Mechatronics at BYU: A Required Low-Level Course for Mechanical Engineers

Mark B. Colton

BYU | MECHANICAL ENGINEERING
IRA A. FULTON COLLEGE

Our Goal

Engineers who can
understand, model, design,
build, and program
dynamic mechatronic
systems

ME Mechatronics Approaches

High-Level Approach	**Low-Level Approach**
System-level integration, modeling, and control	Subsystem-level design
Commercial kits and controllers	Individual components and single-chip microcontrollers
High level of software abstraction	Low level of software abstraction

ME Mechatronics Approaches

High-Level Approach	**Low-Level Approach**
System-level integration, modeling, and control	Subsystem-level design
Commercial kits and controllers	Individual components and single-chip microcontrollers
High level of software abstraction	Low level of software abstraction

ME Mechatronics Approaches

High-Level Approach

System-level integration, modeling, and control

Commercial kits and controllers

High level of software abstraction

Low-Level Approach

Subsystem-level design

Individual components and single-chip microcontrollers

Low level of software abstraction

ME Mechatronics Approaches

High-Level Approach	**Low-Level Approach**
System-level integration, modeling, and control	Subsystem-level design
Commercial kits and controllers	Individual components and single-chip microcontrollers
High level of software abstraction	Low level of software abstraction

ME Mechatronics Approaches

High-Level Approach	**Low-Level Approach**
System-level integration, modeling, and control	Subsystem-level design
Commercial kits and controllers	Individual components and single-chip microcontrollers
High level of software abstraction	Low level of software abstraction
Students gain experience in modeling, control, and system integration	Students gain experience in hardware design and underlying software principles

ME Mechatronics Approaches

High-Level Approach

System-level integration, modeling, and control

Commercial kits and controllers

High level of software abstraction

Students gain experience in modeling, control, and system integration

Low-Level Approach

Subsystem-level design

Individual components and single-chip microcontrollers

Low level of software abstraction

Students gain experience in hardware design and underlying software principles

Our Approach

Single-chip microcontrollers instead of commercial controllers (e.g., Arduino or cRIO)

Register-level C programming instead abstracted software (e.g., MATLAB or LabVIEW)

Design instead of analysis or modeling

Circuit design (including PCBs) instead of commercial modules

Low-Level Approach

Subsystem-level design

Individual components and single-chip microcontrollers

Low level of software abstraction

Students gain experience in hardware design and underlying software principles

Our Approach

Single-chip microcontrollers instead of commercial controllers (e.g., Arduino or cRIO)

Register-level C programming instead abstracted software (e.g., MATLAB or LabVIEW)

Design instead of analysis or modeling

Circuit design (including PCBs) instead of commercial modules

Low-Level Approach

Subsystem-level design

Individual components and single-chip microcontrollers

Low level of software abstraction

Students gain experience in hardware design and underlying software principles

Our Approach

Single-chip microcontrollers instead of commercial controllers (e.g., Arduino or cRIO)

Register-level C programming instead abstracted software (e.g., MATLAB or LabVIEW)

Design instead of analysis or modeling

Circuit design (including PCBs) instead of commercial modules

Low-Level Approach

Subsystem-level design

Individual components and single-chip microcontrollers

Low level of software abstraction

Students gain experience in hardware design and underlying software principles

Our Approach

Single-chip microcontrollers instead of commercial controllers (e.g., Arduino or cRIO)

Register-level C programming instead abstracted software (e.g., MATLAB or LabVIEW)

Design instead of analysis or modeling

Circuit design (including PCBs) instead of commercial modules

Low-Level Approach

Subsystem-level design

Individual components and single-chip microcontrollers

Low level of software abstraction

Students gain experience in hardware design and underlying software principles

Our Approach

Single-chip microcontrollers instead of commercial controllers (e.g., Arduino or cRIO)

Register-level C programming instead abstracted software (e.g., MATLAB or LabVIEW)

Design instead of analysis or modeling

Circuit design (including PCBs) instead of commercial modules

Low-Level Approach

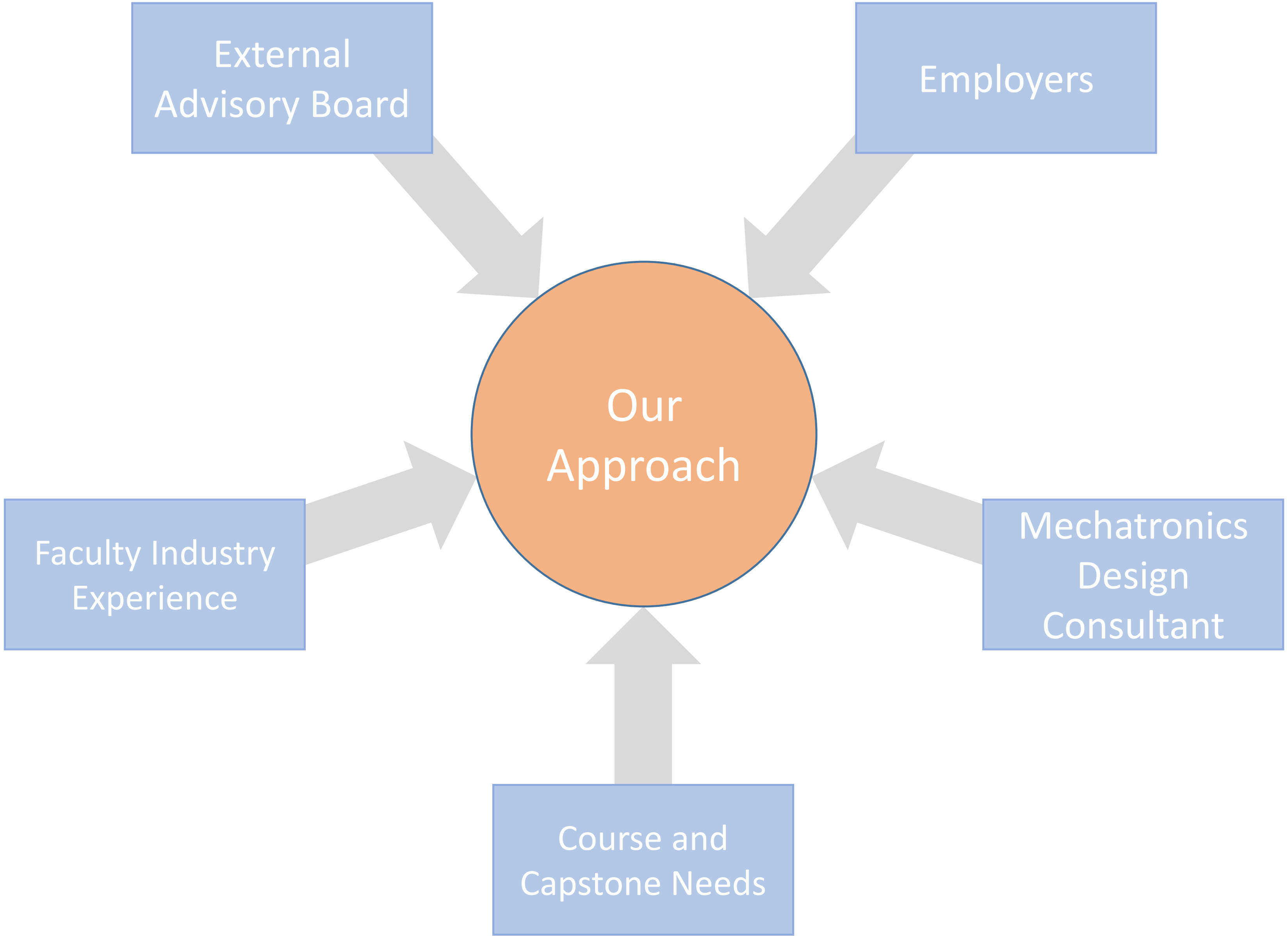
Subsystem-level design

Individual components and single-chip microcontrollers

Low level of software abstraction

Students gain experience in hardware design and underlying software principles

Justification



External Advisory Board

Employers

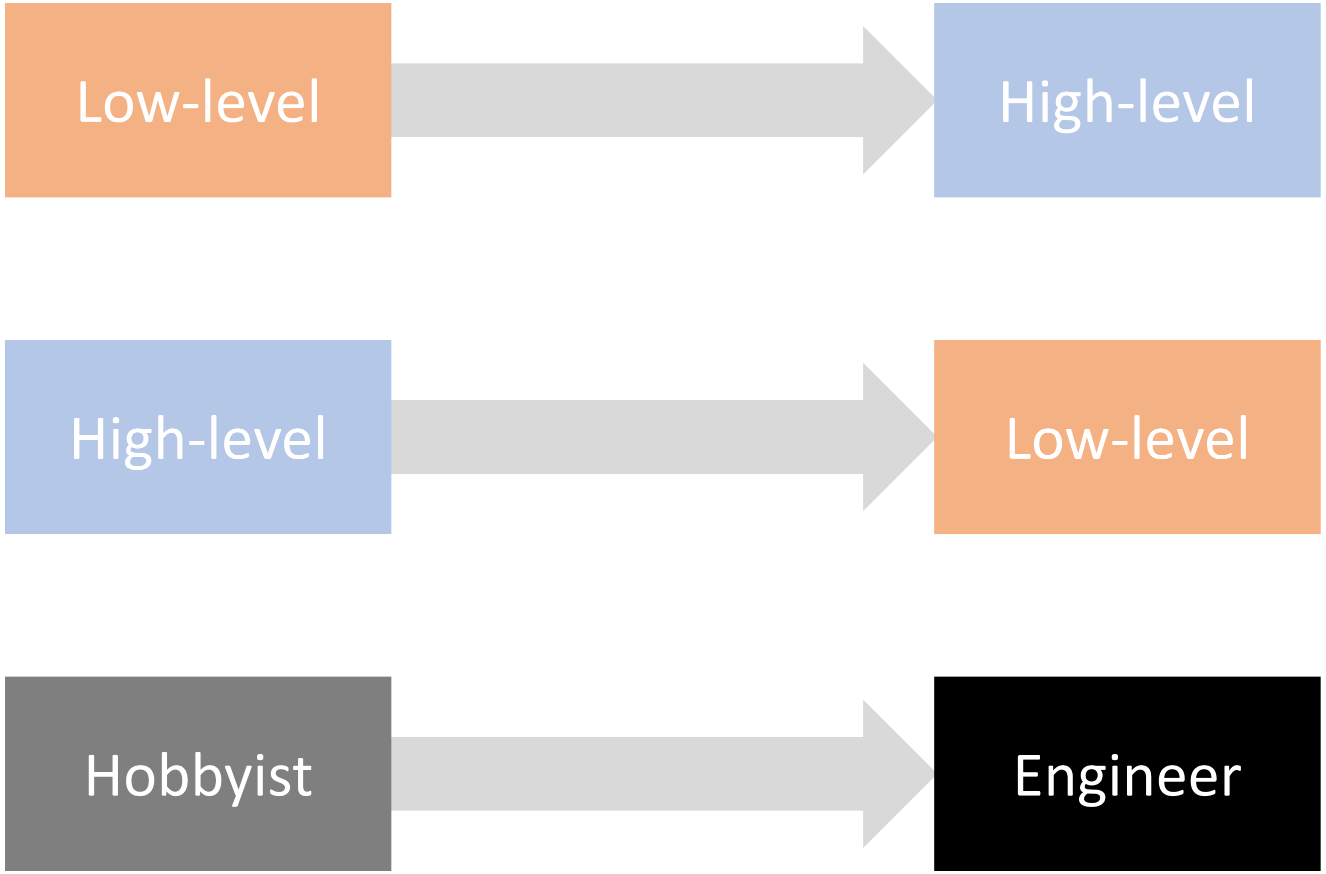
Our Approach

Faculty Industry Experience

Mechatronics Design Consultant

Course and Capstone Needs

Justification

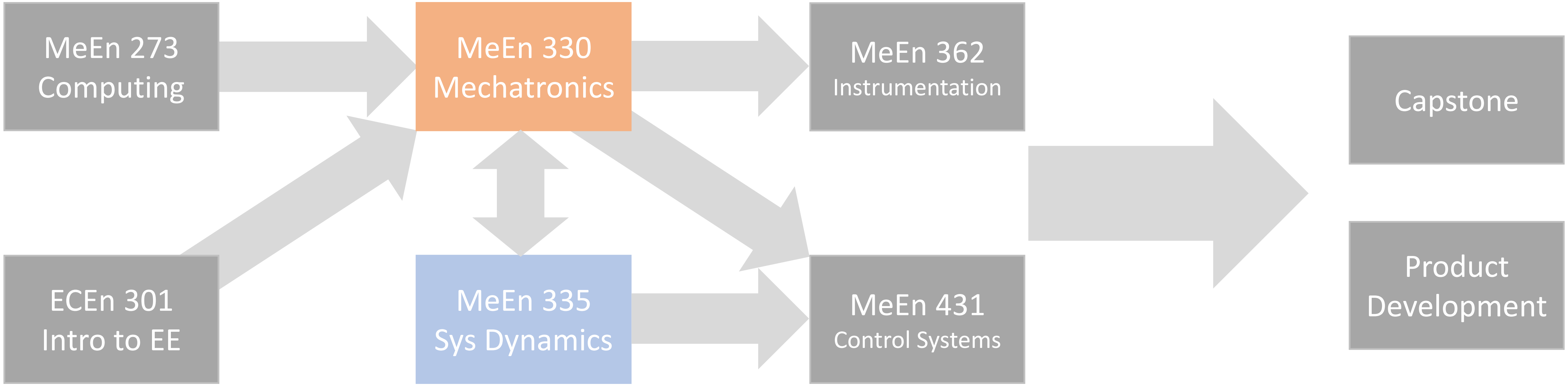


Low-level → High-level

High-level → Low-level

Hobbyist → Engineer

Our Course



MeEn 273
Computing

MeEn 330
Mechatronics

MeEn 362
Instrumentation

Capstone

ECEn 301
Intro to EE

MeEn 335
Sys Dynamics

MeEn 431
Control Systems

Product
Development

Course Outcomes

1. Students should have an understanding of microcontroller architectures, memory, and peripherals, including timers, counters, interrupts, and analog-to-digital converters.
2. Students should be able to program microcontrollers using a high-level programming language.
3. Students should know how to interface digital and analog circuits and sensors with a microcontroller.
4. Students should understand analog-to-digital and digital-to-analog conversion.
5. Students should understand basic serial and parallel communication options for microcontrollers.
6. Students should gain familiarity with various electromechanical actuators, including DC motors, stepper motors, solenoids, and servomotors.
7. Students should be able to interface motors with a microcontroller and implement motor driver circuits.
8. Students should understand and be able to implement pulse-width modulation as a method for controlling motors.
9. Students should have experience using real-world design and prototyping tools, including printed circuit board design software, breadboards, soldering, and mechanical prototyping tools.
10. Students should be able to read data sheets and select electronic components to meet design requirements.
11. Students should be able to integrate microcontrollers, electronic components, and mechanical components into a complete mechatronic system.

Course Outcomes

Microcontroller hardware and programming

Sensors, electronics, and digital and analog I/O

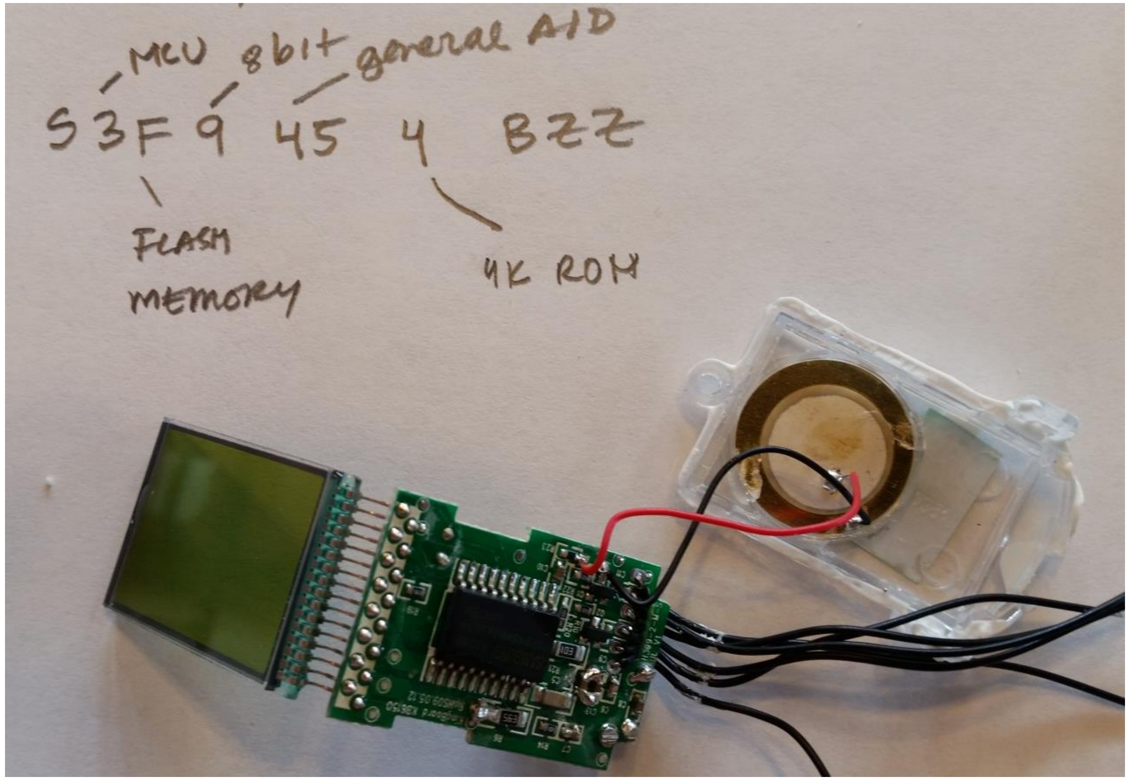
Understanding, interfacing, and driving actuators

Real-world design, component selection, and prototyping

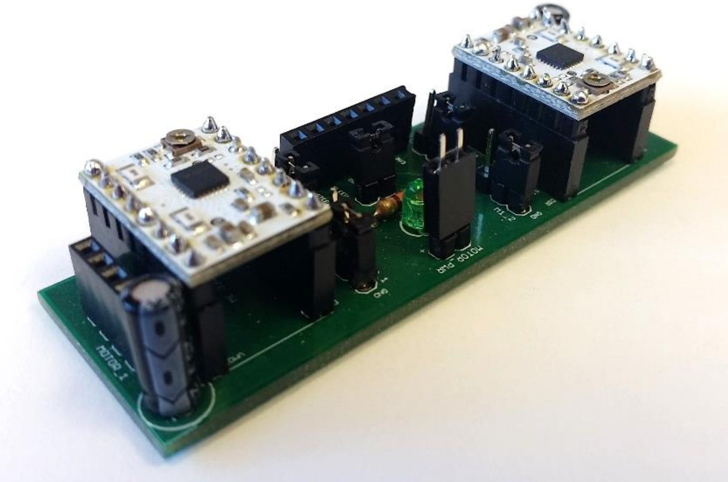
Mechatronic system integration

1. Students should have an understanding of microcontroller architectures, memory, and peripherals, including timers, counters, interrupts, and analog-to-digital converters.
2. Students should be able to program microcontrollers using a high-level programming language.
3. Students should know how to interface digital and analog circuits and sensors with a microcontroller.
4. Students should understand analog-to-digital and digital-to-analog conversion.
5. Students should understand basic serial and parallel communication options for microcontrollers.
6. Students should gain familiarity with various electromechanical actuators, including DC motors, stepper motors, solenoids, and servomotors.
7. Students should be able to interface motors with a microcontroller and implement motor driver circuits.
8. Students should understand and be able to implement pulse-width modulation as a method for controlling motors.
9. Students should have experience using real-world design and prototyping tools, including printed circuit board design software, breadboards, soldering, and mechanical prototyping tools.
10. Students should be able to read data sheets and select electronic components to meet design requirements.
11. Students should be able to integrate microcontrollers, electronic components, and mechanical components into a complete mechatronic system.


COOL
5
READY


MCU
8bit
general A/D
S3F9 45 4 BZZ
FLASH MEMORY
4K ROM

PCB Design

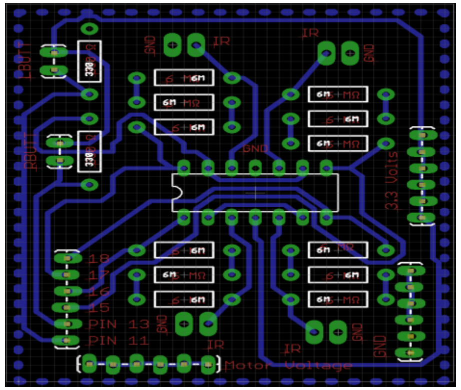


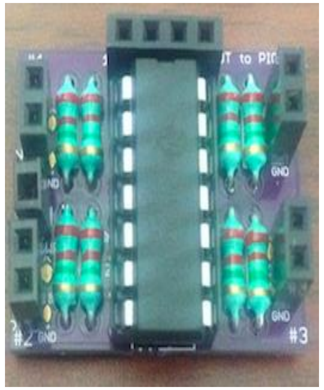
Unique for required ME course

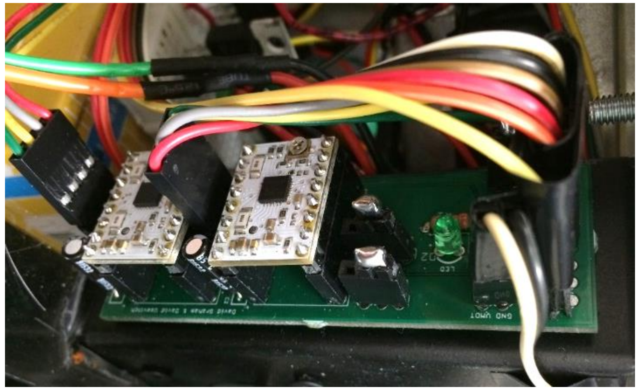
Follows industry trend

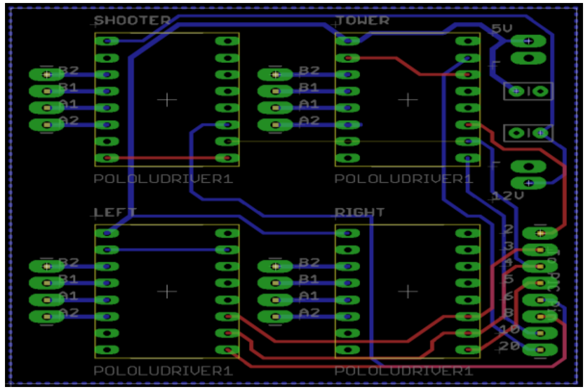
Taught first week of class, used throughout semester

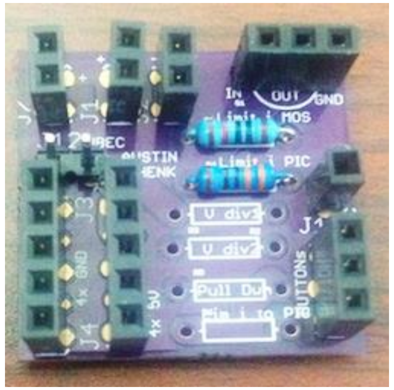
Prepares students for other “ME jobs” (thermal and vibration analysis)



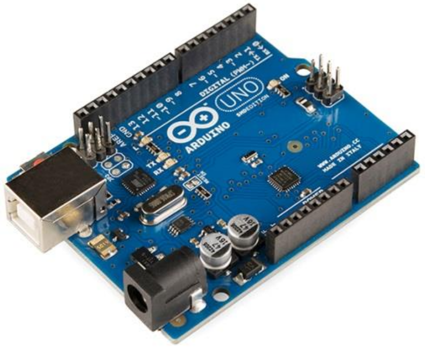








Microcontrollers



Single-chip PIC24F instead of Arduino, etc.

- Unusual (unique?) for required ME course

Students design and build their own board

Requires intimate knowledge of the hardware

- Registers
- Electrical characteristics

Why?

- Better teaches certain fundamentals
- Prepares students for product development

Microcontrollers

Single-chip PIC24F instead of Arduino, etc.

  Unusual (unique?) for required ME course

Students design and build their own board

Requires intimate knowledge of the hardware

- Registers
- Electrical characteristics

Why?

- Better teaches certain fundamentals
- Prepares students for product development

Microcontrollers

Single-chip PIC24F instead of Arduino, etc.

Unusual (unique?) for required ME course

Students design and build their own board

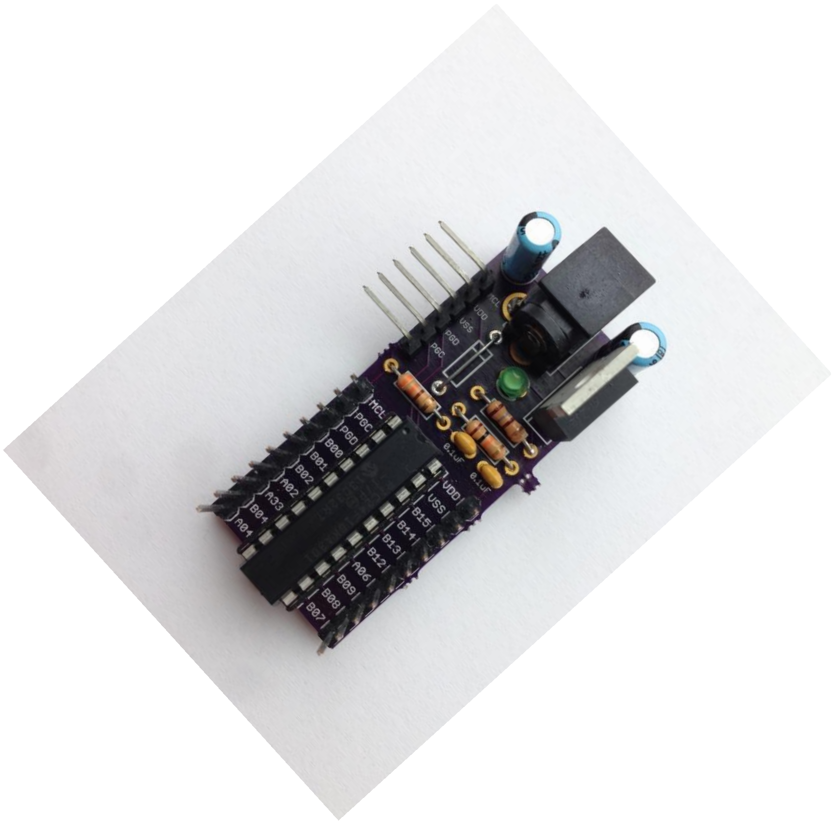
Requires intimate knowledge of the hardware

Registers

Electrical characteristics

Why?

Better teaches certain fundamentals
Prepares students for product development



Microcontrollers

Single-chip PIC24F instead of Arduino, etc.

Unusual (unique?) for required ME course

Students design and build their own board

Requires intimate knowledge of the hardware

Registers

Electrical characteristics

```
x = analogRead(analogPin);
```

Why?

Better teaches certain fundamentals
Prepares students for product development

Microcontrollers

- Single-chip PIC24F instead of Arduino, etc.
  - Unusual (unique?) for required ME course
- Students design and build their own board
- Requires intimate knowledge of the hardware
  - Registers
  - Electrical characteristics
- Why?
  - Better teaches certain fundamentals
  - Prepares students for product development

```
_ADON = 0;    // AD1CON1<15> -- Turn off A/D during config
_ADSIDL = 0;  // AD1CON1<13> -- A/D continues in idle mode
_MODE12 = 1;  // AD1CON1<10> -- 12-bit A/D operation
_FORM = 0;    // AD1CON1<9:8> -- Unsigned integer output
_SSRC = 7;    // AD1CON1<7:4> -- Auto conversion
_ASAM = 1;    // AD1CON1<2> -- Auto sampling
_PVCFG = 0;   // AD1CON2<15:14> -- Use VDD as positive
              // ref voltage
_NVCFG = 0;   // AD1CON2<13> -- Use VSS as negative
              // ref voltage
_BUFREGEN = 1;// AD1CON2<11> -- Result appears in buffer
              // location corresponding to channel
_CSCNA = 0;   // AD1CON2<10> -- Does not scan inputs
              // specified in AD1CSSx registers (instead
              // uses channels specified by CH0SA bits in
              // AD1CHS register) -- Selecting '0' here
              // probably makes writing to the AD1CSSL
              // register unnecessary.
_SMPI = 0;    // AD1CON2<6:2> -- Each conversion sent to
              // buffer
_ALTS = 0;    // AD1CON2<0> -- Sample MUXA only
_ADRC = 0;    // AD1CON3<15> -- Use system clock
_SAMC = 1;    // AD1CON3<12:8> -- Auto sample every A/D
              // period TAD
_ADCS = 0x3F; // AD1CON3<7:0> -- A/D period TAD = 64*TCY
_CH0NA = 0;   // AD1CHS<7:5> -- Use VDD as negative input
_CH0SA = ???; // AD1CHS<4:0> -- Use ANx as positive input
AD1CSSL = 0;  // AD1CSSL<15:0> -- Skip all channels on
              // input scan -- see the CSCNA bits in
              // AD1CON2
_ADON = 1;    // AD1CON1<15> -- Turn on A/D

x = ADC1BUF2;
```

Microcontrollers

Single-chip PIC24F instead of Arduino, etc.
    Unusual (unique?) for required ME course
Students design and build their own board
Requires intimate knowledge of the hardware
    Registers
    Electrical characteristics
Why?
    Better teaches certain fundamentals
    Prepares students for product development

Course Structure

No homework

No exams

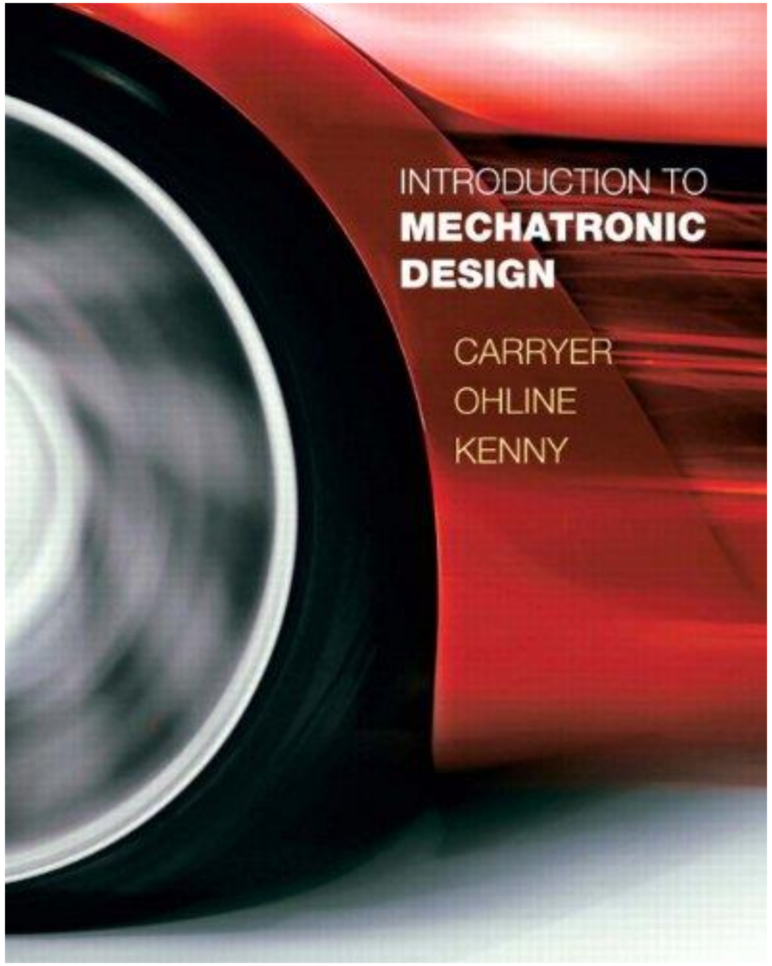
Weekly online quiz to check understanding of text and datasheets

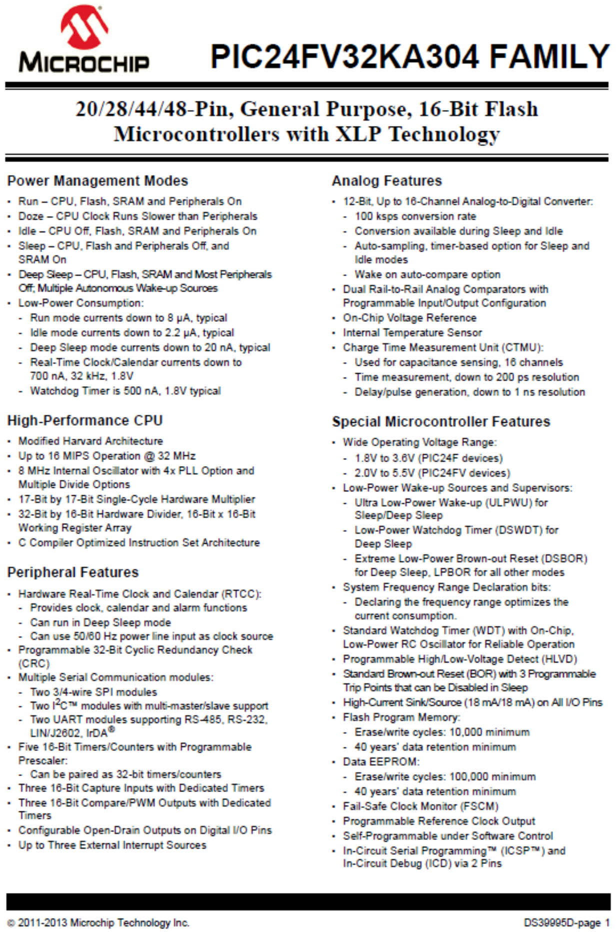
Weekly labs

Semester-long project

Applied and hands-on in a curriculum that is otherwise high-level and theoretical

Texts





MICROCHIP

PIC24FV32KA304 FAMILY

20/28/44/48-Pin, General Purpose, 16-Bit Flash Microcontrollers with XLP Technology

Power Management Modes

- Run – CPU, Flash, SRAM and Peripherals On
- Doze – CPU Clock Runs Slower than Peripherals
- Idle – CPU Off, Flash, SRAM and Peripherals On
- Sleep – CPU, Flash and Peripherals Off, and SRAM On
- Deep Sleep – CPU, Flash, SRAM and Most Peripherals Off; Multiple Autonomous Wake-up Sources
- Low-Power Consumption:
  - Run mode currents down to 8 µA, typical
  - Idle mode currents down to 2.2 µA, typical
  - Deep Sleep mode currents down to 20 nA, typical
  - Real-Time Clock/Calendar currents down to 700 nA, 32 kHz, 1.8V
  - Watchdog Timer is 500 nA, 1.8V typical

High-Performance CPU

- Modified Harvard Architecture
- Up to 16 MIPS Operation @ 32 MHz
- 8 MHz Internal Oscillator with 4x PLL Option and Multiple Divide Options
- 17-Bit by 17-Bit Single-Cycle Hardware Multiplier
- 32-Bit by 16-Bit Hardware Divider, 16-Bit x 16-Bit Working Register Array
- C Compiler Optimized Instruction Set Architecture

Peripheral Features

- Hardware Real-Time Clock and Calendar (RTCC):
  - Provides clock, calendar and alarm functions
  - Can run in Deep Sleep mode
  - Can use 50/60 Hz power line input as clock source
- Programmable 32-Bit Cyclic Redundancy Check (CRC)
- Multiple Serial Communication modules:
  - Two 3/4-wire SPI modules
  - Two I^{2}C™ modules with multi-master/slave support
  - Two UART modules supporting RS-485, RS-232, LIN/J2602, IrDA®
- Five 16-Bit Timers/Counters with Programmable Prescaler:
  - Can be paired as 32-bit timers/counters
- Three 16-Bit Capture Inputs with Dedicated Timers
- Three 16-Bit Compare/PWM Outputs with Dedicated Timers
- Configurable Open-Drain Outputs on Digital I/O Pins
- Up to Three External Interrupt Sources

Analog Features

- 12-Bit, Up to 16-Channel Analog-to-Digital Converter:
  - 100 ksps conversion rate
  - Conversion available during Sleep and Idle
  - Auto-sampling, timer-based option for Sleep and Idle modes
  - Wake on auto-compare option
- Dual Rail-to-Rail Analog Comparators with Programmable Input/Output Configuration
- On-Chip Voltage Reference
- Internal Temperature Sensor
- Charge Time Measurement Unit (CTMU):
  - Used for capacitance sensing, 16 channels
  - Time measurement, down to 200 ps resolution
  - Delay/pulse generation, down to 1 ns resolution

Special Microcontroller Features

- Wide Operating Voltage Range:
  - 1.8V to 3.6V (PIC24F devices)
  - 2.0V to 5.5V (PIC24FV devices)
- Low-Power Wake-up Sources and Supervisors:
  - Ultra Low-Power Wake-up (ULPWU) for Sleep/Deep Sleep
  - Low-Power Watchdog Timer (DSWDT) for Deep Sleep
  - Extreme Low-Power Brown-out Reset (DSBOR) for Deep Sleep, LPBOR for all other modes
- System Frequency Range Declaration bits:
  - Declaring the frequency range optimizes the current consumption.
- Standard Watchdog Timer (WDT) with On-Chip, Low-Power RC Oscillator for Reliable Operation
- Programmable High/Low-Voltage Detect (HLVD)
- Standard Brown-out Reset (BOR) with 3 Programmable Trip Points that can be Disabled in Sleep
- High-Current Sink/Source (18 mA/18 mA) on All I/O Pins
- Flash Program Memory:
  - Erase/write cycles: 10,000 minimum
  - 40 years' data retention minimum
- Data EEPROM:
  - Erase/write cycles: 100,000 minimum
  - 40 years' data retention minimum
- Fail-Safe Clock Monitor (FSCM)
- Programmable Reference Clock Output
- Self-Programmable under Software Control
- In-Circuit Serial Programming™ (ICSP™) and In-Circuit Debug (ICD) via 2 Pins

© 2011-2013 Microchip Technology Inc.

DS39995D-page 1

Labs

PCB design

Electronics

Microcontrollers and peripherals (digital I/O, ADC, timers, interrupts, PWM, etc.)

Actuators (DC motors, servos, steppers, solenoids)

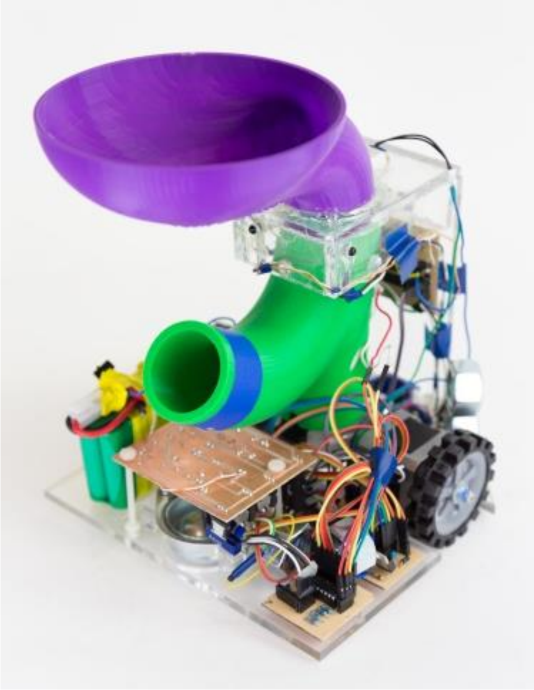
Sensors (various IR, encoders, touch)

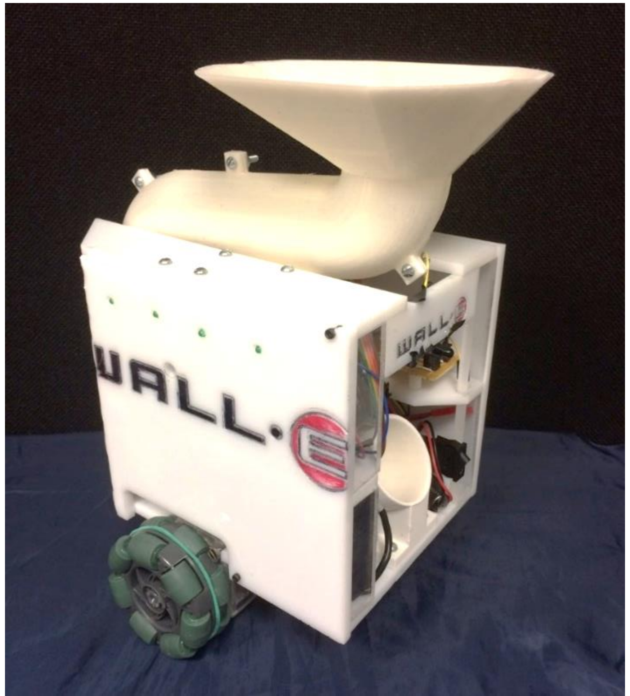
Project

Autonomous robot competition

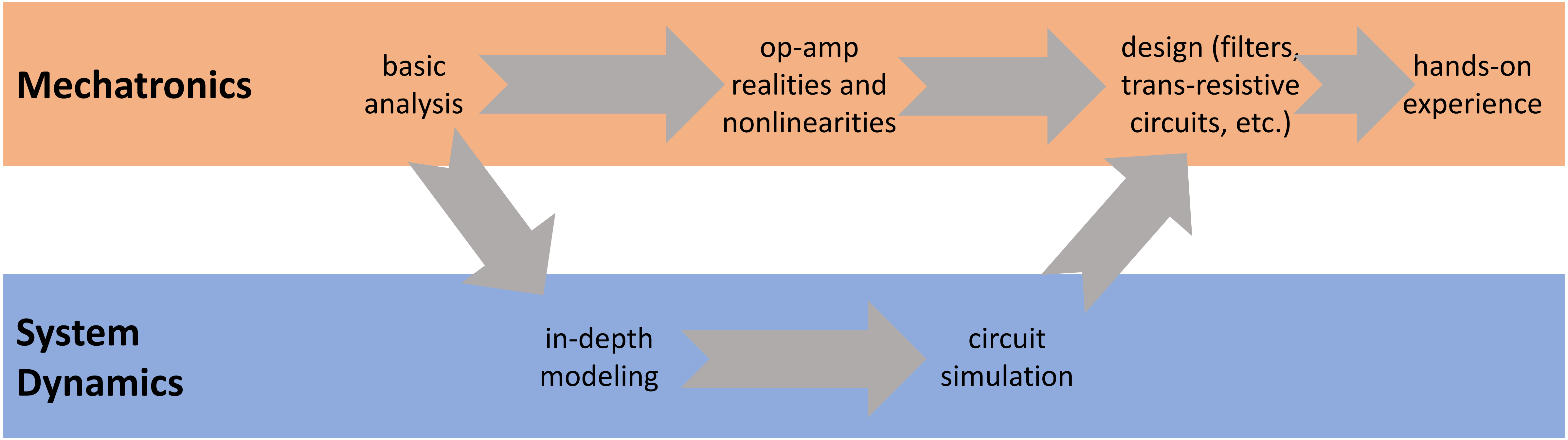
Semester-long, in teams

Design, construct, test, repeat





ME330/ME335 Envelope: Op-Amps



Mechatronics

basic analysis

op-amp realities and nonlinearities

design (filters, trans-resistive circuits, etc.)

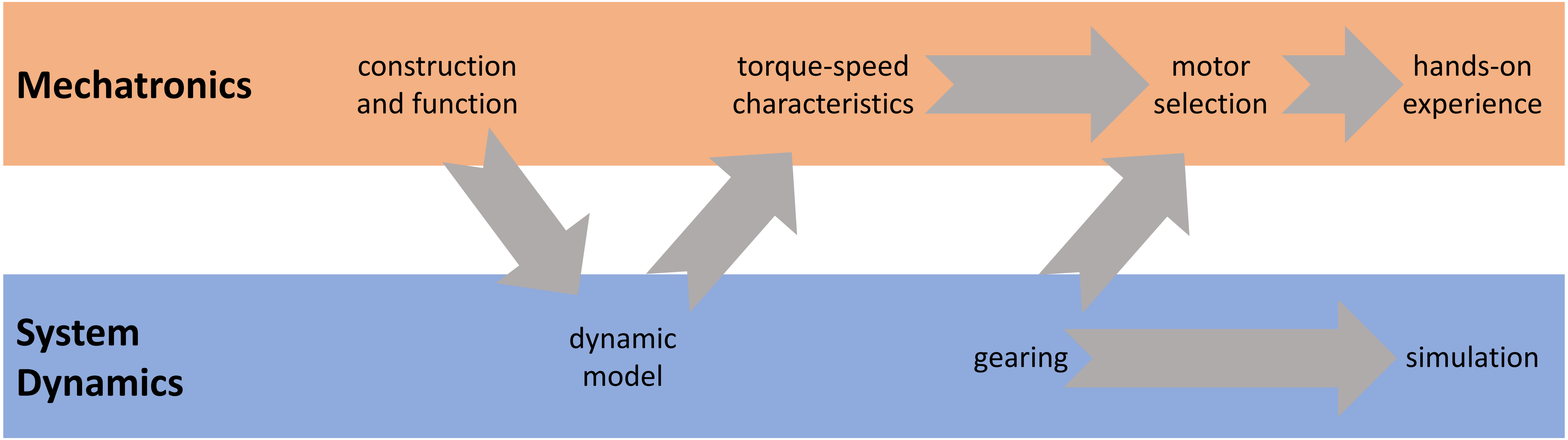
hands-on experience

System Dynamics

in-depth modeling

circuit simulation

ME330/ME335 Envelope: DC Motors



Mechatronics

construction and function

torque-speed characteristics

motor selection

hands-on experience

System Dynamics

dynamic model

gearing

simulation

Outcomes

Fun

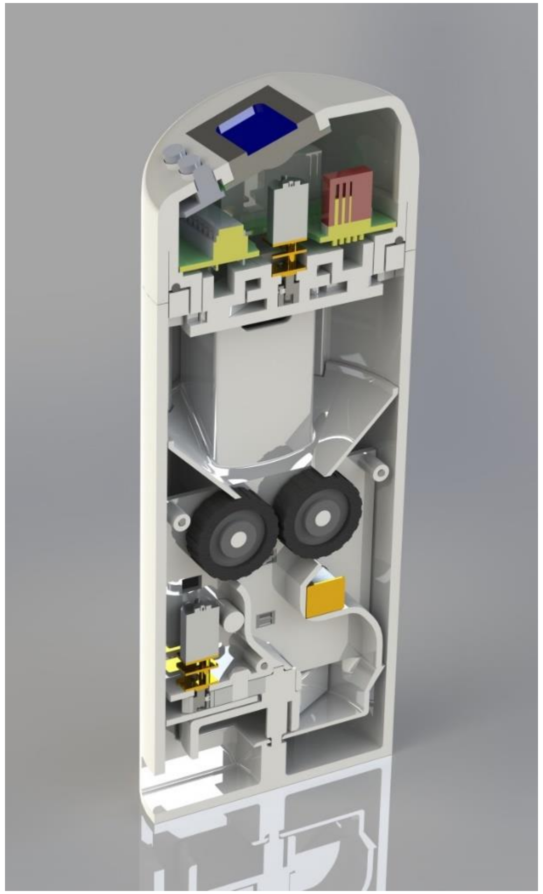
Frustration

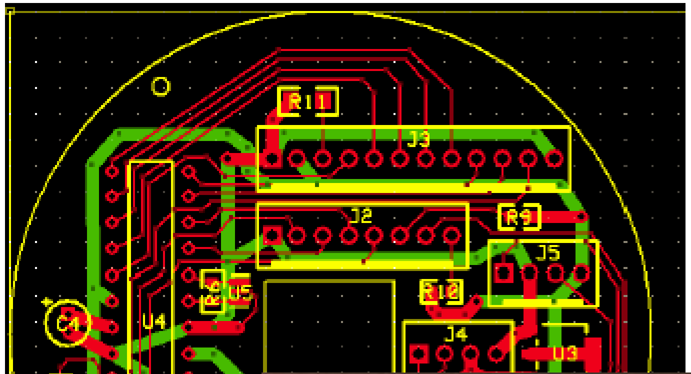
Computing

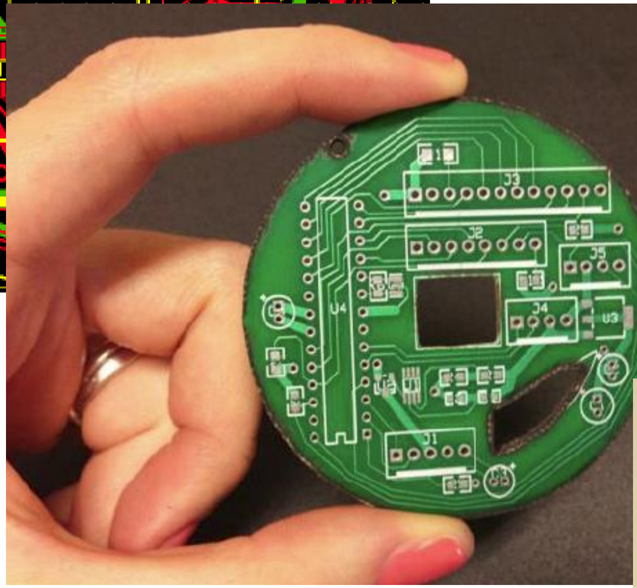
Capstone

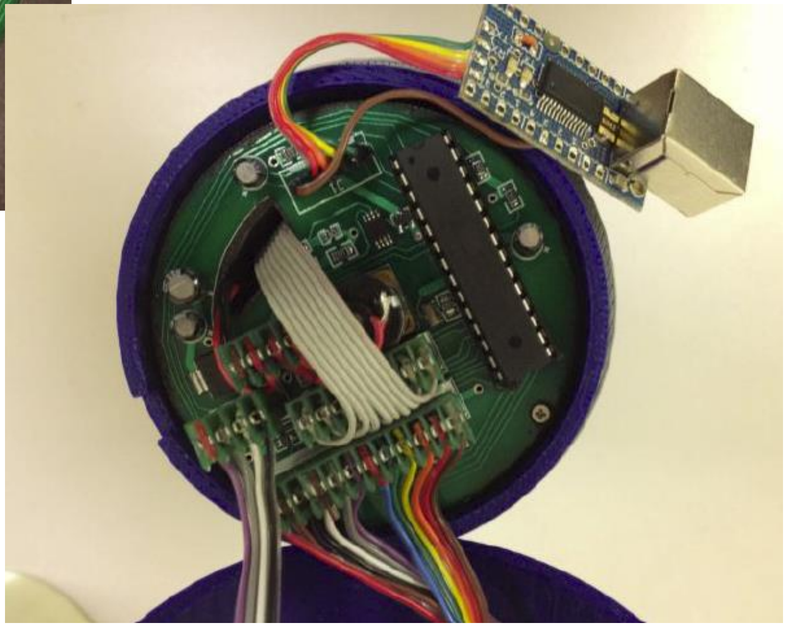
Employers

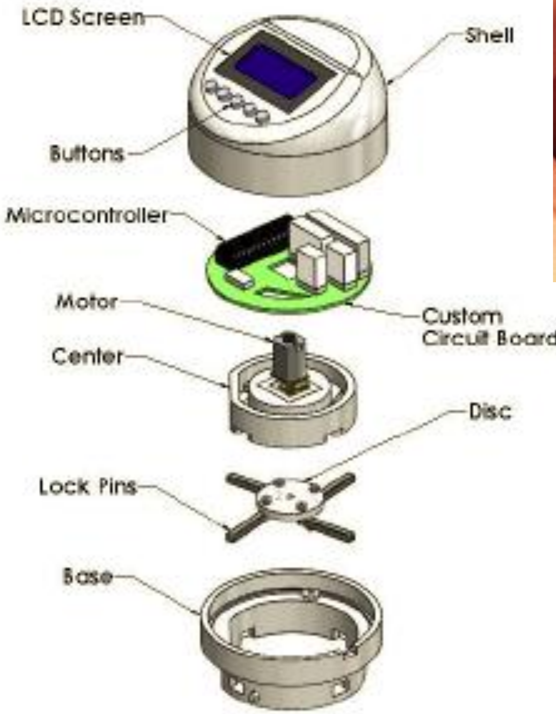
Student comments




R11
J3
J2
R9
J5
R12
J4
U3
U4
U5
C4
R8
R7
U2






LCD Screen
Shell
Buttons
Microcontroller
Custom Circuit Board
Motor
Center
Disc
Lock Pins
Base

Take-aways

A low-level mechatronics course for ME students…

- … may better prepare them for product development
- … may reinforce certain topics better than a high-level course
- … may prepare them to work in interdisciplinary teams
- … requires that high-level topics (modeling, analysis, control) be taught in other courses
- … has many challenges in terms of student preparation, scalability, and pedagogy

colton@byu.edu