Microsoft Research Faculty Summit 2012
ADVANCING THE STATE OF THE ART
Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, Colin Miller
Microsoft Research
Custom devices in ubicomp research
Custom devices often need to be:

- developed quickly by a small team
- fully functional and usable
- self-contained
- usable in the wild
- able to be produced in quantity
Existing tools have *some* of the qualities we look for for.
Making a custom hand-held videogame device in 24 hours
Connect hardware modules (5 minutes)

- Mainboard
- Four-way switch
- Input potentiometer
- Colour OLED display (128x128 resolution)
- USB power source and programming socket
```csharp
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 (3 hours)
3D printing (6 hours)
Assembly (20 minutes)
The .NET Gadgeteer Platform

Modular Hardware

Software Tools

Physical Design

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

    myButton.

    // Do one
    Debug.Print

    ButtonPressed
    ButtonReleased
    DebugPrintEnabled
    Equals
    GetHashCode
    GetType
    IsPressed
    ToString
```
The .NET Gadgeteer Platform

Modular Hardware

Software Tools

Physical Design

void ProgramStarted()
{
    // Initialize GTM.Modules and
    myButton = new GTM.Button(GTM.
    myLed = new GTM.MulticolorLE
    myButton.

    // Do one
    Debug.Print
    }
At the heart of every Gadgeteer project is a **mainboard**. A mainboard is made up of a programmable processor (ARM7 / ARM9 / Cortex M4), memory, and a number of sockets that Gadgeteer **modules** can plug into.
Modules: sensors

- Seeed Studio Compass
- Seeed Accelerometer
- Seeed Studio Soil Moisture Sensor
- Seeed GPS
- Seeed Temperature and Humidity Sensor
- Seeed Studio Gyroscope
- Seeed Studio Barometer
- Sytech 3-Axis Accelerometer
- Seeed Pulse Oxymeter
- GHI PIR Sensor
- Seeed Current Sensor
- GHI Light Sensor
Modules: communication

- GHI RS232
- Seeed Cellular Radio
- GHI Serial-USB
- Sytech Ethernet and SD
- GHI CAN (Dual-Wire)
- GHI XBee Adapter
- GHI Bluetooth
- GHI Ethernet J11D
- GHI Ethernet ERC28
- GHI WiFi RS21
Modules: display and user input

- Seeed OLED Display
- GHI Display T35
- GHI LED7R
- GHI Video Out
- GHI Multicolor LED (DaisyLink)
- Sytech LCD Touch Panel 4.3
- GHI Camera
- Sytech Serial Camera
- GHI Potentiometer
- GHI Button
- Sytech Button LED
- GHI Joystick
Modules: power and actuation

- Sytech USB Device
- GHI USB Client SP
- GHI USB Client DP (Dual-Power)
- SolderMonkey LittleStep
- Seeed Relays
- GHI Motor Driver L298
Modules: storage and audio

- GHI Micro SD Card
- GHI USB Host
- GHI SD Card
- Sytech Ethernet and SD

- GHI Music
Modules: extensibility
Sockets have types, which specify their electronic interface capabilities.
### Socket Type A

<table>
<thead>
<tr>
<th>Pin 1</th>
<th>Pin 2</th>
<th>Pin 3</th>
<th>Pin 4</th>
<th>Pin 5</th>
<th>Pin 6</th>
<th>Pin 7</th>
<th>Pin 8</th>
<th>Pin 9</th>
<th>Pin 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>+3.3V</td>
<td>+5V</td>
<td>AIN (G)</td>
<td>AIN (G)</td>
<td>AIN</td>
<td>GPIO</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
</tbody>
</table>

*Pinout specified by the socket type A definition.*

**AIN**
- Analog input pin.

**GPIO**
- A general-purpose digital input/output pin, operating at 3.3 Volts.

**(G)**
- In addition to another functionality, a pin that is also usable as a GPIO.

**[UN]**
- Modules must not connect to this pin if using this socket type. Mainboards can support multiple socket types on one socket, as long as individual pin functionalities overlap in a compatible manner, so that a pin from one socket type can overlap with a [UN] pin of another.

**!**
- Interrupt-capable and software pull-up capable GPIO (the pull-up is switchable and in the range of 10,000 to 100,000 ohms).

**+3.3V**
- Connection to the +3.3V power net.

**+5V**
- Connection to the +5V power net.

**GND**
- Connection the power ground net.
**GPIO**  
A general-purpose digital input/output pin, operating at 3.3 Volts.

- Modules must not connect to this pin if using this socket type. Mainboards can support multiple socket types on one socket, as long as individual pin functionalities overlap in a compatible manner. A pin from one socket type can overlap with a [UN] pin of another.
- Interrupt-capable and software pull-up capable GPIO (the pull-up is switchable and in the range of 10,000 to 100,000 ohms).
- Socket type * should not appear on a mainboard, only on DaisyLink modules. The [MS] pins on this socket type can optionally support reflashing the firmware on the module.

### Socket types definition table

<table>
<thead>
<tr>
<th>TYPE</th>
<th>LETTER</th>
<th>PIN 1</th>
<th>PIN 2</th>
<th>PIN 3</th>
<th>PIN 4</th>
<th>PIN 5</th>
<th>PIN 6</th>
<th>PIN 7</th>
<th>PIN 8</th>
<th>PIN 9</th>
<th>PIN 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 GPIO</td>
<td>X</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>GPIO</td>
<td>GPIO</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>7 GPIO</td>
<td>Y</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>GPIO</td>
<td>GPIO</td>
<td>GPIO</td>
<td>GPIO</td>
<td>GPIO</td>
<td>GPIO</td>
<td>GND</td>
</tr>
<tr>
<td>Analog In</td>
<td>A</td>
<td>+3.3V</td>
<td>+5V</td>
<td>AIN (G!)</td>
<td>AIN (G)</td>
<td>AIN</td>
<td>GPIO</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>CAN</td>
<td>C</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>TD (G)</td>
<td>RD (G)</td>
<td>GPIO</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>USB Device</td>
<td>D</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>D-</td>
<td>D+</td>
<td>GPIO</td>
<td>GPIO</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>Ethernet</td>
<td>E</td>
<td>+3.3V</td>
<td>+5V</td>
<td>[UN]</td>
<td>LED1 (OPT)</td>
<td>LED2 (OPT)</td>
<td>TX D-</td>
<td>TX D+</td>
<td>RX D-</td>
<td>RX D+</td>
<td>GND</td>
</tr>
<tr>
<td>SD Card</td>
<td>F</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>DAT0</td>
<td>DAT1</td>
<td>CMD</td>
<td>DAT2</td>
<td>DAT3</td>
<td>CLK</td>
<td>GND</td>
</tr>
<tr>
<td>USB Host</td>
<td>H</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>D-</td>
<td>D+</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>I2C</td>
<td>I</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GPIO</td>
<td>[UN]</td>
<td>SDA</td>
<td>SCL</td>
<td>GND</td>
</tr>
<tr>
<td>UART+Handshaking</td>
<td>K</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>TX (G)</td>
<td>RX (G)</td>
<td>RTS</td>
<td>CTS</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>Analog Out</td>
<td>O</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>GPIO</td>
<td>AOUT</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>PWM</td>
<td>P</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GPIO</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>SPI</td>
<td>S</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>GPIO</td>
<td>GPIO</td>
<td>CS</td>
<td>MOSI</td>
<td>MISO</td>
<td>SCK</td>
<td>GND</td>
</tr>
<tr>
<td>Touch</td>
<td>T</td>
<td>+3.3V</td>
<td>+5V</td>
<td>[UN]</td>
<td>YU</td>
<td>XL</td>
<td>YD</td>
<td>XR</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>UART</td>
<td>U</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>TX (G)</td>
<td>RX (G)</td>
<td>GPIO</td>
<td>[UN]</td>
<td>[UN]</td>
<td>[UN]</td>
<td>GND</td>
</tr>
<tr>
<td>LCD 1</td>
<td>R</td>
<td>+3.3V</td>
<td>+5V</td>
<td>LCD R0</td>
<td>LCD R1</td>
<td>LCD R2</td>
<td>LCD R3</td>
<td>LCD R4</td>
<td>LCD VSYNC</td>
<td>LCD HSYNC</td>
<td>GND</td>
</tr>
<tr>
<td>LCD 2</td>
<td>G</td>
<td>+3.3V</td>
<td>+5V</td>
<td>LCD G0</td>
<td>LCD G1</td>
<td>LCD G2</td>
<td>LCD G3</td>
<td>LCD G4</td>
<td>LCD G5</td>
<td>BACKLIGHT</td>
<td>GND</td>
</tr>
<tr>
<td>LCD 3</td>
<td>B</td>
<td>+3.3V</td>
<td>+5V</td>
<td>LCD B0</td>
<td>LCD B1</td>
<td>LCD B2</td>
<td>LCD B3</td>
<td>LCD B4</td>
<td>LCD EN</td>
<td>LCD CLK</td>
<td>GND</td>
</tr>
<tr>
<td>Manufacturer Specific</td>
<td>Z</td>
<td>+3.3V</td>
<td>+5V</td>
<td>[MS]</td>
<td>[MS]</td>
<td>[MS]</td>
<td>[MS]</td>
<td>[MS]</td>
<td>[MS]</td>
<td>[MS]</td>
<td>GND</td>
</tr>
<tr>
<td>DaisyLink Downstream*</td>
<td>*</td>
<td>+3.3V</td>
<td>+5V</td>
<td>GPIO!</td>
<td>GPIO</td>
<td>GPIO</td>
<td>[MS]</td>
<td>[MS]</td>
<td>[MS]</td>
<td>[MS]</td>
<td>GND</td>
</tr>
</tbody>
</table>

GPIO: A general-purpose digital input/output pin, operating at 3.3 Volts.

[UN]: Modules must not connect to this pin if using this socket type. Mainboards can support multiple socket types on one socket, as long as individual pin functionalities overlap in a compatible manner. A pin from one socket type can overlap with a [UN] pin of another.

* Interrupt-capable and software pull-up capable GPIO (the pull-up is switchable and in the range of 10,000 to 100,000 ohms).

* Socket type * should not appear on a mainboard, only on DaisyLink modules. The [MS] pins on this socket type can optionally support reflashing the firmware on the module.
Connecting a module to a mainboard

Match socket type letters when connecting modules to the mainboard.
The .NET Gadgeteer Platform

<table>
<thead>
<tr>
<th>Modular Hardware</th>
<th>Software Tools</th>
<th>Physical Design</th>
</tr>
</thead>
</table>

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

ButtonPressed, ButtonReleased, DebugPrintEnabled, Equals, GetHashCode, GetType, IsPressed, ToString
Based on the .NET **Micro** Framework:

- Subset of .NET
- Programming in C# and Visual Basic
- Interactive debugging

.NET Gadgeteer adds:

- Gadgeteer Core Libraries
- Visual Studio Graphical Designer
- Framework for mainboard and module drivers
A template for a Microsoft .NET Gadgeteer application allowing the user to specify a device design comprising a .NET Gadgeteer-compatible mainboard and modules, and write and debug C# software for that device.
void ProgramStarted()
{
    // Associate events with event-handling methods
    button.ButtonPressed += new Button.ButtonEventHandler(button_ButtonPressed);
    camera.PictureCaptured += new Camera.PictureCapturedEventHandler(camera_PictureCaptured);
}

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

void camera_PictureCaptured(Camera sender, Gt.Picture picture)
{
    // Show the picture on the display
    display.SimpleGraphics.DisplayImage(picture, 0, 0);

    // Save the picture to the SD card
    sdCard.GetStorageDevice().WriteFile("picture.bmp", picture.PictureData);
}
Hardware module driver wizard
The .NET Gadgeteer Platform

- Modular Hardware
- Software Tools
- Physical Design

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

    myButton.

    // Do one
    Debug.Print
}
```

- ButtonPressed
- ButtonReleased
- DebugPrintEnabled
- Equals
- GetHashCode
- GetType
- IsPressed
- ToString
Hardware design guidelines

The keep-out area should be clearly delimited in the silkcreen 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 silkcreen to make space for other labeling or silkcreen elements. Under no circumstances should you place components inside the keep-out area.

All mounting holes should be placed on a 5-mm grid, that is, the distance between adjacent holes should be a multiple of 5 mm, as shown in the following illustration.

Corners

Corners should be rounded, with a 7-mm-diameter curve that is concentric with a mounting hole’s keep out area, as shown in the following illustration.

If a corner does not include a mounting hole, the corner does not need to be rounded. However, we recommend that you maintain the same 7-mm rounding diameter for consistency.
Standardized mounting holes
3D models of hardware modules
Integration with 3D CAD (SolidWorks)
Parametric enclosure templates
Adding and positioning 3D models
Automatic mounting feature generation
Automatic cut-out generation
Laser-cut enclosure based on the *Jigsaw Box* template
3D-printed enclosure based on the *Project Box* template
Custom devices in ubicomp research

DIY Biology (Kuznetsov et al, DIS 2012), Telematic Dinner Party (Barden et al, DIS 2012), How to Nudge In Situ (Kalnikaitė et al, UbiComp 2011), PreHeat (Scott et al, UbiComp 2011), Serendipitous Displays (Helmes et al., Interact 2011), Sonic Mementos (Petrelli et al, CHI 2010)
Home heating using occupancy sensing & prediction

PreHeat (Scott et al, UbiComp 2011)
Form-factor iteration
Production and deployment of 50 devices
TEI’11 Studio: from sketch to 3d-printed demo
Getting .NET Gadgeteer out of the lab
Open source repository for project documentation, software and hardware designs

Introducing .NET Gadgeteer!

Microsoft .NET Gadgeteer is an open-source toolkit for building small electronic Framework and Visual Studio or Visual C# Express. .NET Gadgeteer combines th programming, solderless assembly of electronics with a kit of peripherals, and is construction using computer-aided design. This powerful combination allows or be iteratively designed, built and programmed in a matter of hours rather than c description of the platform: http://channel9.msdn.com/Blog/Clint/NET-Gadgeteer

The .NET Gadgeteer project is an open collaboration between Microsoft, hobbyist. This website is targeted at those interested in developing .NET Gadgeteer compatible hardware. For http://netgadgeteer.com/gadgeteer/. If you already have hardware and are looking for: visit the hardware vendor’s website.
http://netmf.com/gadgeteer/

Hardware index, example projects, forums
Huge thanks to the following Microsoft teams:

- Microsoft Research Connections who are supporting Gadgeteer outreach activities
- The MSR Advanced Prototyping team who built the Visual Studio Designer
- The Microsoft Garage members who beta tested Gadgeteer

And to interns and collaborators who have used the platform in their research.