EC 535: Introduction to Embedded Systems

Lecture 1: Overview

Eshed Ohn-Bar
ee shed

Introduction

- What are embedded systems?
- Challenges in embedded computing system design.
- Design methodologies how to approach constrained setup in embedded

Definition

Special purpose vs generalized computer

- Embedded system: is a special-purpose computer system designed to perform one or a few dedicated functions, often with computing constraints.
  - multiple cm syst trying to work together to do a task

- Take advantage of application characteristics to optimize the design:
  - Don't need all the general-purpose bells and whistles.
  - But need to understand the application well.

Definition

- Embedded system: is a special-purpose computer system designed to perform one or a few dedicated functions, often with computing constraints.

air VAC: specific place doing specific thing

hard constraints / other types : computational

- Take advantage of application characteristics to optimize the design:
  - Don't need all the general-purpose bells and whistles.
  - But need to understand the application well.

- Dedicated functionality: Specific tasks. Often custom software, but could also be hardware, customized to the application.
- Real-time: May need to respond within a certain time constraint.
- Resources: Limited in memory, computing, energy, within a device.

real world syst come with mult. constraints

Where Can We Find Embedded System?

everywhere
actuation, monitoring
autonomous vehicle can have 100s of emsys
Microcontroller sensors can have emsys

Avionics

Communication

Transportation

Medical Instrument

Office Equipments

Consumer Electronics

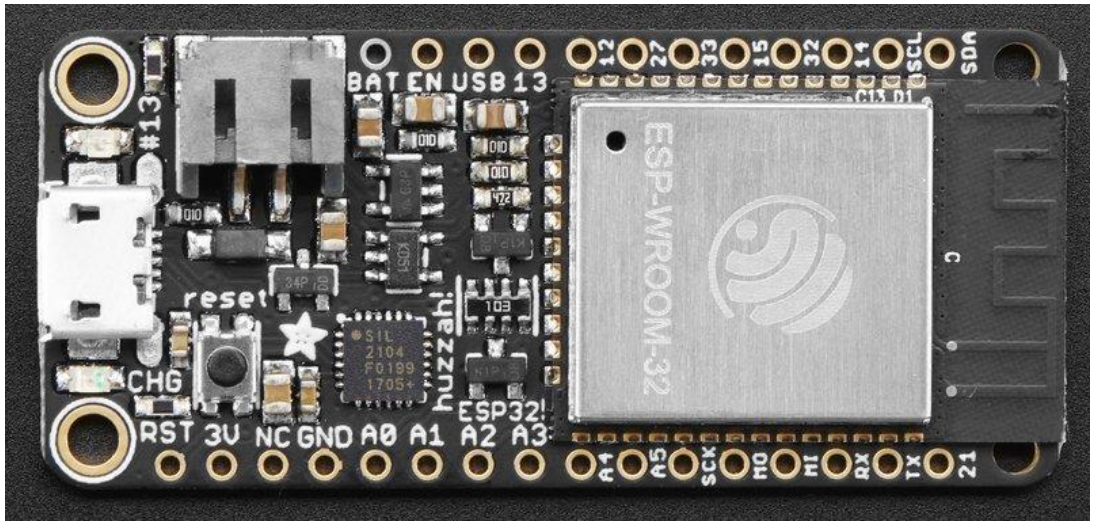
and have algorithm for cleaning in SW optimization could also be in HW

Household Appliances

take minimal computer cheap easy
can attach various sensors actuators...

Microcontrollers: Low cost, minimal computer

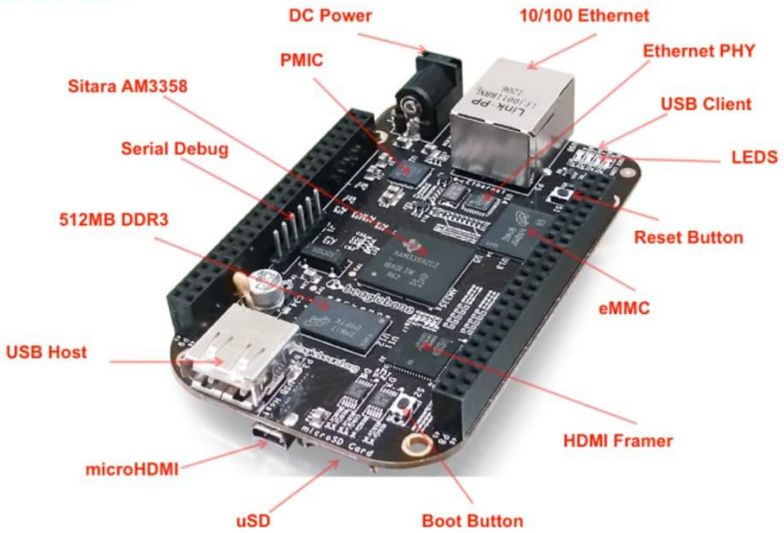
different specs for all microcontrollers



$22.00

Arduino Uno Rev3

$20
ESP32

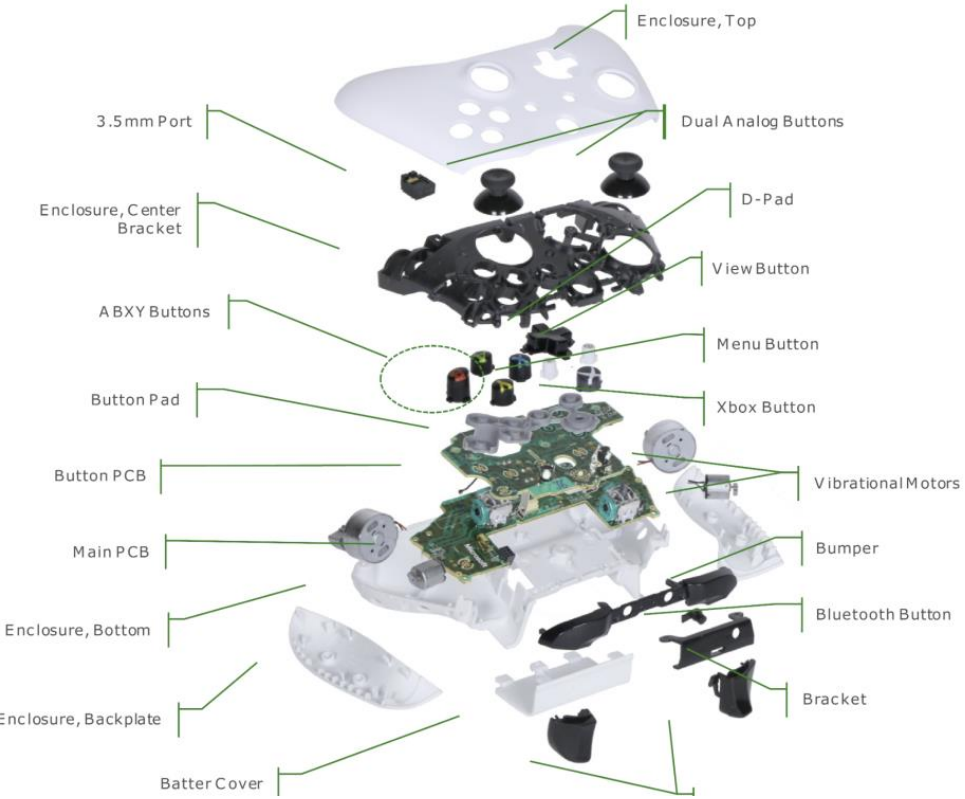


$60
BeagleBoard

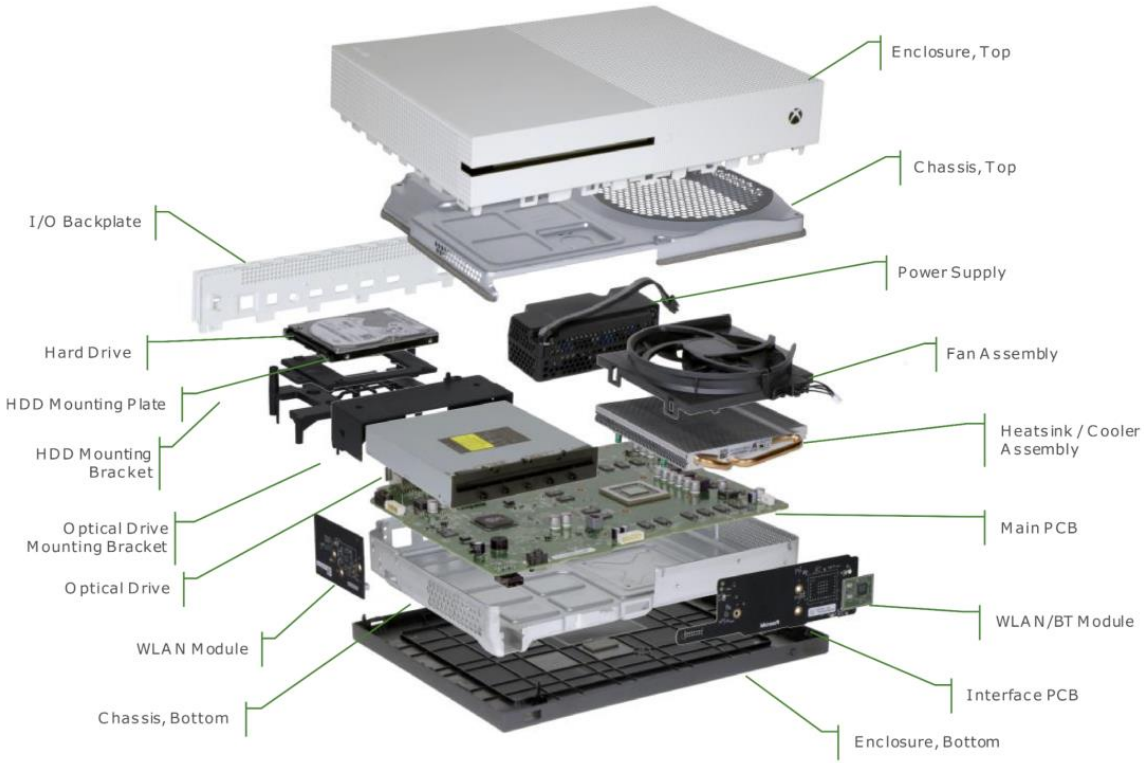
An Embedded System Example: Xbox

Optimized for gaming but can also be optimized for different specs

Controller - Exploded View

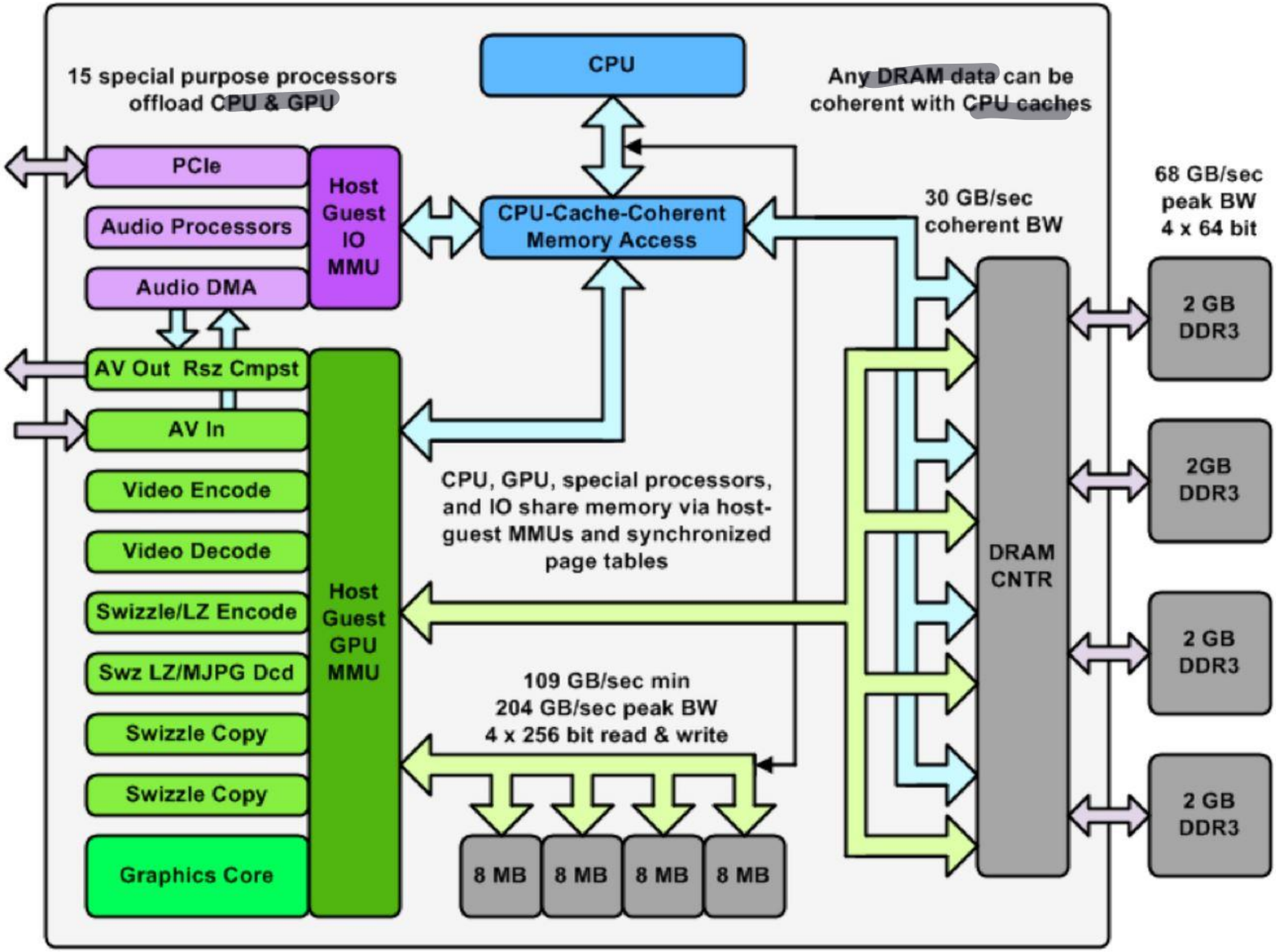


Console - Exploded View



Architecture optimized for specialized task it aims to do

An Embedded System Example: Xbox



Automotive Embedded Systems

high # syst. can also need to rely on other cmsys ∴ communication

- Today's high-end automobiles have > 100 microprocessors:
  - 4-bit microcontroller checks seat belt;
  - Microcontrollers run dashboard devices;
  - 16/32-bit microprocessor controls engine;
  - Millions of lines of code!

- **More than 30% of the cost is in electronics**

robots need ensyst as well

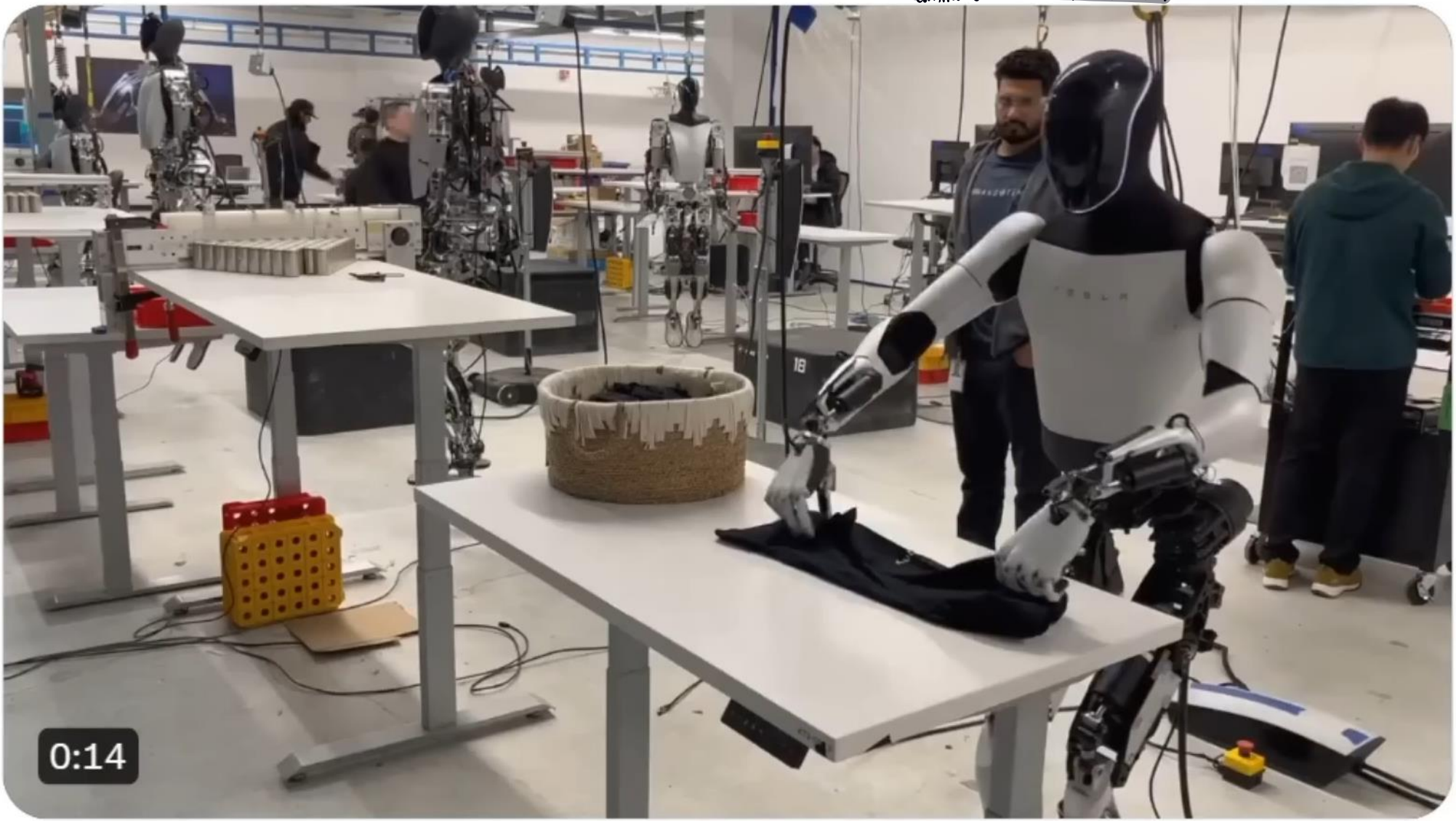
achieving overall task

☆ constrained

constraints: real time, cost, safety

IOT: Internet of things

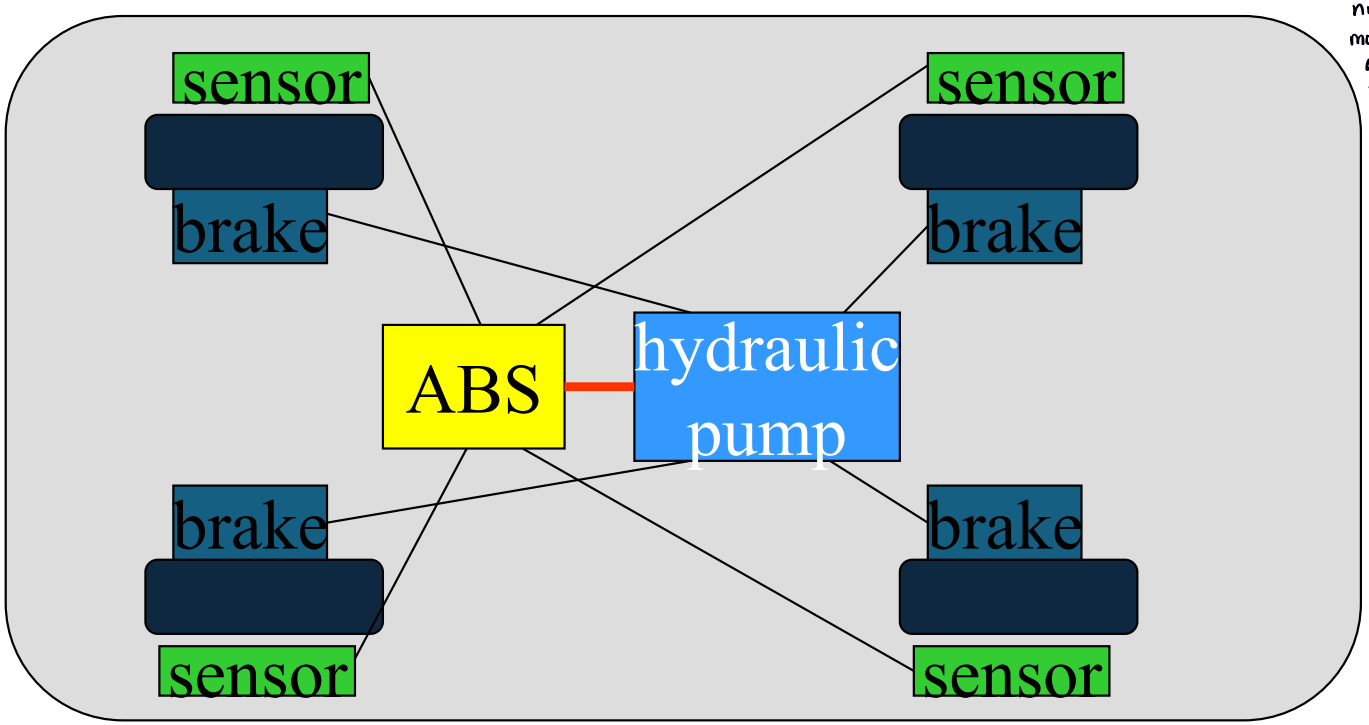
Motion: have to carry all computational resources on you



Brake and Stability Control System

- Anti-lock brake system (ABS): pumps brakes to reduce skidding.

ABS



When skidding: pump the brake
Now don't need to pump the brake
monitors skidding and can auto pump breaks
avoids sweeving out of control
* only if ABS is functional

has to operate fast and accurately.
don't need to have entire pc allocated to do this

SW and HW come together to optimize smart

- Stability control (ESC) further controls the engine.

Other applications of embedded systems?

1. Pedometer
2. Smart home environment

Microprocessor Varieties

computer chips, each one will have own optimal functions
FPGA: running parallel better than arduino

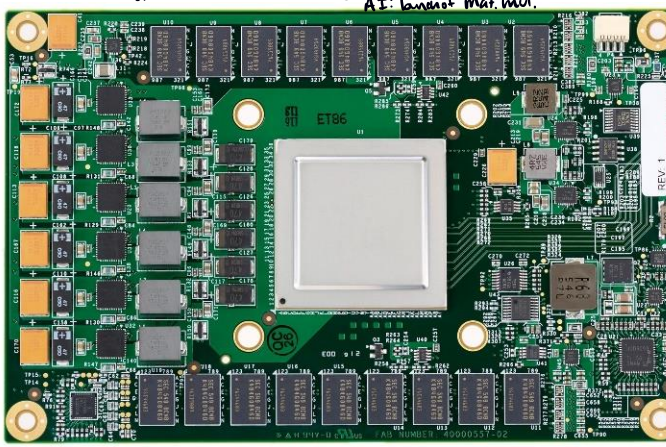
- Microcontroller: includes I/O devices, on-board memory.
  - Examples (among many, many products in the market):
    - Arduino Uno R3
    - Adafruit Gemma M0
    - STM32 F3 Discovery
    - FPGA with softcore processors, e.g., MicroBlaze
- Digital signal processor (DSP): microprocessor optimized for digital signal processing.

Multiplication and accumulation optimized on HW level

  - Examples:
    - Philips Trimedia, TI C6000™, Analog Devices ADSP-21xx series
- Application-specific Processors/SoCs: processors optimized for a particular type of task.
  - Examples:
    - Intel Nervana Neural Network Processors (NNPs)
    - Canon DIGIC series image processor
    - Google's Tensor Processing Units (TPUs)

designed own chip for AI

optimized on HW for accessing memory
AI: hardcoat mat. mul.

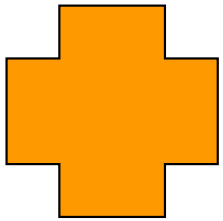


Processor technology

Microprocessors usually harder to work with than computers but not hard

ISA: defines everything a processor can do
↳ can also be optimized!

- Processors vary in their customization for the problem at hand

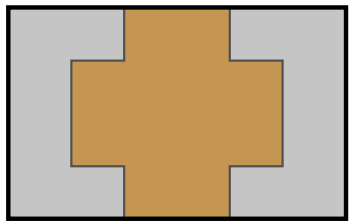


Desired functionality

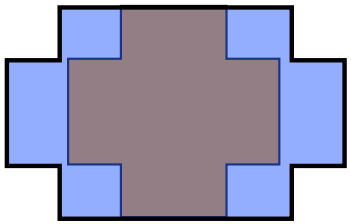
```
total = 0
for i = 1 to N  loop
   total += M[i]
end loop
```

Configurable aspects of general processors
quite a bit of overhead saved by emphasizing specific task
many ways to optimize

overhead can be reduced

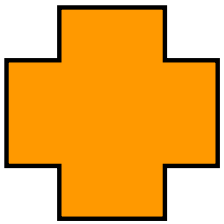


General-purpose processor



Application-specific processor

Specializes what cores can do
ASIP



Customized Coprocessor

Advantages of Processors vs. Custom Logic

more static circuits are less flexible
but lowers latency

★ think about different aspects of embsys

different mp allow you to
change diff things at cost of other things

- Flexibility
  - Quick turn around for bug fix
  - Can upgrade through software (firmware)
- Lower Cost
  - To develop software vs. hardware
  - To manufacture
- Acceptable Performance
  - Heavy pipelining, wide issue width, large cache
  - Experienced design teams
  - Aggressive VLSI technology

Characteristics of Embedded Systems

optimize all different ESP32 because there are so many

- Sophisticated functionality. ★ and specific!
- Real-time operation. design in SW and HW
- Low manufacturing cost.
- Low power budget. Microprocessors in class are power hungry because board
- Short time-to-market and small teams.

don't necessarily have infinite time to do these things

time-to-market : 6 months

team should be working max for 6 months

Functional Complexity

want to optimize so tradeoff for this

- Often have to run multiple algorithms.
  - Example: A Video Disc player
    - Blu-ray Disc, DVD, video CD, audio CD, JPEG image CD, MP3 CD, MPEG-4, DivX
    - Digital rights management
  - Requiring domain-specific knowledge.

- Often provide sophisticated user interfaces.
  - Multiple levels of user menus
  - Support for multiple languages
  - Graphics
  - Speech, handwriting

hard real time : ABS - failure
Soft real time : thermostat - degraded performance

Real-time Operation

- Must finish operations by deadlines.
  - Hard real time: missing deadline causes failure.
  - Soft real time: missing deadline results in degraded performance.

- Many systems are multi-rate: must handle operations at widely varying rates.
  - Example: Audio 44KHz and Video 30fps

have different time constraints

Cost and Power Consumption

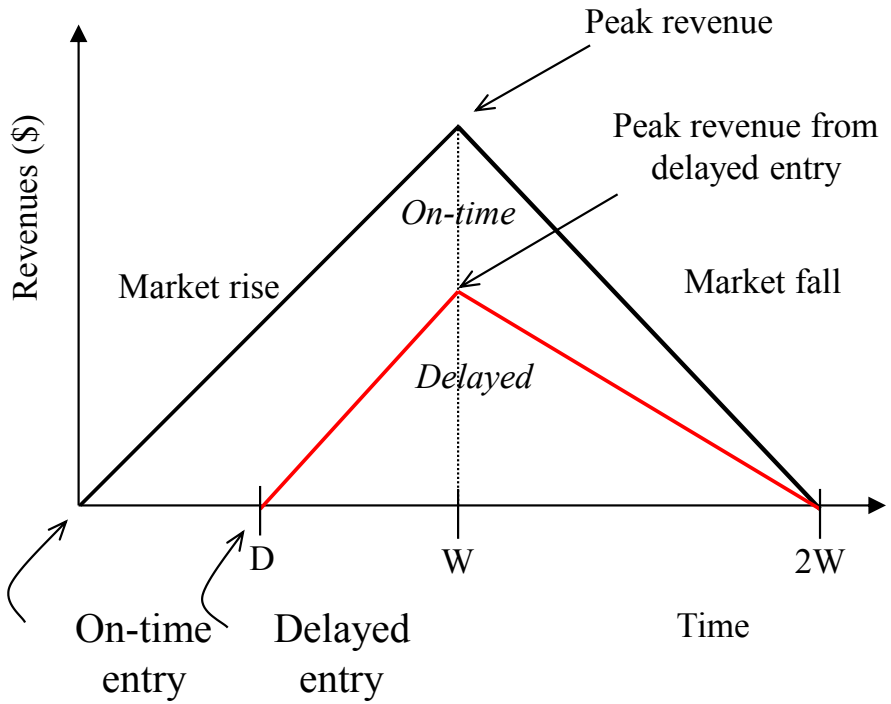
- Many embedded systems are mass-market items that must have low manufacturing costs.
  - Limited memory, microprocessor power, etc.
- Power consumption is critical in battery-powered devices.
  - Excessive power consumption increases system cost even in wall-powered devices.

Power

- Custom logic is a clear winner for low power devices.
- Modern microprocessors offer features to help control power consumption
  - Idle/sleeping mode
  - Dynamic voltage/frequency adjustment
- Software design techniques can help reduce power consumption.
- Can combine processor and custom logic.

Time-to-market

- Often must meet tight deadlines.
  - 6 month market window is common.



Challenges in Embedded System Design

series of decisions to meet

- How much hardware do we need?
  - How many processors? How big are they? How much memory?
- How do we meet performance requirements? real time sys. needs to be fast enough
  - What's in hardware? What's in software? how to optimize HW?
  - Faster hardware or cleverer software?
- How do we minimize cost and power?
  - Turn off unnecessary logic? Reduce memory accesses?
- How do we ship in time?
  - Buy solution? From scratch? Off-the-shelf chips? IP-reuse?

Optimize software then wanting to change chip means remanufacturing required.

Challenges (cont'd)

- How do we know that it fully works?
  - Is the specification correct?
  - Does the implementation meet the spec?
  - How do we test for real-time characteristics?
  - How do we test on real data?
- How do we locate the problem if it doesn't work?
  - Observability?
  - Controllability?

Usually goes through testing and validation on many levels
how to test AFBS? network on ms level of decisions: stress test

debug tools: LED, serial log

relying heavily on emsys

Adjusting control settings on embedded systems is hard because having predictable environments isn't always feasible; and simulation is actually pretty different than real world experiences

Required Designers

because of so many constraints: optimize many areas

SW and HW have to be bridged

Need to optimize many different areas in order to optimize throughput and balance so need to optimize both SW and HW

- Expertise with both **software and hardware** is needed to optimize design metrics
  - Not just a hardware or software expert.
  - A designer must be comfortable with various technologies in order to choose the best for a given application and constraints.
  - A designer must be able to communicate with teammates of various backgrounds.

Firmware vs circuits and chips
(SW) (HW)

Design Methodologies

– how to approach designing multi-constrained systems

git : CAP

How to approach designing multi-constrained systems

in order to design a system, need a framework to follow : this is the procedure

- A procedure for designing a system.
- Understanding your methodology helps you ensure you didn't skip anything.
- Compilers, software engineering tools, computer-aided design (CAD) tools, etc., can be used to:
  - help automate methodology steps;
  - keep track of the methodology itself.

workflow tools to automate methodology. dont want to do everything manually

need to follow procedure !

want to automate parts of design aspect

dont want to do everything manually

workflow tools

Two different approaches

Top-down vs. Bottom-up

what does the system do?
break down into steps

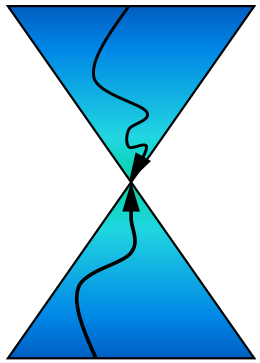
- Top-down design: optimized for final goal
  - start from most abstract description;
  - work to most detailed.
- Bottom-up design: integrate into working prototype
  - work from small components to big system.
- Real design uses both techniques, more or less.

top down: final product / goal?
work down how to achieve this goal

bottom up: what do I have, what can I do with these parts? How to integrate them into larger parts?

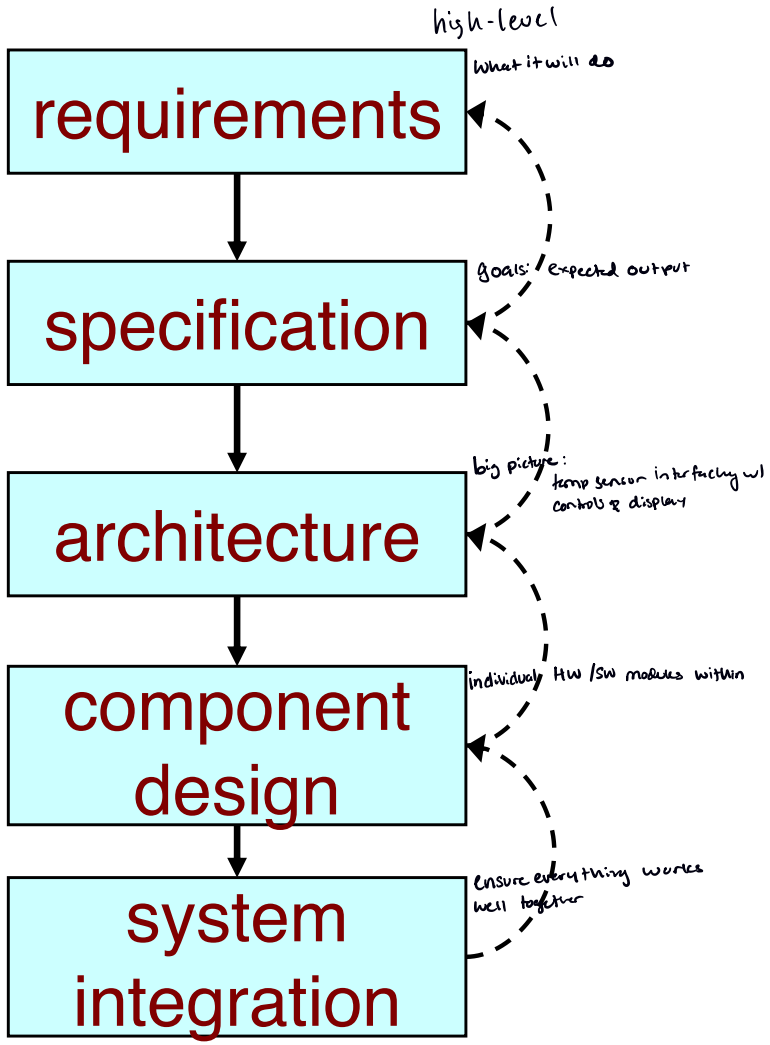
use both approaches in same project

can use bottom-up to integrate already existing components



Levels of Abstraction

levels of abstraction
HL
requirements — what will it do
↓
specification — expected output (goals)
↓
architecture — big picture:
↓
component design — individual HW and SW modules within
↓
system integration — ensure everything works well together



Requirements

- Plain language description of what the user wants and expects to get.

- May be developed in several ways:
  - talking directly to customers;
  - talking to marketing representatives;
  - providing prototypes to users for comment.

EK210
Figure out what the vision is from users?
What is the requirements for this to properly represent what a customer expects!

Functional vs. Non-functional Requirements

functional: output as function of input

- Functional requirements:
  - output as a function of input.

- Non-functional requirements: non-functional: other speed??
  - time required to compute output;
  - size, weight, etc.;
  - power consumption;
  - reliability;
  - etc.

Specification

output given certain input

output given certain input

- A more precise description of the system:
  - should not imply a particular architecture;
  - provides input to the architecture design process.
- May include functional and non-functional elements.
- May be executable or may be in mathematical form for proofs.

Architecture Design

We have the specification picture. What do we need for architecture to satisfy the specifications?

- What major components do we need to satisfy the specification?
- Hardware components:
  - CPUs, peripherals, etc.
- Software components:
  - major programs and their operations.
- Must take into account functional and non-functional specifications.

Designing Hardware and Software Components

- Must spend time architecting the system before you start coding.

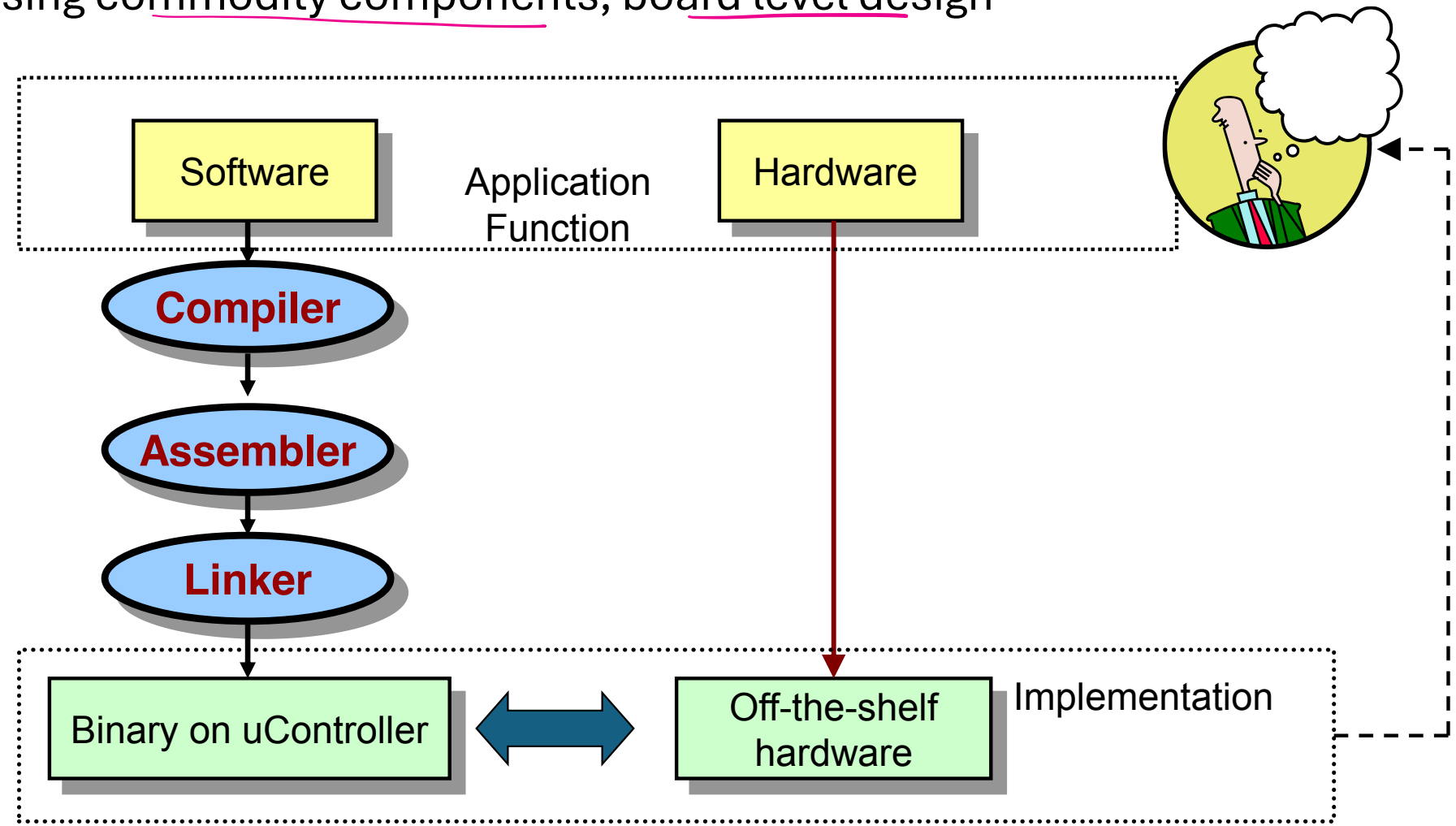
figure out how system will connect before coding.
don't go in blindly

- Some components are ready-made, some can be modified from existing designs, others must be designed from scratch.

Traditional Embedded System Design

many ways of doing this
lots of human effort to optimize green boxes system

- SW and HW partitioning is decided at an early stage
- Using commodity components, board level design



System Integration

has to worry about putting things together

how to properly integrate components? many bugs arrive from how certain signals outputted from different parts of the system communicate with eachh other

- Put together the components.
  - Many bugs appear only at this stage.

- Have a plan for integrating components to uncover bugs quickly, test as much functionality as early as possible.

test as much as possible to see that everything is functional as expected

Summary

- Embedded computers are all around us.
  - Many systems have complex embedded hardware and software.
- Embedded systems pose many design challenges: design time, deadlines, power, etc.
- Design methodologies help us manage the design process.

What class covers

Topics to Cover

- Embedded Computers
  - Instruction set architecture/microarchitecture
  - Memory architecture
  - Input/output devices I/O
- Embedded Software/Real-time Operating System
  - Scheduling
  - Linux device drivers
- Design Techniques
  - Specification methods/computation models
  - Verification techniques
  - Hardware-software co-design techniques how to design with both in mind
- Secure Embedded Systems
  - Common vulnerabilities
  - ...and how to avoid them

RTOS: optimized for real time processing
Certain task can run in timely manner

Recommended

Embedded systems on the web

- Berkeley Design technology, Inc.: http://www.bdti.com
- EE Times Magazine: https://www.eetimes.com/
- Embedded Linux Journal: https://www.linuxjournal.com/tag/embedded
- Embedded.com: http://www.embedded.com/
- Circuit Cellar: http://www.circuitcellar.com/
- Electronic Design Magazine: https://www.electronicdesign.com/
- Sensors Magazine: https://www.fierceelectronics.com/sensors
- Embedded Systems Tutorial: http://www.learn-c.com/
- Collections of embedded systems resources (older / not recently updated)
  - http://www.ece.utexas.edu/~bevans/courses/ee382c/resources/
  - http://www.ece.utexas.edu/~bevans/courses/realtime/resources.html

Today's Assignments

- Review C programming
  - http://www.cprogramming.com/tutorial.html#ctutorial
  - http://crasseux.com/books/ctutorial/
  - http://www.howstuffworks.com/c.htm
  - C for C++ programmers: http://people.cs.uchicago.edu/~iancooke/osstuff/ccc.html
  - Many other textbooks and online C tutorials are available.
- Check room access, request access in Access Management Portal if needed
  - PHO 307
  - Make sure blackboard access.

have acuss to BB and gradescope

Course Reviews

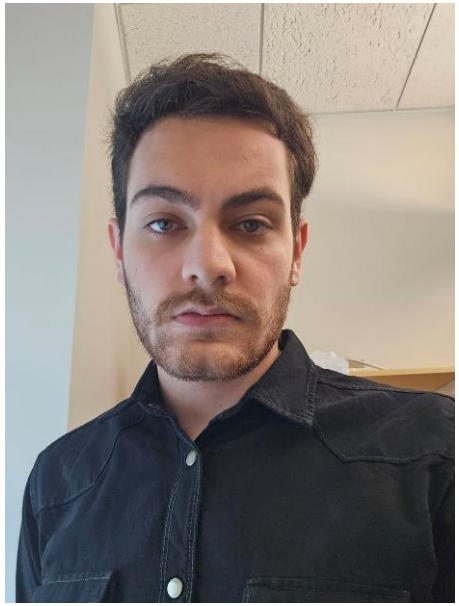
- "Too much lab and homework assignments!"
- "Too much time spent on searching on the internet"
- "Forces students to figure out some of the material themselves, which ends up less efficient for people who comes from a rather weak background"
- "Be prepared to do a lot of self-learning, programming, and debug for hours"
- "the course content scale up fast in terms of difficulty, one of the hardest courses had to take. "

· lots of C programming
· component of self-learning

Administration

- **Instructor**
  - **Prof. Eshed Ohn-Bar**
  - **Office:** ~~PHO 532~~
  - **Email:** eohnbar@bu.edu
    - **Please write "EC535" in subject line**
  - **Office Hours:**
    - **Thursdays 11am-noon (starting next week)**

Administration



- **Teaching Assistants**
  - **Amin Khodaverdian**
    - **Email: aminhaji@bu.edu**
  - **Labs, Office Hours**
  - **Check Blackboard for the latest schedule and links**
    - Mondays and Wednesdays 5pm-7pm at PHO305/307.
    - *Check Blackboard for most up-to-date schedule.*
  - Ask for access in Zaius to PHO307 (can also for PHO305)

Grading Policy

- Exam: 40% most likely a final
  - Date/Time: TBD
- Homework Assignments: 10%
- Lab Assignments: 15%
  - Labs 1-3 individually on emulator, then moving to hardware
- Project: 25%
  - On hardware
- Participation: 10%
  - Written responses to reading or lab-prep questions, in-class exercises

Homework

- HW to be assigned on Tuesdays or Thursdays
  - Typically due a week later
  - Electronic submission: Deadline is strictly enforced. No late submissions.

- Programming assignments
  - C (or C++) knowledge necessary
  - Unix/Linux experience helpful.
  - HW1 coming up
    - Review C (asap!):
      - Fundamentals (functions, loops, data types, etc.)
      - Bit-wise operators, logical operators
      - Basic data structures and declarations: array, list, struct
    - In-class component, take-home component

- Reading assignments

Lab/Project

- Individual labs on emulator for labs 1-3
- Labs on boards for lab 4 & project (lab 5)
- Arm-based development boards running Linux

  (Other programmable platforms may also be permitted– always ask first!)

Photo: BeagleBone Black

Lab Hours

Must go to lab!!

- Fridays: 10:10am-11am
  - Tutorials
  - Lab help
  - Hands-on exercises

- **Mandatory to attend**
  - We may skip a week or two

mandatory