Lecture 1: Course Introduction

ME/AE 6705

Introduction to Mechatronics

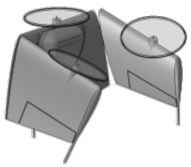
Dr. Jonathan Rogers

My Background

- Personal
  - Professor at GT since 2013
  - Washington, DC metro area
  - Wife: Hillary
- Education
  - B.S. from Georgetown Univ. (Physics, History, Math) 2006
  - M.S. from Georgia Tech (Aerospace Eng.) 2007
  - Ph.D. from Georgia Tech (Aerospace Eng.) 2009
- Other
  - FAA-Licensed commercial pilot
  - Hobbies: running, guitar, baseball, flying

Aerial Robotics and Experimental Autonomy Lab

- Research at the intersection of dynamics, control and estimation, and vehicle design.



- **Robot Vehicle Design and Flight Dynamics**
  - *Design of novel autonomous vehicle concepts*
  - *Analysis and tradeoffs of various vehicle designs*
  - *Ex: Modular vertical lift aircraft*

- **Control and Estimation for Complex Systems**
  - *Fault-tolerant control, cooperative control*
  - *System identification methods*
  - *Ex: Expert system controller for helicopter autorotation*
  - *Ex: Information-Theoretic system identification*

- **Drone Prototyping and Flight Testing**
  - *Px4 autopilot stack, ROS2 integration*
  - *Ex: Drone autonomous landing on a moving ground vehicle*

Georgia Tech

Georgia Tech

What is Mechatronics

- Mechatronics is the synergistic integration of **mechanical engineering, electronics,** and **control theory** in the design and manufacturing of products and processes

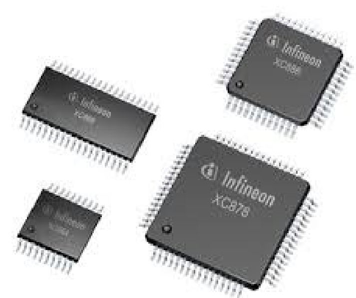
- Mechatronics uses a balance of **theoretical analysis** and **hardware implementation** in system design

What is Mechatronics?

- In simpler terms, Mechatronics studies the intersection between:
  - Sensors
  - Actuators
  - Microprocessors
  - Software
  - Control Theory

Hardware Elements

Software Elements



Mechatronics vs. Robotics

- What is the difference between Mechatronics and Robotics?

Mechatronics vs. Robotics

- What is the difference between Mechatronics and Robotics?
  - Robotics encompasses broad range of theoretical and applied areas of study
    - *Path planning, vision systems, mechanical design, controls, dynamic analysis, vibration, etc.*
  - Mechatronics can be seen as <u>subfield</u> of Robotics dealing with hardware-software integration, with emphasis on actuators and sensors

What This Class Is

- Will learn theory and implementation behind microprocessor control of mechanical systems
- Strong emphasis on understanding electrical and software fundamentals
  - Toolset learned in this class should be *portable* across microprocessor families
- Learn both system modeling and control system design
  - Model real-world systems, design controllers
  - Design and implement mechatronic device

What This Class Is Not

- Class is not about integrating open source hardware and software components
- Class is not about using arduino
- Class is not about learning to be a "Maker"

- *Focus of this class is on understanding underlying fundamentals of electro-mechanical interface, software, <u>in rigorous engineering context</u>*

Skills You Will Need

- Understanding of basic electrical components – what they do, how they are used
  - Resistors, capacitors, transistors, op-amps, etc.
- Understanding of basic electrical circuits
  - I will “refresh your memory” as needed
- Rigid body dynamics
- Ordinary differential equations
  - Formulate and solve
- Laplace transforms, basic linear controls

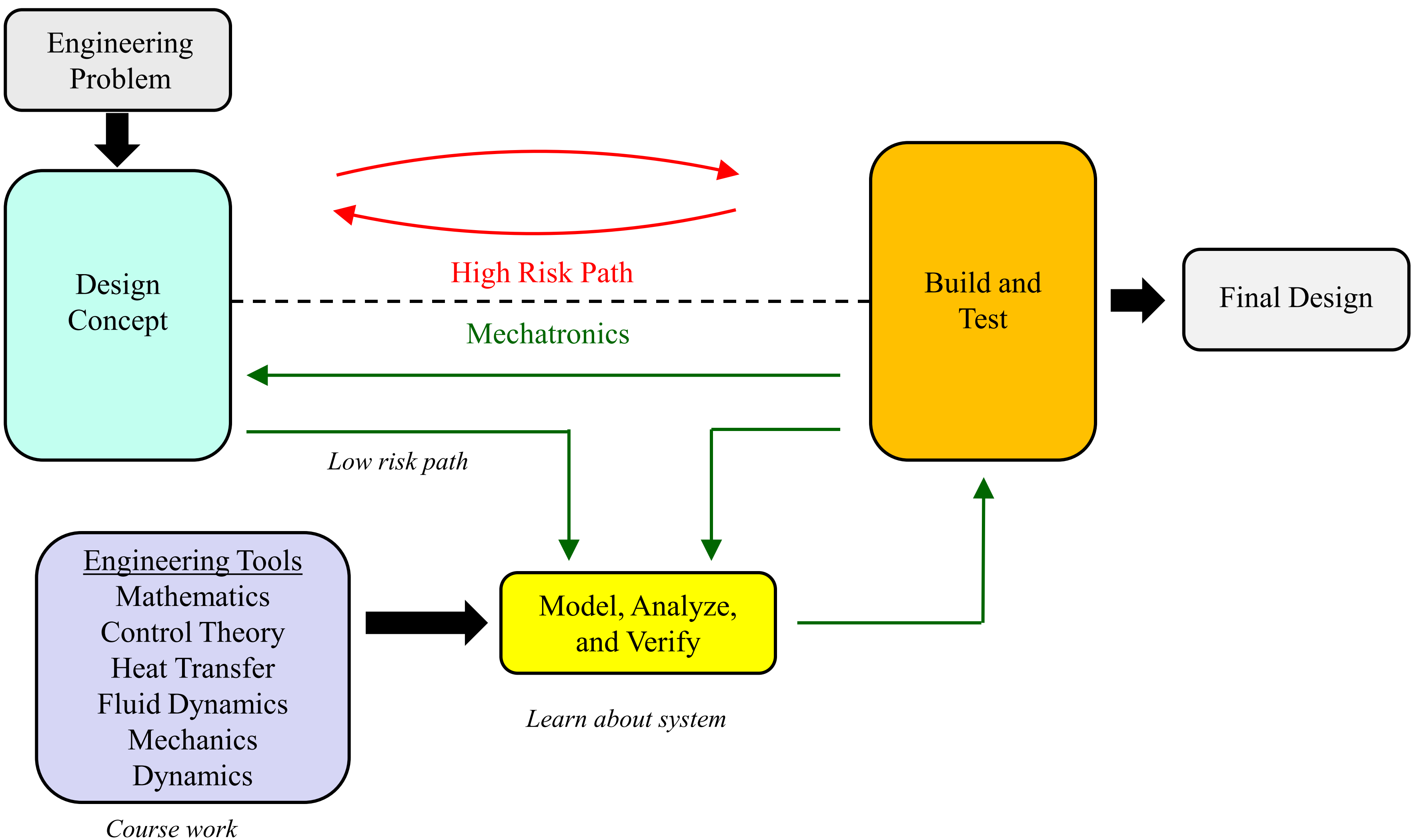
Skills You Will Learn

- C language programming
  - For embedded systems
- Designing and constructing signal conditioning circuits
  - Soldering!
- Interdevice communications
  - Sensor to processor, processor to actuator, processor to computer, computer to processor
- Control system and filter implementation
  - PID control implementation

Synthesis of Analysis and Hardware Design in Mechatronics

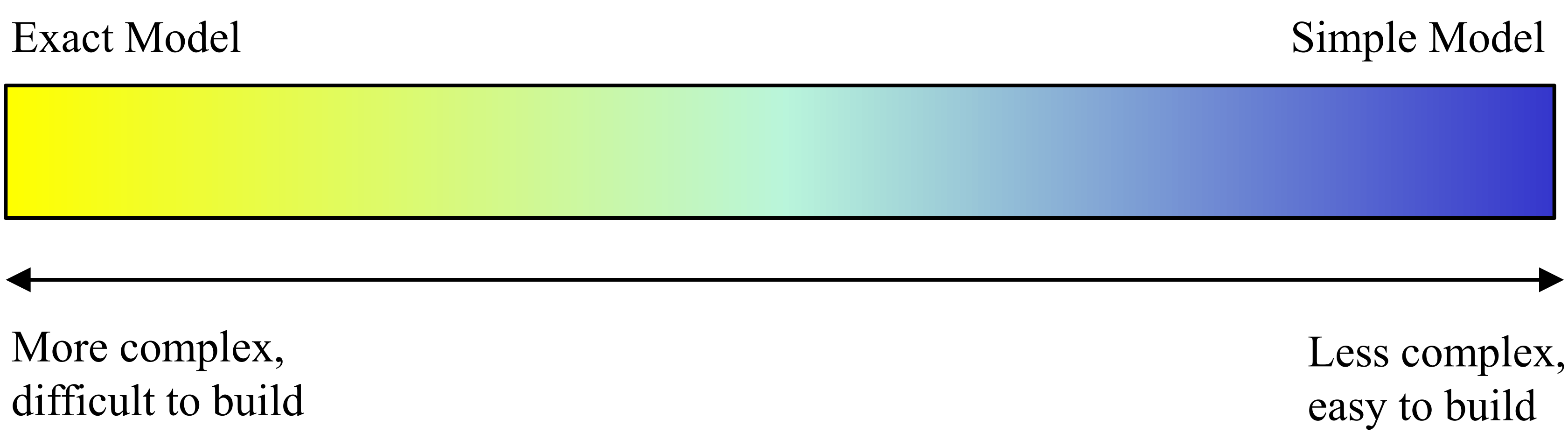
- Balance between **engineering analysis** and **hardware experimentation** is critical to success in Mechatronics
  – Engineering analysis = methods you have learned in classes
- Your ability to perform rigorous engineering analysis is what separates you from hobbyists, makers, etc.
  – Mathematics should be viewed as tool to improve your design rather than something to be avoided

Synthesis of Analysis and Hardware Design in Mechatronics

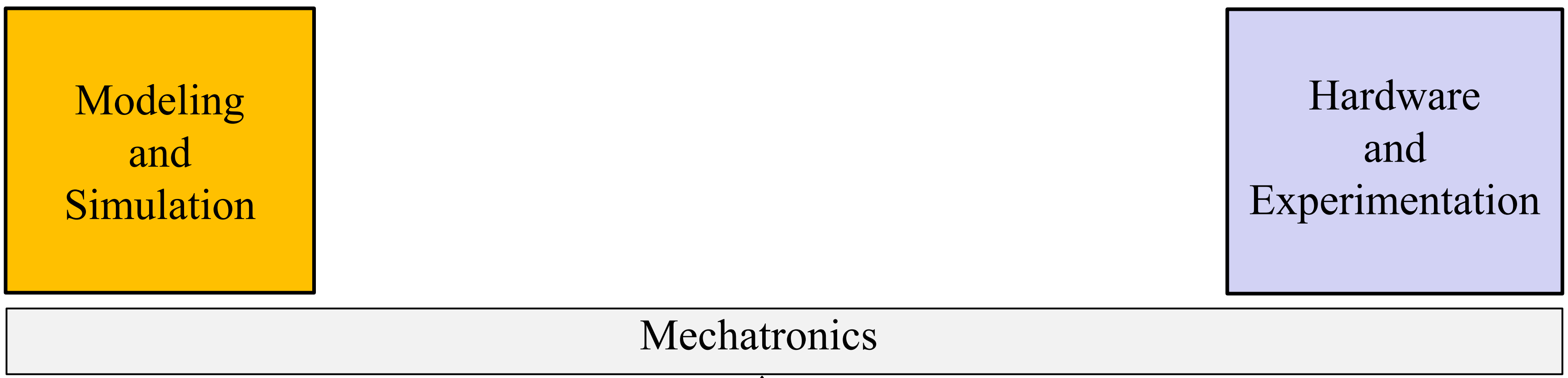


System Modeling

- System modeling is critical aspect of Mechatronics
- Balancing model complexity and utility is important
  - If extremely complex, can be too time consuming and not work effort
  - If too simple, will not accurately represent system
  - Desire model that <u>captures all relevant dynamics</u>



Balance of System Modeling and Hardware Experimentation



Computer simulation without experimental verification is at best misleading, and at worst meaningless!

Hardware development without analysis is at best time consuming, and at worst useless!

Georgia Tech

ME/AE 6705 Course Components

System Modeling/Analysis

- Integer and floating point mathematics
- Circuit analysis/design
- PID control theory
- System block diagrams
- Laplace transform analysis
- Digital control
- Filtering algorithms

Hardware Implementation

- Microcontroller architectures
- C programming
- Serial communications
- Analog-to-digital conversion
- PWM actuator control
- Sensor characterization and signal conditioning
- Interrupts and clocks
- Memory structures

Georgia Tech

What is a Microcontroller?

- A microcontroller (MCU, short for microcontroller unit) is a "small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals"

- "Microcontrollers are designed for **embedded applications**, in contrast to the microprocessors used in personal computers or other general purpose applications."

Microcontroller vs Microprocessor

Microcontroller

- Includes input and output devices (serial comm, analog-to-digital converters, GPIO, etc.)
- Slower clock rate
- Smaller memory (kB’s)
- A lot cheaper (~$10’s)
- Low power

Microprocessor

- Very high clock rates (low latency operation)
- Large memory spaces (GB’s)
- More expensive (~$100’s)
- Higher power
- Does not include input/output

What is a Microcontroller?

- Embedded Applications vs Computers

Use Microcontrollers

Use Microprocessors

What is a Microcontroller?

- Microcontroller is just integrated circuit – with lots of pins!

- For prototyping purposes, kind of useless
- Only useful if you design a board and include pads for this chip

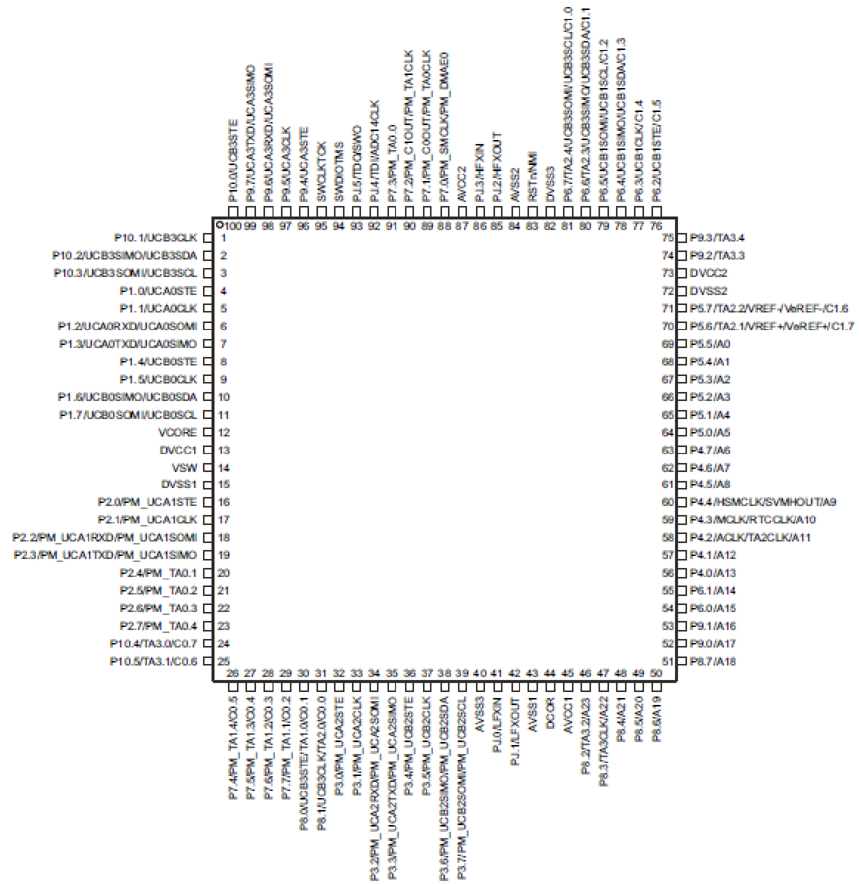
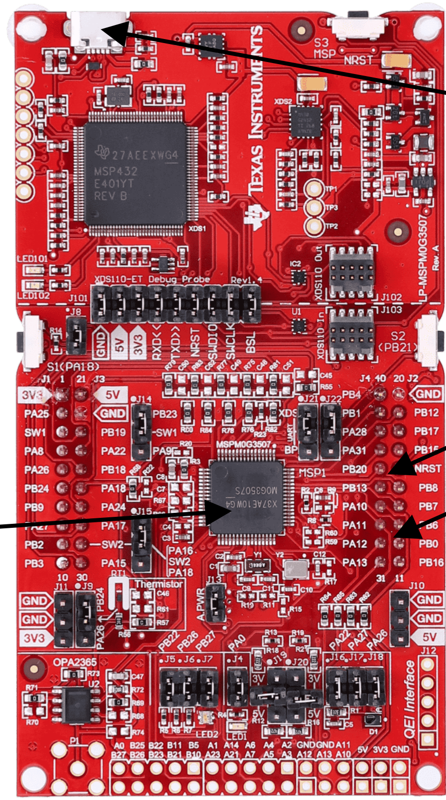


Figure 4. MSP432P401RIPZ Pinout

What is a Microcontroller?

- When incorporated into development kit, MCU becomes very useful for prototyping.

“Launchpad” Development Board



USB connection to connect to PC

Pins used to connect sensors, power, etc to MCU

TI MSPM0 32-bit microcontroller

What Does n-bit Processor Mean?

- Microcontrollers (MCUs) come in many models which may be 8-bit, 16-bit, or 32-bit
  - Microprocessors are all 64-bit

What Does n-bit Processor Mean?

- Microcontrollers (MCUs) come in many models which may be 8-bit, 16-bit, or 32-bit
  - Microprocessors are all 64-bit
- MCUs use data in 1-byte (8-bit) chunks, perform operations on groups of bytes called **words**
  - A 32-bit MCU uses a word size of 4 bytes
  - Word size is width of “bus” that passes data within MCU

What Does n-bit Processor Mean?

- Microcontrollers (MCUs) come in many models which may be 8-bit, 16-bit, or 32-bit
  - Microprocessors are all 64-bit
- MCUs use data in 1-byte (8-bit) chunks, perform operations on groups of bytes called **words**
  - A 32-bit MCU uses a word size of 4 bytes
  - Word size is width of “bus” that passes data within MCU
- All operations and memory accesses are performed on words
  - Thus, 32-bit processor can get data from memory faster since it can grab it in 4-byte chunks

What Does n-bit Processor Mean?

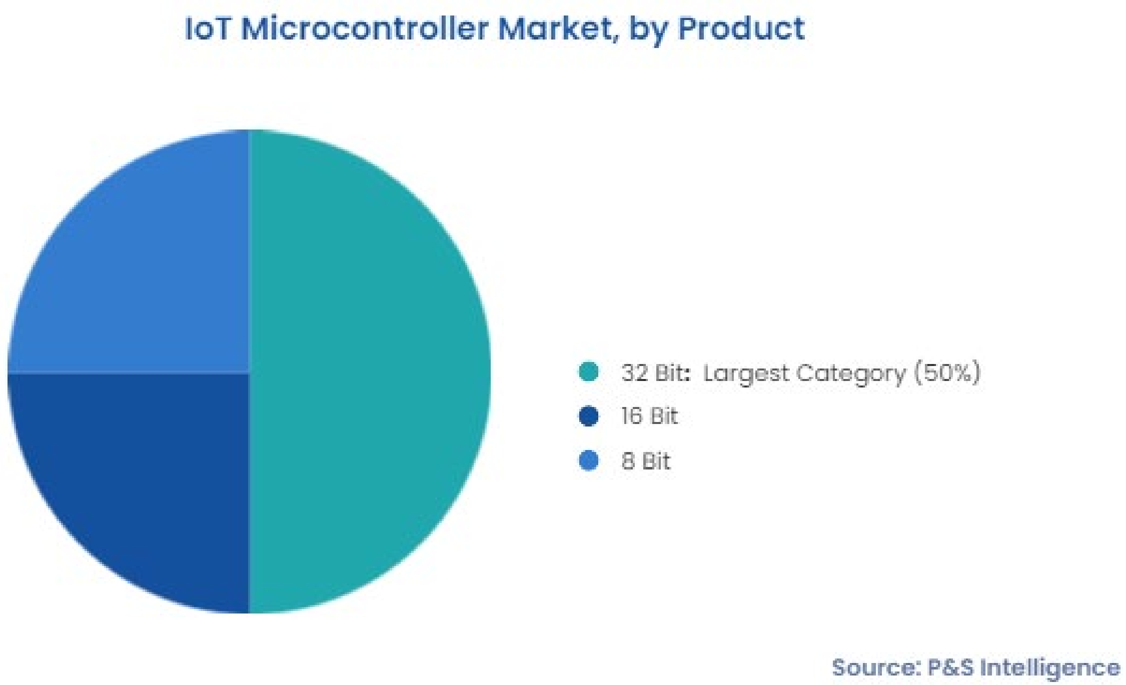
- Furthermore, registers in MCU are size of word
  - 32-bit MCU stores floating point (decimal) numbers using 32 bits – is precise to about 6 decimal places
  - 8-bit MCU stores floating point (decimal) numbers using 8 bits – cannot adequately represent a floating point number
  - *Thus, 32-bit MCU is both <u>faster</u> and <u>more precise at arithmetic</u> than 8-bit processor*

What Does n-bit Processor Mean?

- Higher precision (32-bit) processors are:
  - Higher performance (speed, latency of operations)
  - More expensive
  - Can draw more power
  - Can be more complex to program
- Lower precision (8-bit) processors are:
  - Less capable/lower performance
  - Less expensive
  - Can draw less power
  - Can be less complex to program

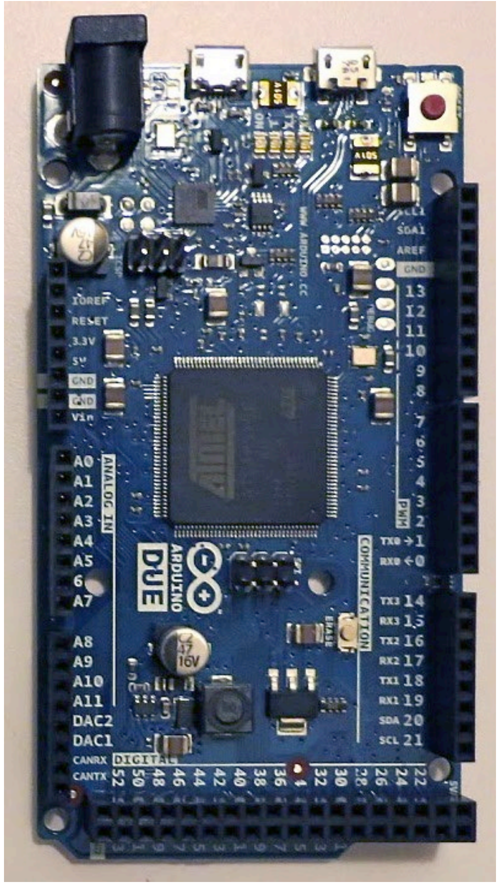
What Does n-bit Processor Mean?

- Market share of different types of MCUs (2024)



Various MCU Development Kits

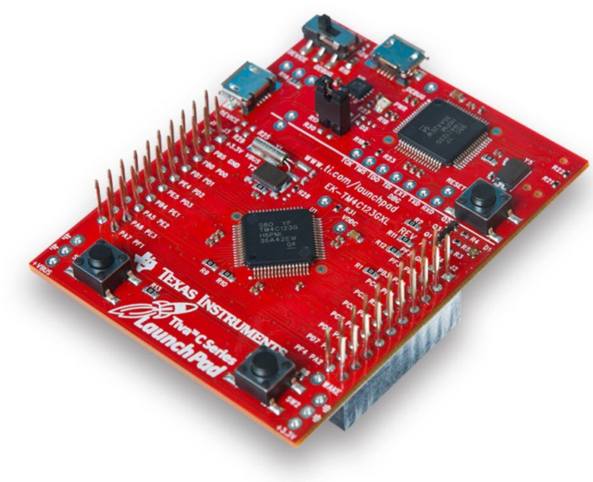
- Arduino Due



Manufacturer	Arduino (board) / Atmel (MCU)
Microcontroller	AT91SAM3X8E
Architecture	ARM 32Bit Cortex-M3
Speed	84 Mhz Max
SRAM	96 KBytes
Flash	512 KBytes
Cost	$36-$50
Software Development	ATmel Studio 6 (free)
Debugger	Additional ($60-$100)
Additional Sensors	Available as multiple shields
Pros	Popular platform. Useful to students Free Professional IDE Direct Support from ATmel
Cons	More Expensive No Onboard debugger No Onboard buttons/sensors No DSP Extension Shield Libraries written in Arduino Code

Various MCU Development Kits

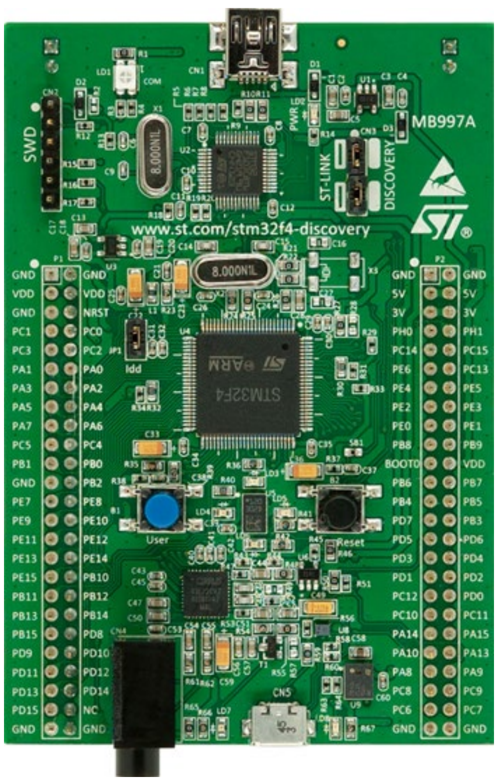
- Texas Instruments Tiva C Launchpad



Manufacturer	Texas Instruments (board + MCU)
Microcontroller	TM4C123GH6PM
Architecture	ARM 32Bit Cortex-M4
Speed	80 Mhz Max
SRAM	32 KBytes
Flash	256 KBytes
Cost	$12
Software Development	Code Composer Studio (free for TIVA C)
Debugger	On-board debugging functionality
Additional Sensors	Available as single booster shield ($50)
Pros	Low Cost Professional IDE for free Onboard buttons Sensor expansion pack On-board debugger FPU Core
Cons	Not a popular prototyping tool or popular hobbyist/entry level platform

Various MCU Development Kits

- STM32F4 Discovery



Manufacturer	ST (board + MCU)
Microcontroller	STM32F407
Architecture	ARM 32Bit Cortex-M4
Speed	180 Mhz Max
SRAM	192 KBytes
Flash	1 MBytes
Cost	$15
Software Development	Keil
Debugger	On-board debugging functionality
Additional Sensors	Available as multiple shields (Not relevant shields. Only Camera, LCD, Wifi)
Pros	Low Cost Accelerometer MEMS and Switches on board Onboard Debugger FPU Core
Cons	Professional IDE limited by size Not popular as prototyping tool

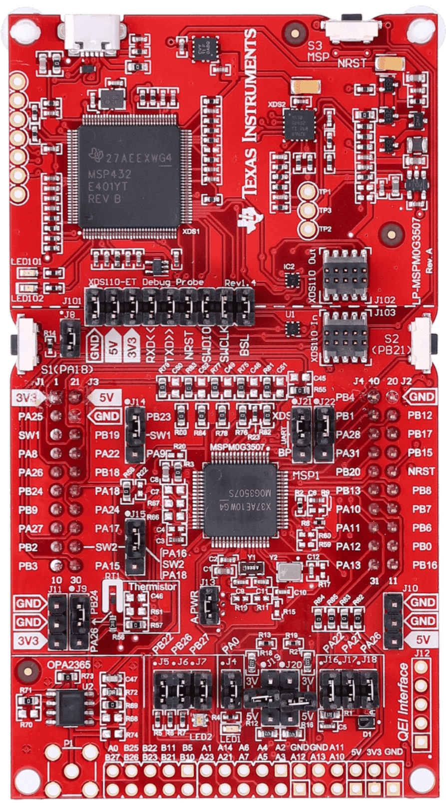
Various MCU Development Kits

- Microchip PIC18FJ Development Board

Manufacturer	Microchip (board + MCU)
Microcontroller	PIC18F87J50
Architecture	8-bit PIC
Speed	48 MHz
SRAM	3 KBytes
Flash	128 KBytes
Cost	$39
Software Development	MPLAB IDE
Debugger	PC-based debugging
Additional Sensors	None.
Pros	Simple, entry level board to learn the PIC architecture Relatively straightforward to program
Cons	Limited processor capability

Various MCU Development Kits

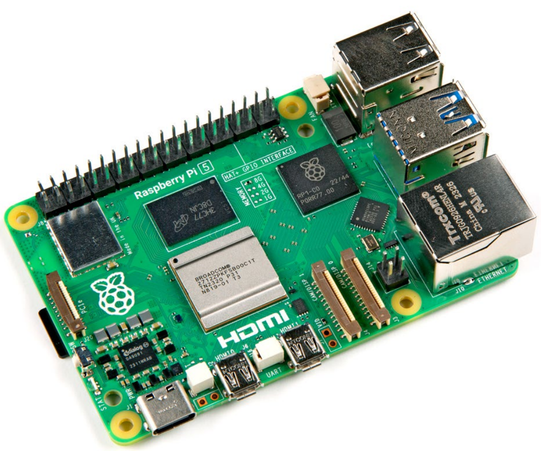
- Texas Instruments LP-MSPM0G3507 Launchpad



Manufacturer	Texas Instruments (board + MCU)
Microcontroller	MSPM0G3507
Architecture	ARM 32Bit M0+
Speed	80 MHz
SRAM	32 KBytes
Flash	128 KBytes
Cost	$17
Software Development	Code Composer Studio (CCS)
Debugger	On-board Debugger
Additional Sensors	Available as Booster pack modules from TI
Pros	32-bit processor with large memory capacity Relatively easy to use IDE and support
Cons	Relatively new, not a lot of open-source code available

Embedded (Single-Board) Computers

- Another class of embedded computers have been developed, called “single-board” computers
- These have similar functionality to a laptop, but in a very small package
- They run complete operating system with a “Desktop”
- **Raspberry Pi 5**

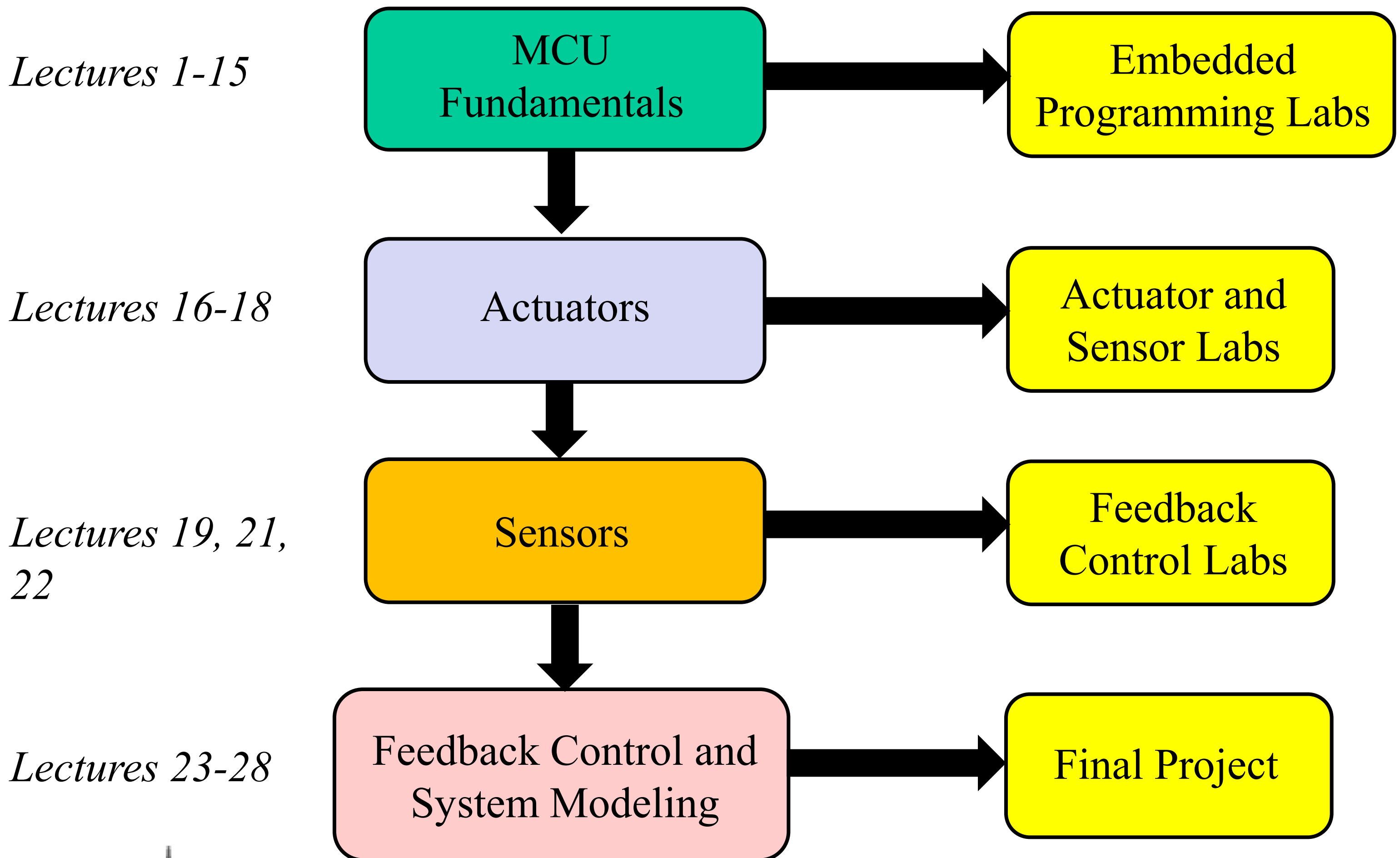


Manufacturer	Raspberry Pi Foundation
Microprocessor	64-bit Quad-Core Arm Cortex-A76 (along with GPU)
Speed	2.4 GHz
SRAM	8 GB
Storage	Micro-SD
Cost	$80
Operating System	Raspberry Pi OS (based on Debian Linux), Ubuntu, others

Embedded Computers vs MCUs

- Embedded computer advantages over MCUs
  - Provide higher performance
  - Easier to prototype – port PC code directly to it
- Embedded computer disadvantages over MCUs
  - Usually more expensive
  - Usually require more power
- *Embedded computers becoming more attractive and common in variety of <u>prototyping</u> applications where power is not big concern*
  - *Embedded computers not usually used in production systems except when very high computing power needed*

Overview of ME4405



Hands-on Learning

- Only way to really learn Mechatronics is by getting hands-on experience
- Course will involve 10 total Labs/Homeworks which will be assigned, on a weekly basis
  - All labs will be done individually, no groups
  - Purpose of labs is not to write detailed reports, but to complete task successfully (i.e., make device work)
  - Lab highlights:
    - *Measure temperature data with MCU and stream data to PC*
    - *Drive wheel system to desired speed using feedback control*

Hand-on Learning

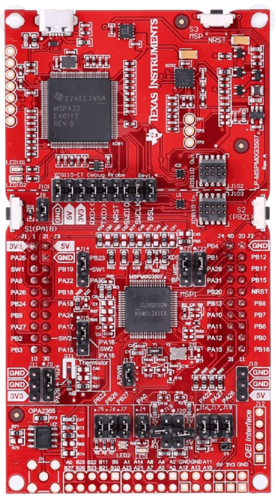
- Students will complete a final project for the course
  - All projects done individually
  - Design, construct, and demonstrate a mechatronic system
  - Must include MCU, at least one actuator, one sensor, feedback control, and some element of system modeling
  - You come up with project idea, I approve it
  - Will demonstrate operation of device to class in final “mini-expo”

Hands-on Learning

- **C programming** is key element of this course
- I will teach core elements of embedded C programming during 3 lectures at beginning of course
- Don't expect anyone to have prior knowledge of C
- Why is C language important?
  - MCU programming is all done in C
  - Will be beneficial – it is important for embedded systems engineers to know C
  - Python is used for prototyping but not production systems

Hardware to Purchase

- All students will need to purchase a Texas Instruments MSPM0G3507 Launchpad Evaluation Kit (LP- MSPM0G3507)
  - Cost is $16.99
  - Kits available from:



https://www.ti.com/tool/LP-MSPM0G3507#overview

Syllabus Review