A new way to make devices

Nicolas Villar
Sensors and Devices Research Group
Microsoft Research
Cambridge, UK
.NET Gadgeteer is a new toolkit for quickly constructing, programming and shaping new small computing devices (gadgets)

“From idea to working device quickly and easily”
Low threshold
Simple gadgets should be very simple to build

High ceiling
It should also be possible to build sophisticated and complex devices
3 Key Components

Modular Hardware
3 Key Components

Modular Hardware

Object-Oriented Programming

```java
void ProgramStarted()
{
    // Initialize GTM.Modules and
    myButton = new GTM.Button(GT);
    myLed = new GTM.MulticolorLED;

    myButton.
    // Do one
    Debug.Print
    ButtonPressed
    ButtonReleased
    DebugPrintEnabled
    Equals
    GetHashCode
    GetType
    IsPressed
    ToString
}```
3 Key Components

Modular Hardware

Object-Oriented Programming

Digital Design and Fabrication
Some History

• We originally built Gadgeteer as a tool for ourselves (in Microsoft Research) to make it faster and easier to prototype new kinds of devices

• Since then, it has proven to be of interest to other researchers – but also hobbyists and educators

• With the help of colleagues from all across Microsoft, we are working on getting Gadgeteer out of the lab and into the hands of others
Modular Hardware
Modules
The Mainboard
Modules – User Interface

- USB Host
- Rotary Encoder
- D-Pad
- Button
- Multicolor LED
- OLED Display
- 480x272 Touchscreen
Modules – Sensors

- Ultrasonic Ranger
- Passive IR
- Temp. & Humidity
- RGB Sensor
- Accelerometer
Modules – Multimedia

- Audio
- Camera
Modules – Networking

- ZigBee
- 802.15.4
- Ethernet
- USB
- Serial
- WiFi
- 868MHz
Modules – Storage

- USB Host
- SD Card
Modules – Actuators

Motor & Servo Control

Relay - low voltage

Relay – mains voltage
Modules – Power Supply

DC Power

Battery
Modules – Extensibility

- Analog IO
- Digital IO
- Breakout
## Gadgeteer Socket Pin-Mapping Table

<table>
<thead>
<tr>
<th>Basic Socket Types (modules list all that they are compatible with)</th>
<th>3 GPIO</th>
<th>7 GPIO</th>
<th>Analog In</th>
<th>CAN</th>
<th>USB Device</th>
<th>Ethernet</th>
<th>SD Card</th>
<th>USB Host</th>
<th>I2C</th>
<th>UART+Handshaking</th>
<th>Analog Out</th>
<th>PWM</th>
<th>SPI</th>
<th>Touch</th>
<th>UART</th>
<th>LCD 1</th>
<th>LCD 2</th>
<th>LCD 3</th>
<th>Manufacturer Specific</th>
<th>DaisyLink Downstream*</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 GPIO</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>7 GPIO</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>Analog In</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>CAN</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>Ethernet</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>SD Card</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>USB Host</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>I2C</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>UART+Handshaking</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>Analog Out</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>PWM</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>SPI</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>Touch</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>UART</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>LCD 1</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>LCD 2</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>LCD 3</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>Manufacturer Specific</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
<tr>
<td>DaisyLink Downstream*</td>
<td>+3.3V</td>
<td>+3V</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
<td>[NC]</td>
</tr>
</tbody>
</table>
Module Design Guidelines

The keep-out area should be clearly delimited in the silk screen on both sides of the PCB, as shown in the following illustration. For small modules, where space is tight, it is possible to interrupt the keep-out delimiter silk screen to make space for other labeling or silk screen elements. Under no circumstances should you place components inside the keep-out area.

Multiple Connectors
A module might include more than one connector in the following two instances:
- The module needs to provide connection to more than one mainboard socket at a time.
- The module is compatible with the .NET Gadgeteer DelayLink Protocol and can be delay-chained with other modules.

Modules with Multiple Socket Connectors
In this case, connectors can be placed anywhere on the board, following the guidelines under "Connectors" earlier in this document. If possible, connectors should be placed side by side near an edge of the board, as shown in the following illustration.

If the dimensions of the board prohibit a side-by-side arrangement of connectors, the next best option is to place a second connector immediately behind the first, in the same orientation, as shown in the following illustration.
Object-Oriented Programming

```csharp
void ProgramStarted()
{
    // Initialize GTM.Modules and
    myButton = new GTM.Button(GTM);
    myLed = new GTM.MulticolorLED;

    myButton.

    // Do one
    Debug.PrintPressed
    Debug.PrintReleased
    Debug.PrintEnabled
    Debug.PrintEquals
    Debug.PrintHashCode
    Debug.PrintGetType
    Debug.PrintIsPressed
    Debug.PrintToString

    // Initialize GTM.Modules and
    myButton = new GTM.Button(GTM);
    myLed = new GTM.MulticolorLED;
}
```
Software Development Libraries

- Gadgeteer uses the Microsoft .NET Micro Framework (NETMF), which provides a simple and powerful way to write software for small devices.

- Software is developed and debugged in Visual Studio, and code is in managed, object-oriented C#.

- The SDK provides classes encapsulating functionality for individual hardware modules as well as other utility functions.
.NET Micro Framework

- C# Application and User Libraries
- Class Libraries (Display, Networking, I/O, File System...)
- Runtime Component Layer (Hardware Abstraction + CLR)
- Hardware
Other NETMF devices

FEZ (GHI Electronics)

Netduino (Secret Labs)
Coding aids

Namespace alias to make list of modules easy to find

 Enums to provide clear lists of parameter options

Unabbreviated names

Thread safe event model

In-line help

Auto-generated event handlers
Coding aids

Many method signatures avoiding need to specify default values

Provide simple and complex APIs when appropriate

void button_ButtonPressed(GTM.Button sender, GTM.Button.ButtonState state)
{
    camera.TakePicture();
}

High level object-oriented API

Include many parameters since local vars are easier

Templates and examples

Examples go here
Gadgeteer Cloud Portal
For Web-Of-Things Apps
Gadgeteer Cloud Portal (Azure Service)

Other Web Services (e.g. Twitter)
Digital Design and Fabrication
Cardboard prototyping
Cardboard prototyping
Digital design and rapid manufacture
Falling cost and increasing availability of 3D printers
Digital design and rapid manufacture

Digital Design  →  3D Printer  →  Physical Object
Digital design and rapid manufacture

Digital Design → Online 3D Printing Service → Physical Object

imaterealise
• We want to make it easier to give shape to Gadgeteer devices by using digital fabrication technologies

• First step: integration with 3D CAD modelling software (e.g. Solidworks)
Add CAD models of modules
Auto-generate mounting fixtures
Auto-generate openings
A 24 Hour Prototyping Exercise:
Making a Hand-Held Videogame
Hardware configuration (~5 minutes to assemble)

- Four-way switch to control placement of puzzle piece
- Knob to rotate piece
- Colour OLED display (128x128 resolution)
- USB power source and programming socket
public class Piece
{
    public Point[] positions;
    public Point displacement;
    public Color color;

    public Piece(Point[] positions, Point displacement, Color color)
    {
        this.positions = positions;
        this.displacement = displacement;
        this.color = color;
    }

    public void Rotate(bool clockwise)
    {
        for (int i = 0; i < positions.Length; i++)
        {
            Point oldpos = positions[i];
            positions[i].x = clockwise ? -oldpos.y : oldpos.y;
            positions[i].y = clockwise ? oldpos.x : -oldpos.x;
        }
    }

    public Piece Clone()
    {
        Piece clone = new Piece((Point[])positions.Clone(), new Point(displacement.
        return clone;
    }
}
Enclosure design in Solidworks (~3 hours)
Enclosure 3D printed
(~6 hours)
Assembly (~30 minutes)
Next steps: Getting .NET Gadgeteer out of the lab

• .NET Gadgeteer software, hardware design and design guidelines released as open source project:
  http://gadgeteer.codeplex.com/

• Community site (in development):
  http://netmf.com/gadgeteer
Next steps: Getting .NET Gadgeteer out of the lab

• Working with a number of hardware manufacturers who will build, distribute and sell the hardware modules

• Initial availability expected end of July

• Started kit priced around $250

• More modules to become available from different manufacturers during the rest of the year
More information

Please get in touch if you are interested in using .NET Gadgeteer for research or teaching

gadgeteer@microsoft.com