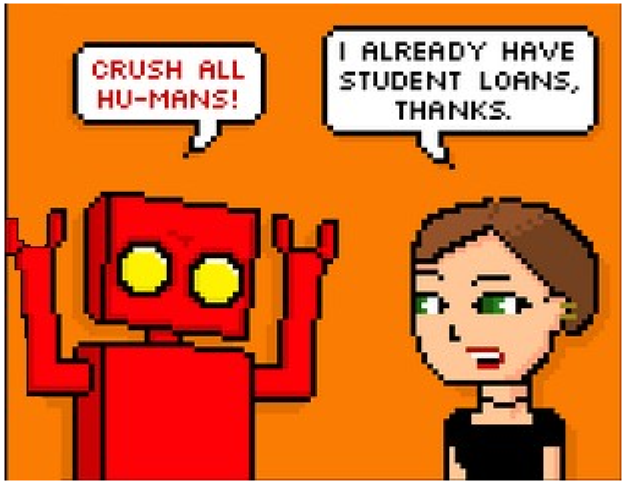
The Internet of Things and Embedded Development

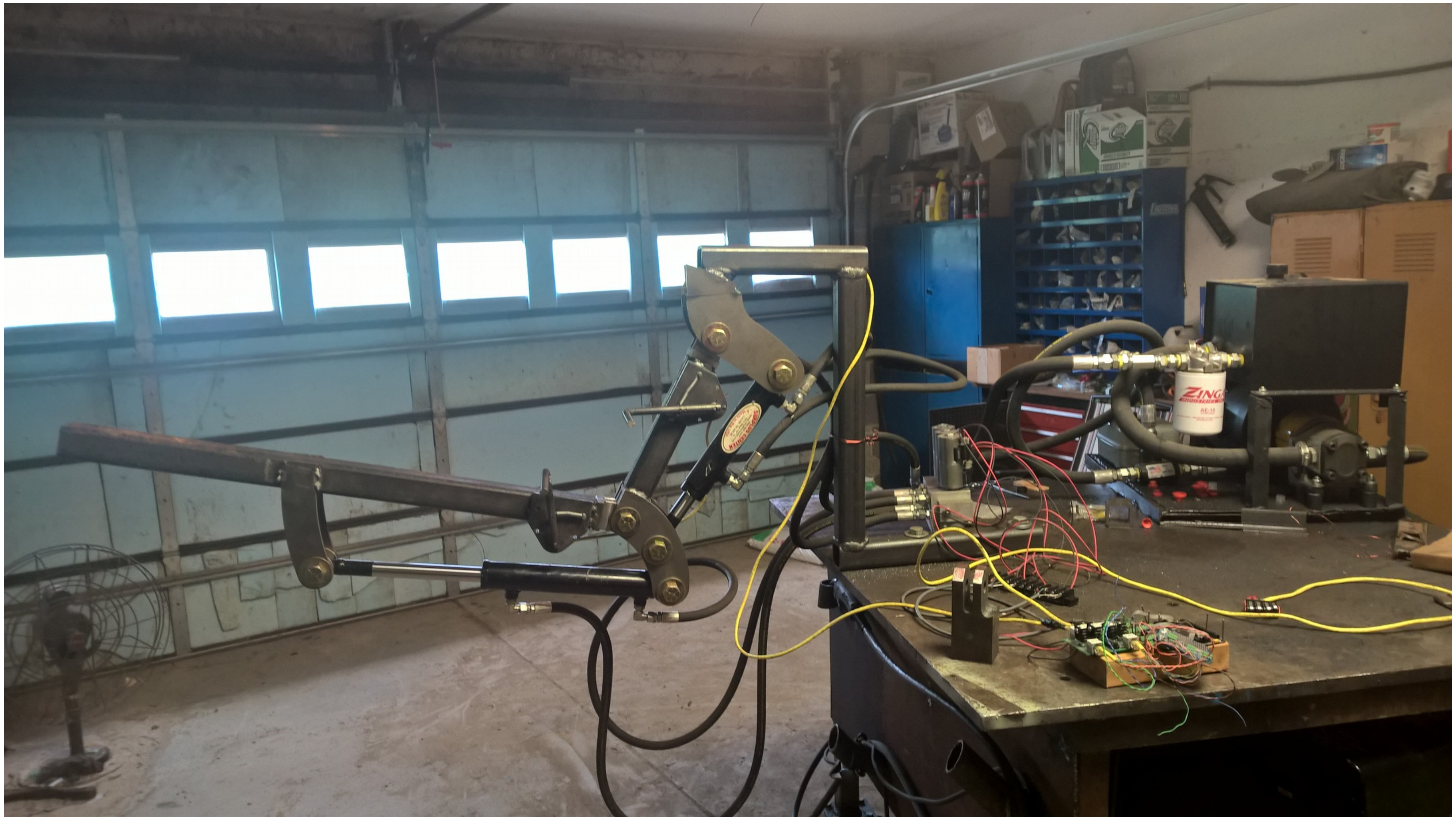
Or
How to Contribute to the Robot Apocalypse
with the .NET Micro Framework



Michael Phelps
Phelps Consulting.
Think software, not swimwear.

Who am I to talk about this?

Hydraulic exoskeleton arm prototype, 2014-2015

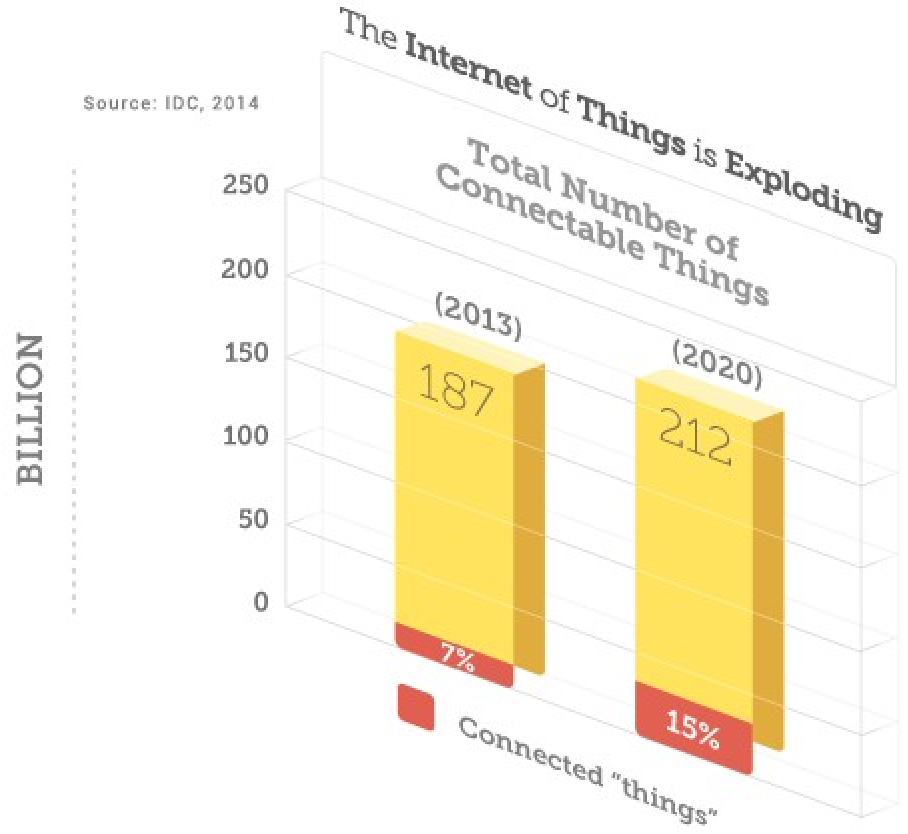


What's a “Thing”?

- A device with computer hardware and software, designed for a dedicated function.
- Interacts with the environment or user by acquiring data and acting on it in some way.

What is an “Internet of Things”

- An ecosystem of things that talk to other things via networks.
  - Over wifi, bluetooth, LAN
  - Local (Intranet)
  - Remote (Internet)
- Market is expanding



How to be an IoT Developer

- Have you ever used one of these?
  - Web service
  - Database
- Congratulations, you are already an IoT developer.

Thanks for coming!

IoT is really Embedded Development

- The meat of IoT is in the
  - Devices that do the work.
- Data transmission (the “Internet” part) simply adds transmission and persistence.
  - Analysis and aggregation of the (big) data

Device Platform – Full OS

- Runs a complete OS like Linux, Windows, Embedded/CE
- Raspberry Pi - Linux. Code in Python, C, C++, many others. Win10 soon!
- BeagleBoard – Linux. Code in BoneScript, a JS library in NodeJs.
- Lower performance, lots of overhead.
- Big resources.

Device Platform – Bare Metal

- Runs without OS
- You are responsible for basic services like memory allocation, file access, networking
- C/C++, Assembler
- Really hard
- High performance
- Industrial systems
- Tiny resources

Device Platform - Maker

- Offers thin firmware layer as an API to access device resources. Abstracts away much of the direct hardware access.
- Generally still offers direct register access in some way.
- Gaining traction due to ease of use.
- Medium resources

Maker Platforms

- Arduino – Everyone get “The Arduino Starter Kit”! Limited C/C++, but pretty easy. Great intro to electronics.
- Netduino – NetMF, Arduino compatible.
- GHI FEZ – NetMF, great vendor support, MS partner for Gadgeteer. Open source and proprietary boards. Production ready options available.
- Many others

Meet the.NET Micro Framework

- Created by MS around ~2004. Basis was the SPOT smartwatch. (Smart Personal Objects Technology)
- SPOT is dead, but .NetMF came out of it.
- For 64KB RAM or better.
- Limited *clone* of the full .NET framework
- C#, VB only.
- Not machine code.
  - Interpreted
  - no JIT
  - no unsafe code.
- Open source, part of .NET Foundation

Getting Started

- Get a development board
- Visual Studio
- NetMF SDK – version will be specific to the board you choose.
- Vendor specific extensions
- Device Firmware

- Each vendor has a getting started page that tells you what to install, and in what order.

Gadgeteer

- Rapid prototyping platform.
- Standardizes electrical connectors.
- Wrapper class libraries and designers for many devices.
- Hides the infinite loop, uses an event dispatcher.

Blinky

- Blinky is the embedded equivalent of “Hello World!”
- Tests:
  - Your dev environment works
  - The device can power on and initialize
  - You can deploy to the device/communicate
  - Basic hardware functionality
- Demo!

“I”s and “O”s

- There are some basic objects for getting input and sending output.
  - InputPort – Reads a binary input.
  - OutputPort – Sends a binary output.
  - AnalogInput – Reads an analog input (resolution is ADC dependent). Value is either an int or double between 0 and 1 inclusive.
  - PWM – Pulse Width Modulation. Method of simulating an analog output without a DAC.

Interrupts

- Interrupt Port – Fires event on input changes.
- Interrupts prevent the need to poll.
- Improve efficiency

Other Types of “I” and “O”

- Complicated messaging protocols are supported in software depending on your device.
  - I2C (Inter-Integrated Circuit) – One way chip communications on a single PCB.
  - SPI (Serial Peripheral Interface) – Two way chip communications on a single PCB.
  - CAN (Controller Area Network) – Multi master communication bus. Noise tolerant. Good for connecting devices over long distances (max 1000 meters) and in industrial environments.

More complicated demo

- Gadgeteer Designer
- Get SPOT IO objects and methods
- Get Gadgeteer IO objects and methods
- Show encapsulation of IO in a class.

- Demo!

Key Framework Features

- No Generics. No plans due to performance hit.
  - Lots of casting if you use the collection types.
  - Use raw arrays of the correct type.
- No Linq. Microlinq may help.
- Extension methods (may need to define ExtensionAttribute)
- Lambda syntax

NetMF is Not Real Time

- Real time (“Time Critical”) means:
  - Guaranteeing that inputs and outputs are handled based on a time constraint.
  - Generally accepted to be **ms** or **us** response time.
- There are always at least two NetMF threads
  - Application code (what you write)
  - Garbage collector (can preempt you any time it wants)
- Thread scheduler is simple 20ms time slices.
- Heavy GC or threading = lost data.
- GC and Threading Demos!

Faking Real Time

- GC can be avoided mostly like full framework.
  - Avoid boxing
  - Statelessness
  - Reusable object pools. Circular buffers are good.
  - Avoid strings (immutability)
  - Structs are treated like reference types!
- For a real example, search for “netmf quadcopter”.
- Lots of testing!

Testing? Failures? Who cares?

- What are the ramifications of code failure for these types of devices?
  - Toys/games
  - Appliances – like an alarm clock or coffee maker
  - Automated industrial machines
  - Machines with an operator/passenger, medical devices

Failures in Toys

- Nothing lost.
- Mad users.

Failures in appliances

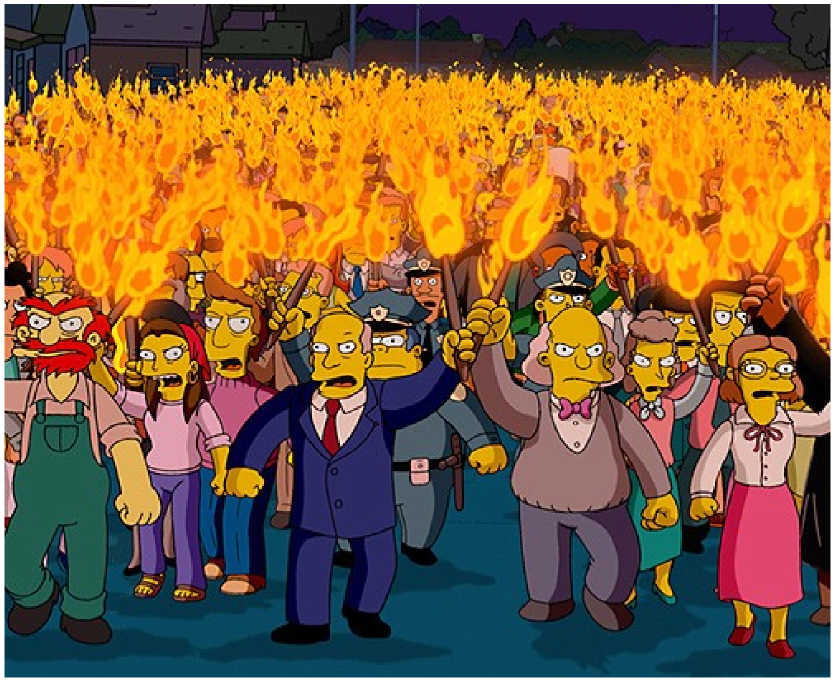
- Alarm clock – user doesn't wake up
- Coffee maker – user doesn't wake up
- Washing machine – property damage

- Mad users
- Small claims (but not always)

Failures in Industrial Machines

- Machine crash
- Production line stoppage
- Real time requirements may apply

- Mad shareholders
- Machine damage
- Lost products and materials
- Big lawsuits



Failures in Passenger Devices

- Machine/operator interference
- Medical devices – over/under dosage
- Aircraft - crashes
- Real time requirements may apply

- Injured or dead users
- Mad governmental authorities
- Mad insurance companies
- Mad public/watchdog groups
- Criminal court
- Prison

Security

- Security is as important as in hosted software
- Users (Hackers) have your device and your code.
- Security deficiencies in current devices
  - Some ATMs can flash firmware from a USB stick, allowing arbitrary code to be loaded.
  - Some insulin pumps have unprotected wireless interface. An attacker can control all settings, including dosage.
  - Vehicles with OnStar can be hacked via the diagnostic port, allowing remote control of throttle, brakes, locks, etc.

Methods for Testing

- On actual device (post production)
  - Testing delayed until after manufacturing.
  - Defect cost is high, potentially dangerous.
  - Feedback loop is long.
- Unit Testing
  - Short feedback
  - Nobody dies
  - Mock the device IO
  - Deploy tests as POST if possible

MFUnit

- Limited, but effective.
- Runs in the NetMF emulator.
- Note that GHI assemblies throw exceptions when used with MFUnit. Put code to be tested into a class library that is pure NetMF only.

- Demo!

Mocking & Dependency Injection

- InputPort, OutputPort, etc. are sealed, no interface
- Gadgeteer also sealed, no interface
- What to do?

MFMock

- Wraps core IO objects with interface layer
- Gives basis for dependency injection
- Mock inputs with multiple data samples
- Mock outputs with record of changes
- Allows 100% coverage for SPOT

- MFMock Demo!

Programming Recommendations for Non Trivial Projects

- Gadgeteer - startup and pin assignments.
- Use the Native SPOT IO objects.
- Use NetMF class library projects.
- Test with MFUnit and MFMock.
- Wrap other objects for mocking support.

Challenges for Nontrivial Devices

- There are many fields of discipline involved. You might need people for one or more of the following.
  - Electrical Engineering
  - Mechanical Engineering
  - Custom Domain Expertise
  - Programming
  - Manufacturing

Electrical Engineering

- Determine what ICs, passive, active components needed.
- PCB design
- FCC noise compliance
- Radio transmissions and antenna design
- Device interconnects and protocols
- Amplifiers
- Optical isolation & other device protections

Mechanical Engineering

- Linear motion
  - Hydraulics/Pneumatics
  - Solenoids
- Rotary motion
  - Motors (AC, DC, Stepper)
  - Servos
- Positional feedback
  - Proportional sensors
  - Encoders

Custom Domain Expertise

- Pace maker – bioelectrical signals
- Insulin pump – biochemistry
- Paint matching machine – color analysis
- Segway – balancing physics.
- Thermostat – HVAC and thermodynamics

Programming

- Device code
- Troubleshooting and debugging logs
- In-field software and firmware updates
- Cloud services
- On-premise services

Manufacturing

- PCB printing, population, soldering
- Device components and enclosure
- Assembly
- Packaging and shipping

Windows 10 IoT

- Windows 10 IoT Is part of Universal Windows Platform.
- Most of what we covered is still relevant.
- “Small Device” for W10 IoT is 256MB Ram, 2GB storage.
- Can run on Raspberry Pi 2 right now, in preview.
- New API To support IO.
- Full .NET capabilities in .NET Core. Meaning generics, Linq, etc.
- NetMF still around, continues to exist for smaller devices. Not intended to be supplanted by Win10 IoT.

Networking

- Connecting is trivial.
- Communicating is limited. Simpler is better.
- WebAPI can do heavy lifting for you.
- Consider authentication.
- Higher powered devices can use SSL.
- Demo!
  - Networking & Azure Web API
  - Client Application

Just the beginning

- Subjective opinion time!
- This market is huge. There is room for both embedded devs and enterprise devs.
- At the beginning of the market like
  - Smart phones in 2007.
  - Internet in early/mid 1990's.
  - Sliced bread in 1928.

The End

- For real this time.
- Questions now? Stick up your hand!

- Questions later?
  - @MikeVPhelps
  - mvphelps@gmail.com

- GitHub
  - https://github.com/mvphelps/MFMock
  - https://github.com/mvphelps/Presentations