ECE – 304
Junior Design Project

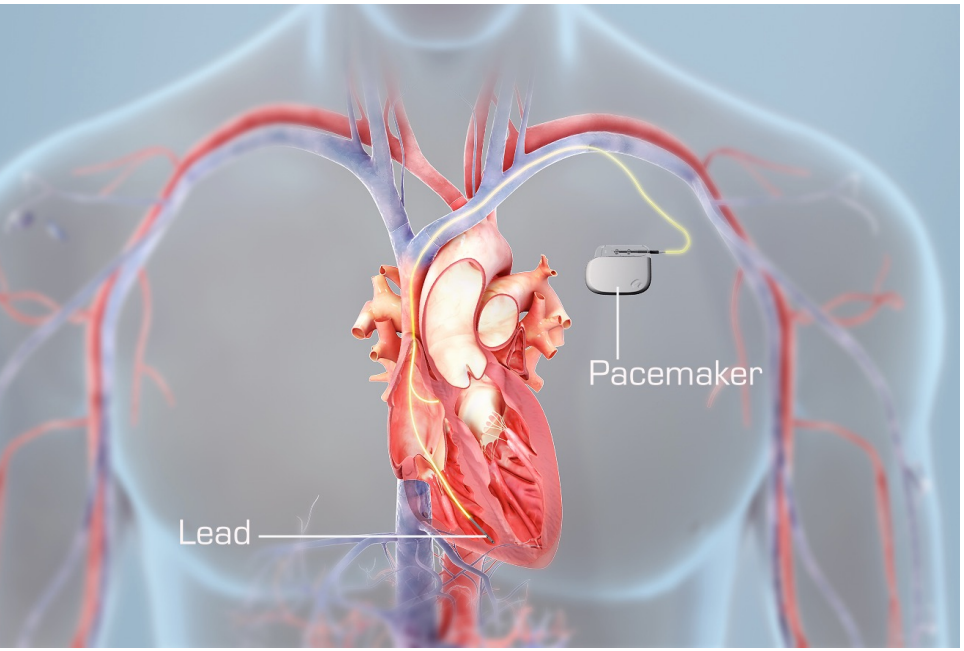
Spring 2022

David McLaughlin, Professor
he/him
Department of Electrical & Computer Engineering
University of Massachusetts at Amherst

Lecture 1 - 1/28/22 – Course Overview

Cardiac Pacemaker

A cardiac pacemaker is a device that helps monitor and regulate heartbeat. It is commonly employed to overcome faulty electrical signaling in the heart that causes the heartbeat to become abnormally slow and irregular.



http://www.scientificanimations.com/all-you-need-to-know-about-pacemakers/

https://hoardingwoes.wordpress.com/2012/02/04/pacemakers-sustained-life-during-hospice/

Slide Courtesy of Prof. Yu Chen, UMass Dept of Biomedical Engineering

Pacemaker design requirements (the device shall …)

Functional

- measure heart rate
- stimulate heart with electrical pulses when rate irregularity is detected
- maintain function during battery change
- update software functions wirelessly
- reject interference/false triggers

Performance/Characteristic

- Generate xx second voltage pulse of yy amplitude, sourcing at least xx mA of current within xx seconds of detecting irregularity
- size: volume, weight < xxx
- power consumption: battery lasts for xx years
- not cause infection
- operate when patient is awake, exercising, sleeping, and ill

requirements are design-agnostic

after product is developed, all requirements are verified:
inspection, demonstration, analysis, test

Problem statement – what is needed? what are we trying to create?
Requirements – what functions must it perform? how well? what else?
Preliminary Design – research, analyse alternatives, synthesize solutions, tradeoffs, …
Preliminary Design Review – present to peers/management before proceeding
Detailed Design – subsystem make vs buy, detailed designs, prototypes
Midway Design Review – identify risks, test & integration plans, prototype demo
Comprehensive Design Review – key issues resolved
Final Design Review – last checks
Low Rate Initial Production
Field Experience
Next level of prodution

Design Activities

- Research similar/existing
- Tradeoff studies
- Simulations
- Synthesize solutions
- Analysis
- Make vs buy decisions
- Detailed subsystem design
- Supply chain
- Risk analysis
- Integration & Test plans
- Requirements verification plans
- Build & code prototypes

Senior Design Project

Year-long project
Teams of 3 or 4
Design, build, test, demonstrate something new that your team has concieved of.

Entire ECE Dept is involved:
each team has a faculty mentor & 2 faculty evaluators for the entire year
PDR, MDR, CDR, FPR, Demo Day

W I D E range of results, from excellent & novel ideas, innovative designs,

http://www.ecs.umass.edu/ece/sdp/sdp21/teams.html

…to some teams that barely get something to work at all.

Why do some teams have difficulty?

Some SDP Teams Are Hampered by:

- Poorly-scoped problem: too complex or too simple; problem statement (system descrption, requirements) is missing or not adhered-to. [Faculty mentors and the review process is in place to mitigate against this risk]

- Lack mature understanding of the engineering design process (time allocation & scheduling; planning; risk analysis & mitigation; team coordination & responsibiliy sharing; requirements management)

- Team Problems: uneven contribution & ineffective communication; internal conflict over design scope

- Teams/team members unfamiliar with embedded microcontroller design. Some CompE's think it's the EE's job. Some EE's think it's the CompE's job; Some ECEs are afraid of MCUs.

Description: This 2-credit course prepares you for parts of ECE Senior Design Project by

- ❑ exposing you to the engineering system design process
- ❑ giving you practice designing and implementing an embedded sense/response system using programmable 8-bit microcontrollers.

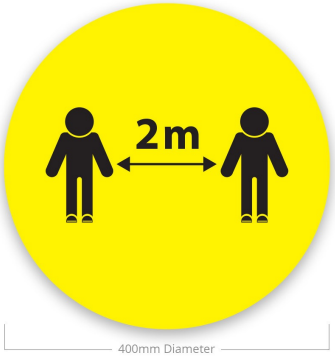
This will be fun. You will emerge non-fearing embedded design.

Course Plan:

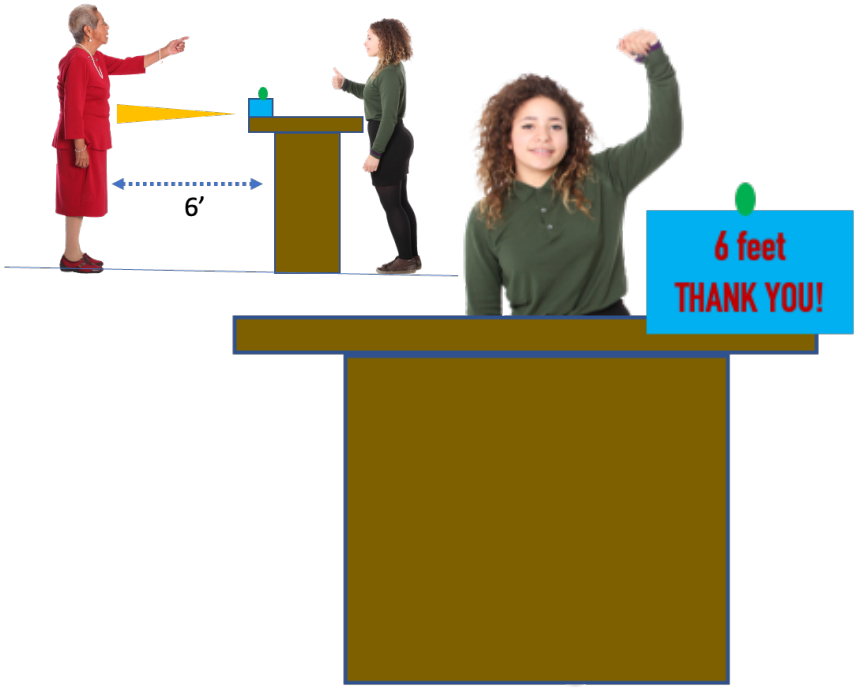
- At the beginning of the semester, you will receive a kit containing electronic parts that you will use to design and build an electronic system for **managing the approach to a reception counter during covid.**

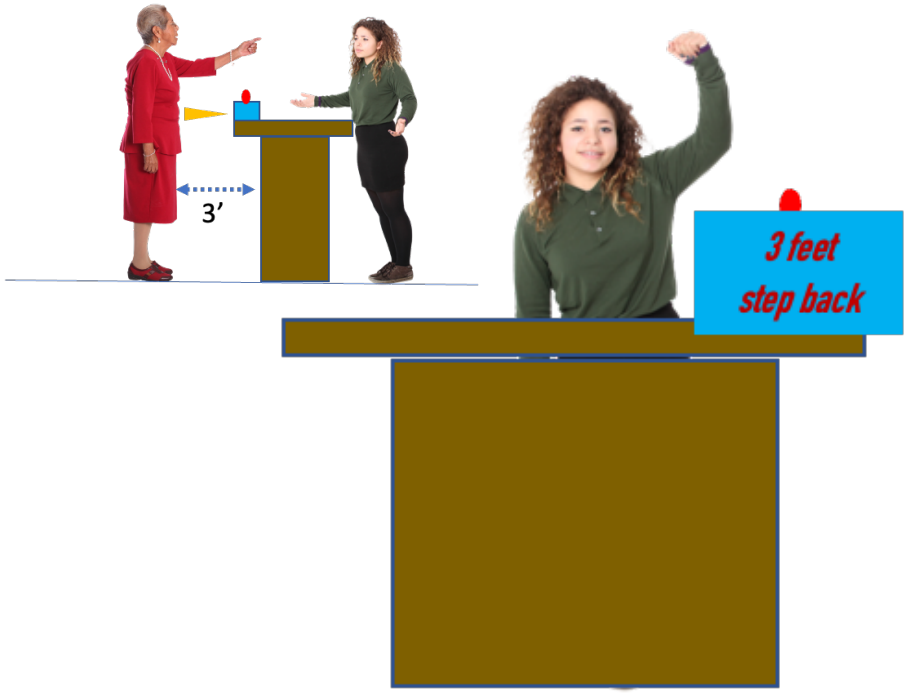
- Working individually, you will develop a set of systems engineering design artifacts and deliverables and complete a series of design and development milestones according to the course schedule.

Please stay 6’ from the counter....



Safe Reception Distance System (SRDS)





Safe Reception Distance System (SRDS). You are to design, build, and test an electronic system to manage approach to a reception counter to maintain social distancing in a time of COVID. Your system shall automatically measure the distance of approaching persons and give instructions for them to keep a safe distance (6'). The system shall also provide a display to the person behind the counter summarizing the number of counter-approaches during the past time period and providing a list of the minimum distance of the most recent customers.

During the semester, you will complete two full iterations of the design/build/test/demonstration of your system. Requirements for the two iterations are:

First iteration:

1. Laboratory bench-top system meets functional but not performance requirements.
2. Use of an Arduino Uno development board and the Arduino Integrated Development Environment (IDE) is allowed (encouraged).
3. Couter person activates SRDS at beginning of shift with a button.
4. AC (wall-outlet) power is allowed.

Second iteration:

1. Counter-top deployment and demonstration at scale.
2. System meets functional requirements (detect approaching persons, provide instructions, provide information to person behind counter, etc...)
3. System meets performance requirements (ie, instructions are visible to persons approaching from specified distance; information display to counter person is visible).
4. A bare ATmega328P IC on a breadboard, rather than any development board is required.
5. Counter person activates SRDS without physical contact.
6. MCU must be programmed ANSI C (i.e., no Arduino Code).
7. System shall be battery operated with a minimum battery lifetime of 30 days.

Readings: There is no textbook required for this course. The datasheets for the ATmega328P microcontroller will be useful throughout the second half of the course, and URLs for these resources will be provided. Many open-source resources exist online for students who want help with the Arduino programming part of the course.

Background Knowledge: Prior experience with C programming programming using the AVR ATmega328P MCU is a pre-requisite for this course.

Support: Marston 221 lab will be available for drop-in help & replacement parts on a schedule TBD.

Semester Schedule

(subject to change)

JDP Schedule Spring 2022			
	Date	**Class Meeting Topic**	**Due COB**
Week 1	1/28/22	Lecture 1 - Course Intro	
Week 2	2/4/22	Lecture 2 - Project Specifications	collect kit
Week 3	2/11/22	Q&A with the pretend customer	
Week 4	2/18/22	Lecture 3 - Arduino & Sonar Demo	Build1 PDR report due
Week 5	2/25/22	Lecture 4 - Test plans, issues	
Week 6	3/4/22	No Class	Built1 Test plan & EVM1
Week 7	3/11/22	Build1 Report Due, no Class Meeting	Build1 report
	3/18/22	Spring Break	
Week 8	3/25/22	Lecture5 - avr-gcc tools demo	
Week 9	4/1/22	Lecture 6 - Risk Mgt, Q&A with pretend customer	
Week 10	4/8/22	Build2 PDR Due, no Class Meeting	Build2 PDR report due
Week 11	4/15/22	Senior Design Project (SDP) Topics	EVM2
Week 12	4/22/22	Senior Design Project (SDP) Topics	
Week 13	4/29/22	Build2 Report Due, no class meeting	Build2 report

ECE304 JDP Grading	Spring 2022		
Item		**% grade**	
Build1 preliminary design review	PDR1	10%	Build1 project specification & block diagram
Test plan 1	TP1	5%	Build1 test & requirement verification plan
Earned value management report 1	EVR1	5%	Earned value management (hours planned & spent) report
Build1 final report	FR1	30%	Video demo of functioning build1 system & requirements verification report
Build2 Preliminary design review	PDR2	10%	Updated project spec, block diagram, test & requirement verification plan. Risk management plan, go-forward schedule
Earned value management report 2	EVR2	5%	Earned value management report
Build2 final report	FR2	35%	Video demo of functioning build2 system; requirements verification report; final EVM chart
	Total	100%	

PDR1: Project Specification & Block Diagram (10%, due in 3 weeks)

- Paragraph description of your system in non-technical, design-agnostic terms
- Set of ~ 10 requirements specifying what functions your system will perform and how well it will perform them
- Diagram showing major hardware & software subsystems and how they interconnect

TP1: Test & Requirement Verification Plan (5%, due in 5 weeks)

- Description of the deployment of your system (eg, components assembled on a table-top or deployed on a counter)
- Description of methods you will use to verify that your system meets the requirements specified in your Project Specification (methods include: inspection, demonstration, analysis, test)

EVR1: Earned Value Management Report (5%,due in 5 weeks)

- 1 page table or plot documenting planned vs actual hours spent week by week.

FR1: Final Design Review (30%, due in 6 weeks)

- Video demonstrating functioning system
- Report describing requirements verification

- This is the first pass, or the first iteration of the design/build process.
- Building the system around the Arduino Uno dev board & IDE is encouraged at this stage

PDR1: Preliminary Design Review (10%, due in 9 weeks)

- Updated Project Specification
- Updated Block Diagram
- Updated test & requirements verification plan
- Risk management plan
- Go-forward schedule

EVR1: Earned Value Management Report (5%,due in 10 weeks)

- 1 page documenting planned vs actual hours spent week by week

FR2: Final Design Review (35%, due in 12 weeks)

- Video demonstrating functioning system
- Report describing requirements verification

- This is the 2nd pass, or the 2nd iteration of the design/build process.
- Building the system around the bare ATmega328p on bread board & ANSI C is required at this stage (no Arduino code).

Earned Value Management

Concepts:

- Assume 3 hours of effort per 1 academic credit per week (minimum)
- ECE-304 is 2 credits, so assume 6 hours per week or 78 hours over the semester.
- We will spend 13 hours in class, so spend 65 hours on the project.

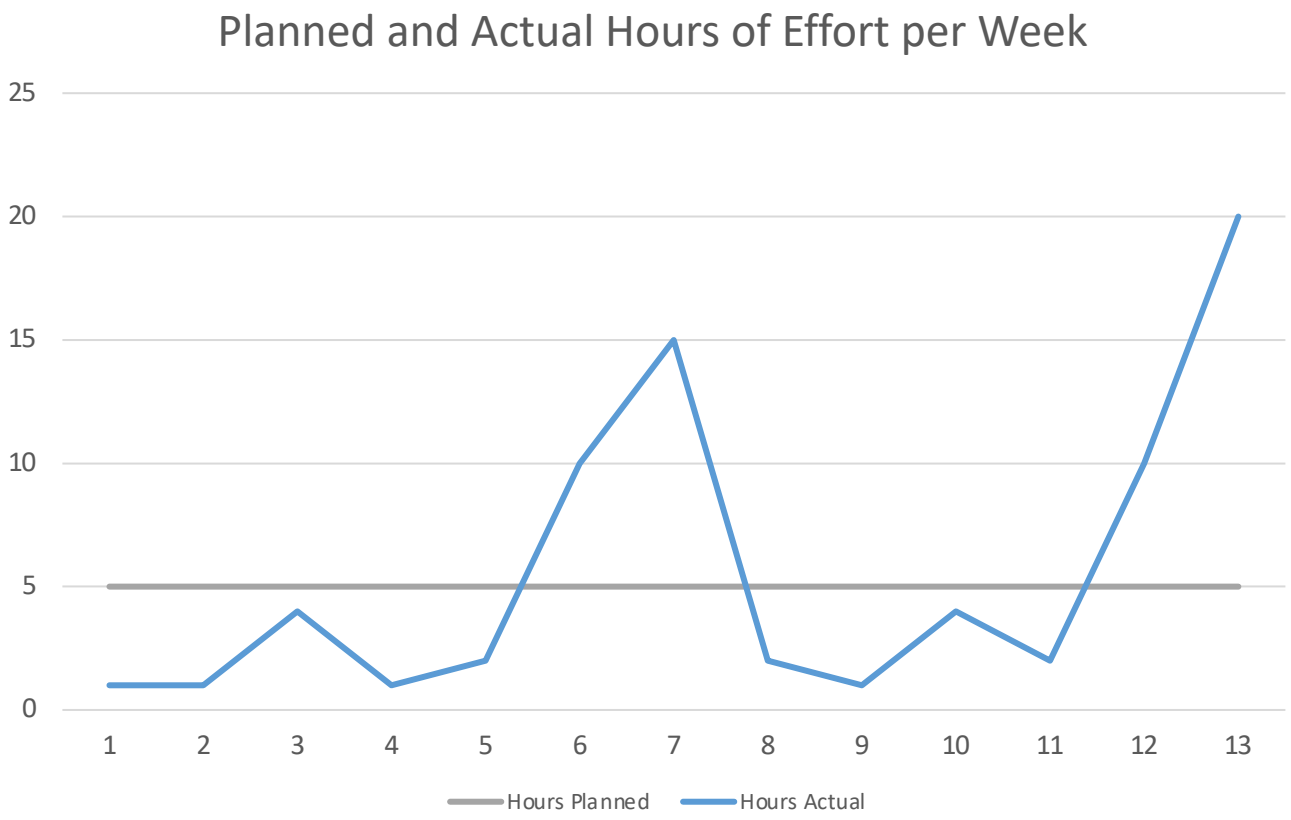
Let's pretend:

- I'm the customer/payer and I've contracted with your company for engineering design, test, and development services for this job.
- Your department charges $100 per hour of work against the cost number associated with this project. (This translates to $6.5k in labor costs if you spend 65 hours over 13 weeks.)
- Contract type:
  - If we set up a firm, fixed price (FFP) contract for, say, $6.5k, and you spend <65 hours, your company makes a profit on this. If you spend > 65 hours, your company loses money.
  - If we set up a cost plus fixed fee (CPFF) of, say, 10%, I pay you for your actual hours worked, plus a 10% adder.
- In both cases, organizations want to see a week-by-week documention of hours spent and accomplishments made.

Earned Value Management

- Firm Fixed Price
  - Customer/sponsor, doesn't care about how many hours you worked. Cares about results.
  - Weekly earned value = % of the total performance of the project (based on agreed-upon milestones and verified requirements. This is often referred to as "selling off requirements"). Each week, you tell me how much value you acrued, and I pay you.
  - Your company wants to know how many hours you worked, since this tells them how long it takes to get things done. Very valuable for bidding on future jobs.
- Cost plus fixed fee
  - Both company and customer/sponsor want to see a weekly tally of hours spent, since this is the basis of payment.
  - We also need to agree on some performance milestones, otherwise you could work forever and bankrupt me!
- For either contract type, knowledge of weekly hours & performance is key to measuring project health & pricing & managing future contracts.

But this is all we want for this course: data to enable you to create this plot at the end of the semester. It will help you gauge effort & performance on future projects.



Perspectives

“Having a requirements-based solution is always more effective than saying, ‘Hey, I have this neat gadget! Let’s see what problems this can solve’ ” – Matt Tyhach, UMass MS 2008, Manager, Raytheon

Plan before building.

Take guesses when needed. Form hypotheses. Then follow thru. Don’t be afraid to say “This is what I originally thought. Here’s why. This is what I learned. This is how I view it now.” --- This is one of the most important skills to have as an engineer & leader. Own your errors & your process.

Obviously, you need to know the tech so that you’re not afraid or unable to plan & try things out.

Experiment. Research. Try. Retry. Discard. Save. Review. Reflect.

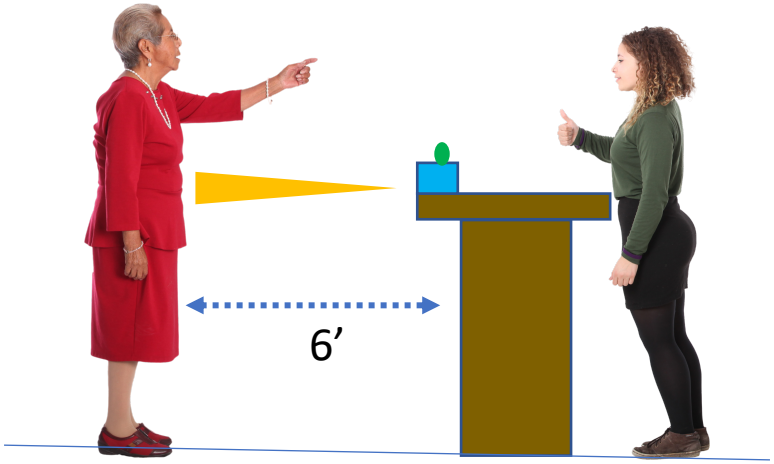
Your EVM actuals are you most important take-away from the course.

Assignment

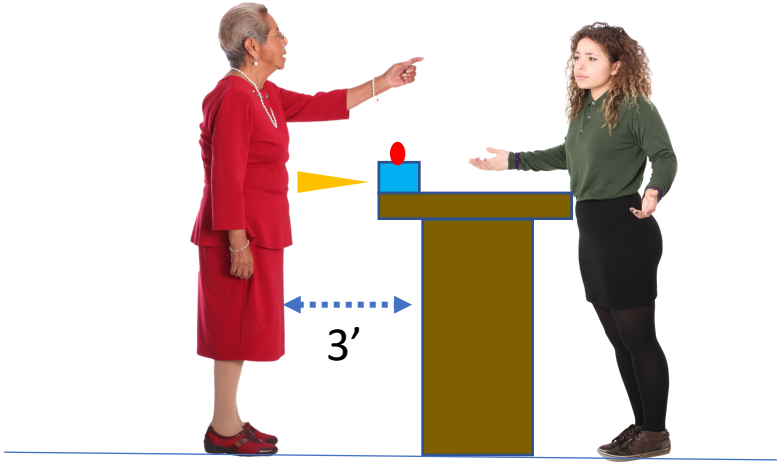
- Reread & think about the semester assignment.
- Begin thinking about requirements (the system shall do <something>; this is how well it should do it)
- Pick up your kit from Marston 221. Inventory everything. (If you wait and discover at 11 pm on a Sunday 3 months from now that your kit doesn't have a zazzawassawhatsit#505, then what will you do?)
- Go online and check out all the parts of your kit so that you'll know what's available to you.
- Come prepared with any Questions you have about the assignment next week.

What do you think the key design-driving requirements will be?


6 feet
THANK YOU!


6’


3 feet
step back


3’