Lecture 01
Introduction

CE346 – Microcontroller System Design

Branden Ghena – Spring 2026

Some slides borrowed from:
Josiah Hester (Northwestern), Prabal Dutta (UC Berkeley)

Northwestern

Welcome to CE346/CS346!

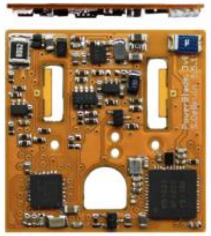
- Focus on hardware/software systems and their design
  - Hardware/Software co-design
    - How do you write software that interacts with hardware?
    - How do you choose hardware to support software needs?
  - Sensors and Sensing
    - What can sensors do and how do they work?
    - How do you write applications that sense the world?

Outline

- **Who and Why**
- Embedded Systems
  - Microcontrollers
- Course Overview
- Class Hardware
- Project Ideas

Branden Ghena (he/him)

- Assistant Faculty of Instruction
- Education
  - Undergrad: Michigan Tech
  - Master's: University of Michigan
  - PhD: University of California, Berkeley
- Research
  - Resource-constrained sensing systems
  - Low-energy wireless networks
  - Embedded operating systems: Tock
- Teaching
  - Computer Systems
    - CS213: Intro to Computer Systems
    - CS343: Operating Systems
    - CE346: Microprocessor System Design
    - CS433: Wireless Protocols for the IoT



Tock

Things I love

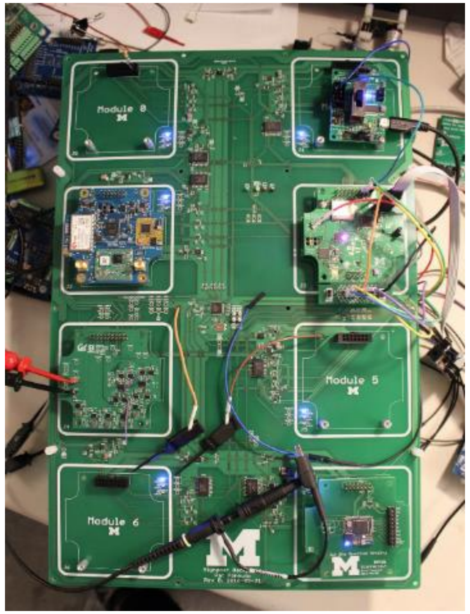
Cosmere

Frieren
Beyond Journey's End

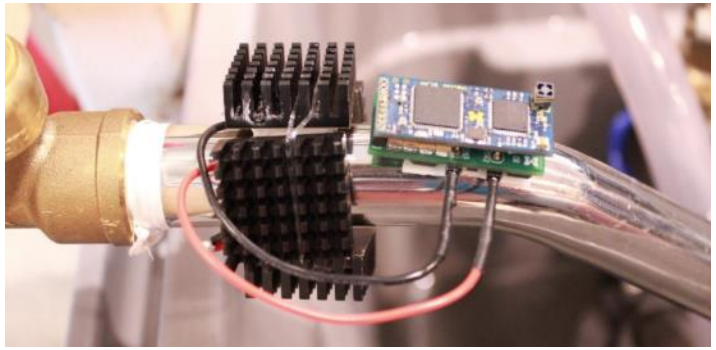
twitch

Critical Role

Research area: resource-constrained embedded systems



- Most interesting to me: the interfaces
  - Hardware and software
  - Applications and OS
  - Communication



Faculty: now I can choose what to teach!

- Goal: provide classes that teach more advanced embedded systems topics
  - Hopefully, generally useful to other nearby domains of CS and ECE too!

- Result: this course!
  - Course goal: introduce students to hardware-software interactions
    - Practical hands-on experience with microcontrollers and sensors
    - Open-ended project where students can choose their specific focus

Today’s Goals

- What are the goals of this course?
- Why do I think embedded systems are so important?
- How is the course going to operate?
- Discuss hardware used for the course and some project ideas.

Asking questions, four ways

1. You can always ask questions during lecture!
   - I’ll let you know if I need to move on for now and answer you after class

2. We’ll take breaks during lecture
   - I’ll pause after each break to see if any questions came up

3. I will hang out after class for questions
   - Plenty of time to answer everyone

4. You can always ask questions on Ed Discussion too

   The class message board app

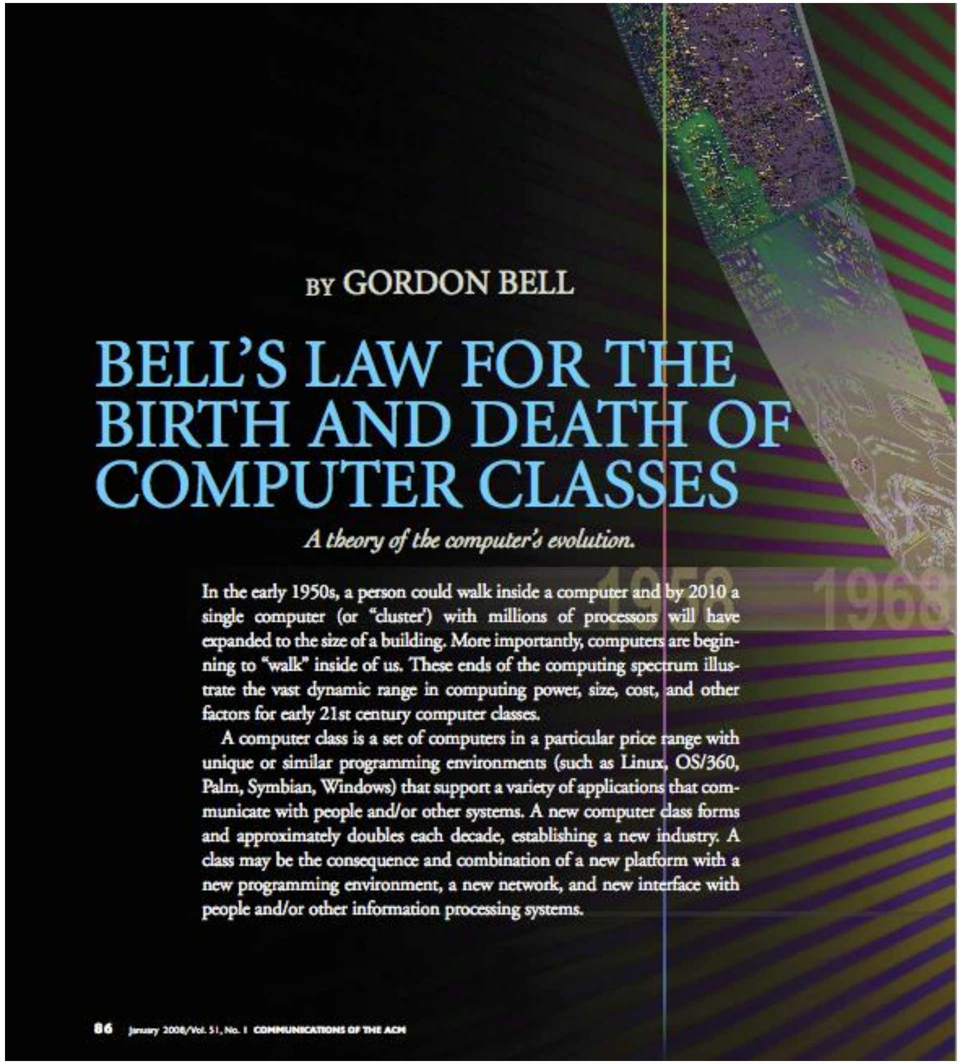
Outline

- Who and Why
- **Embedded Systems**
  - Microcontrollers
- Course Overview
- Class Hardware
- Project Ideas

Bell’s Law: A new computer class every decade

“Roughly every decade a new, lower priced computer class forms based on a new programming platform, network, and interface resulting in new usage and the establishment of a new industry.”

- Gordon Bell [1972,2008]



BY GORDON BELL

BELL’S LAW FOR THE BIRTH AND DEATH OF COMPUTER CLASSES

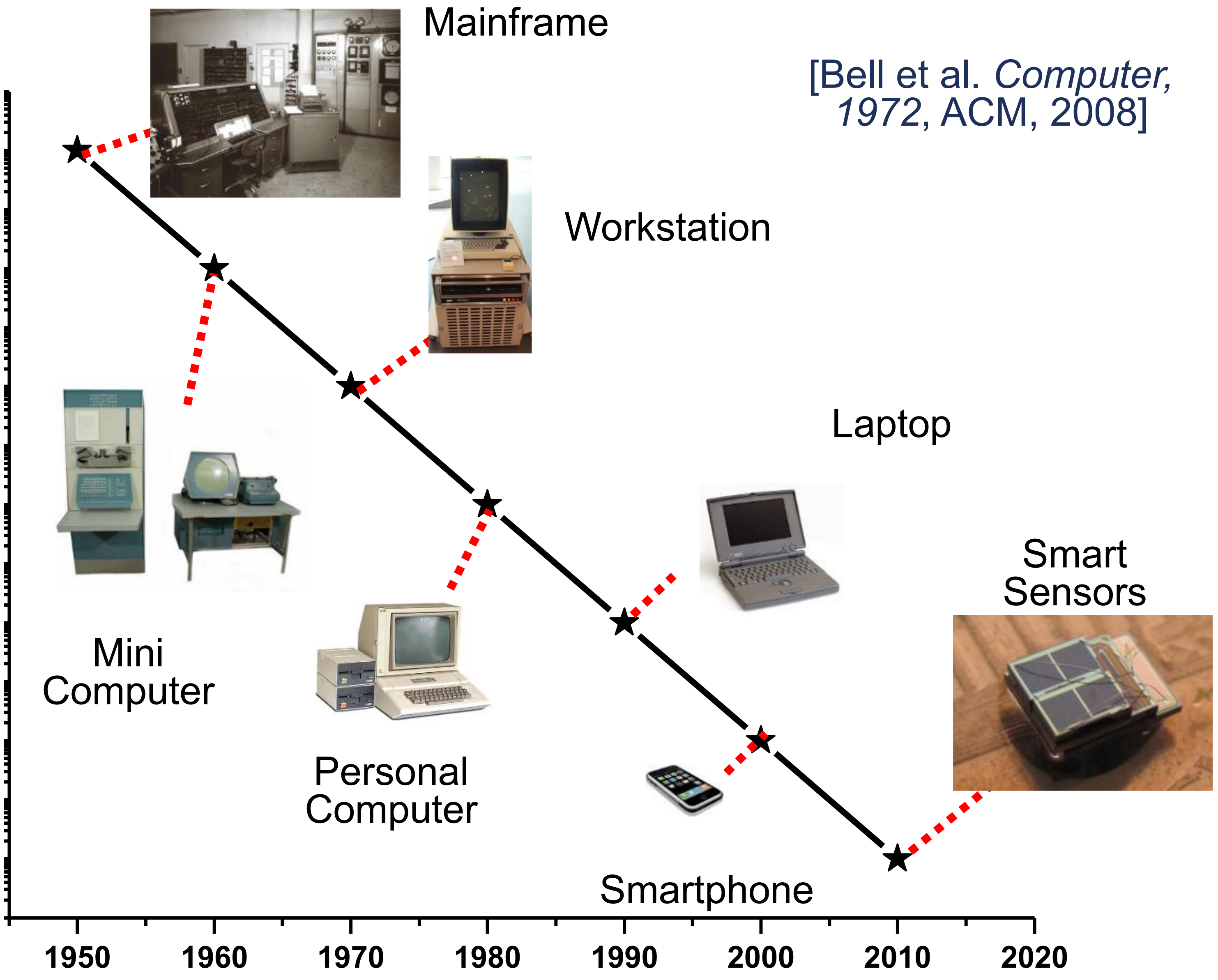
A theory of the computer’s evolution.

In the early 1950s, a person could walk inside a computer and by 2010 a single computer (or “cluster”) with millions of processors will have expanded to the size of a building. More importantly, computers are beginning to “walk” inside of us. These ends of the computing spectrum illustrate the vast dynamic range in computing power, size, cost, and other factors for early 21st century computer classes.

A computer class is a set of computers in a particular price range with unique or similar programming environments (such as Linux, OS/360, Palm, Symbian, Windows) that support a variety of applications that communicate with people and/or other systems. A new computer class forms and approximately doubles each decade, establishing a new industry. A class may be the consequence and combination of a new platform with a new programming environment, a new network, and new interface with people and/or other information processing systems.

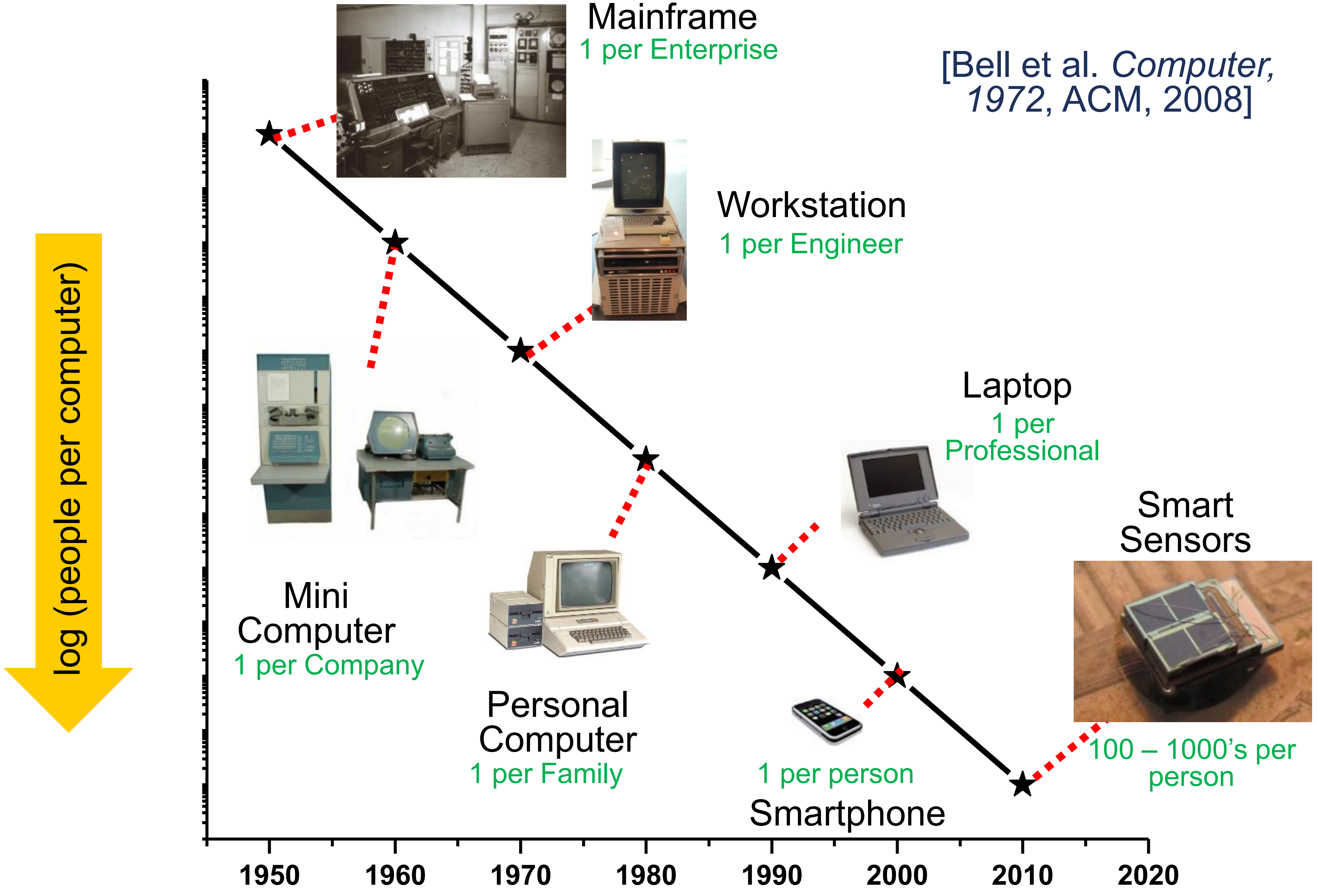
86 January 2008/Vol. 51, No. 1 COMMUNICATIONS OF THE ACM

Classes of computation

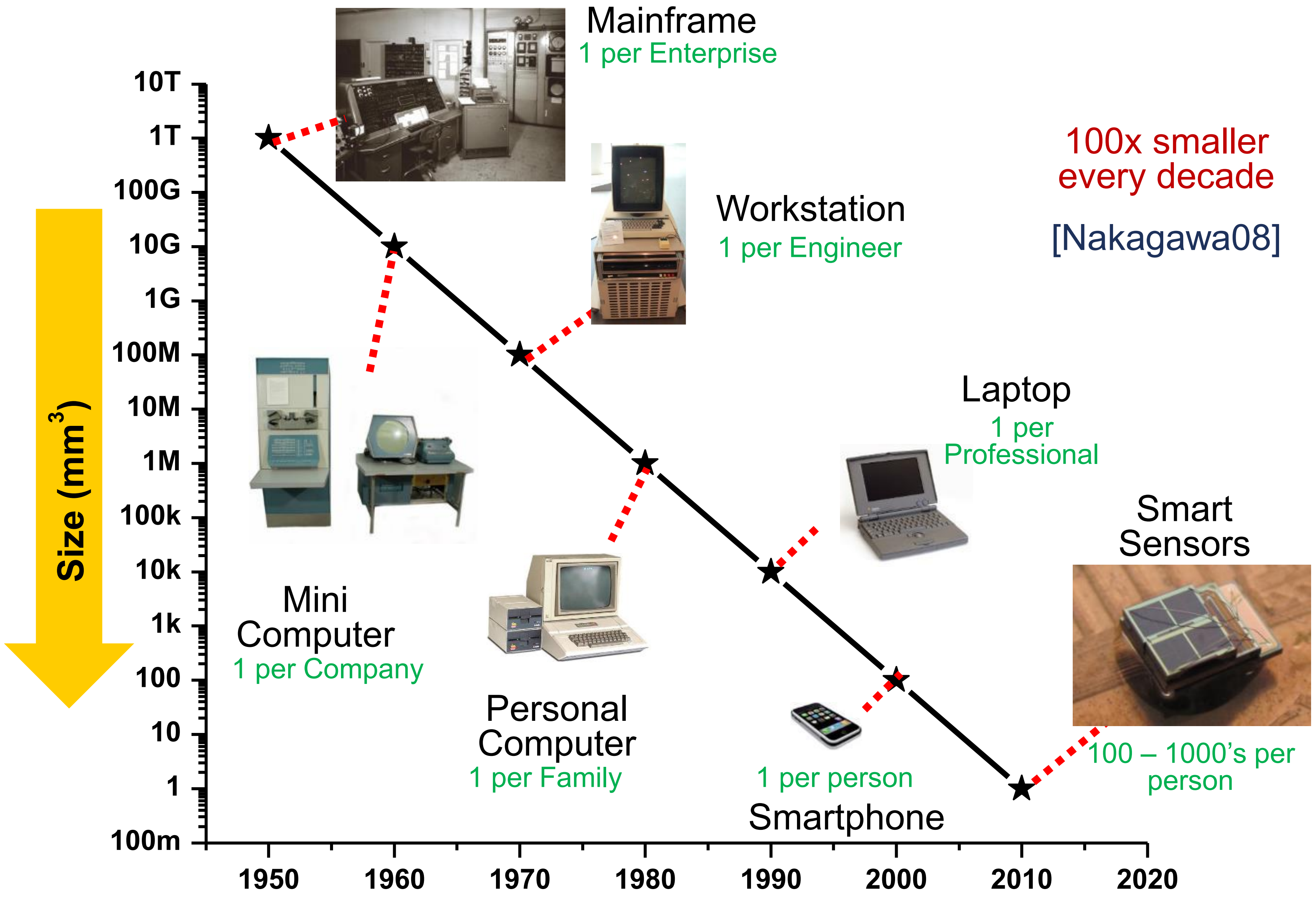


[Bell et al. *Computer, 1972*, ACM, 2008]

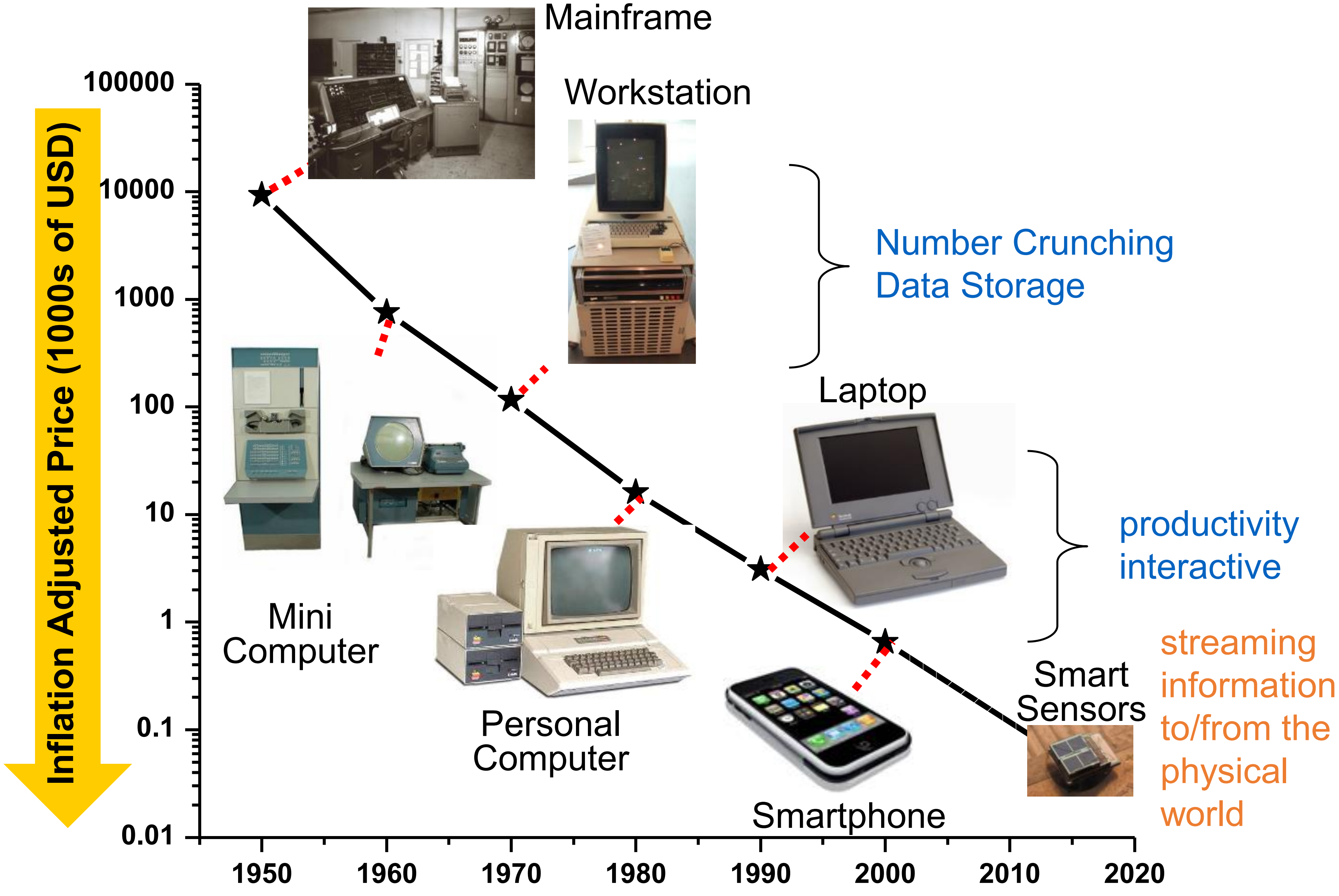
Number of computers per person grows over time



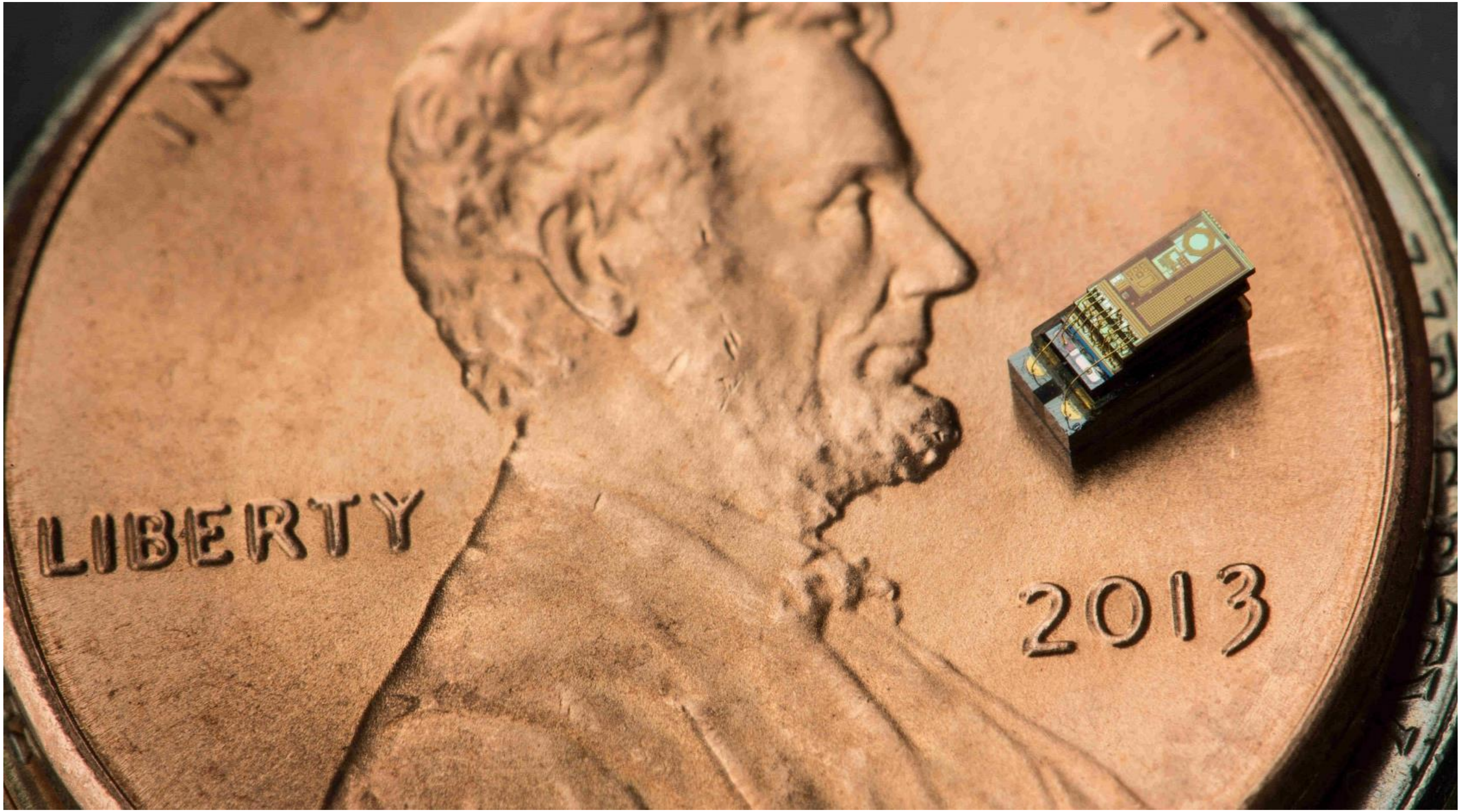
Computer volume shrinks by 100x every decade



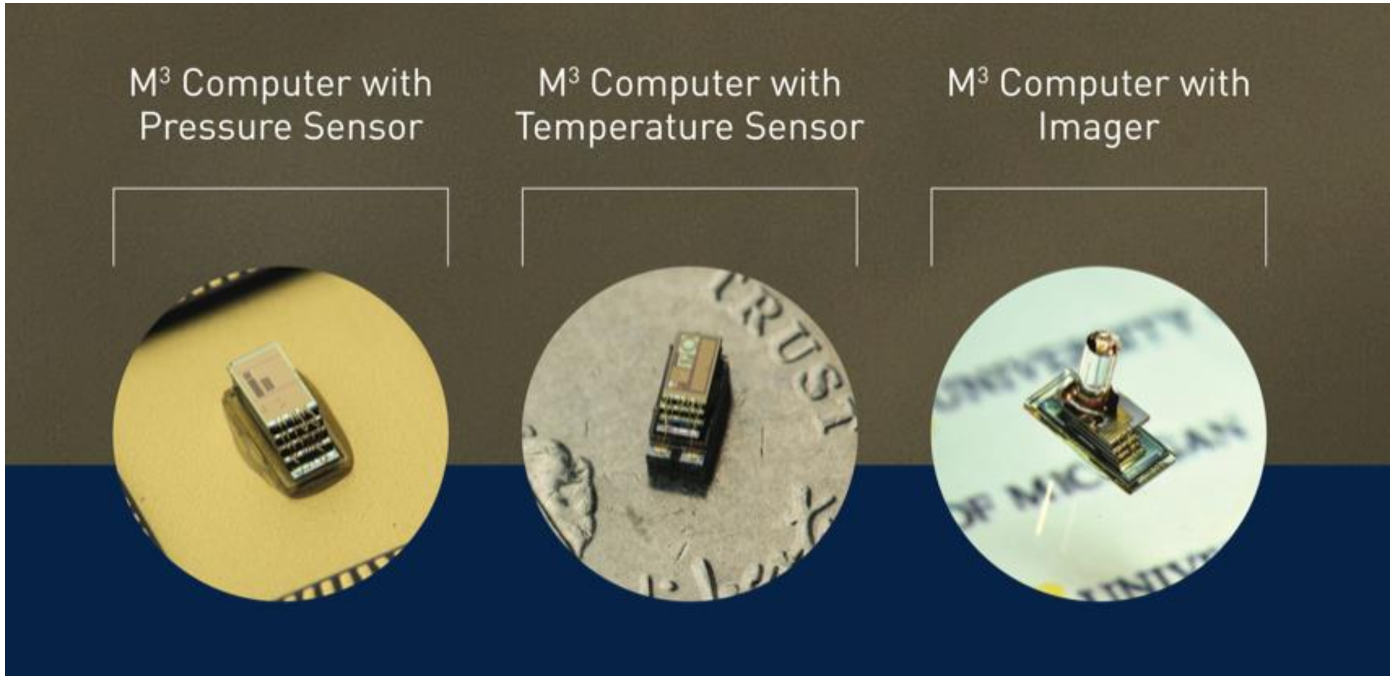
Price falls dramatically, enabling new applications

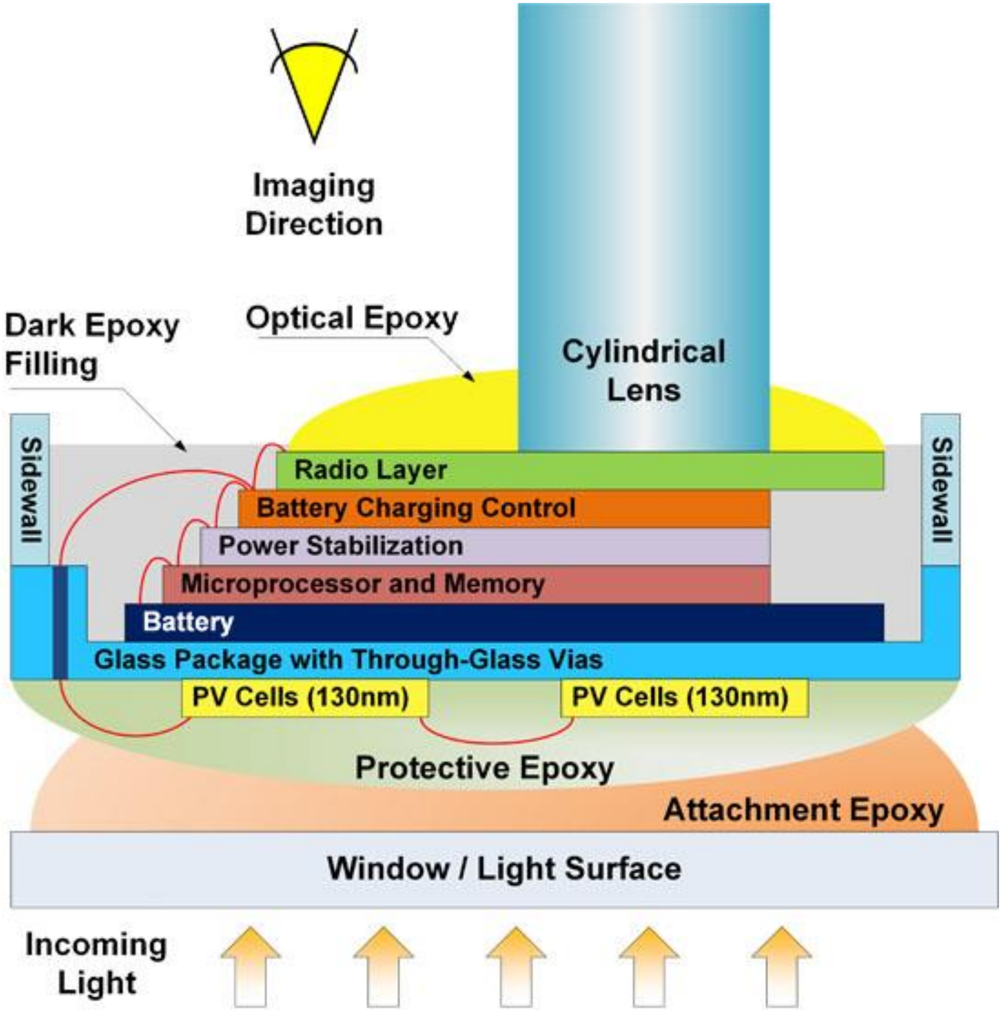


Michigan Micro Mote (2015)



M^3 Use cases and design





https://ece.engin.umich.edu/stories/michigan-micro-mote-m3-makes-history-as-the-worlds-smallest-computer

Key idea: computers can be built into other things

- Computers have become small enough that they don’t need their own rooms or even their own boxes to house them
- They can be built into other things we use in everyday life

What is an embedded system?

- A computer built into a device
  <u>such that the *device* is interacted with</u>, **not** the computer
  - E.g., not a desktop or laptop
  - (although many of those deal with overlapping hardware/software issues)

- Includes many domains
  - Robotics
  - Industrial processes
  - Smart home
  - Smart city
  - Wearables and health sensing
  - Internet of Things

Discussion: identify some embedded systems

- What devices that you might not usually consider as computers actually have embedded computers in them?
  - Talk with the others around you
  - Goal: come up some unique ideas

  - We’ll share ideas with the class afterwards

Thought experiment: high-capability computing

- What if the smart lightbulb was powered by an entire desktop?
  - 16-core x86-64 processor, 64 GB RAM, 1 TB SSD

- Could that still be an embedded system?

- Why don’t we see that in practice?

Thought experiment: high-capability computing

- What if the smart lightbulb was powered by an entire desktop?
  - 16-core x86-64 processor, 64 GB RAM, 1 TB SSD

- Could that still be an embedded system?
  - **Yes**

- Why don’t we see that in practice?
  - **Cost**

Trend: embedded computers instead of custom hardware

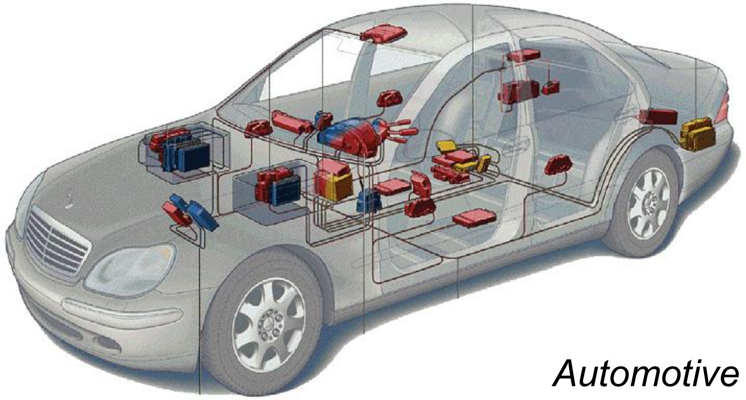
- Some embedded devices could be a state machine in custom hardware instead

- However, computers are increasingly common in those cases
    1. Embedded computers are increasingly cheap
    2. More software developers than hardware developers

- Example: thermostats existed without computers
    - But are cheaper to make *with* computers these days

Related area: Cyber-Physical Systems

- Systems that are part computational and part real-world
  - Example: autonomous vehicles
- Combines multiple fields to handle this problem
  - Embedded Systems
  - Electronics
  - Controls
  - Software Engineering
  - Computer Theory



Automotive

Related area: Internet-of-Things devices

- Device + Computation + Networking

What makes resource-constrained embedded systems interesting?

- Focus on the real world
  - You can actually see the purpose and effects of your applications
  - Easily explainable to non-engineer humans

- Challenging limitations
  - Limited memory and processing
  - Energy concerns

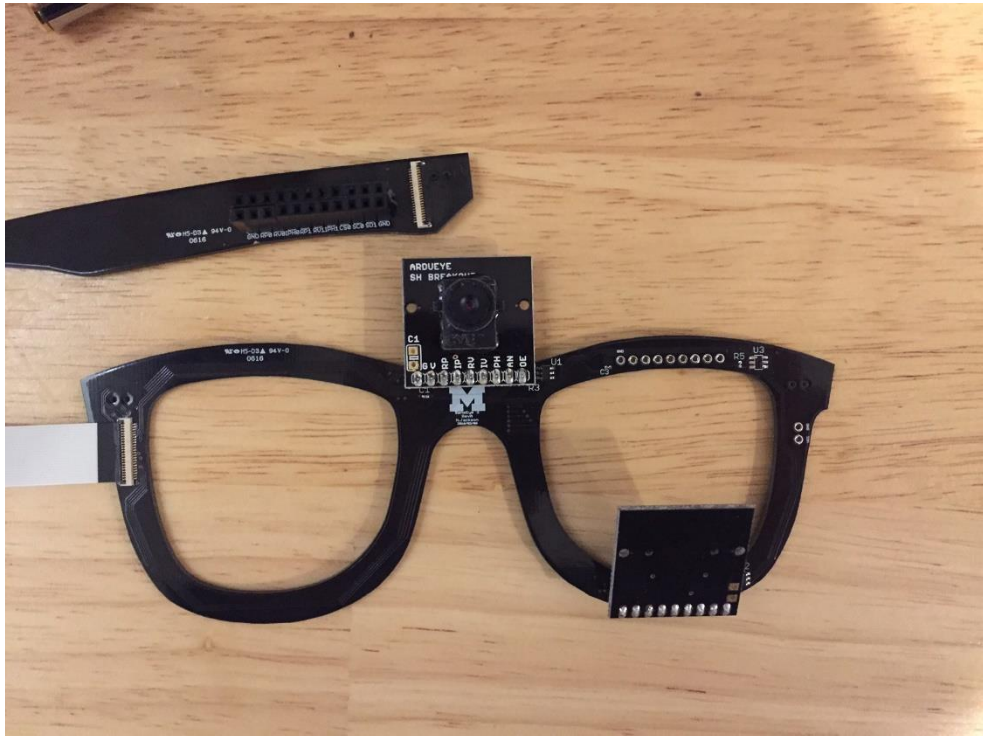
What makes resource-constrained embedded systems frustrating?

- Challenging limitations
  - Limited memory and processing
  - Energy concerns

- Full-stack development means problems could be *anywhere*
  - Hardware problems
  - Firmware problems
  - Software problems

What makes resource-constrained embedded systems frustrating?

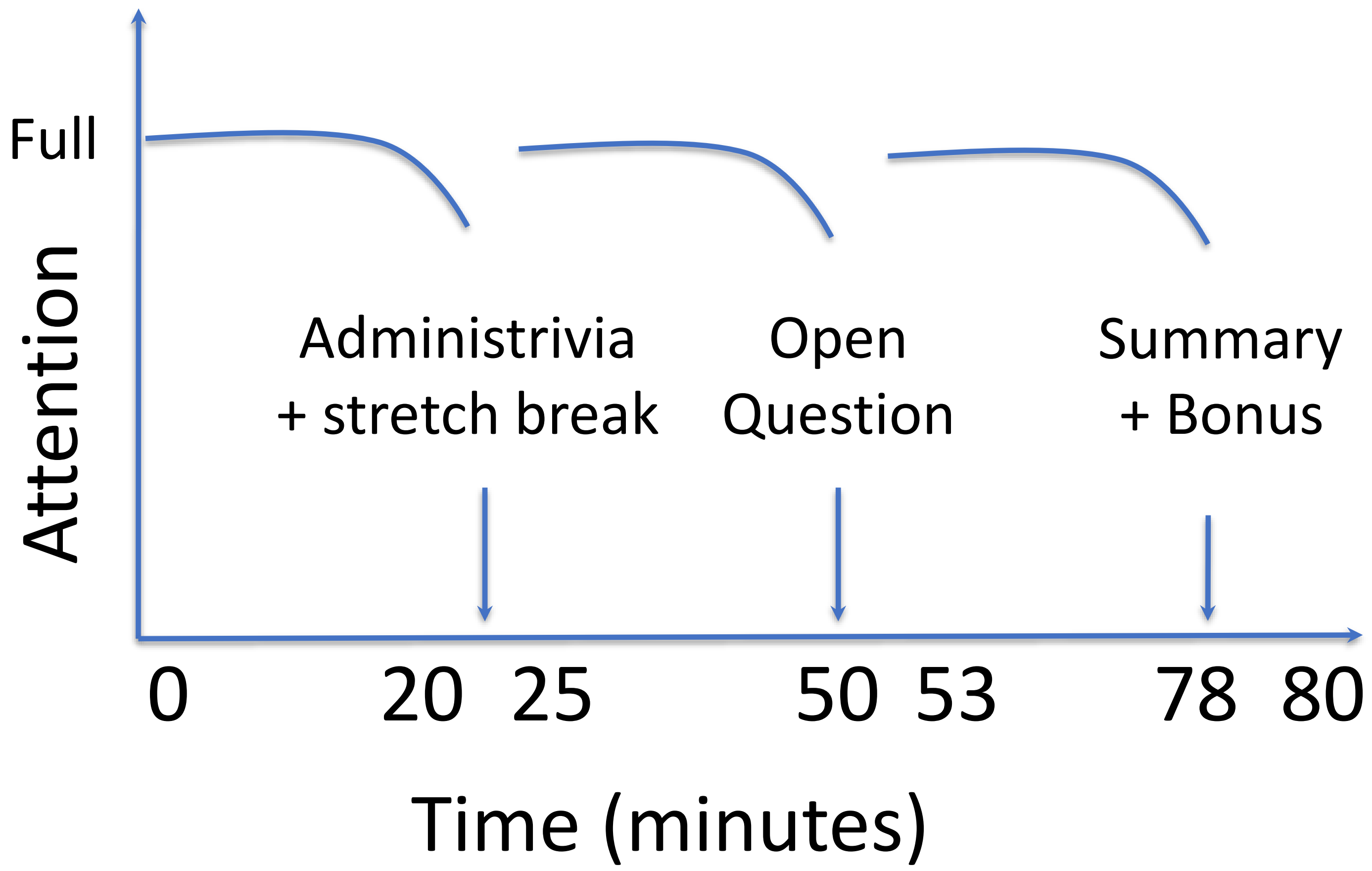
- Example: my first grad project, eye-tracking glasses
  - Captured images of eyes and world, display on a nearby desktop
  - System stack-up:
    - Camera
      -> Analog-to-Digital Converter
      -> FPGA (hardware to manage camera)
      -> Linux driver
      -> Linux app
      -> Network
      -> Visualizer app



  - One morning, I came to lab and nothing worked…

  - This WILL happen to you

Break + Architecture of a lecture



Outline

- Who and Why
- **Embedded Systems**
  - **Microcontrollers**
- Course Overview
- Class Hardware
- Project Ideas

What’s inside a computer?

- Macbook Air (circa 2013, 2^{nd} gen)
  - Modern computers are mostly similar

  - Intel Core i5 processor
  - 128 GB Flash storage
  - 4 GB DDR3 RAM
  - 1440x900 pixel display

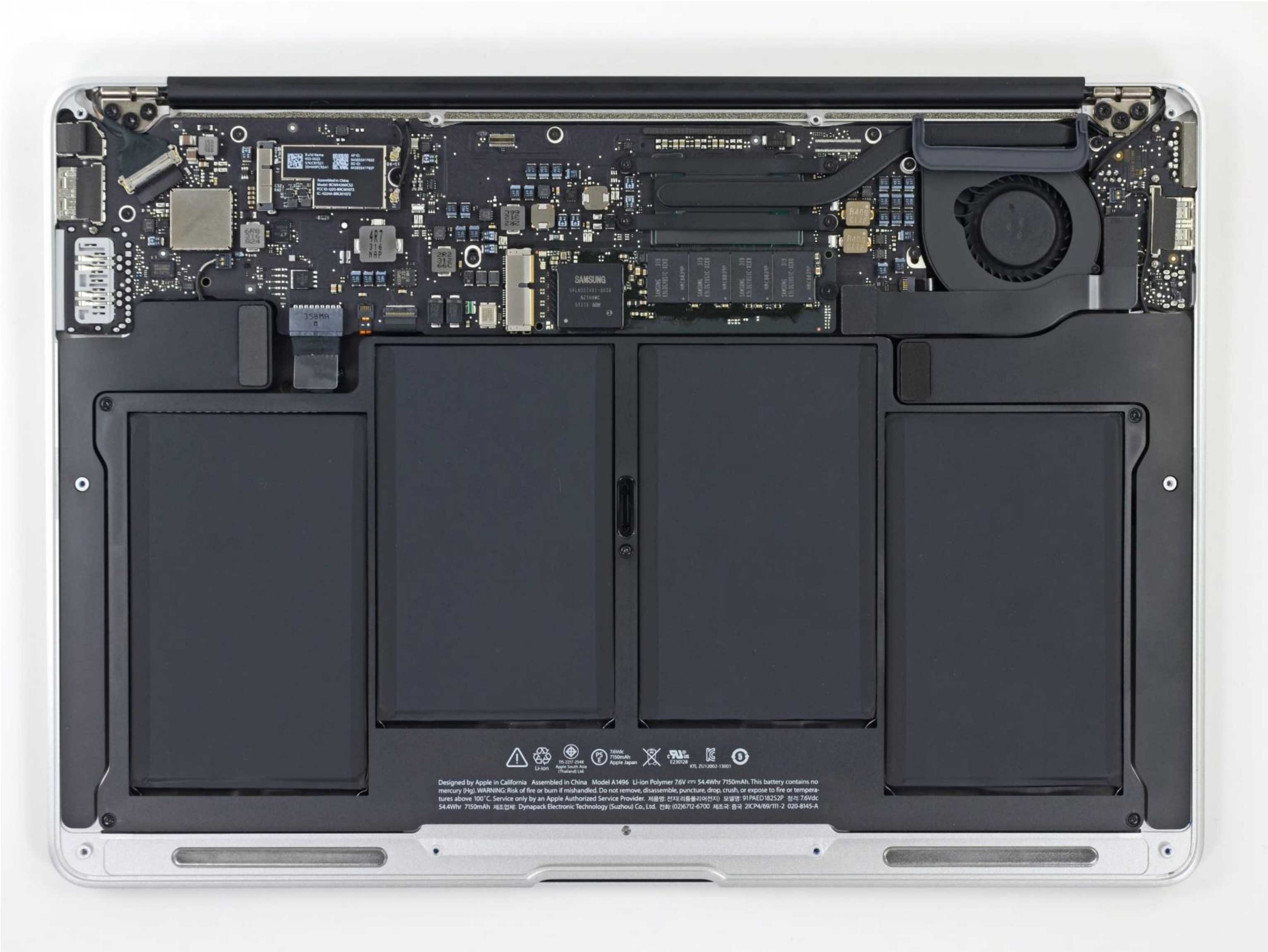
- Picked because there is a teardown I like of an embedded device from the same year

https://www.ifixit.com/Teardown/MacBook+Air+13-Inch+Mid+2013+Teardown/15042

Unscrewing the bottom cover

- Top half is the motherboard
  - Holds and connects all the parts of the computer

- Bottom half are battery packs



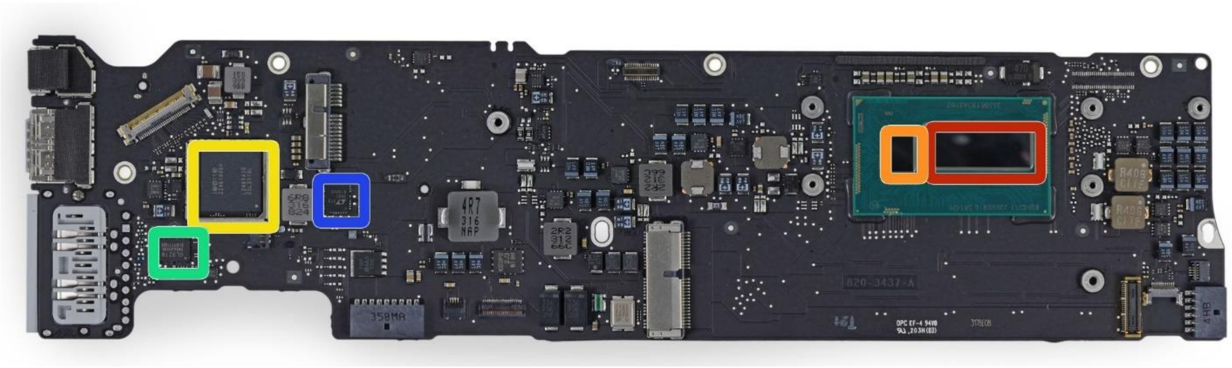
Non-volatile long-term storage: SSD

- The SSD is a module that connects through PCIe

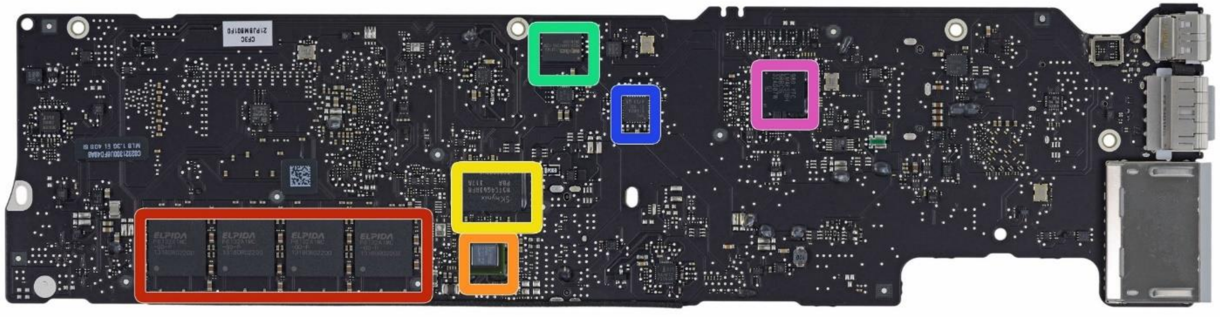
- In modern laptops the SSD is often soldered directly onto the motherboard

Investigating the motherboard

Front



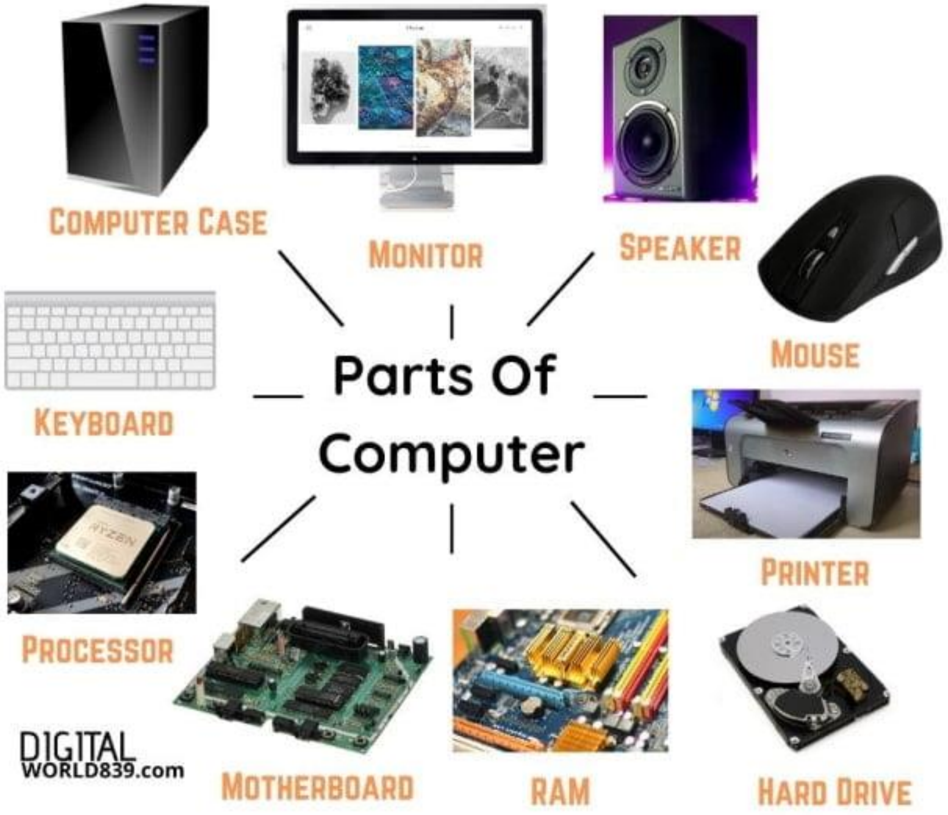
Back



- Red (front, right)
  - Intel core i5 processor
- Red (bottom, left)
  - Elpida LPDDR3 RAM, 4 GB
- Other stuff are mostly radios, USB/Thunderbolt controllers, and power supply

Generalizing computer design

- Computers *usually* need
  - Processor
  - Memory (RAM)
  - Storage (Flash/SSD)
  - External communication
    - USB, Thunderbolt, SATA, HDMI, WiFi
  - Power management
    - Maybe batteries and charging
  - Something to connect it all: motherboard



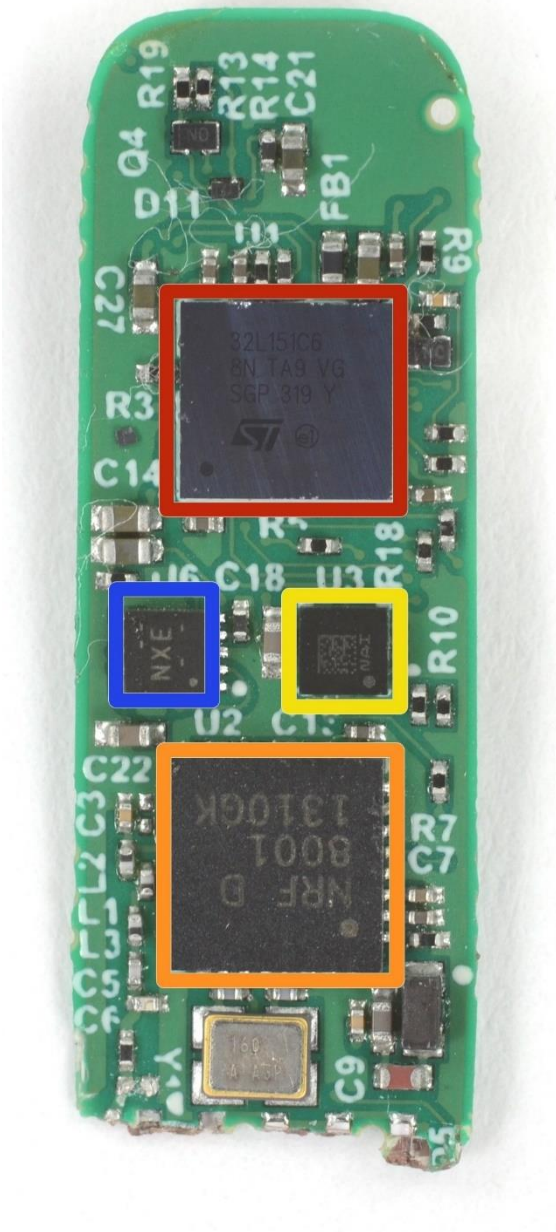
What's inside a Fitbit?

- Fitbit flex (circa 2013)

- Features
  - Counts your steps
  - Reports via Bluetooth Low Energy
  - Lights up some LEDs based on your goals
  - Vibrates when its battery is low

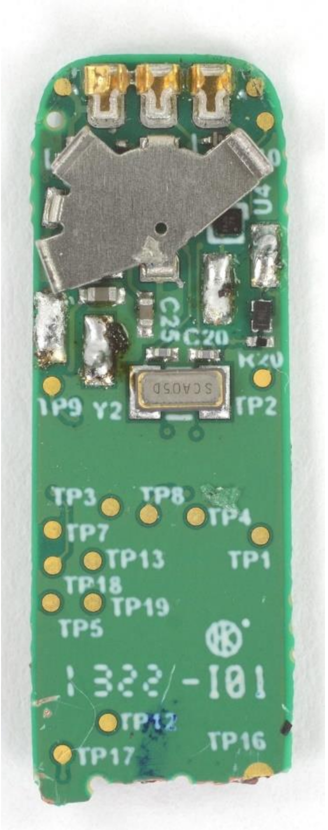
https://www.ifixit.com/Teardown/Fitbit+Flex+Teardown/16050

Fitbit teardown



Fitbit circuit board front

The back is uninteresting



- Red (top)
  - STMicro 32L151C6 Microcontroller
- Blue (left)
  - TI BQ24040 Battery Charger
- Yellow (right)
  - STMicro LIS2DH Accelerometer
- Orange (bottom)
  - Nordic nRF8001 Bluetooth Low Energy Radio

Fitbit as a computer

- Computers *usually* need
  - Processor
  - Memory (RAM)
  - Storage (Flash/SSD)
  - External communication
    - USB, Thunderbolt, SATA, HDMI, WiFi
  - Power management
    - Maybe batteries and charging
  - Something to connect it all: motherboard

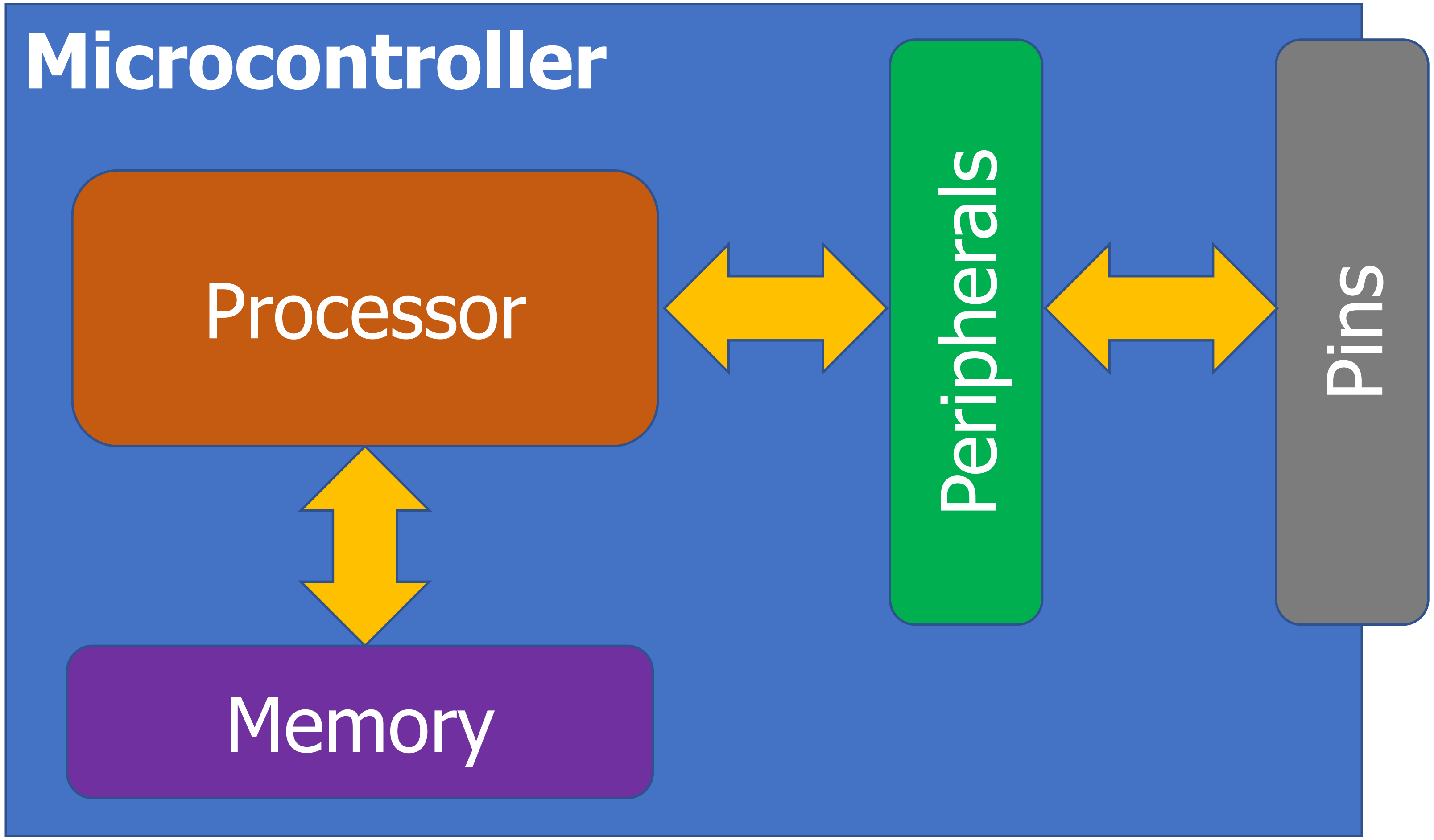
- Microcontroller
- Bluetooth radio
- Vibratory motor
- Battery and power management
- Circuit board

Sidebar: you could make a Fitbit yourself

- All those parts are commercially available, so you could make your own Fitbit if you wanted to

1. Make a circuit board
2. Buy those parts and solder them on
3. Write some embedded software
4. Make a plastic case
5. … build a big software backend for everything
6. You've got a Fitbit!

Generic microcontroller block diagram



Peripherals

- Hardware modules that perform some action
- Common examples
  - Control digital input and output pins
  - Read analog inputs
  - Send messages over various simple wire protocols (UART, SPI, I2C)
  - Set and check timers
- Less common examples
  - Cryptography accelerators
  - Complicated wire protocols (USB, CAN)
  - Wireless radios (BLE, 802.15.4, WiFi)
- We'll spend most of class learning various peripherals

How is a microcontroller different?

- A very constrained computer
  - Simple processor
    - 16 or 32 bits (usually 32-bit these days)
    - Processor speed in MHz
    - Single core, pipelined processor
    - No cache, or maybe a very small instruction cache
  - Memory measured in kB
    - Code executes right from read-only Flash (which is part of the address space)
  - Sometimes no OS support at all
    - "bare-metal" programming
  - More focus on peripherals for interacting with the world

Break + Question

- Why do laptops/desktops use external memory chips instead of building it into the processor (like a microcontroller)?

Break + Question

- Why do laptops/desktops use external memory chips instead of building it into the processor (like a microcontroller)?
  - Flexibility
    - If it's built into the processor, you're limited to whatever they picked
  - Fabrication issues
    - DRAM and CMOS gates are somewhat different manufacturing processes (affects yield and costs)
    - DRAM takes up a lot of space (multiple chips for many GBs of RAM)
  - Some systems are doing this
    - Apple M4 includes separate RAM silicon dies in the same processor package

Outline

- Who and Why
- Embedded Systems
  - Microcontrollers
- **Course Overview**
- Class Hardware
- Project Ideas

Course Staff and Office Hours

- Four PMs who previously took the class
  - Satviki Madaan
  - Konrad Lukasiewicz
  - Katie Jiang
  - Jayden Wool

  - They will help out during labs and also provide lab office hours

- Office Hours: TBD
  - We'll post a schedule soonish (should be M, T, W, and R)
  - Also by request! (especially during projects)

Course details – how to learn stuff

- Lecture: Tuesdays and Thursdays 3:30-4:50pm
  - Tech LR4

- Provides background on everything we'll be doing in labs
  - So stay up-to-date on them!
  - Lectures are automatically recorded so you can review them
  - Slides are posted to the Canvas homepage right before class

- No textbook for this class
  - Nobody seems to write a good one
  - The datasheet for our microcontroller (nRF52833) will be important though!

Asking Questions

- Class and office hours are always an option!
  - We can do extra questions right after class too
- Ed Discussion:
  - Post questions
  - Answer each other’s questions
  - Find posts from the course staff
  - Post private info just to course staff
- Post on Ed – do NOT send me emails
  - Messages are kept in one place and stay “unanswered”
  - You can post “Private” to just go to course staff
    - Use that feature to request office hour appointments if desired
    - Or to tell me that you’re sick and can’t attend lab

Course grade components

- 42% Labs
  - 6 labs at ~7% each
  - Guided exploration of course concepts
  - Staff gives checkoffs as you complete parts

- 20% Quizzes
  - Four timed quizzes at 5% each
  - Covers lecture material from last two weeks
  - In-class at the end of class, schedule on Canvas

- 38% Final Project
  - Open-ended group project (will explain in a minute)

Class lab sessions

- Lab: Fridays 1:00-2:50pm OR 3:00-4:50pm, Frances Searle 3220
  - Mandatory attendance for these (part of your lab grade)
  - Let me know ASAP if you're sick and will miss
- Labs start next week Friday and are weekly from there
  - Six labs total
  - When labs run out, I'll use the time for project meetings with groups
  - **Optional lab this Friday** for software setup
- Warning: labs won't usually be finished during the lab sessions
  - You'll have to work on them on your own time too
  - We'll have office hours for checkoffs

Labs

1. MMIO and Interrupts
2. Virtual Timers
3. LED Matrix
4. Breadboarding
5. Audio Input/Output
6. I2C Accelerometer/Magnetometer

- Labs will be partner work
  - You choose, but different partner each week
  - MUST work with a partner

- Due one week from start of lab
  - You'll need to get checkoffs in lab or during office hours

Show up to lab session on time

- Labs start right away forming partnerships and starting work

- So showing up late is a problem
  - And I will deduct lab points for it if I have to

Quizzes

- In-class, on-paper, closed notes quizzes
  - Usually about 15 minutes and held at the end of lecture
- Cover the last two weeks worth of material (give or take)
  - So make sure you're up-to-date on what we're talking about
- First quiz is Tuesday, April 14th (third week of classes)

Final projects

- Opportunity for you to apply your interests to this course
  - In groups of 2-3 students (maybe 4 for a really big idea)
  - Groups of three is encouraged

- Demonstrate course knowledge through any application
  - Microbit (99% required) in our C software system (99.9% required)
  - Various hardware I'll have on hand
  - Decent per-team budget for purchasing additional stuff
    (usually around $40 per person in team)

Project Logistics

- Week 4: Proposals due
  - I’ll get feedback to teams I’m concerned about
- Week 6: Project Design Presentations
  - Short presentations in class about your proposed project and design
  - Chance to give each other useful feedback about how to proceed
- Week 8-10: Labs are done and Fridays are used for project office hours
  - And a required project check-in, probably week 9
- Exam Week: Live project demos!!
  - Public demo session (invite friends!)
  - Tuesday of exam week (usually half of the class at a time for a 2-3 hour block)
  - **Required attendance for project demos**

Let’s talk about Generative AI

- Generative AI tools **MAY NOT** be used to develop **LAB** code

- Rationale:
  1. Want you to practice working on systems code
  2. Want you to be **significantly better** than AI tools
  3. We will help you complete the labs (in lab session and office hours)

- You may use AI tools for studying and understanding course material outside of labs
- You may use AI tools for the final project **as long as the source is properly documented**

Break + Details about who is in the class

- 58 total students
  - Grad: 6
  - Undergrad: 52

- Majors
  - Computer Science: 39
  - Computer Engineering: 11
  - Electrical Engineering: 8
  - Mechanical Engineering: 4
  - Data Science: 1
  - Music: 2

- You don’t all know the same things, and that’s okay!!

- This class thrives on a variety of backgrounds
  - Software
  - Hardware
  - Data Interpretation
  - Applications

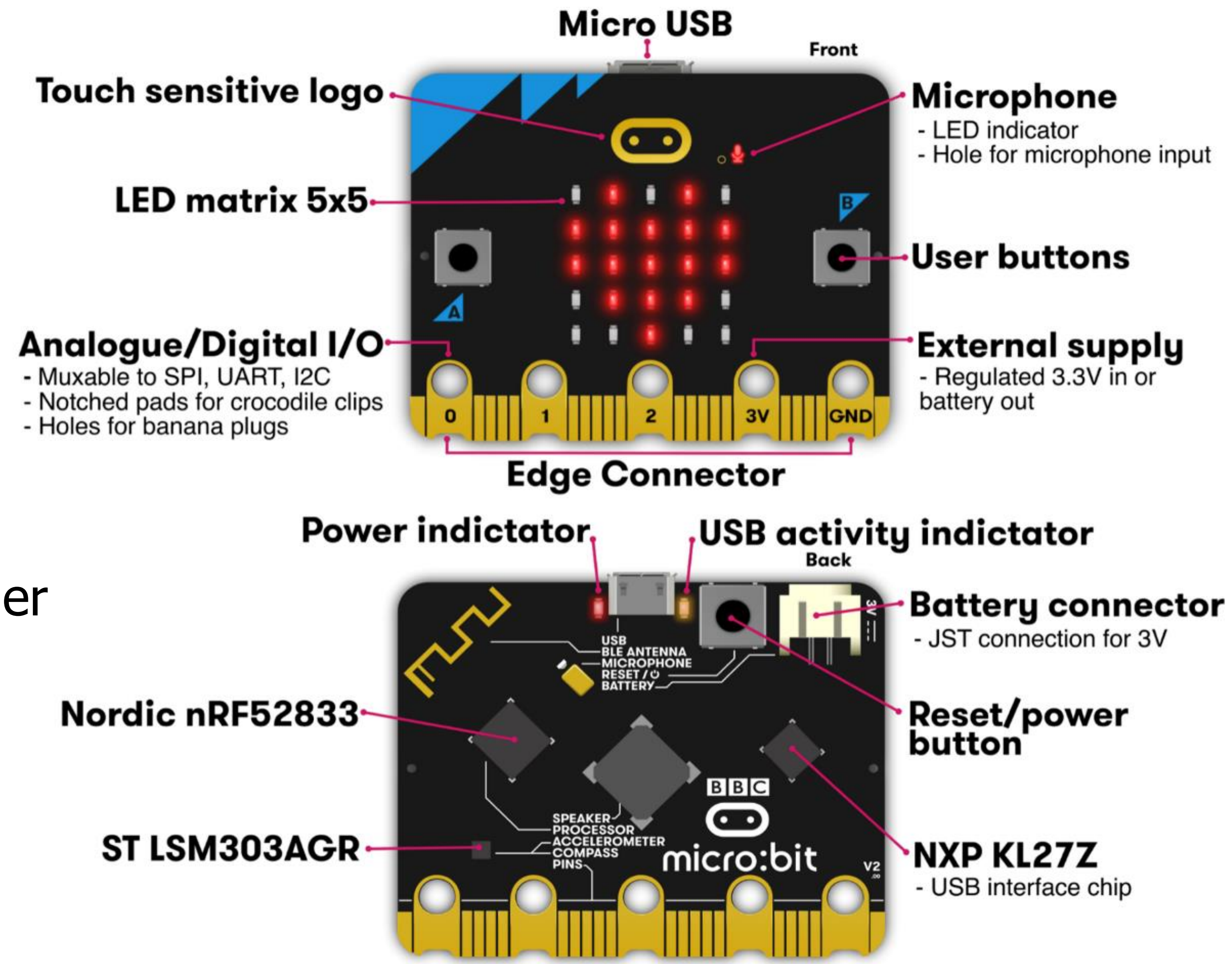
- The best projects often have a mix of backgrounds

Outline

- Who and Why
- Embedded Systems
    - Microcontrollers
- Course Overview
- **Class Hardware**
- Project Ideas

Micro:bit v2

- Legacy from 1980s "BBC Computer Literacy Project"
  - Reimagined today
- Around $20
  - Modern microcontroller AND sensors
- Plan for class:
  - Explore most of its functionality



Getting your own Microbit

- You do NOT need to buy your own Microbit
  - I have enough for everyone in the class to borrow one for the quarter

- If you want your own though, they're pretty cheap:
  - $17.95 Adafruit: https://www.adafruit.com/product/4781
  - $16.50 Sparkfun: https://www.sparkfun.com/products/17287
  - $21.95 Amazon: https://www.amazon.com/BBC-Micro-Programming-Include-Battery/dp/B0BSF5KTLT

Labs will use your own laptops

- All labs will use your own computers

- In the past we used the computers in CG50
  - About one computer would crash per lab session
  - Super crammed in there with no elbow room or walking room
  - And you had to physically go there to work on labs

- Setup for your own computers won't be that hard
  - Native MacOS or Linux works great
  - For Windows, VMWare Workstation + Ubuntu is pretty easy, but requires ~20 GB

- Concern of mine: equal access to labs
  - If you don't have a laptop or don't think it'll work, let me know!

Poll of the room

- MacOS users
- Linux users
- Windows users
  - WSL users?
- Other?

Lab0: Software Setup

- Is posted now on the Canvas homepage!
  - Due next Thursday

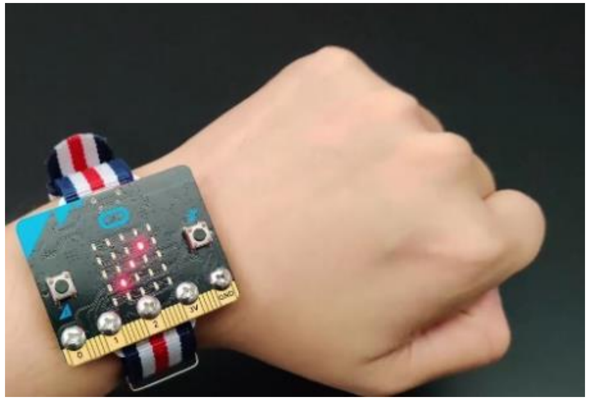
- I’ll be at the lab sessions on Friday if you want help with your setup
  - If you have a weird computer setup
  - Or if you feel uncomfortable installing a bunch of tooling
  - **I will also have hardware to hand out some Microbits!**
    - Will hand out more next week if you don’t come on Friday

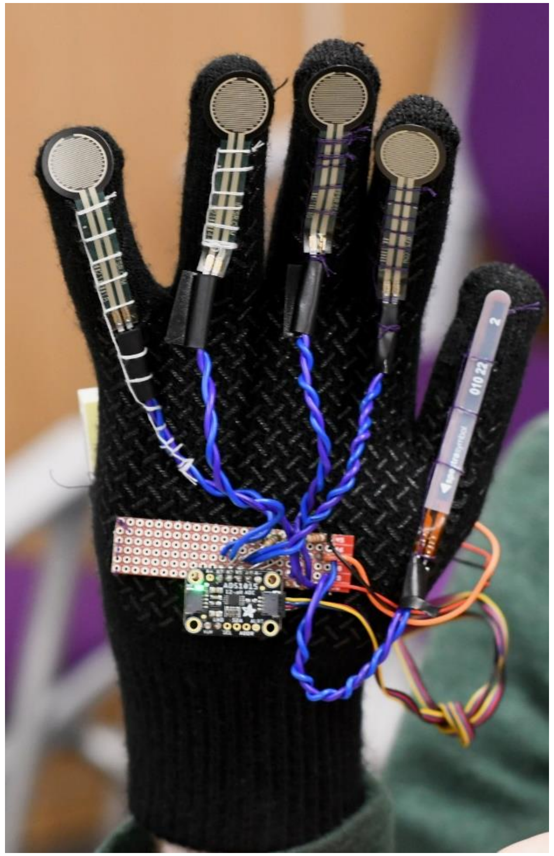
- If you feel comfortable with it, you’re welcome to do this first “lab” on your own
  - And you don’t have to attend lab this Friday
  - But please feel free to come by!

Outline

- Who and Why
- Embedded Systems
  - Microcontrollers
- Course Overview
- Class Hardware
- **Project Ideas**

Project Ideas

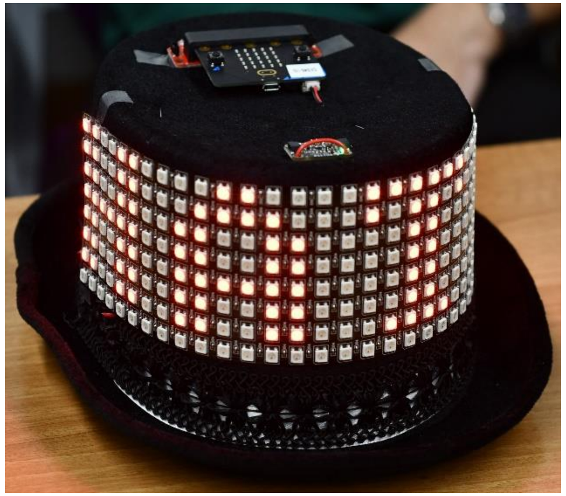


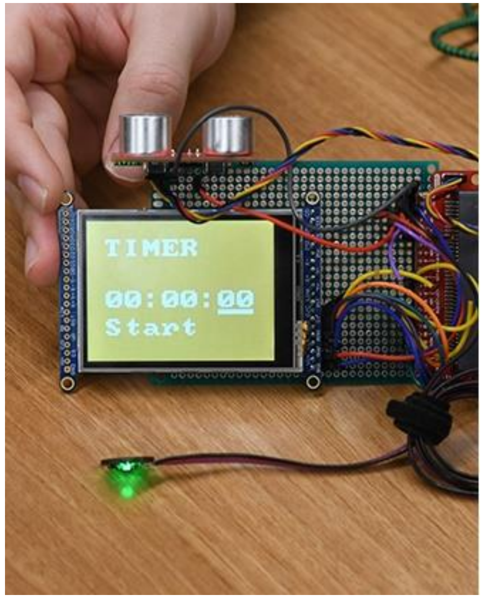


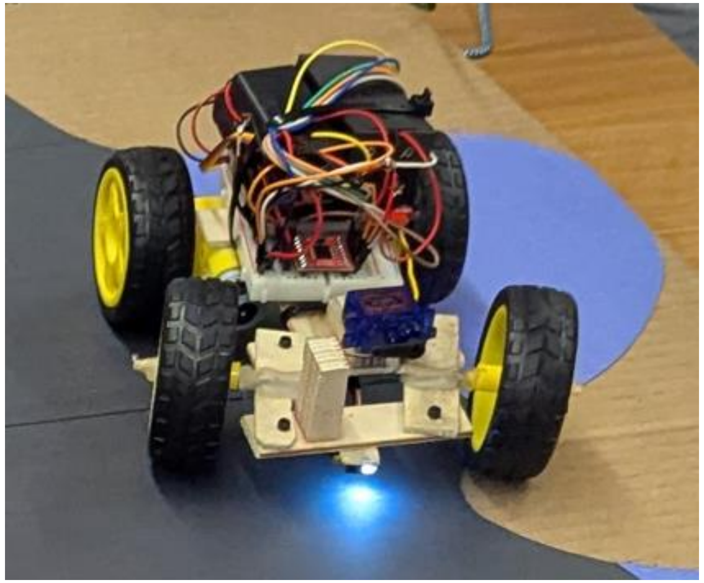
- Some ideas to get you thinking
  - Game with interesting control mechanism
  - Smart gloves
  - Smartwatch
  - Simple robotic systems

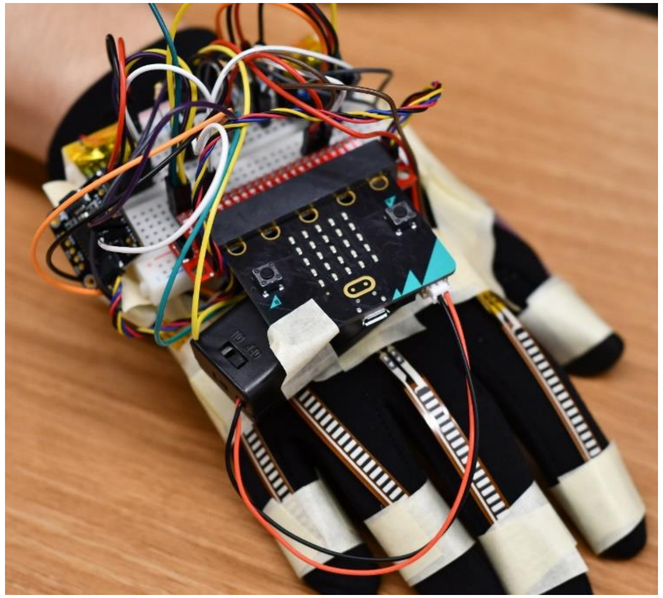
- Projects can use
  - Multiple Microbits
  - A personal computer for some amount of coordination
  - Small screens and displays
  - Lots of different sensors or actuators
    - Go explore Sparkfun and Adafruit

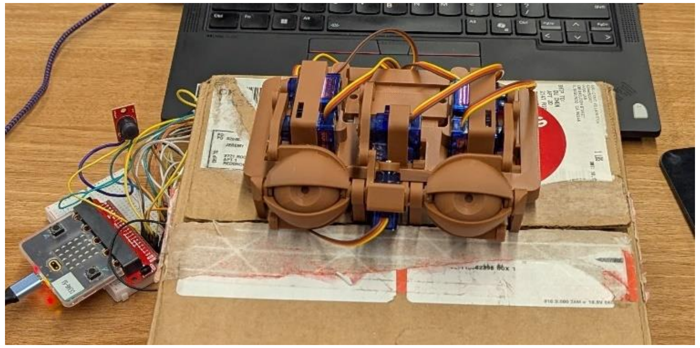
Some awesome projects – interactable objects



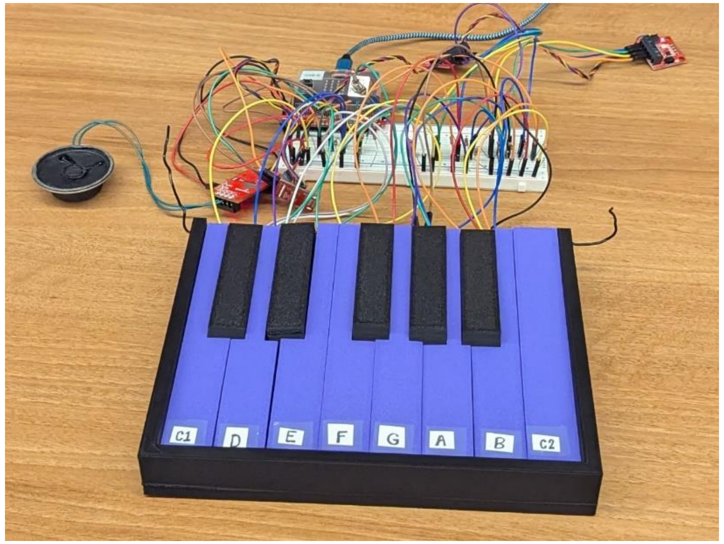


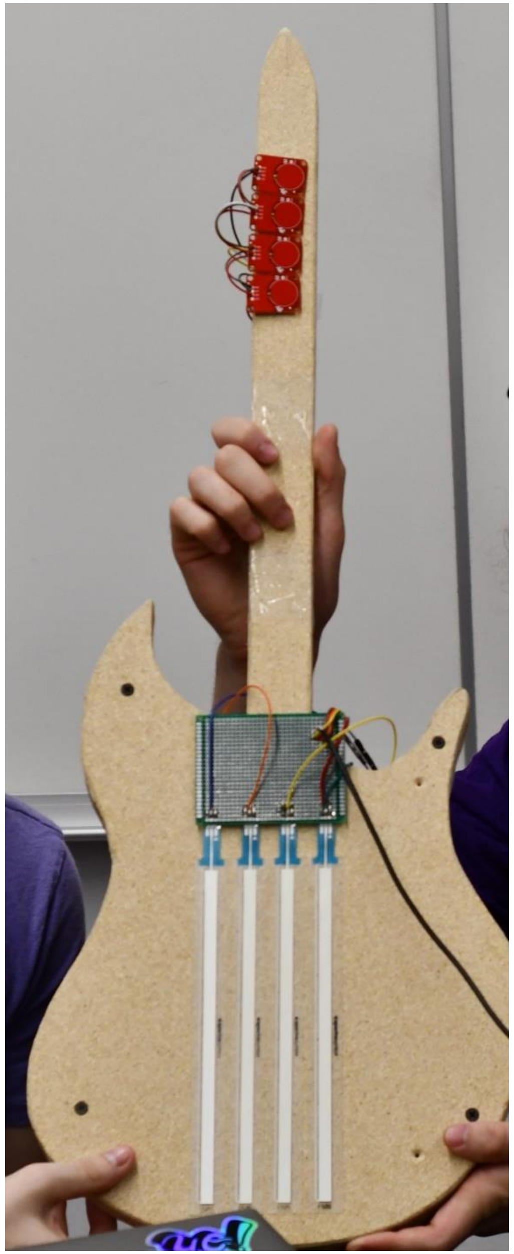


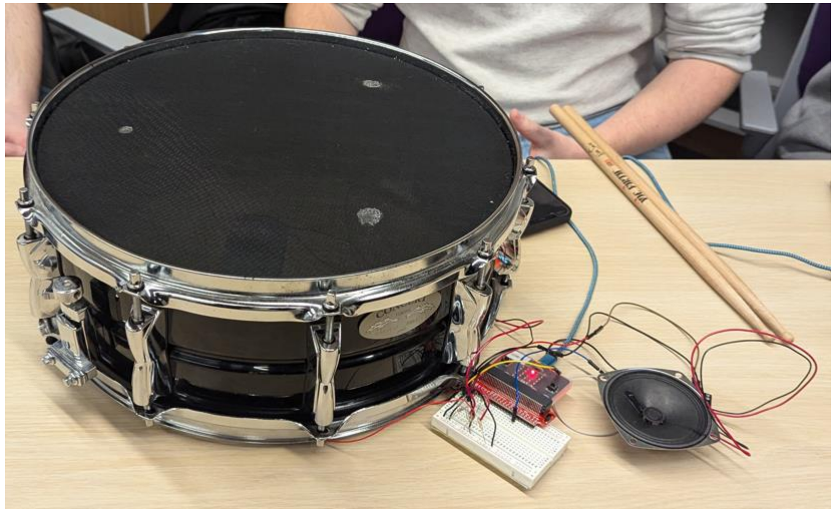


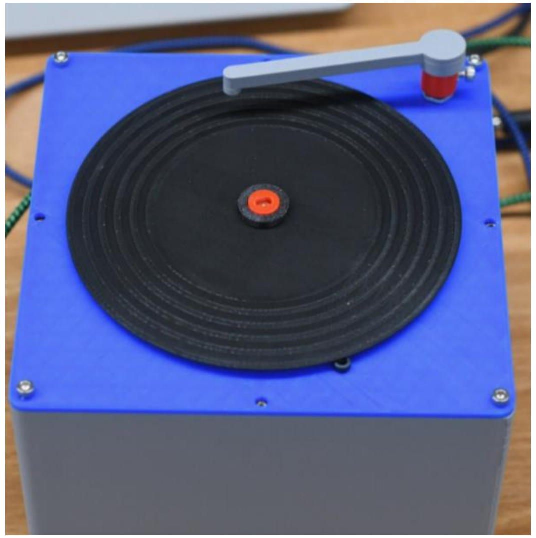


Some awesome projects – electronic music

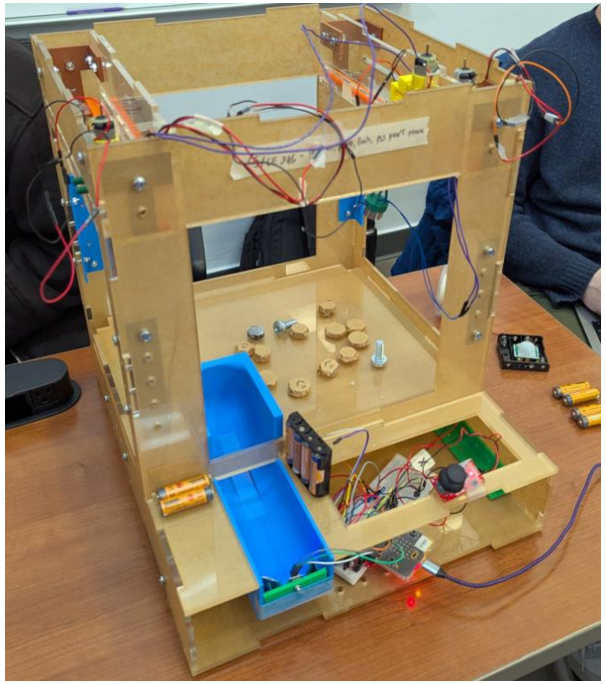


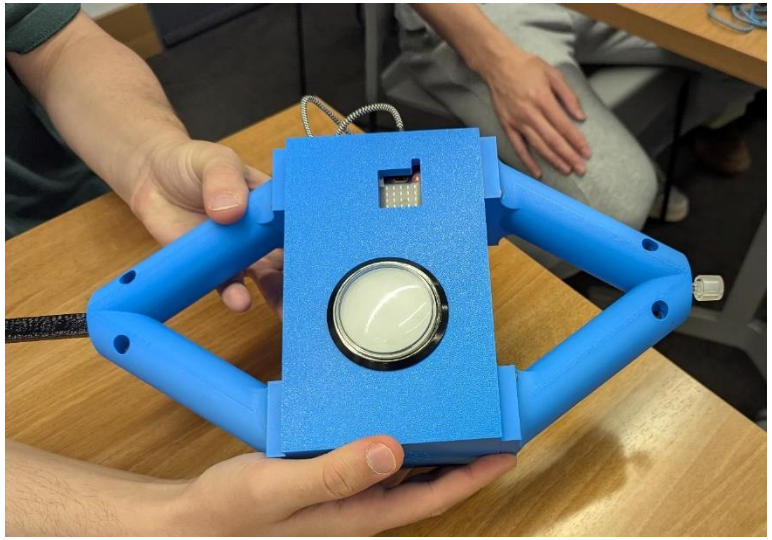


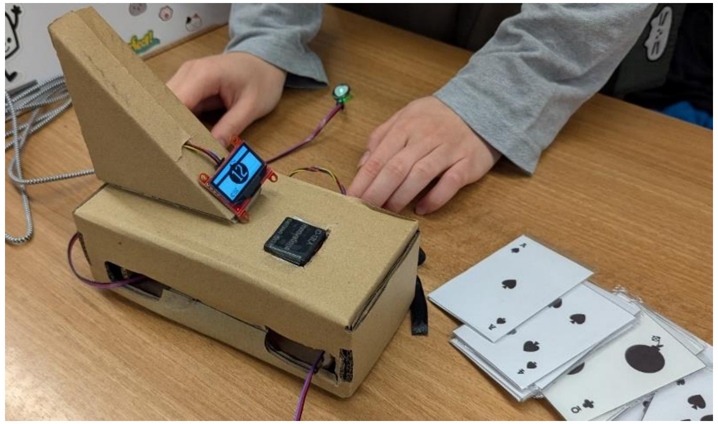




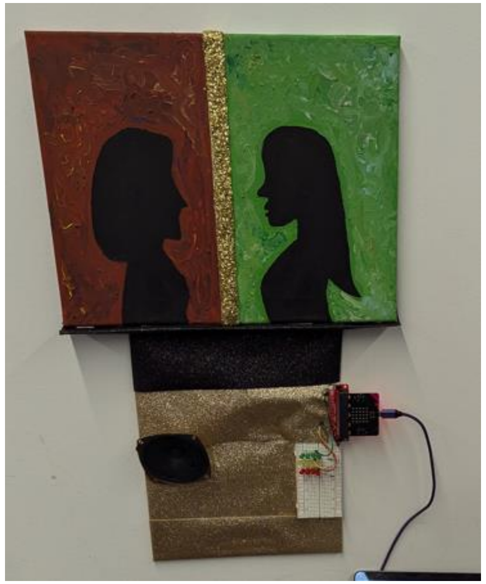
Some awesome projects – playable games

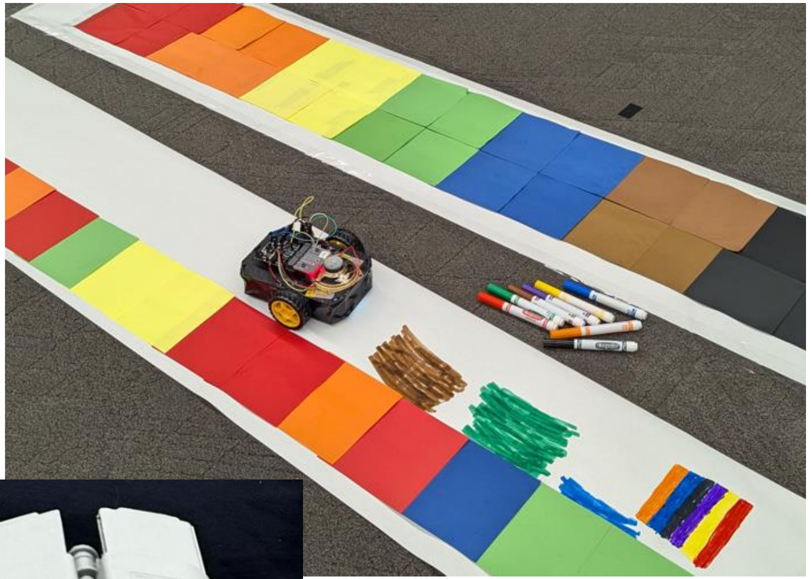


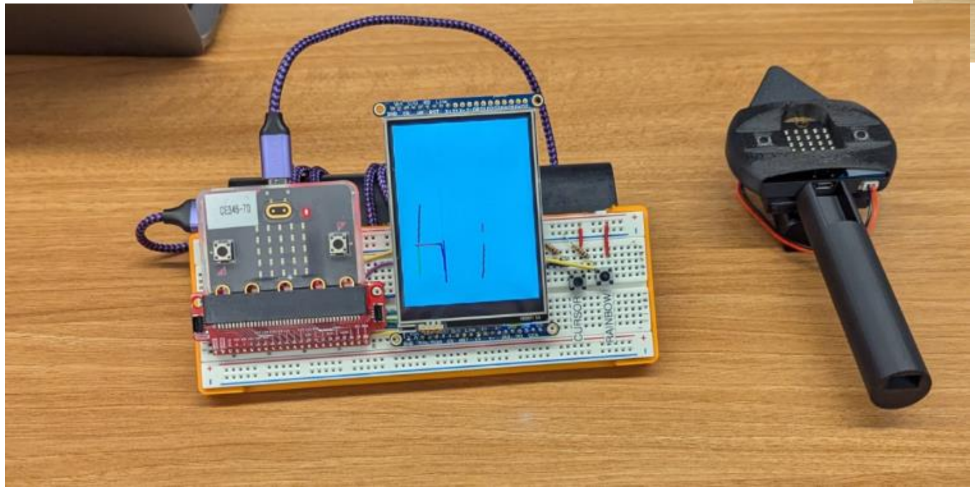




Some awesome projects – interactable art and robotics







Outline

- Who and Why
- Embedded Systems
  - Microcontrollers
- Course Overview
- Class Hardware
- Project Ideas