

Introduction To Embedded Systems

CMPN-445

Basem Ibraheem

Basem@eng.cu.edu.eg

Basemibraheem@gmail.com

Intended Learning Outcomes

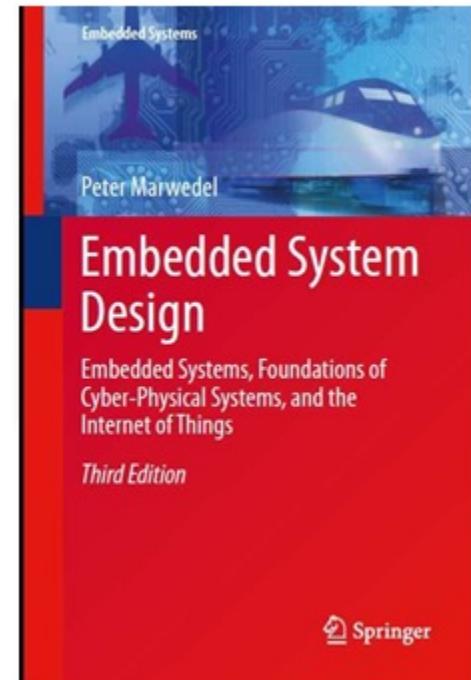
- Develop an understanding of Embedded Systems, their characteristics, applications and challenges
- Learn the different methodologies to model embedded systems.
- Understand the design metrics and the interplay among design trade-offs, and technology issues.
- Differentiate between general operating systems and real time ones and understand the role of the latter in embedded systems.
- Acquire the design skills necessary for co-designing HW and SW.
- Acquire the understanding and design skills of major embedded system components including sensors, actuators, ADCs, DACs & control loops

Course Contents

- ✓ • **Embedded Systems Overview**
- ✓ • **Modeling Embedded Systems**
 - FSMs, Data Flow Diagrams, Petri Nets
- M ✓ • **Embedded Systems Inputs and Outputs**
 - ADCs, DACs, Sensors, Actuators, Storage
 - Processing Units: AVR, PIC, ARM, etc...
 - Memory Hierarchy and Management
 - Design Validation
 - Hardware and Software Engineering

Course References & Readings

- Embedded System Design
 - *Embedded System Design: Embedded Systems, Foundations of Cyber-Physical Systems, and the Internet of Things*
 - Peter Marwedel,
 - Springer, 3rd Edition, 2018



Course Grading

- Tutorials & Assignments (15)
- Midterm (20)
- Final (40)
- Project (20)
- Research & Quizzes (5)

Project Examples

- Face Recognition
- North Seeking Gyro
- Robotic Arm
- Car Parking System
- Logic Analyzer
- RFID Child Safety System
- Remote Home Security and Automation

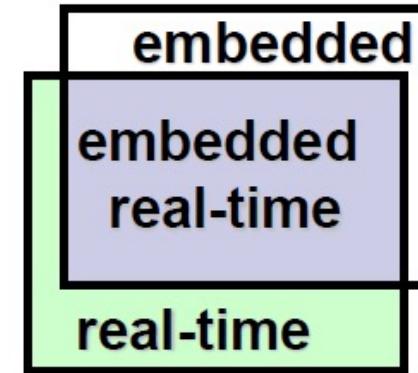
Definition

- Embedded System is a computerized system that is built for the purpose of a specific application
 - “Dortmund“ Definition: [Peter Marwedel]
 - Embedded systems are information processing systems embedded into a larger product
 - Berkeley: [Edward A. Lee]:
 - Embedded software is software integrated with **physical processes**. The technical problem is managing **time** and **concurrency** in computational systems.
 - **Cyber-Physical (cy-phy) Systems** (CPS) are integrations of computation with **physical processes** [Edward Lee, 2006].

Characteristics of Embedded Systems

- Application specific
- Efficient
 - energy, code size, run-time, weight, cost
- Dependable
 - Reliability, maintainability, availability, safety, security
- Real-time constraints
 - Soft vs. hard
- Reactive - connected to physical environment
 - sensors & actuators

↑
Jaww
- Hybrid
 - Analog and digital
- Distributed
 - Composability, scalability, dependability
- Dedicated user interfaces



Inside the PC

- Custom processors
 - Graphics, sound
- 32-bit processors
 - IR, Bluetooth
 - Network, WLAN
 - Hard disk
 - RAID controllers
- 8-bit processors
 - USB
 - Keyboard, mouse



Embedded system metrics

- Some metrics:

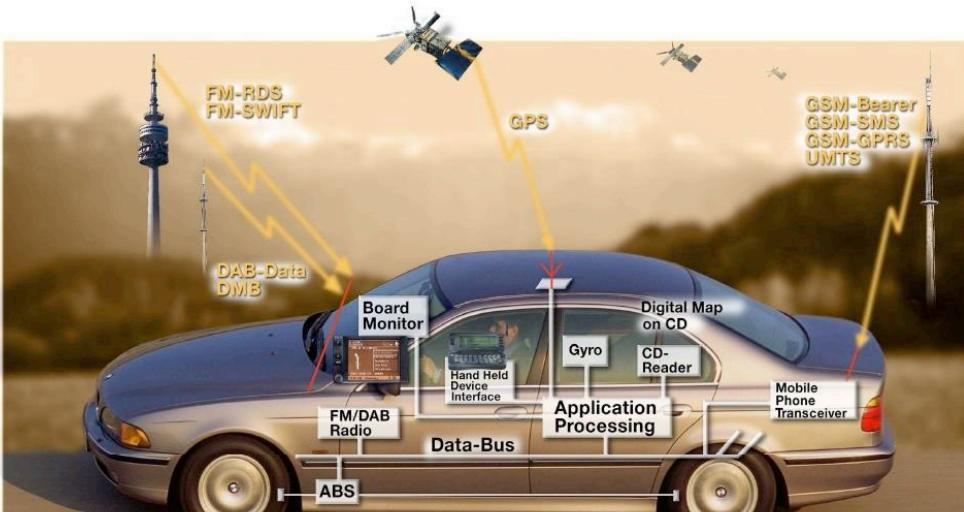
- performance*: MIPS, reads/sec etc.
 - power*: Watts
 - cost*: Dollars
 - Nonrecurring engineering cost, manufacturing cost
 - size*: bytes, # components, physical space occupied
 - Flexibility, Time-to-prototype, time-to-market
 - Maintainability, correctness, safety

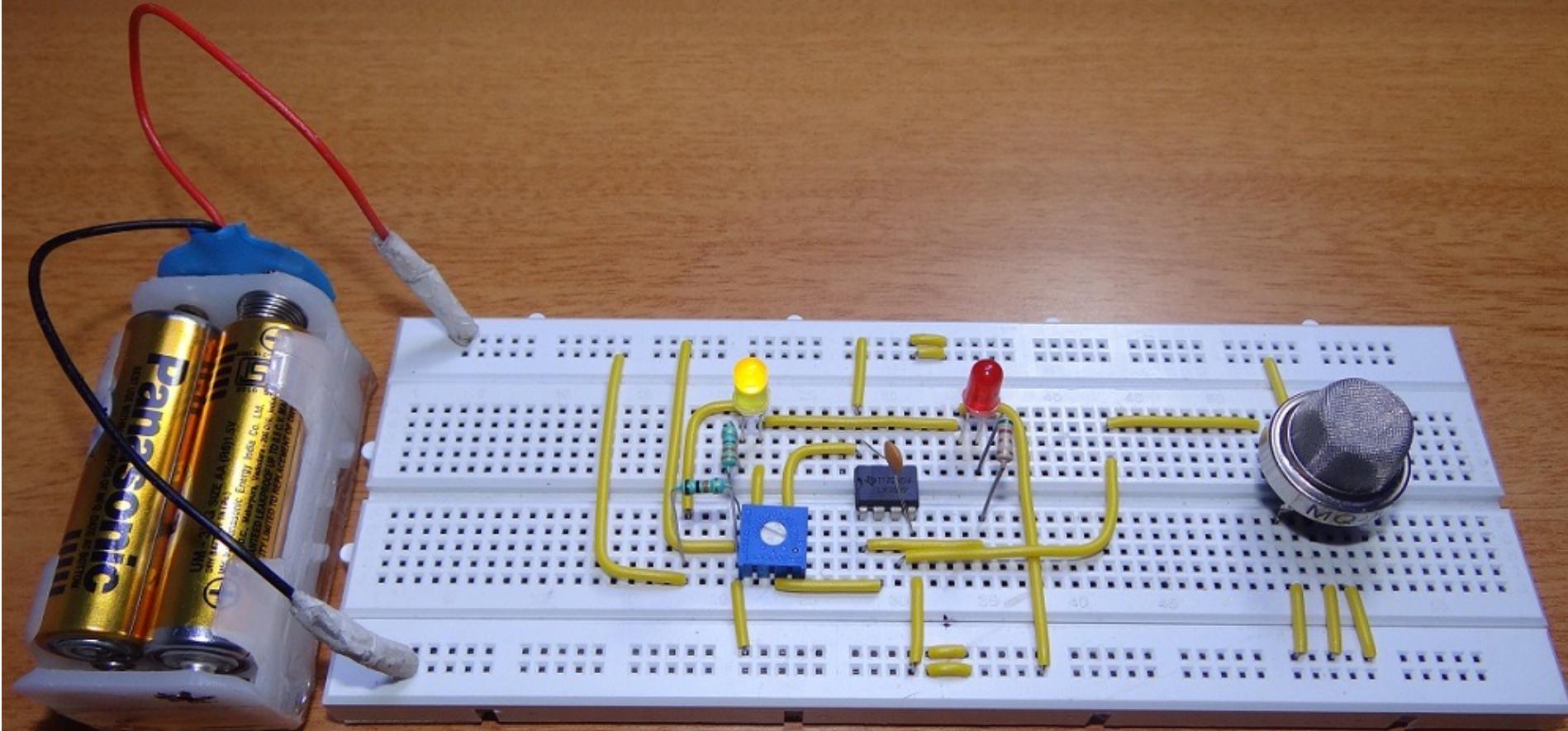
- MIPS, Watts and cost are related

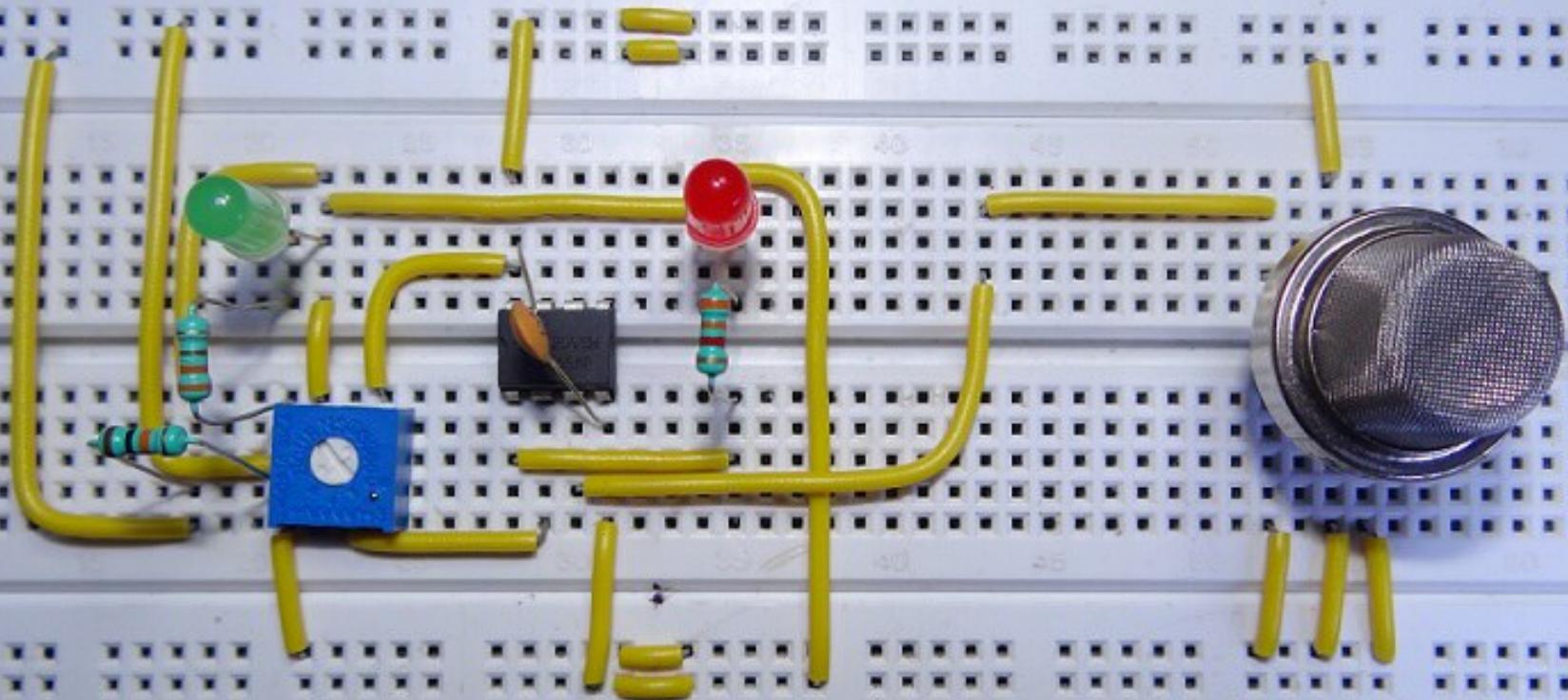
- technology driven
 - to get more MIPS for fewer Watts
 - look at the sources of power consumption
 - use power management and voltage scaling

Projects review

- The Hatcher
- Cell Phone Towers
- Networks







Microcontroller Refreshment

What is a microcontroller?

- A microcontroller is an integrated circuit that is programmed to do a specific task.
- Microcontrollers are really just “mini-computers”.

Where do you find them?

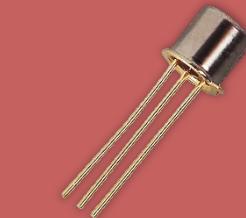
- Microcontrollers are hidden in tons of appliances, gadgets, and other electronics.
 - They're everywhere!



History of Microcontrollers



Vacuum Tube
1939



Transistor
1947



Logic Gate
1960



Microcontroller
1971

Microprocessor –vs- Microcontroller

“Is there a difference between a microprocessor and microcontroller?”

CISC - vs - RISC

+
-
*
÷

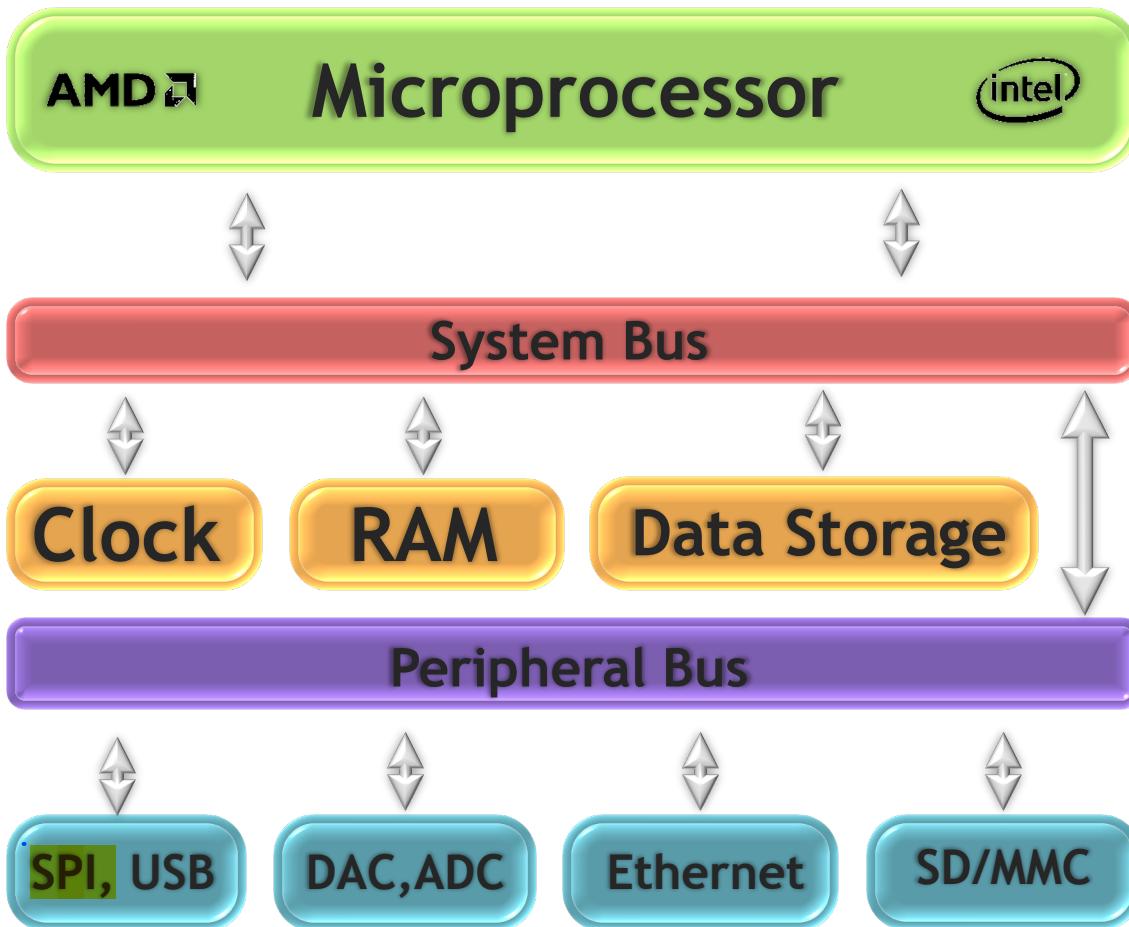


Microprocessor –vs- Microcontroller

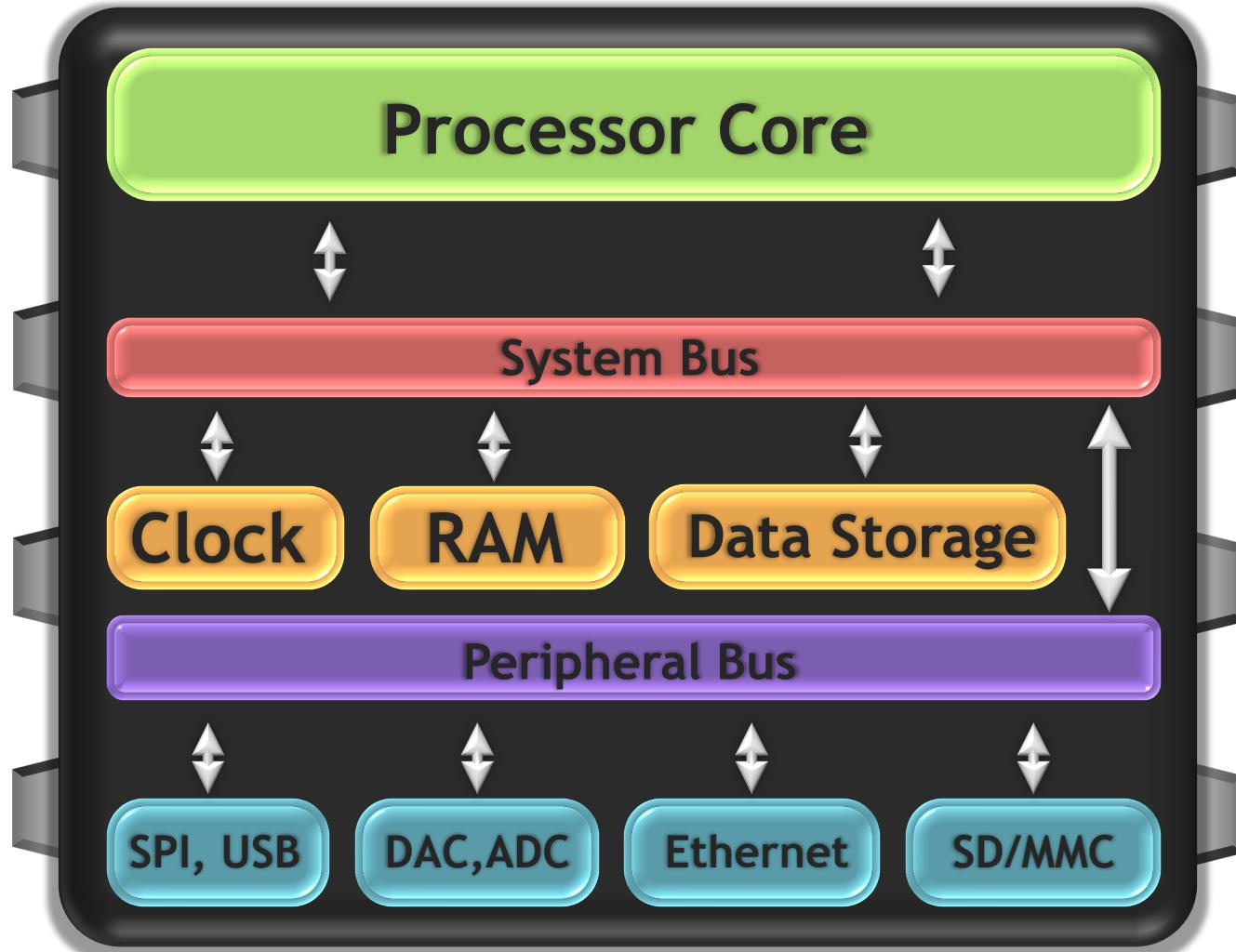
| CISC | RISC |
|---|---|
| Emphasis on hardware | Emphasis on software |
| Includes multi-clock | Single-clock |
| complex instructions | reduced instruction only |
| Memory-to-memory: “LOAD” and “STORE” incorporated in instructions | Register to register: “LOAD” and “STORE” are independent instructions |
| high cycles per second, Small code sizes | Low cycles per second, large code sizes |

| | Microprocessor | Microcontroller |
|----------------|---|---|
| Applications | General computing (i.e. Laptops, tablets) | Appliances, specialized devices |
| Speed | Very fast | Relatively slow |
| External Parts | Many | Few |
| Cost | High | Low |
| Energy Use | Medium to high | Very low to low |
| Vendors |    |     |

Microprocessor



Microcontroller



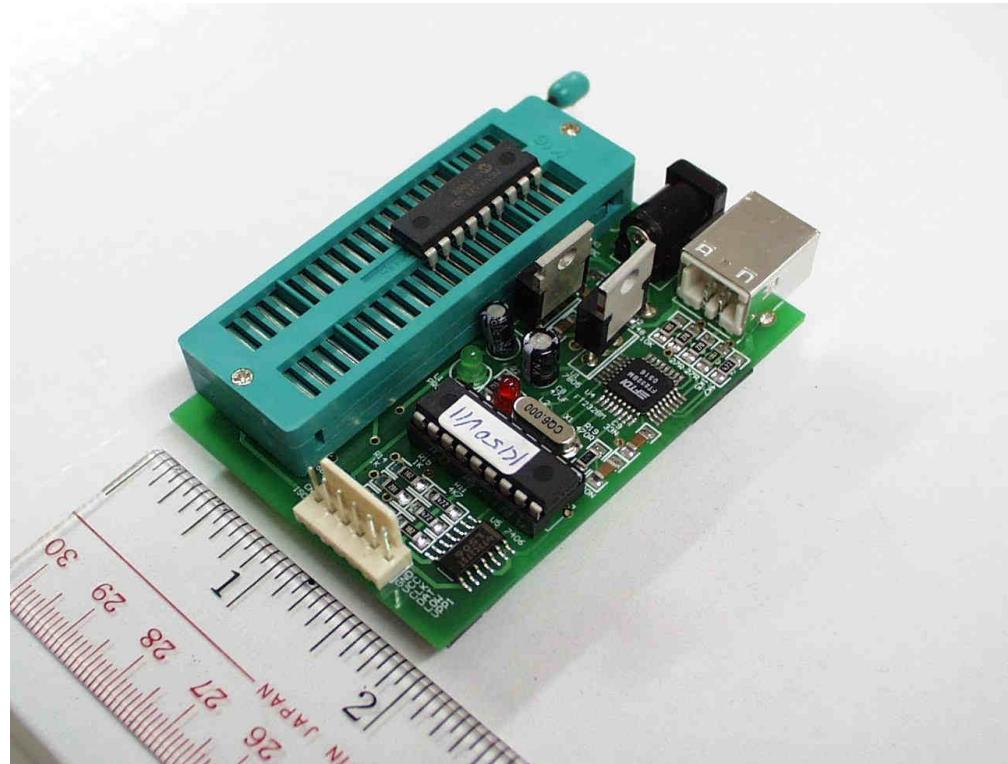
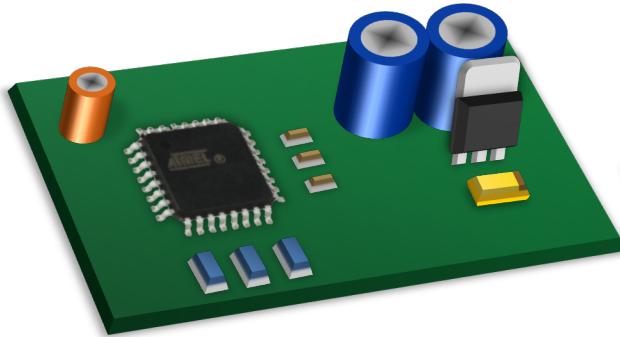
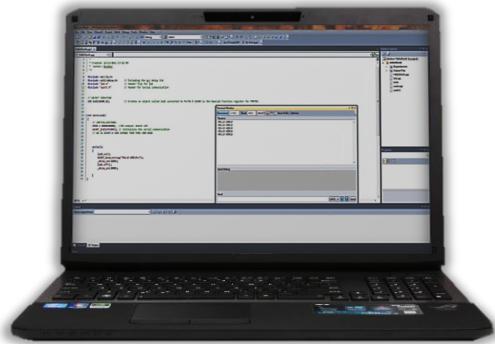
Basic Principles of Operation

- ✓ • Microcontrollers are used for **specific applications**.
- ✓ • They do not need to be **powerful** because most applications only require a **clock** of a **few MHz** and small amount of storage.
- ✓ • A microcontroller needs to be **programmed** to be useful.
- ✓ • A microcontroller is only as useful as the code written for it. If you wanted to turn on a red light when a temperature reached a certain point, the programmer would have to **explicitly specify** how that will happen through his code.

Microcontroller Programming

- 1.) Code is written for the microcontroller in an integrated development environment, a PC program. The code is written in a programming language. (e.g. C, BASIC or Assembly).
- 2.) The IDE debugs the code for errors, and then compiles it into binary code which the microcontroller can execute.
- 3.) A programmer (a piece of hardware, not a person) is used to transfer the code from the PC to the microcontroller. The most common type of programmer is an ICSP (In-circuit serial programmer).

Microcontroller Programming



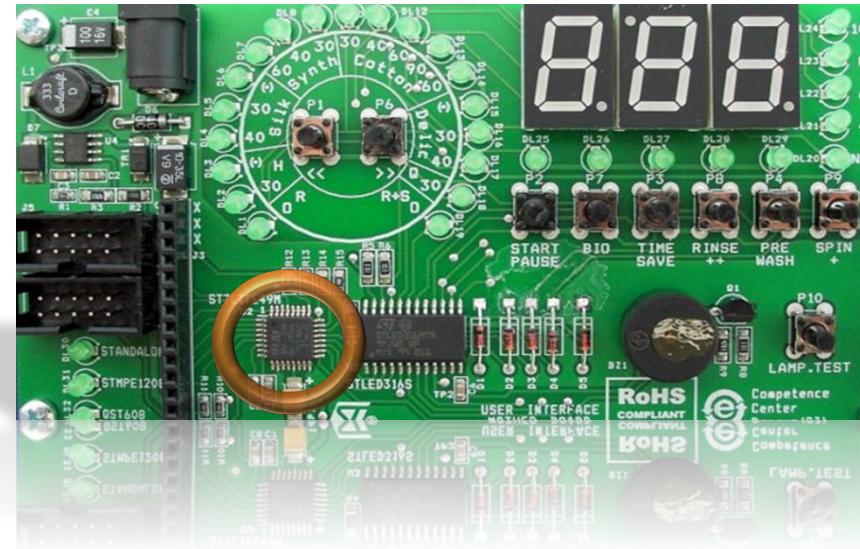
The Analog to Digital Converter (ADC)

- Just about every modern microcontroller contains an ADC(s).
- It converts analog voltages into digital values.
- These digital representations of the signal at hand can be analyzed in code, logged in memory, or used in practically any other way possible.

Microcontroller Applications

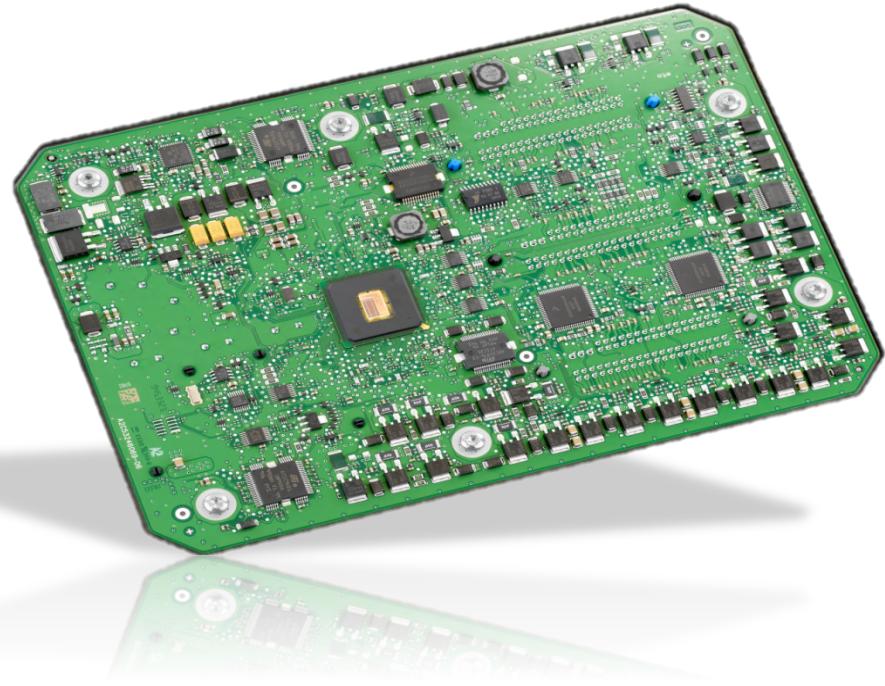
- This is the controller board for a washing machine. If a button is pushed or if a knob is turned, the microcontroller knows how to react to the event.

- Ex. If “start” is pushed, the microcontroller knows to switch a relay which starts the motor.



Microcontroller Applications

- This is the main controller from a Buick Regal. This board has several microcontrollers each for a specific task.
- Ex. A microcontroller may handle dashboard controls or it may even control something more complex like the ignition system.



Microcontroller Applications

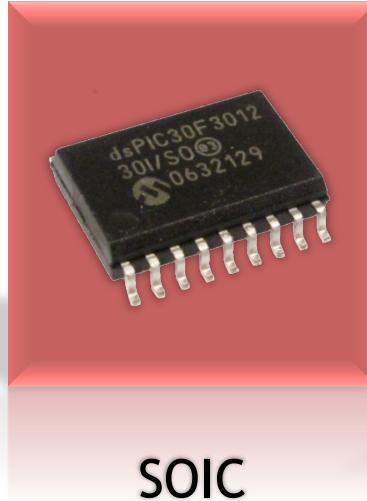
- Many robots use microcontrollers to allow robots to interact with the real world.
- Ex. If a proximity sensor senses an object near by, the microcontroller will know to stop its motors and then find an unobstructed path.



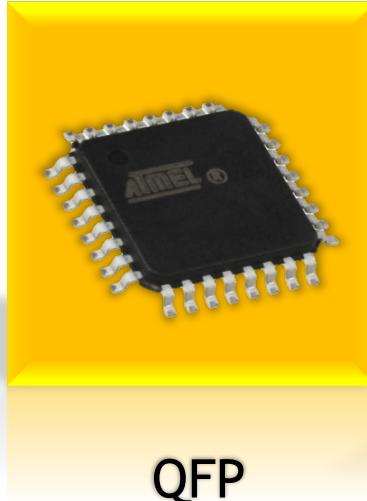
Microcontroller Packaging



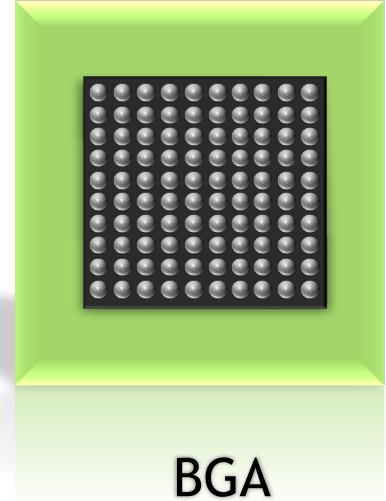
DIP
(Dual Inline Package)
Through hole
8 pins
9mm x 6mm
0.15pins/mm²



SOIC
(Small Outline IC)
Surface Mount
18 pins
11mm x 7mm
0.23pins/mm²



QFP
(Quad Flat Package)
Surface Mount
32 pins
7mm x 7mm
0.65pins/mm²



BGA
(Ball Grid Array)
Surface Mount
100 pins
6mm x 6mm
2.78pins/mm²

Lab Kits

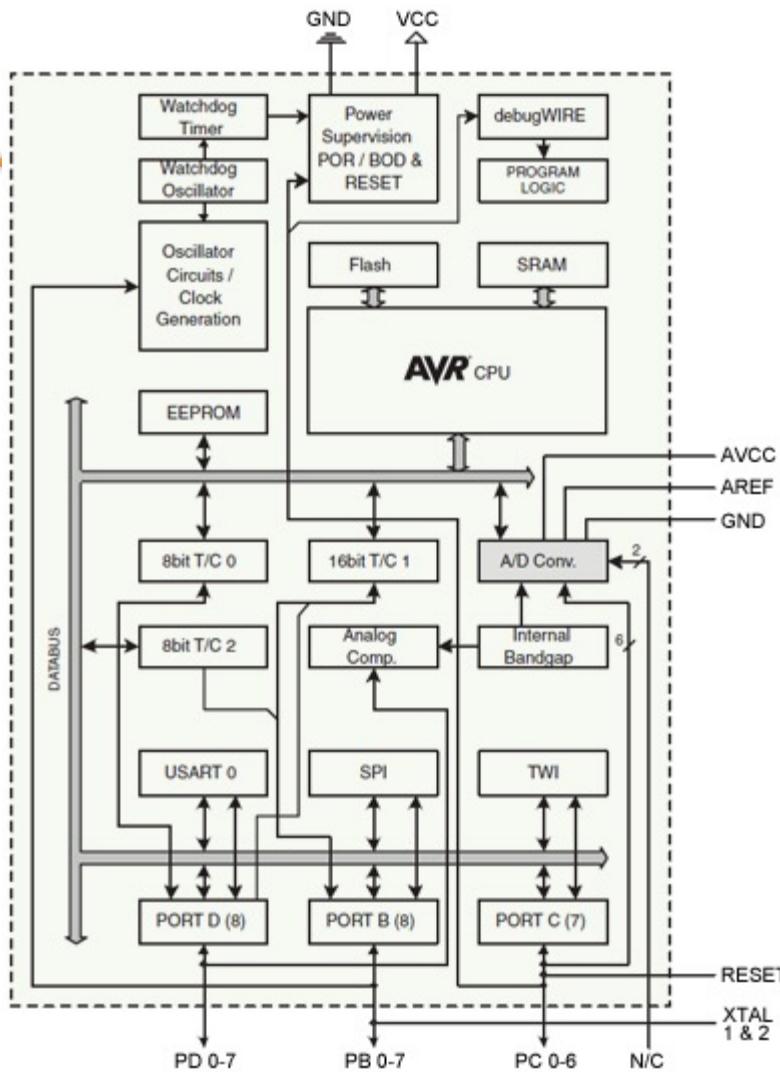
Arduino Uno
Microcontroller Chip
Atmega 328p



Atmega 328p

The Insides

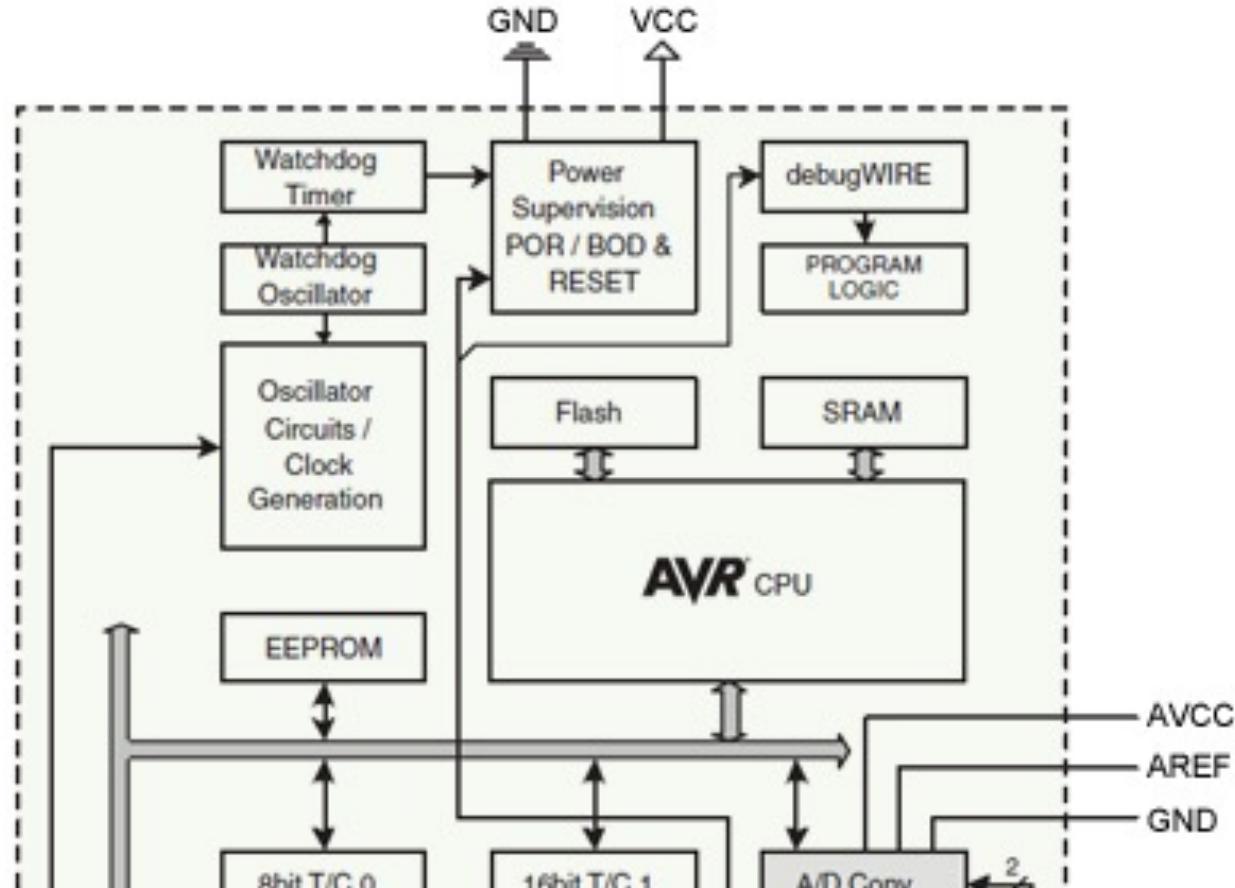
The Data Sheet



Atmega 328p

The Insides

The Data Sheet



Atmega 3

The Insides

The Data Sheet

