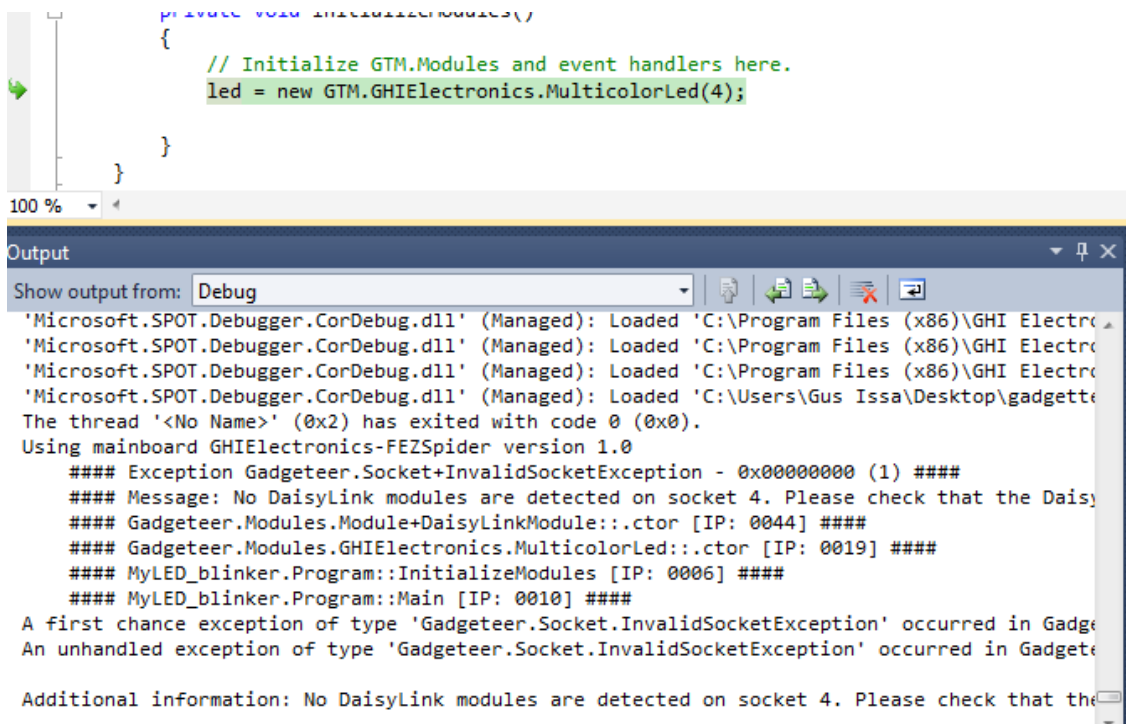
Now click the “Start Debugging” icon or hit F5 key



Visual Studio will now start deploying the needed assemblies to the device and in few seconds the LED will blink in blue color.

Always keep in eye on the “Output” window. For example, I purposely ran the program without the LED and I received this message (exception).





The screenshot shows a Visual Studio window with a C# code file and the Output window. The code file contains the following code:

```
private void InitializeModules()  
{  
    // Initialize GTM.Modules and event handlers here.  
    led = new GTM.GHIElectronics.MulticolorLed(4);  
}  
}
```

The Output window shows the following output:

```
Show output from: Debug  
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\GHI Electro  
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\GHI Electro  
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Program Files (x86)\GHI Electro  
'Microsoft.SPOT.Debugger.CorDebug.dll' (Managed): Loaded 'C:\Users\Gus Issa\Desktop\gadgeteer  
The thread '<No Name>' (0x2) has exited with code 0 (0x0).  
Using mainboard GHIElectronics-FEZSpider version 1.0  
#### Exception Gadgeteer.Socket+InvalidSocketException - 0x00000000 (1) ####  
#### Message: No DaisyLink modules are detected on socket 4. Please check that the Daisy  
#### Gadgeteer.Modules.Module+DaisyLinkModule::.ctor [IP: 0044] ####  
#### Gadgeteer.Modules.GHIElectronics.MulticolorLed::.ctor [IP: 0019] ####  
#### MyLED_blinker.Program::InitializeModules [IP: 0006] ####  
#### MyLED_blinker.Program::Main [IP: 0010] ####  
A first chance exception of type 'Gadgeteer.Socket.InvalidSocketException' occurred in Gadge  
An unhandled exception of type 'Gadgeteer.Socket.InvalidSocketException' occurred in Gadgete  
Additional information: No DaisyLink modules are detected on socket 4. Please check that the
```

For example, you may see a message about LCD configuration the first time you deploy to the device and you need stop VS and deploy again.



## 12. Dasylink

---

Explain dasylink .... to be continued!



## 13. Expanding The Hardware

---

Advanced users

Extender module and breadboards

Accessing the hardware directly...RLP maybe?

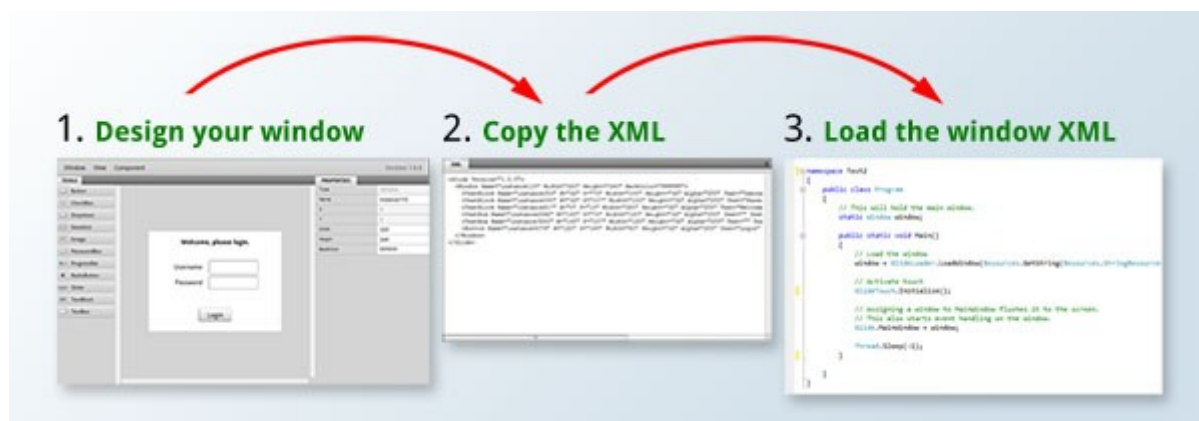


## 14. Graphics

The methods provided by Display modules are usually design with simplicity in mind. But thanks to the native NETMF graphics support, the system is capable with a lot more. I suggest starting with the Display module library but if that didn't get you what you need then take a look at the display tutorial on GHI's community wiki to get an idea of how to use the "Bitmap" object <http://wiki.tinyclr.com/index.php?title=Displays>

NETMF also offers a subset of Windows Presentation Foundation. You can create about anything with WPF but at performance penalties. Also, designing the windows will have to be done programmatically. There is no window designer.

My favorite alternative to all above is Glide, which runs perfectly on any .NET Gadgeteer mainboard. Glide is a graphical library for .NET Micro Framework (NETMF) that uses a graphical screen designer. Glide provides a more responsive experience than NETMF's built-in Windows Presentation Foundation (WPF) with many additional features, from buttons and lists to keyboards and message boxes.



Glide is provided free for non-commercial use. Visit Glide's main page for further details <http://www.ghielectronics.com/glide>



## 15. Designing A Module

---

Designing a mainboard is very complex and involves deep understanding of porting NETMF and then deep understanding of porting .NET Gadgeteer. This will require a whole book itself. On the other hand, designing a module is much simpler and much more needed. Creating a new module involves designing and testing the hardware and also writing a Gadgeteer compatible drivers. I will cover them separately.

### 15.1. The Hardware

If you use EAGLE or Altium designers then the process is much simpler. Microsoft provide many module examples in Altium format and GHI provide many module designs in EAGLE format. These files already include the Gadgeteer socket, already include the correct hole sizes and correct silkscreen, logos and sizes.

The community along with GHI are also available for suggestions and help.

### 15.2. The Software

The software design for a Gadgeteer module involves two steps. One is the actual drivers that will handle the module itself. The second is the files needed for the Gadgeteer designer.

... to be continued!

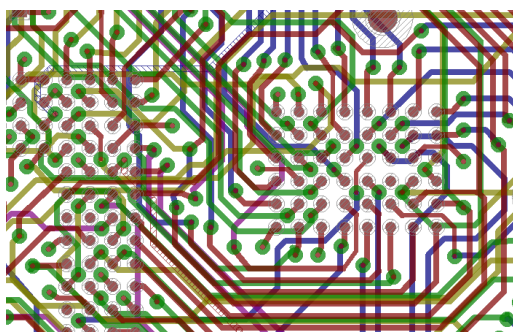


## 16. Moving To Production

### 16.1. Hardware Design Background

While .NET Gadgeteer make prototyping extremely fast and convenient, it may not be suitable for mid to high run volumes. Bringing all needed components on one circuit board makes the final board smaller and brings the cost down. There is also the possibility of expanding the system beyond what available modules can offer.

In fact, moving to production on a custom board is very simple. But first, I want to talk a bit about the complexity of hardware designs. Since I have a very long experience in embedded systems design stages I will explain in details. The most difficult part would be to boot up an embedded processor and then getting its peripherals to work as expected. Thanks to NETMF, this is already covered and abstracted. The second difficult part of the hardware design, and the most expensive step, is selecting components and making prototypes. Manufacturing simple boards for a new design, that you do not even know will work, will cost about 5,000 USD. In the over 10 years I designed hardware, I have never seen a circuit board that went from prototype to production in one run. There is at least second round, average is 3 runs, add another 10,000 USD. Thanks to .NET Gadgeteer, you get to do this at fraction of the cost, basically free. It is free because the kit cost is not really for that one design. The Gadgeteer mainboard and modules can be reused over and over. If needed, the designer can add additional prototype circuit through the standard Gadgeteer socket. The cost and time needed to prototype a small module is much less than designing the whole system, and more importantly, have lower risk.

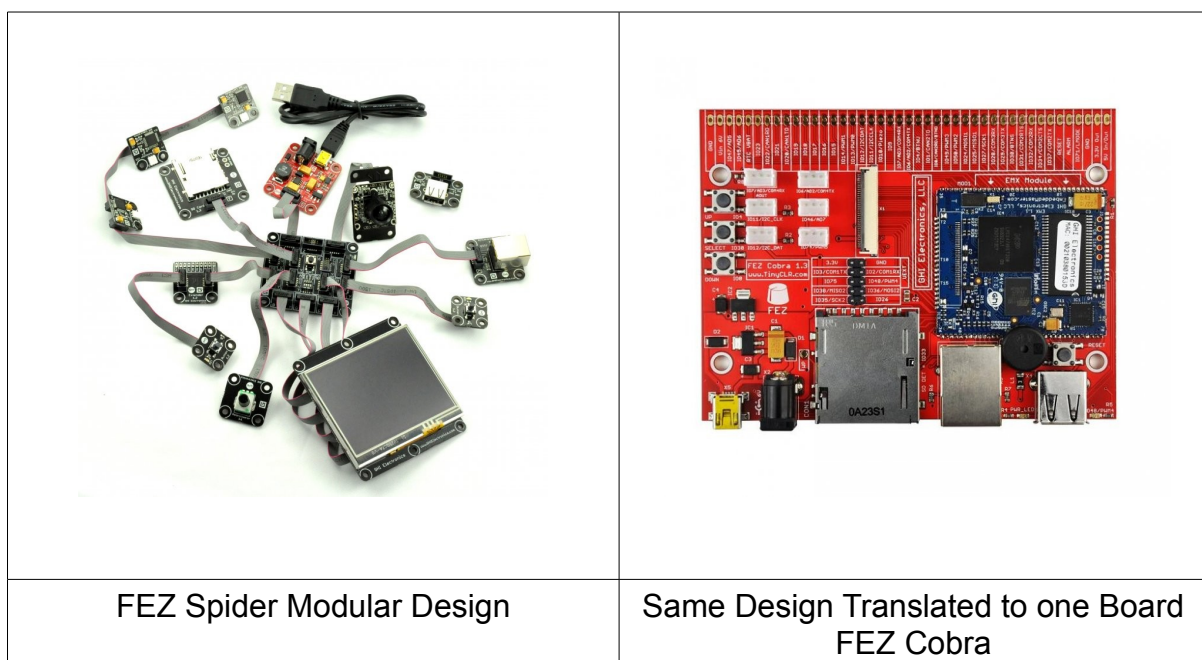


Snapshot of the EMX Module Flash to Processor Routing



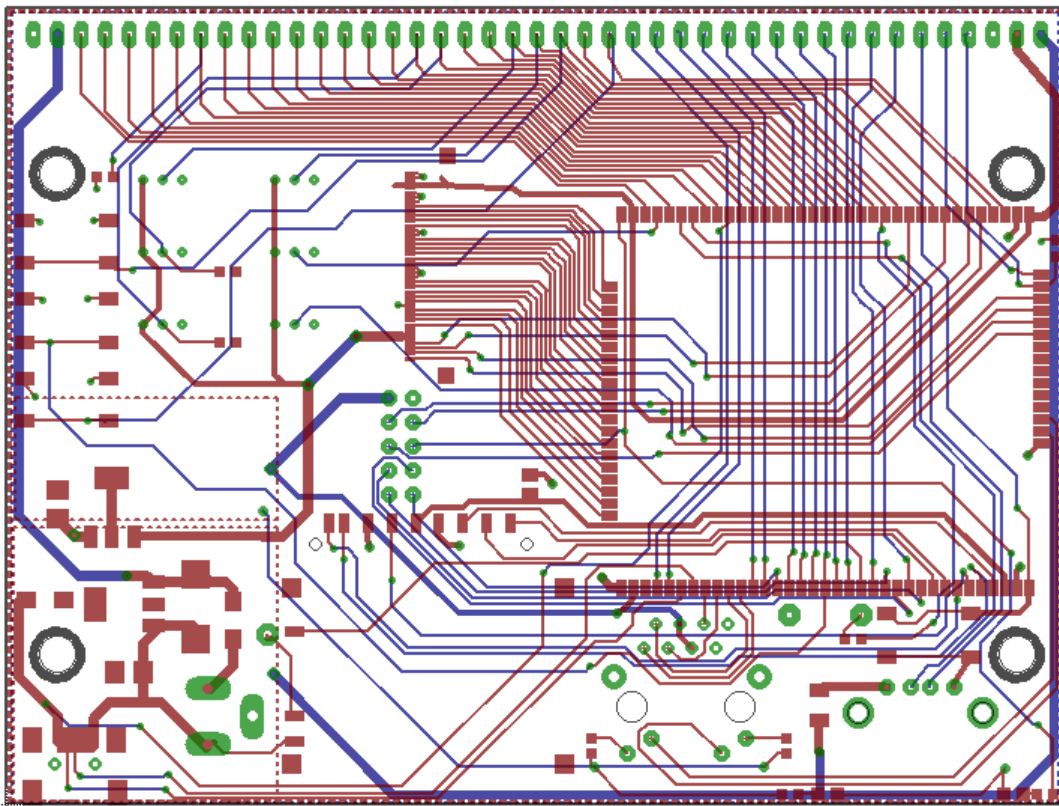
Adding on, thanks to GHI, almost all design files are open source. So, when you have a gadgeteer-based design ready, you can easily take the schematics from each module you used and combine them in one design. This gets even a one step better, easier and cheaper, by using a processor module. The processor module (not to confuse it with Gadgeteer modules) is very complex multilayer board. For example, the heart of FEZ Spider EMX module which is an 8-layer-board. By using a readily processor module, your final design can be a simple 2 layer circuit boards that can be routed easily with minimal experience and low cost software, sometimes free software.

Lets us assume you need a design with Ethernet, USB Client, USB Host and SD card, your design will translate as showing in these pictures



Comparing to the EMX 8-layer circuit layout shown earlier, this is the complete layout of the 2-layered FEZ Cobra design. Routing a circuit like EMX would take at least a month of work where routing FEZ Cobra can be done in one day and it can even be hand soldered if necessary.





## 16.2. Case Study

Let us assume you need to make a nice GPS logging system. I have seen these go for about 300 USD and they do not even include a nice color touch display as a user interface. In summary, you want to design one that looks great, fully features, with color touch display and costs under \$150! In order for you to get the okay of your investor, or your boss, you need to show them a fully functional prototype. I will go through this with you using conventional hardware design stages and through using a Gadgetry-compatible kit, FEZ Spider for example.

### Non-Gadgeteer Design

First of all, we need to know what processor to use. It must also support LCD interface. There are hundreds, or maybe thousands, or processors out there. I give it about 2 weeks to select a processor and another 2 weeks to order the processor's evaluation kit. This is one month and we haven't even started! Maybe you selected LPC2478, the



same processor used on EMX module. Searching DigiKey's website, a known electronics webstore, you will find one kit from the chip maker, NXP. The kit costs \$465.50 when I checked <http://search.digikey.com/us/en/products/OM11076/568-4741-ND/2052461>

Now we need to download the software for the processor. The most common commercial one is MDK, which costs over 3000 USD or 5000 USD according to Digikey! You also need to a JTAG to step in code for debuggine. You can also look into using GCC for free but there is a long learning curve in setting up the tools plus you will most likely lose the debugging capability and have larger firmware size. Basically, after spending over 5000 USD, you are ready to start evaluating the processor, which you are not even sure you will be happy with or not.

If all went well, you need to design a board with the processor, BGA or fine pitch options only, route SDRAM bus, which is critical. This is about a month of work and if you hired someone to do it then it will cost between 5000 USD and 10000 USD.

Finally you need to get the design off to the board house. Board tooling + assembly tooling + assembly is about 3000 USD to 5000 USD.

A good place to check for online quotes on assembly is

<http://www.screamingcircuits.com/>

Adding on, how would the end "sample prototype" or "proof of concept" look like when presented to the investor? Even if it is not in the final enclosure, the design should look presentable.

In summary, the cost for **professionally** making hardware is no less than 30000 USD and averages at about 50000 USD. There are ways to cut corners and save some cost but this is not the discussion here.

I know GHI has spent more than 15,000 USD on compilers/debuggers alone.

### Gadgeteer-Based Design

There is no way for me to explain this without sounding like a sales guy! The 30000 USD minimum design above will cost you \$500, including the cost for software, for debugger and for custom enclosure. The earlier design would need about 6 months to get a prototype ready. You can do the same in one week with Gadgeteer-based design!

All you need is a FEZ Spider Starter Kit, which can be reused in future, and a GPS module. GPS parsing code is already done for you, the display and graphics is already

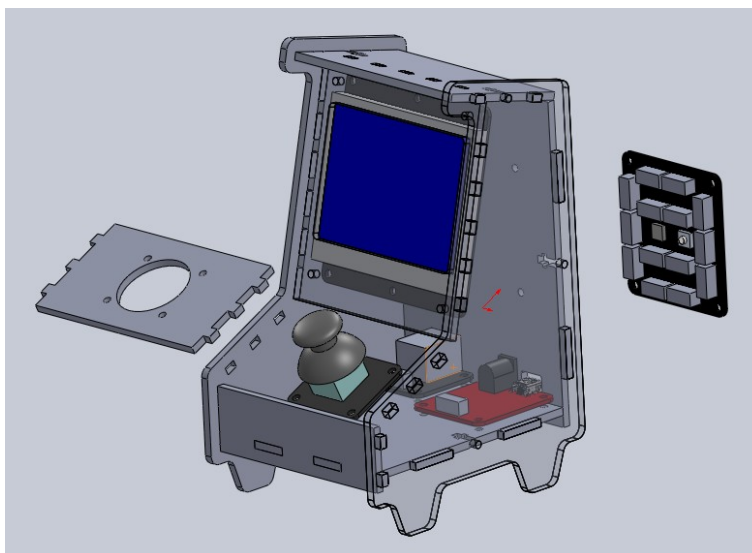


done for you, including JPEG image decoding. You can add SD to save data at not additional work nor cost, even add a USB memory stick as well. The entire code for such application will be under 500 lines of code.

As for software, Visual C# express is state-of-the-art IDE and it is free. The NETMF, Gadgeteer and GHI libraries are all already included with FEZ Spider Starter Kit.

When the design is completed, an enclosure can be 3D printed or laser cut, keeping the complete cost under 500 USD.

A fun example would be the mini arcade Microsoft show on Gadgeteer website. There are some nice videos to watch here <http://tinyclr.com/forum/21/4283/>



## 17. Profiting From Gadgeteer

---

Whether you are a one-man-company or a large firm, the open platform of Gadgeteer makes it a great place for collaboration. Maybe you are an expert in automotive and want to offer a J1850 or FlexRay module. Maybe you are an expert in industrial control and want to offer CANopen module. An expert in aviation and can provide an IMU with needed software? The community always talk about quadcopters. You will be famous overnight for making such a module. This can be taken as far as providing an million-gate-FPGA.

Not only hardware, software components are also needed for the rapid-development Gadgeteer platform offers.

GHI Electronics is doing its best in offering variety of modules and continues to add more. At the same time, GHI welcome those making new modules to expand the “gadgeteer reach”. You can even contact GH about reselling your own modules.





## 18. What Is Next?

One important place to always visit is this page that has all the resources needed and many tutorial links:

<http://www.tinyclr.com/support>

This is a list of other resources that should help in your next invention!

- GHI Electronics website:  
<http://www.ghielectronics.com/>
- The GHI community website:  
<http://www.tinyclr.com/>
- Tutorials and free books:  
<http://www.tinyclr.com/support/>
- The GHI community wiki:  
<http://wiki.tinyclr.com>
- The GHI community code share:  
<http://code.tinyclr.com>
- Facebook:  
<http://www.facebook.com/pages/GHI-Electronics/201366383211427>
- Twitter:  
<http://twitter.com/GHIElectronics>
- YouTube:  
<http://www.youtube.com/GHIElectronics>
- Microsoft's .NET Micro Framework website:  
<http://www.netmf.com/>
- Microsoft's Gadgeteer website:  
[www.netmf.com/gadgeteer/](http://www.netmf.com/gadgeteer/)

