Teaching a Course on Modeling and Simulation for Cyber-Physical Systems using Modelica and FMI Technologies with Hands-on-Laboratories

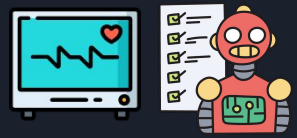
Virtual American Modelica Conference 2020
Sep. 24, 2020

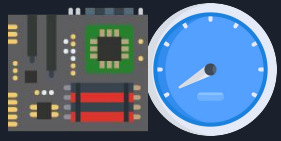
Prof. Luigi Vanfretti
ALSETLab, ECSE
Rensselaer Polytechnic Institute
http://ALSETLab.com luigi.vanfretti@gmail.com

ALSETlab

Outline

- Motivation
  - Cyber-Physical Systems
  - Industry Challenges and Needs
  - My Research Needs
- Course Overview
  - Course Scope
  - Course Topics and Activities
  - Sample Course Projects
  - Integrating Research Projects into the Course
- Development and Roll-Out of Lab Activities
- Conclusions and Future Work





CPS Overview

CPS technologies are transforming the way people interact with engineered systems:

- Just as the Internet has transformed the way people interact with information
- From the Internet of Information to the "Internet of Things"

The "CPS" view and CPS in general help drive innovation and competition in:

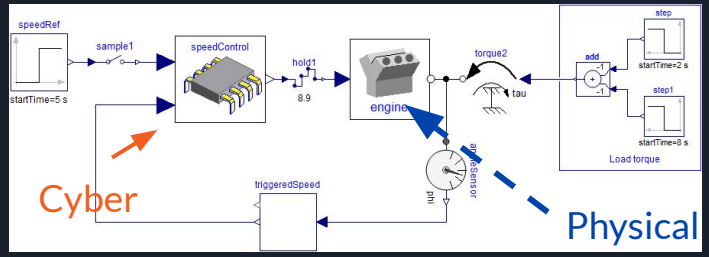
- aeronautics, building design, energy,
- electrical power grids ('smart grid')
- healthcare, manufacturing, and transportation...

Advances in CPS will enable:

- capability, adaptability, scalability, resiliency,
- safety, security, and usability

to expand the horizons of these critical systems.

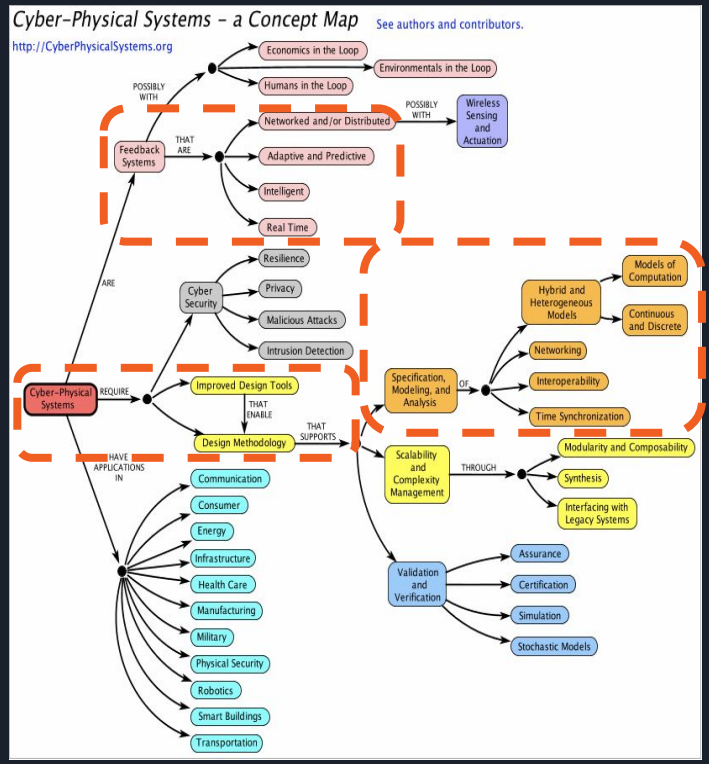
CPS Areas



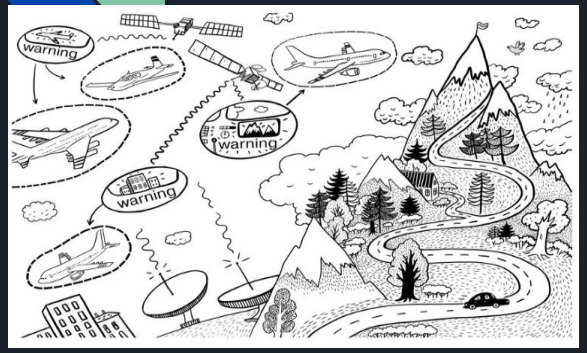
- Cyber-physical systems (CPS) are engineered systems that are built from, and depend upon, the ***seamless integration of computation and physical components***.
  - Physical components / systems: *"follow the laws of nature"*
    - Mechanical, electrical, thermal/HVAC, etc.
  - Cyber components/SYSTEMS: **"follow the rules of algorithms"**
    - Computation (e.g. computers, IT systems), communication protocols & networks

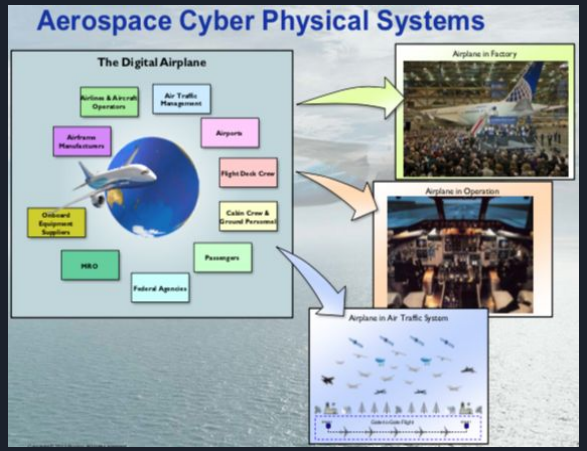
- Smart CPS drive innovation and competition in a range of application domains including ***aeronautics,*** building design, energy, electrical power grids, healthcare, manufacturing, and transportation, etc.

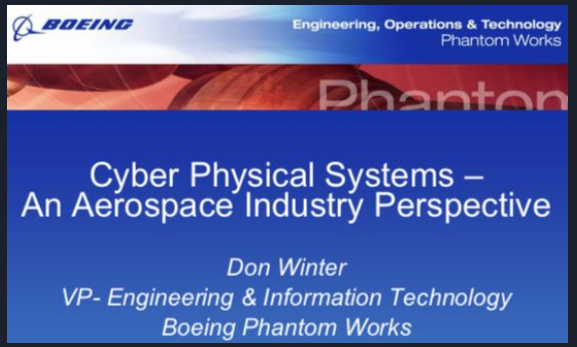
- CPS encompasses a broad spectrum of topics and fields, which would be impossible to cover in a single undergraduate/graduate course (see figure on the right).

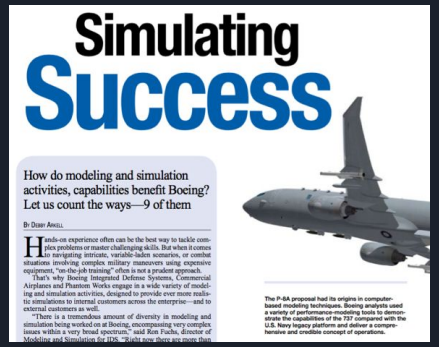


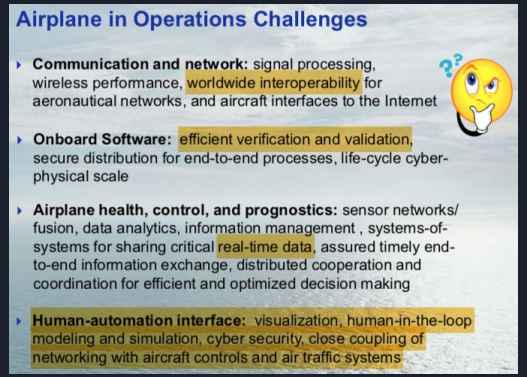
CPS Challenges
Aerospace/Boeing











This course deals with general purpose modeling languages and tools for transdisciplinary, and interoperable, modeling & simulation required in multi-domain CPS, directly aligned with industry's needs and research challenges.

Industry Needs Aerospace/Boeing

- Boeing's relationship with Dassault & the3DS Platform:

Boeing and Dassault Systèmes Announce Extended Partnership

Boeing Extends Use of Dassault Systèmes to Include 3DEXPERIENCE Platform for Manufacturing Operations Management and Product Lifecycle Management

"Boeing will deploy the 3DEXPERIENCE platform in phases and rely on Winning Program, Co-Design to Target, Ready for Rate, Build to Operate and License to Fly industry solution **experiences for aerospace and defense** to deepen its end to end digital collaboration, design, engineering, analysis, **manufacturing planning** and shop floor execution capabilities throughout the enterprise."

- Boeing and similar companies will benefit from research and teaching that involves the development of these skills.

Software Engineer - Simulation Community Lead Promoted

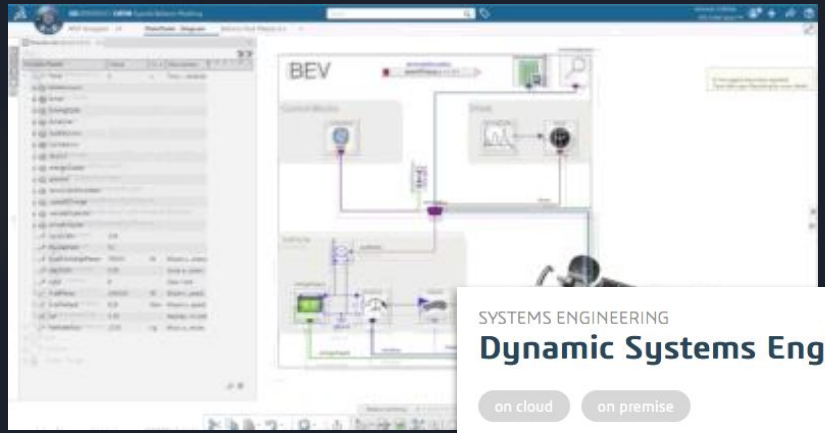
Boeing

St. Louis, MO, US

To succeed in this role you should have experience in modeling & simulation concepts, an understanding of discrete-event and frame-based simulation and an ability to work in a ... jobs.boeing.com

3 connections work here

1 week ago



Skills students need

SYSTEMS ENGINEERING

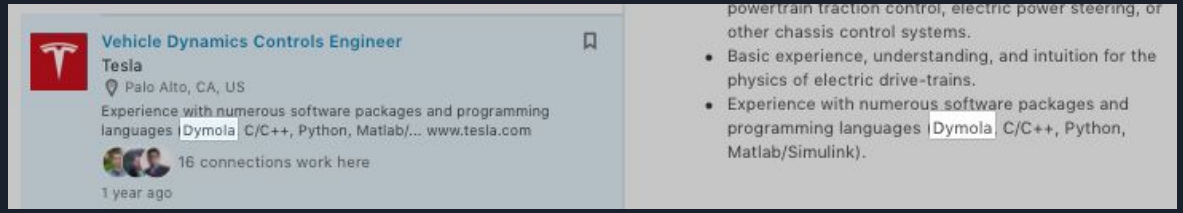
Dynamic Systems Engineer

on cloud | on premise

Accelerate the development and understanding of complex dynamic systems through modeling and simulation.

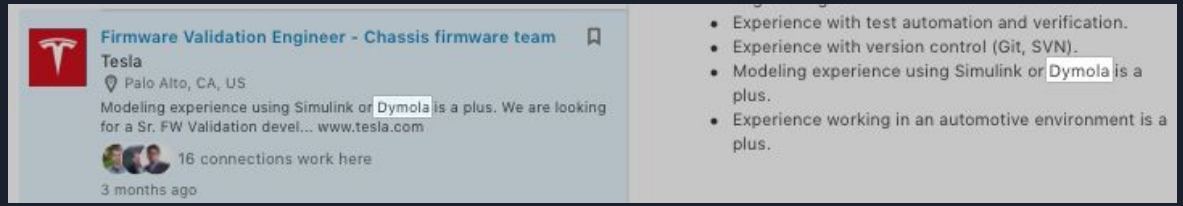
- Rapidly model and simulate the performance of complex products and systems.
- Accelerate understanding and validation of complex systems through early virtual simulation.
- Quickly find solutions to complex multi-physic system design problems.
- Uses Modelica compliant component models to ensure Intellectual Property capture and re-use.
- Size, tune and optimize system components and parameters quickly, accurately and early in the system design process.

Industry Needs Other Examples



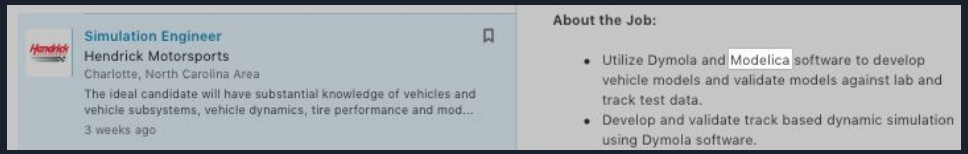
Vehicle Dynamics Controls Engineer
Tesla
Palo Alto, CA, US
Experience with numerous software packages and programming languages (Dymola, C/C++, Python, Matlab/... www.tesla.com
16 connections work here
1 year ago

powertrain traction control, electric power steering, or other chassis control systems.
- Basic experience, understanding, and intuition for the physics of electric drive-trains.
- Experience with numerous software packages and programming languages (Dymola, C/C++, Python, Matlab/Simulink).



Firmware Validation Engineer - Chassis firmware team
Tesla
Palo Alto, CA, US
Modeling experience using Simulink or Dymola is a plus. We are looking for a Sr. FW Validation devel... www.tesla.com
16 connections work here
3 months ago

- Experience with test automation and verification.
- Experience with version control (Git, SVN).
- Modeling experience using Simulink or Dymola is a plus.
- Experience working in an automotive environment is a plus.



Simulation Engineer
Hendrick Motorsports
Charlotte, North Carolina Area
The ideal candidate will have substantial knowledge of vehicles and vehicle subsystems, vehicle dynamics, tire performance and mod...
3 weeks ago

About the Job:

- Utilize Dymola and Modelica software to develop vehicle models and validate models against lab and track test data.
- Develop and validate track based dynamic simulation using Dymola software.

Wolfram Technology Engineer
Wolfram Research, Inc.
Champaign, IL, US
Modelica experience or system modeling skills are a plus. Wolfram, creator of Mathematica, Wolfram|A... my.jobs
Be an early applicant
1 month ago

- Self-motivated and self-disciplined
- Excellent written and oral communication skills
- Able to articulate ideas and instructions
- Desire to work with clients on a daily basis
- Interest in growing into a technology consultant
- Wolfram Language programming skills are a plus
- Modelica experience or system modeling skills are a plus

Research Scientist in Applied Mathematics for Design Optimizatio
PARC, a Xerox Company
Palo Alto, CA, US
We are searching for a research scientist in applied mathematics, specifically with expertise in alg... www.jsco.re
Be an early applicant
3 weeks ago

The Following Experience Is a Plus

- Experience with Category Theory.
- Experience with Functional Programming e.g. Haskell, Scheme.
- Experience with Computational Physics and Computational Geometry.
- Familiarity with Modelica/Matlab Simulink.

Research Scientist AI & Human-Computer Interacton
PARC, a Xerox Company
Palo Alto, CA, US
We are searching for a research scientist with expertise in design of human-computer search methodol... www.jsco.re
Be an early applicant
3 weeks ago

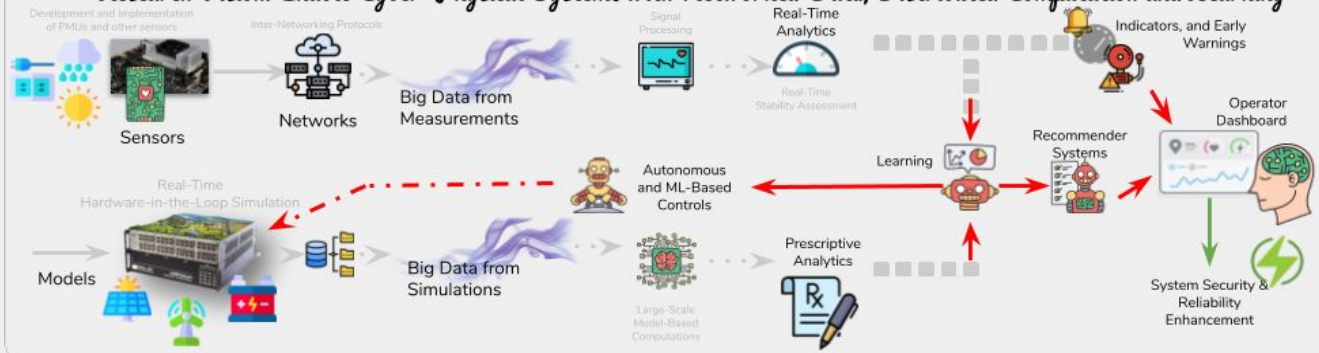
The Following Are Pluses

- Familiarity with Knowledge Representation in AI
- Familiarity with Modelica/ Matlab Simulink.
- Experience with Solid Modeling and CAD

Motivation: My Research Needs!

ALSETlab
Luigi Vanfretti, Department of Electrical, Computer & Systems Engineering @ RPI
http://ALSETLab.com, 518-496-0196, vanfrl@rpi.edu

Research Vision: Enable Cyber-Physical Systems with Networked-Data, Distributed Computation and Learning



Research Scope: Applied research in the development, implementation and testing of innovative concepts, models, methods, and SW/HW tools for

real-time monitoring, stability assessment & control; security and prescriptive analytics of

cyber-physical systems with networked data and distributed computation, with a major specialization on ***electrified systems and power grids.***

Research Areas

Synchronized phasor measurement units (PMUs) and inter-networked sensor Technologies and Applications

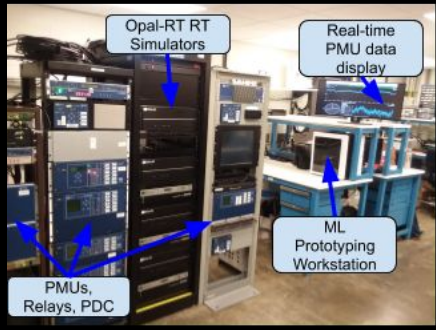
Modeling & Simulation of cyber-physical systems with networked data & distributed computation

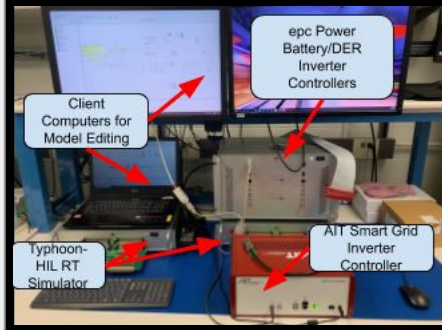
With Applications of:

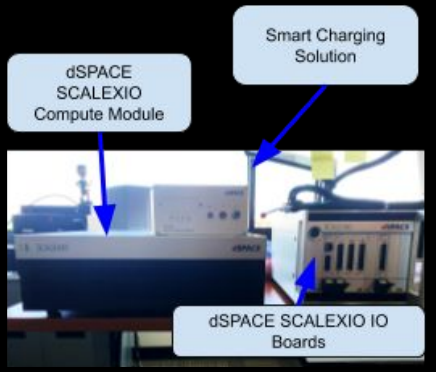
System Identification

AI & Machine Learning

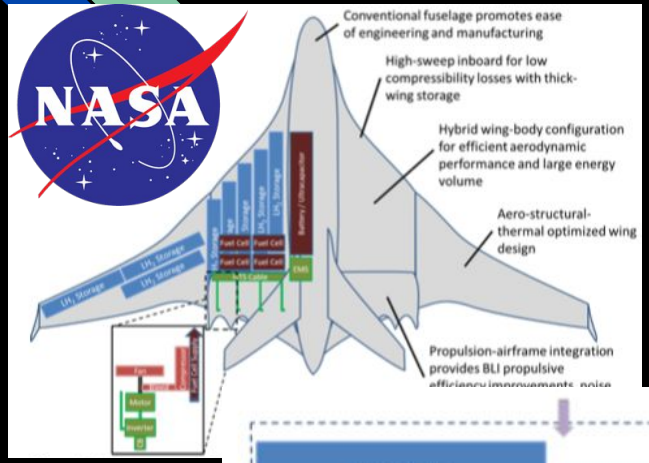
Network Science

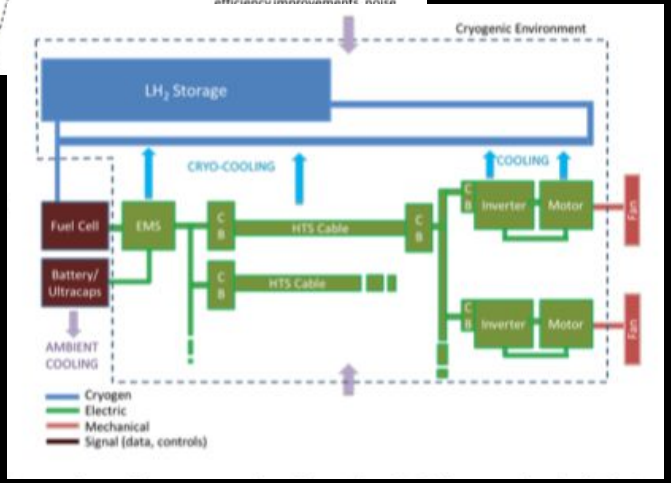






Motivation: My Research Needs!

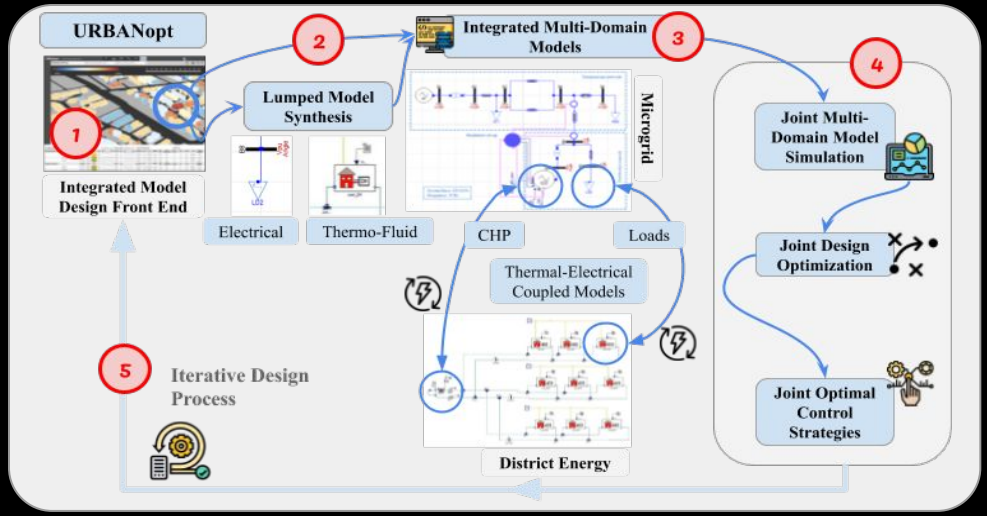




(CHEETA): NASA ULI Center for Cryogenic High-Efficiency Electrical Technologies for Aircraft Online: https://cheeta.dev.engr.illinois.edu

(OpTEN): Optimal Co-Design of Integrated Thermal-Electrical Networks and Control Systems for Grid-interactive Efficient District (GED) Energy Systems

Funded by the Department of Energy, DE-FOA-0001980, Advanced Manufacturing Office FY19 Multi-topic Funding Opportunity



M&S *for* CPS

Course Overview

CPS Course Scope & Motivation

- A 2016 report (link) of the National Academies of Sciences, Engineering and Medicine discusses the needs for students to gain the knowledge and skills required to engineer cyber-physical systems, and provides guidance in curriculum development.
- The course has been designed to address the needs highlighted Foundation 5 of the report above.
- The course makes emphasis on the merging and interactions across the physical and cyber aspects of systems via equation-based object-oriented modeling and simulation.
- The specific areas covered in the 4000-level course are highlighted in **blue**, while the additional course scope for the 6000-level course is highlighted in red.

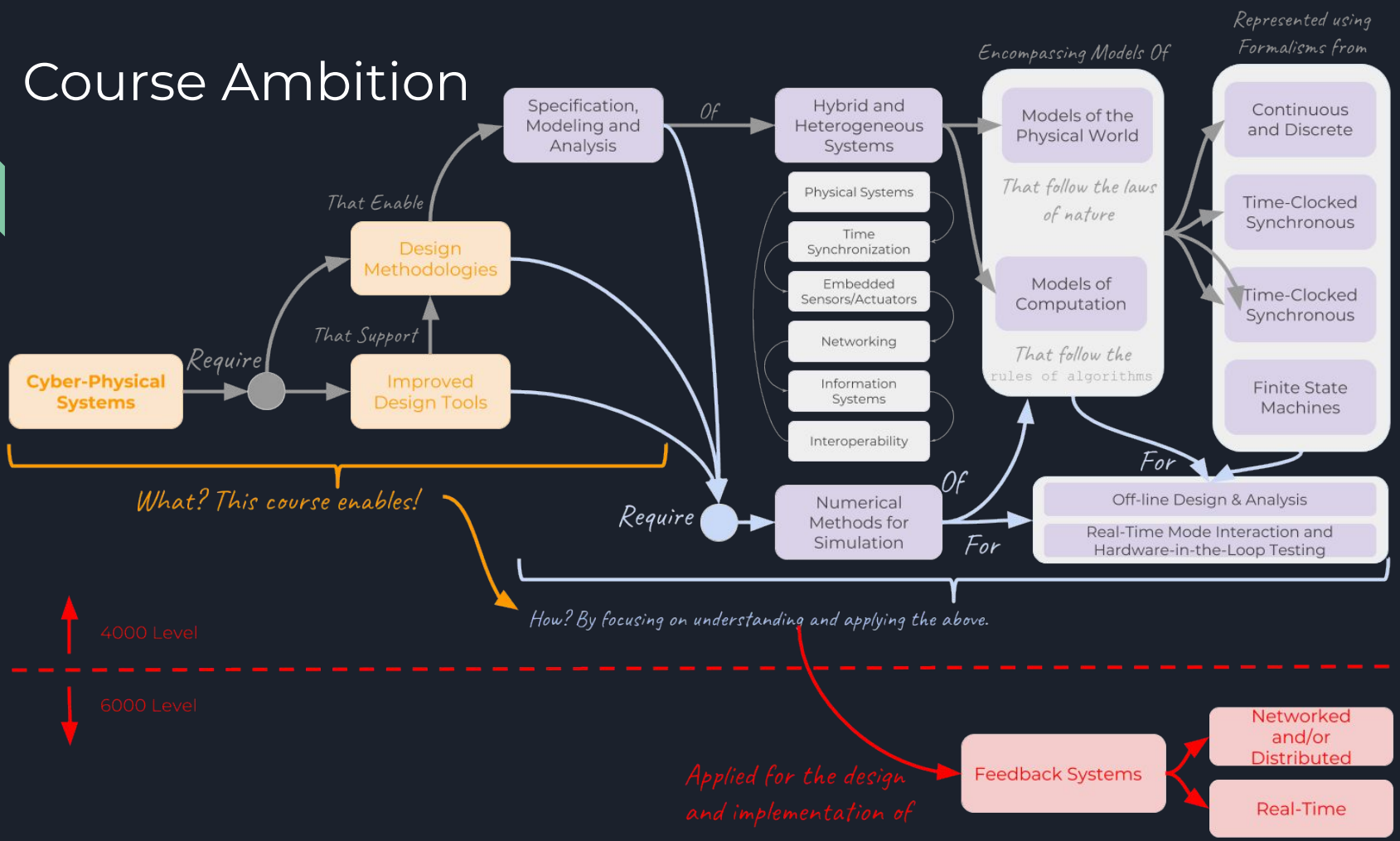
Foundation 5, stresses the need for *modeling of heterogeneous and dynamic systems integrating control, computing, and communication* with an emphasis on uncertainty and heterogeneity. **Such work is especially challenging because physical and cyber modeling use different and often incompatible models. Focusing on the merging and interactions of models across the physical and cyber aspects of systems is necessary.**

Foundation 5. Modeling of Heterogeneous and Dynamic Systems Integrating Control, Computing, and Communication

CPS modeling requires a complete picture of control, communications, and computing with emphasis on representing and accounts for modularity, abstraction, uncertainty, and heterogeneity. Relevant techniques include linear and nonlinear models, stochastic models, and discrete-event and hybrid models, and associated design methodologies based on optimization, probability theory, and dynamic programming are needed. Key concepts include the following:

- Properties of the physical world, including uncertainty and risk;
- Properties of computational devices, including computational and power limits;
- Properties of communication systems, including limitations of wireless communications;
- Error detection and correction;
- Merging physical and computational modeling; and
- Commonalities between signals and systems and finite-state automata.

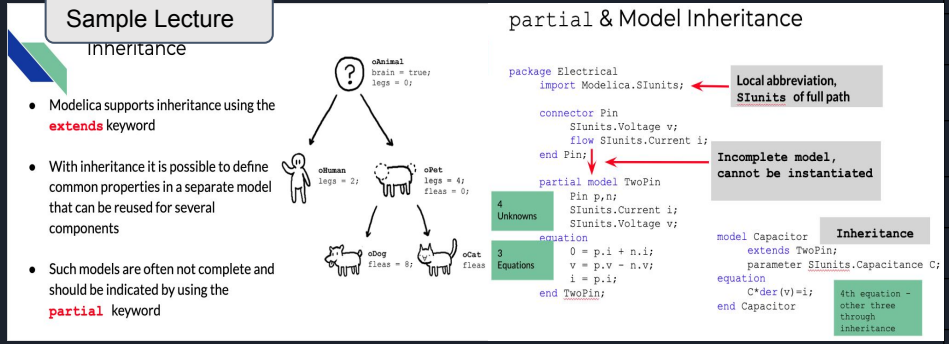
Course Ambition



Course Activities

All skills learned were then applied to projects focused on different components in an aircraft power system

Sample Lecture



Students During Lab. 2

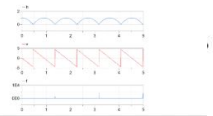
Session	Topic
1	Course Introduction and Introduction to Dymola
2	Building Object-Oriented Graphical Models
3	Simulation and Post-Processing
4	System Models and System-Wide Simulation Configuration
5	Developing Models using Equation-Based Modeling Languages
6	Understanding Equation-Based Modeling
7	Graphical Annotations, Interfaces and Documentation
8	Building Models using Standard Libraries - Electrical, Mechanical, Thermal, … etc.
9	Building Models using Standard Libraries - Multibody, Fluid, StateGraph, etc.
Lab. 1	Lab. 1: Cyber-Physical Modeling and Model Verification using Measurements
10	Development, Debugging and Identifying Numerical Issues
11	Reusable Modeling – Model Architectures, Templates and Interfaces
12	Model Variants and Data Management
13	Discontinuous and Hybrid System Modeling Principles and Specialized Operators for Time and State Event Handling
14	Workflow Automation and Scripting
15	Integrating Dymola with Other Tools
16	The FMI Standard for Model-Exchange and Co-Simulation
17	Real-Time Simulation Principles and Applications
18	Introduction to Control Systems using Modelica Linear Systems 2
Lab. 2	Lab. 2: Real-Time Hardware-in-the-Loop Simulation

Lecture Samples

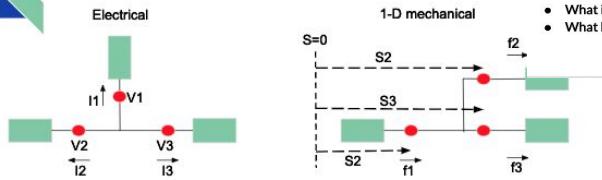
State Event Example

- The simplest model for the bouncing ball can be defined as
  - The motion of the ball is defined by the *height* above the ground and the vertical *velocity*.
- The ball moves continuously between bounces.
- An ideal ball would have a contact force equal to the gravitational force.
- What is the solver doing?
- What happens when "c" is changed?

```
model bouncingball_simplest
  parameter Real c = 1e6;
  parameter Real m = 1;
    // mass
  parameter Real a0 = 9.81;
    // acceleration
  Real a(start=a0), v(start=0), h(start=1), f;
equation
  der(h) = v;
  der(v) = a;
  f = m*(a+a0); // gravitational force
  f = if h > 0 then 0 else -c*h;
// the velocity is reversed, i.e. -c*h
end bouncingball_simplest;
```



Potential and Flow Variables



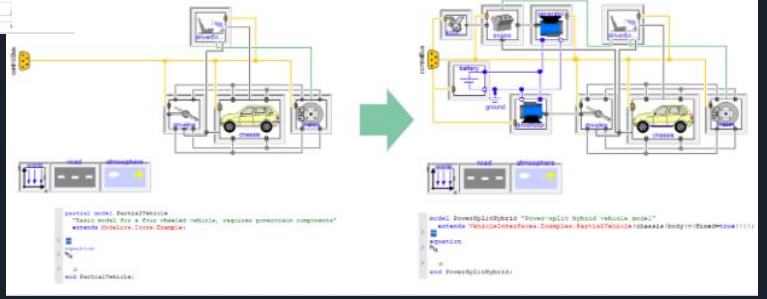
V1 = V2 = V3	**Potential:** n-1 equations per connection set.	S1 = S2 = S3
I1 + I2 + I3 = 0	**Flow:** 1 equation per connection set	f1 + f2 + f3 = 0

```
connect(R1.p, R2.p);
```

```
connect(m1.flang_a, m2.flange_a);
```

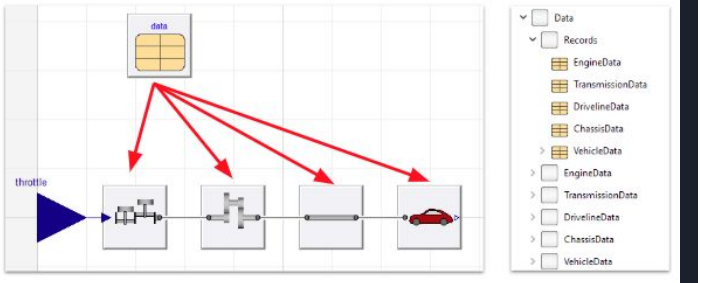
Templates

Now if we have interfaces for each subsystem, we can design a template with replaceable components:



Hierarchical Data Management

- The record can contain one replaceable record for each submodel

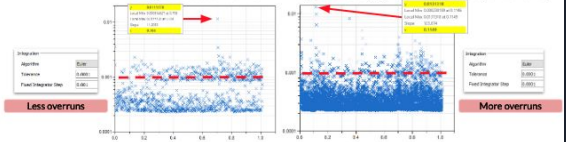


Why is real-time simulation needed?

- Certain applications requires real-time performance
  - Flight simulators
  - Car simulators
  - Grid simulators
- Model-based testing and development
  - Hardware in the loop
    - Test real control system against a model
  - Advantages
    - Desktop testing
    - Cost reduction, tests are expensive

Overruns

- Special event handling for *"inline integration"* in Dymola helps to avoid large overruns associated with event iterations
- If no overruns are allowed:
  - Average execution time << integrator step (much harder requirement)



- Good numerical implementation decreases overruns significantly, in Dymola, this is offered by *"inline integration"*

Homeworks - Interfacing with other tools

Workshop 15

This workshop will cover running dymosim.exe as a standalone program, using external input to dymosim.exe, implementation and use of an external C function, and simple usage of the Dymola-Python interface.

Part I - Interacting with Dymosim.exe

First, a test model is needed.

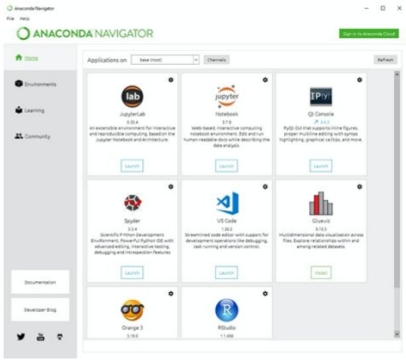
1. Create a new package Workshop15 inside the course package.
2. Duplicate model Modelica.Mechanics.Translational.Examples.Acceleration into your package and call it Acceleration
   When using the Translate button in Dymola the model is compiled into an executable binary file called dymosim.exe. In the next couple of steps you will create the binary and simulate the model manually.
3. Translate your model
4. Type cd in your commands window to see what your working directory is
5. Go to that location in your file explorer and identify what files were created when translating the model. Try sorting by date modified for easy identification.
6. Start windows command prompt. This can be done by searching cmd in your start menu
7. Change the directory to that of the newly created files using the commands *cd* to change the directory and *dir* to list the contents of the current directory
8. Once there, type dymosim.exe -h and press enter to show the syntax for executing dymosim and all the available options.

```
dymosim.exe started
  Usage: dymosim.exe [-ehilst] [-hdfoprvc] [file_in [file_out]]
         simulate, trim, linearize a model with dymosim
  Use at most ONE of the ACTION options (default: -s)
    -e       list command examples
    -h       list this command summary
    -i       generate Dymosim input file
    -l       linearize at initial value
    -s       simulate
    -t       trim (= implicit for -l,-s; initial value calculation)
    -m       multi-simulation
    -M       multi-simulation and stdout redirected to nul
  Use any of the additional options
    -i1 file  set default input file for multi-simulation (default multIn.csv) #5 means file-descriptor 5
    -o2 file  set default output header file for multi-simulation (default multOutHeader.csv)
    -d file  set default output file for multi-simulation (default multOut.csv)
    -b       generate file in binary format (only for -i, -t)
    -c cmd   execute command cmd after simulation end
    -d file  use input defaults from "file"
    -f file  store final states on "file" (default: "dsfinal.txt"; no log for file="nolog" only for -s)
    -n       if -t successful, use default weights in output file (only for -t)
    -o       generate input file in old format (only for -i)
    -p       precede output lines by program name (useful for multi-processes)
    -r       generate Dymosim input file using timer defaults (only for -i)
    -u file  read input signals from "file" (default: "dsu.txt")
    -v       verify input (store complete input in "dsinver.txt")
    -w file  write log output (default: "dslog.txt"; no log for file-"nolog")

Binary model export NOT activated.
```


Part III - Scripting using the Dymola-Python Interface

In this part of the workshop, you will learn to simulate examples of Modelica models using the Python interface for Dymola. In particular, the examples will be run by Python scripts in Jupyter Notebooks, an application that comes with Anaconda package. Please install Anaconda if you have not done so already.

After setting the Python interface for Dymola as indicated in the installation instructions, we can start testing the tools. First, launch Anaconda and you should get the following home page:



Then you can open Jupyter Notebook by clicking on the corresponding "*Launch*" button.
A browser web page should open with a list of accessible local folders:

```
In [9]: dymola.simulateExtendedModel("Modelica.Mechanics.Rotational.Examples.CoupledClutches",
                                     stopTime=1.5,
                                     outputInterval=0.001,
                                     method="Dassl",
                                     tolerance=0.0001,
                                     initialNames=["J1.J","J2.J"],
                                     initialValues=[2, 3],
                                     finalNames=["J1.w","J4.w"])
```

What do you get as output?

9. Now we can plot again the velocity of the inertia components:

```
In [10]: dymola.plot(["J1.w", "J2.w", "J3.w", "J4.w"])
         plotPath = "C:\\Users\\lab1\\Desktop\\DominionCourse\\plot2.png"
         dymola.ExportPlotAsImage(plotPath)
```

Compare this plot with the one obtained at point 6 and with the results at the previous point. What do you notice?

10. Let's check another extension of the "simulateModel" function that is "simulateMultiExtendedModel". The function handles a number of simulations. For each simulation it is possible to set parameters and start-values before simulation and to get the final values at end-point of simulation. This function is useful if, for example, we want to study the best parameter setup or the robustness of a parameter setup for a static simulation.
    Use the following commands and check the output:

```
In [13]: dymola.simulateMultiExtendedModel("Modelica.Mechanics.Rotational.Examples.CoupledClutches",
                                           initialNames=["J1.J","J2.J"],
                                           initialValues=[[2,3],[3,4],[4,5]],
                                           finalNames=["J1.w", "J4.w"])
```

Multithreading

The Python interface supports **multithreading**.

Each Dymola instance needs to have a unique port number. To set the port, *DymolaInterface* must be instantiated with the port argument.

To automatically find a free port, *DymolaInterface* can be instantiated either without the port argument, or setting port to -1.

Two Dymola instances that are run simultaneously cannot share the same working directory (to set the working directory, the interface method *cd* is used).

Homeworks - Student Solution

Mod and Sim for Cyber-Physical Systems

Workshop 15

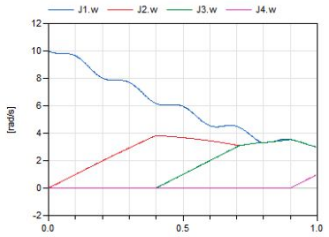
Jacob Montenieri

Scripting using the Dymola-Python Interface

After going through the installation instructions for the Python interface for Dymola, we can open Anaconda and then open a Jupyter Notebook. Adding the following code, we get the following result:

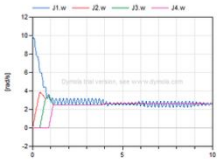
```
import platform

from dymola.dymola_interface import DymolaInterface
from dymola.dymola_exception import DymolaException

osString = platform.system()
isWindows = osString.startswith("Win")

dymola = None
try:
    # Instantiate the Dymola interface and start Dymola
    dymola = DymolaInterface("C:\\Program Files\\Dymola 2020\\bin64\\Dymola.exe")

    # Call a function in Dymola and check its return value
    result = dymola.simulateModel("Modelica.Mechanics.Rotational.Examples.CoupledClutches")
    if not result:
        print("Simulation failed. Below is the translation log.")
        log = dymola.getLastErrorLog()
        print(log)
        exit(1)
    dymola.plot(["J1.w", "J2.w", "J3.w", "J4.w"])
    if (isWindows):
        plotPath = "C:/Temp/plot.png"
    else:
        plotPath = "/tmp/plot.png"
    dymola.ExportPlotAsImage(plotPath)
    print("OK")
except DymolaException as ex:
    print("Error: " + str(ex))
finally:
    if dymola is not None:
        dymola.close()
        dymola = None
```

OK



After doing this, we can use the following code instead of the previous cell to plot the inertia components and get the following result:

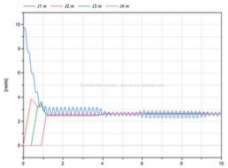
```
dymola.plot(["J1.w", "J2.w", "J3.w", "J4.w"])
plotPath = "C:\\Users\\montej7\\Documents\\Dymola\\der\\plot.png"
dymola.ExportPlotAsImage(plotPath)
```



Now if we want to plot the result in the Notebook we can also do this by using the following cell in replacement of the previous one and get the following result:

```
import platform
import scipy.io
import matplotlib.pyplot as plt
data = scipy.io.loadmat('C:\\Users\\montej7\\Documents\\Dymola\\der\\dsres.mat')
t = data['data_1'][0]
w1 = data['data_1'][4]
w2 = data['data_1'][25]
w3 = data['data_1'][44]
w4 = data['data_1'][63]
plt.plot(t, w1, label='w1')
plt.plot(t, w2, label='w2')
plt.plot(t, w3, label='w3')
plt.plot(t, w4, label='w4')
plt.xlabel('t [s]')
plt.ylabel('w [rad/s]')
plt.grid()
plt.legend()
```

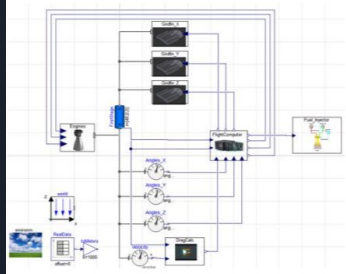
(Opening the plot with Dymola):



As we can see, this is the same plot as above.

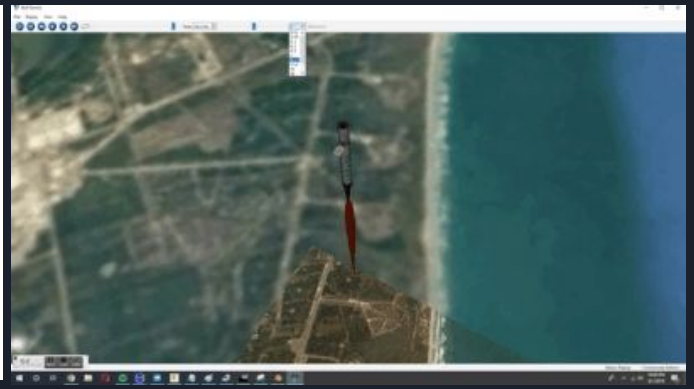
Sample Project F'18 - Christian Canham

Top Level Model

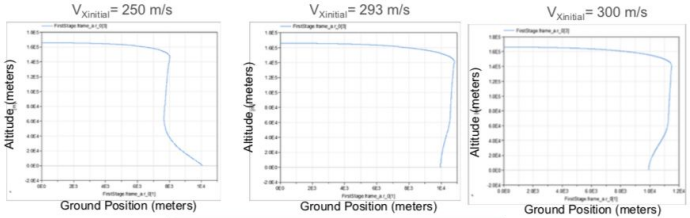


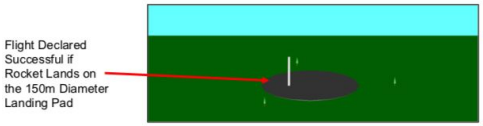
- Design Process
  - Develop landing rocket only in the Z-axis
  - Develop landing rocket in the X-Z plane with initial horizontal velocity and tilt
- First stage modeled with cylindrical multibody part
  - 48 meter length, 3.7 meter diameter, 0.3 g/cm^3
- Flight Computer
  - Inputs: clock, velocity, position, and angle data
  - Outputs: 3 grid fins angles, rocket engine thrust
- Fuel Injector shows rocket propellant needed for flight
- Drag takes the effects of air resistance into account

4



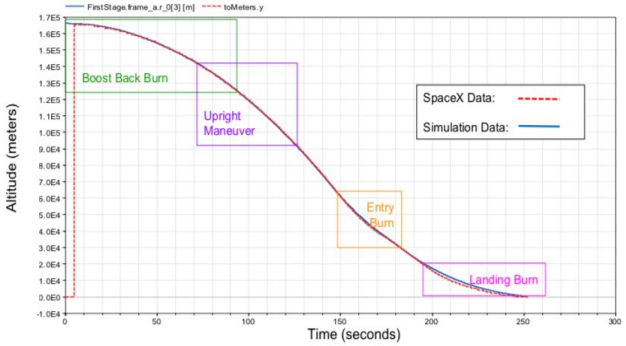
Simulation Flight Path





11

Comparison to Real Data



13

Sample Project Summer ’19 - Jose Ilton (KAUST)

Library for modeling simultaneous lightwave energy and data transmissions

Jose Ilton de Oliveira Filho[1]

[1]CEMSE, King Abdullah University of Science and Technology, Saudi Arabia

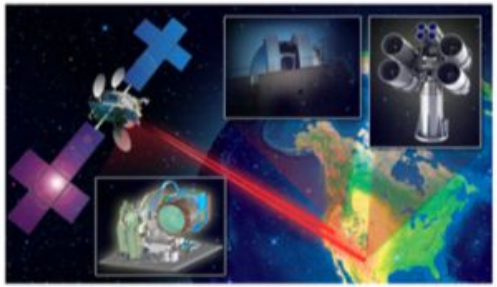


Figure 1. Nasa Laser Communication Relay Demonstration (LCRD) project

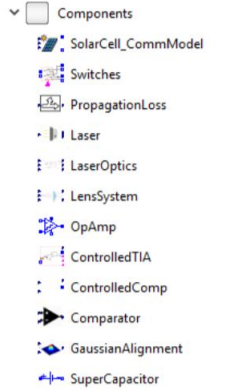


Figure 3. Components subfolder

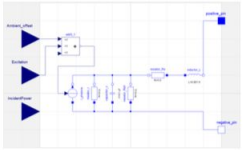
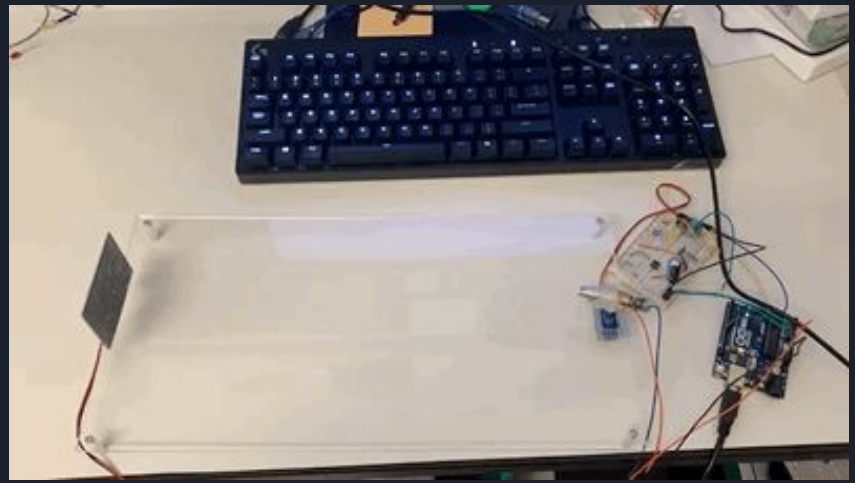


Figure 4. Solar cell model



Integrating Research Projects into the Course

Determining project topics:

- Evaluated the CHEETA framework and identified subsystems that could be modeled as a simple model using available Modelica libraries and materials. Defined parameters and details necessary for creating that model in the scope of the CHEETA aircraft.
- Students created their own project proposals from the materials we provided.

Project: LH2 Storage
ECSE 4961/6961

Background information and Objective:

- Design LH2 storage system
  - Use simple fuel cell model from the fuel cell libraries to develop an LH2 storage system
  - Should include a DC/DC converter for DC regulation and inverter as output of the storage system; should also study direct DC/AC connection from LH2 storage.
    - Up to 400 kW for converters
    - Use resistors as heat load if necessary
  - Focus on 100 gal system of LH2 maintaining a temperature of -423 deg F (source)

Suggested Libraries to Start With:

- Modelica Standard Library
- Modelon Liquid Cooling Library
- Modelon Fuel Cell Library
- Modelon Fuel Systems Library

Our assignment

Approach/Methodology:

1. After designing your DC/DC to DC/AC system and DC/AC system, study the thermal losses in the network.
   a. Study under different loading conditions
   b. Derive the total efficiency of the system under normal operation
2. Begin analysis with 25% of the nominal rating of the LH2 storage and then increase up to 100% in 10% intervals.
   a. Derive the total efficiency under these conditions at each storage level.
   b. Summarize the key design parameters influencing the operation of LH2 storage.

Subject and Background:

Subject Title: The Study of Liquid Hydrogen Fuel Cell Modeling

Background: The use of liquid hydrogen serving as a means to power small aircraft, such as Unmanned Aerial Vehicles, or UAVs, has become a popular product as it is an efficient and long-lasting way to provide electrical power. "The hydrogen chemical energy is converted to electrical energy through a series of fuel cells, which drive the ultra-efficient electric propulsion system" [1]. With the use of electrical power, the UAVs become quiet, easy to control, and reduce its thermal signature [2]. Naval Research Laboratory has been designing an UAV called the Ion Tiger UAV with the use of liquid hydrogen, or LH2, to extend the duration of flight and output high, efficient energy to power their UAV. However, there is always a cost considering that are conditions that must keep LH2 stable, such as, by keeping the temperature within the UAV to 20 Kelvins in a well-insulated tank and keeping the tank at a pressure of 50psi that way its chemical energy can be converted to electrical energy.

Goal and Objective:

Goal: To use the Modelica Library Packages to develop a thermal model of liquid hydrogen fuel cell providing power to a system.

Objectives:

- To learn how a liquid hydrogen fuel cell works.
- To learn how to use the Modelica Fuel Cell library and other libraries to develop model.
- To analyze different models and connection layouts for different fuel cell power supply systems.

Approach and Methodology:

The first task is to research how the fuel cells work in regard to their thermal properties and to use the Modelica Fuel Cell, Heat Exchanger, and Liquid Cooling Libraries as a start to develop the model. On top of that, to research any modeling techniques to help develop the model.

Next, with the research gathered, create a simple fuel cell model with liquid hydrogen as the fuel source and use the proper equations to simulate the model correctly.

Furthermore, enhance the model to simulate it thermal properties assuming that the liquid hydrogen is in a 20 Kelvin atmosphere.

With all enhancements, test the model to show its thermal properties where it is converting chemical energy in the fuel cells to electrical, providing power to the system.

Finally, simulating the model to show visualized data of the fuel cells of its thermal properties to improve the overall understanding of the thermal properties of the fuel cells.

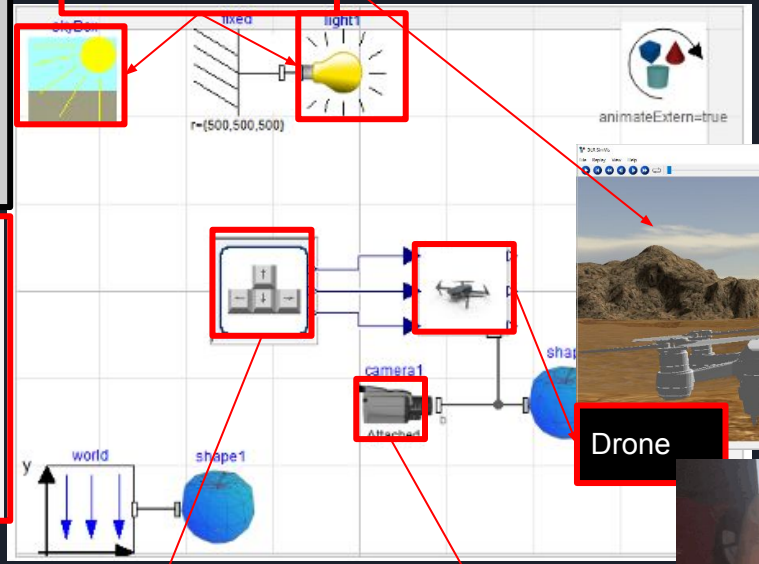
Student proposal

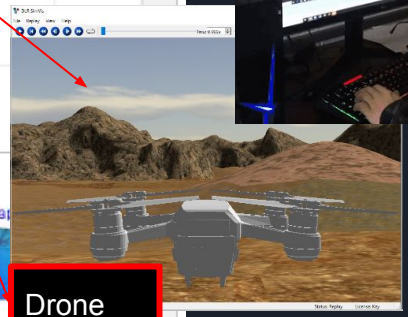
CHEETA-inspired Project F'19: Jake Monteneri

Virtual reality-based design with visualization

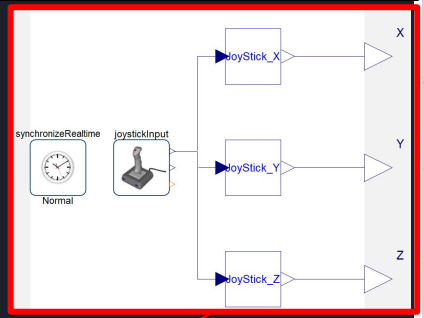
- Expose students to the Visualization library and integrating CAD models to allow for better visual representations
- Expose students to methods for design interaction (e.g. keyboard, joysticks)

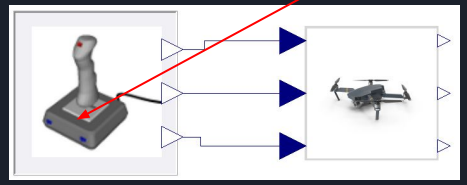
Environment





Drone





Keyboard input

Camera follows drone around environment

CHEETA-inspired Project F'19: Jordan Grey

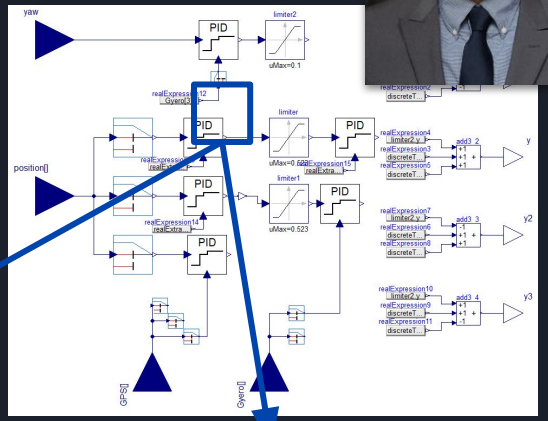
Multi-domain drone modeling and simulation using synchronous control

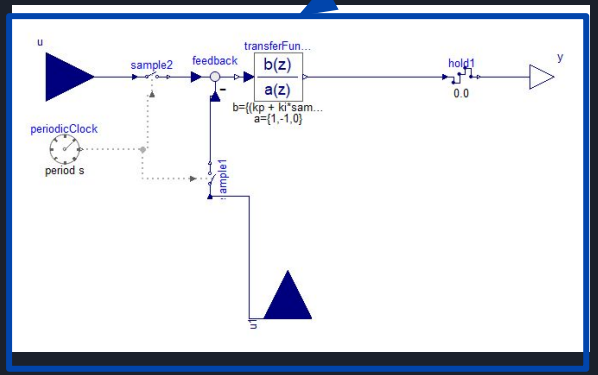
Goals:

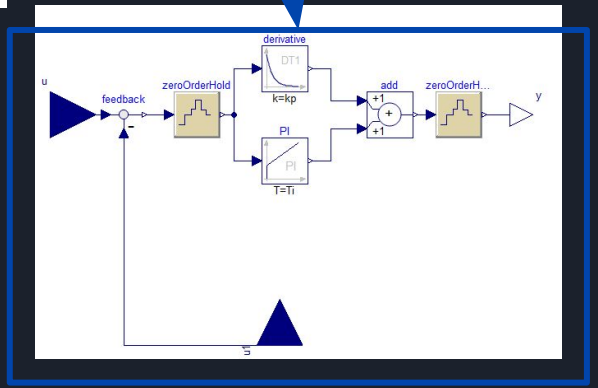
- Implement different drone models using a previously established drone model to provide a proof of concept for reusable models in other aircraft applications
- Use Modelica Synchronous Library to improve simulation performance for control components

Outcomes:

- Implemented a control system with faster simulation time, which will be helpful in modeling EMS
- Added mechanical models for the MavicAir and Phantom drone so system identification studies can be expanded to other drone models.







Synchronous Discrete PID block block

Discrete PID

M&S *for* CPS

Development and Roll-Out of Lab Activities

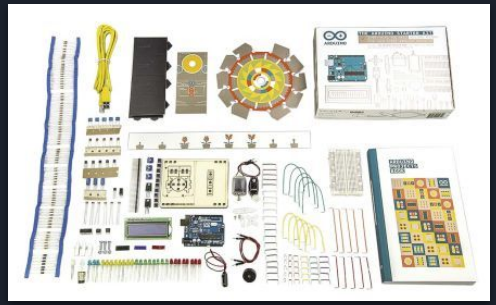
Supported through Boeing's Corporate Grant for Teaching

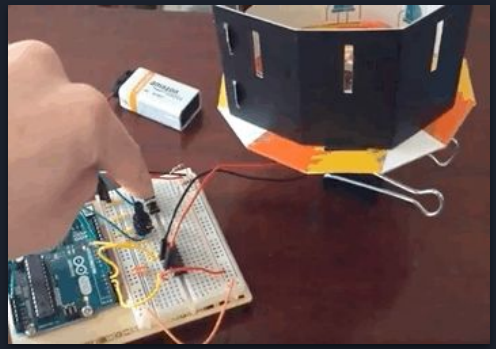
Development of Lab. Sessions

Lab. 01: Digital Twins for the Arduino Starter Kit Projects

- ***Learning Objectives:***
  - Challenge the students to apply the course skills, while at the same time, comparing their models with an actual cyber-physical system.
  - Build a simple CPS system: Arduino (controller/software) + physical system
  - Model the physical system, and learn to integrate the software with the model.
  - Learn to identify and explain differences of models (digital twins) vs. reality (measurements)!

- ***Lab. Tasks:***
  - Students build a HW prototype and take measurements.
  - Students create a complete CPS model from scratch, integrating software (Arduino controller code) with the model of one of the projects.
  - Contrast the simulation with the model, identify differences, etc.

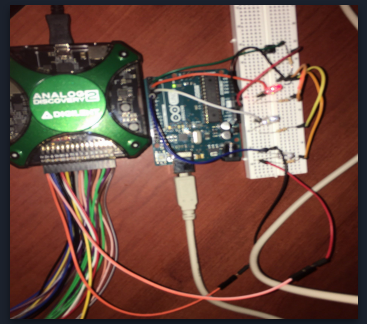


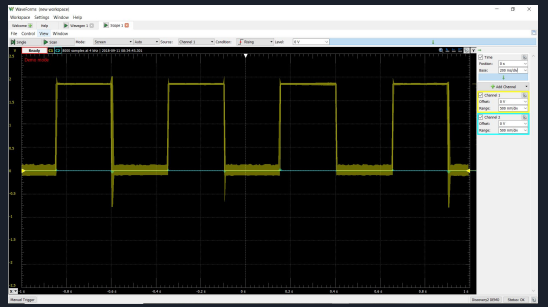


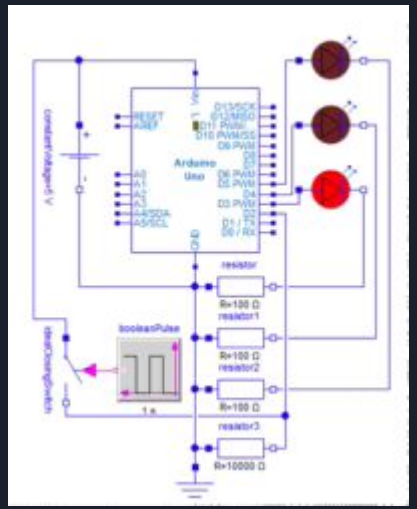
Lab. 01: A Digital Twin for the Arduino Starter Kit Projects

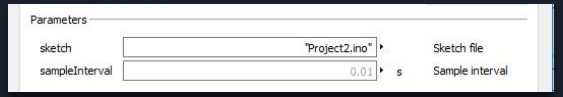
The lab is organized as follows:

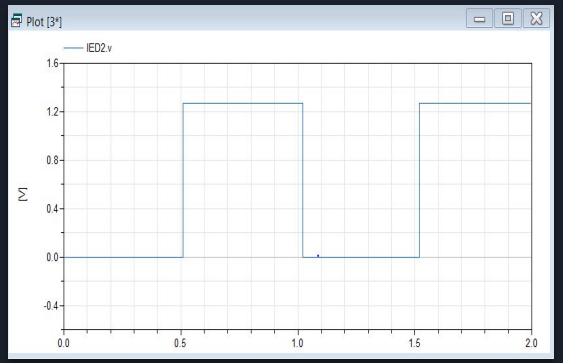
- ***Before the lab:*** The students are asked to put together the prototype for the “Spaceship Interface” project before arriving to the lab, together with measurement points.
- ***During the lab:***
  - An introduction on the use of the Modelica Arduino library is given.
  - The “Spaceship Interface” project Tutorial is provided in the following order:
    - First the circuit portion
    - Next the Arduino is coupled with the circuit, and the Arduino software is integrated with the model.
  - Measurements are compared to the model, followed by a discussion of the differences.
- ***After the lab:*** The students are assigned a different project from the Arduino Starter Kit and are asked to repeat the process above for a different system.





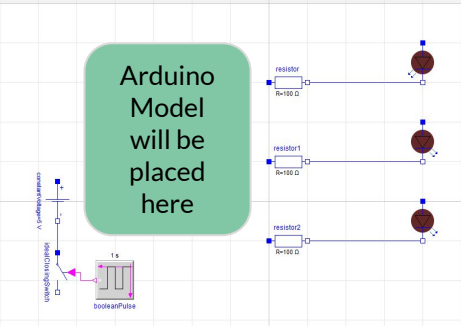






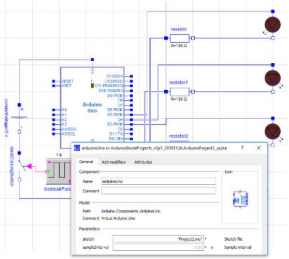
Lab. 01: A Digital Twin for the Arduino Starter Kit Projects

Lab Introduction

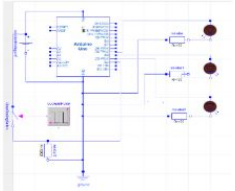


Interfacing the Arduino code to the Model

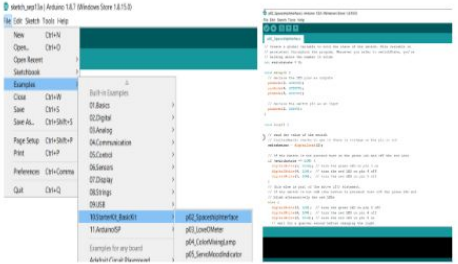
- The software part
  - Once you have built the circuit, you need to put the "sketch" (.ino) file in the right directory → "MySketch.ino"
- To simulate the model with "MySketch.ino"
  - Save your sketch as ./Arduino/Resources/Sketches/MySketch.ino
  - Add the block Arduino.Components.ArduinoUno to your model
  - Double-click the block and set the parameter sketch to "MySketch.ino"
  - Click to re-translate the model when the sketch has changed



Lab Guide - Showing how the Arduino's code is interfaced with the model.



14. Now, open the Arduino IDE and open or copy the code for the Spaceship Interface example in the Arduino IDE. To do this, go to File -> Examples -> 10.StarterKit_BasicKit -> p02_SpaceshipInterface

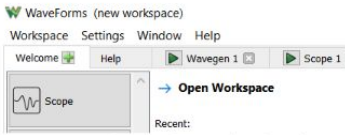


15. Save this file in the same folder as your Spaceship Interface model file
16. Your folder should now have both the sketch file from the Arduino IDE and your Diagram model from Dymola as seen below.

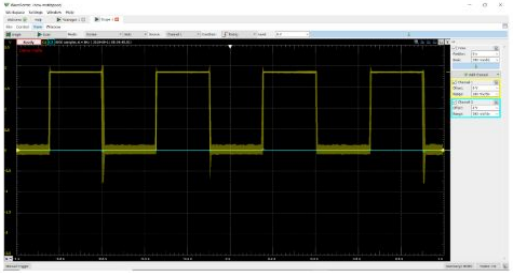
ArduinoProject2	6/4/2018 2:11 PM	MO File	4 KB
Project2	7/2/2018 4:34 PM	INO File	2 KB

17. Now, double click on your ArduinoUno board to edit the sketch file.

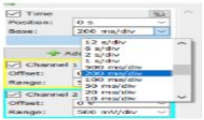
Lab Guide - Showing how to acquire and compare measurements and the model's simulation



29. Navigate to the scope and measure the Voltage of your LED over time as you hold down the push button on your circuit. Do this by pressing the green arrow play button in scope. Press the stop button once you have collected enough data.

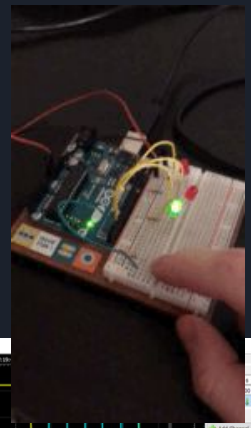


30. Take a screenshot or use snipping tool to capture the image of the graph as seen above.
31. If you need to change the time interval of data collection. Go to the right side of the screen in the time task bar and change the base to a greater time per division in the drop down menu.



Test Run of Lab 01 Guide

Independent Study Course Re-Run S'19
Full Course Taught at KAUST Summer 2019



Jack Frey

661318804

CPS Arduino Lab

3/31/19

Spaceship Interface

The spaceship interface is a simple Arduino project designed for beginners. It consists of 3 LEDs and a switch. Without pressing the switch, a single green LED (LED3) is lit. When the switch is pressed, the other two LEDs (LED1 and LED2) flicker back and forth until the switch is released, while LED3 is off. Because of the simplicity of this example, it was a good choice for first time modelling of an Arduino project in dymola.

Following the Arduino starter kit handbook and using the Arduino components package for dymola, a model of the spaceship interface was created.

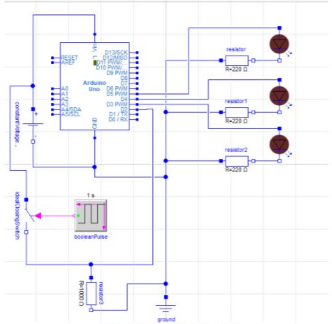


Figure 1 - Spaceship Interface Dymola Model

This model was simulated, so that results could be compared to the real-world equivalent. Initially, I chose to compare results relating to the voltage drop across the push button. The Dymola simulation results are shown below for the push button:

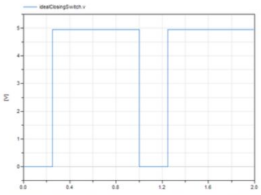


Figure 2 - Dymola Simulation Results for Voltage Drop Across Push Button

And now the results from the Analog Discovery 2 on the real-world experiment.

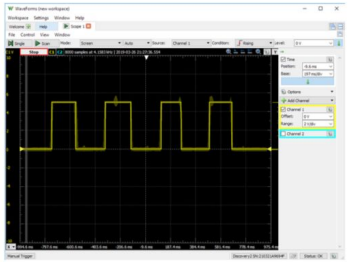


Figure 3 - Real-World Results for Voltage Drop Across Push Button

Comparing the results above, it is evident from the plots that they are very similar. The voltage drop across the dymola switch is slightly higher than that in the real-world one, but this is to be expected, as the dymola switch is an ideal switch with no losses or resistance at all. This could be accounted for in the dymola model by using a different switch model with more realistic characteristics if desired.

Next, and lastly, I chose to compare results for the 3 LEDs in the simulation and the real-world model.

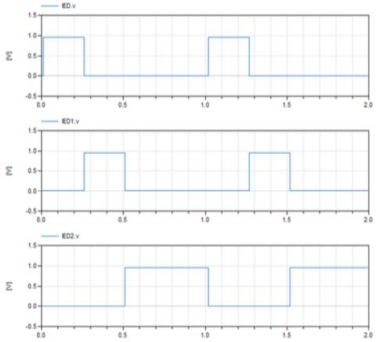


Figure 4 - Three LED Voltage Drops for Dymola Model

For the real-world results, the Analog Discovery 2 only supports two simultaneous channels, so LED1 is not included – however, it is just the opposite of LED2 when the pushbutton is pressed so its results are not really needed.

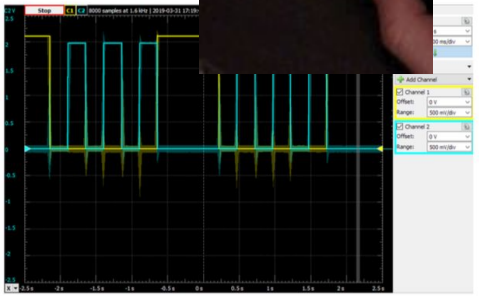


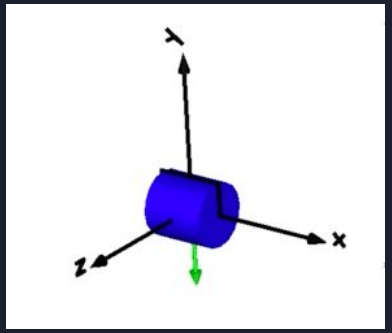
Figure 5 - LED Voltage Drops for Real-World Experiment

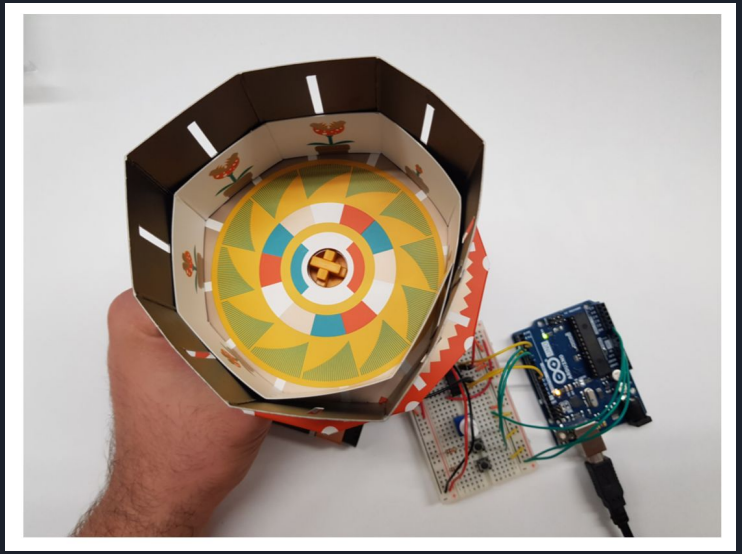
Comparing these results, a few things stand out. The behavior of the LEDs in both results are the same, but the voltage differentials are different. In the dymola results, the voltage drop across each LED is about half of what it is in the real-world results. Also, in the real-world results, the voltage drops are different for LED2 (channel 2) and LED3 (channel 1). This should not be the case because their respective circuits are the same, with the same resistor in place and setup. Regarding this voltage drop difference, it is most likely some small error with the Arduino, Analog Discovery, or most likely a small resistance difference between the green LED and red LED. Regarding the difference in voltage drop between the real-world results and the Dymola simulation however, I believe that there is a difference in resistance of the LED or the Arduino with respect to the other type of model. Another possible reason is that the voltage on the Dymola Arduino pins is different than that on the real Arduino pins. In order to alleviate these issues, delving deeper into the specifics of the modelica Arduino modelling package is necessary.

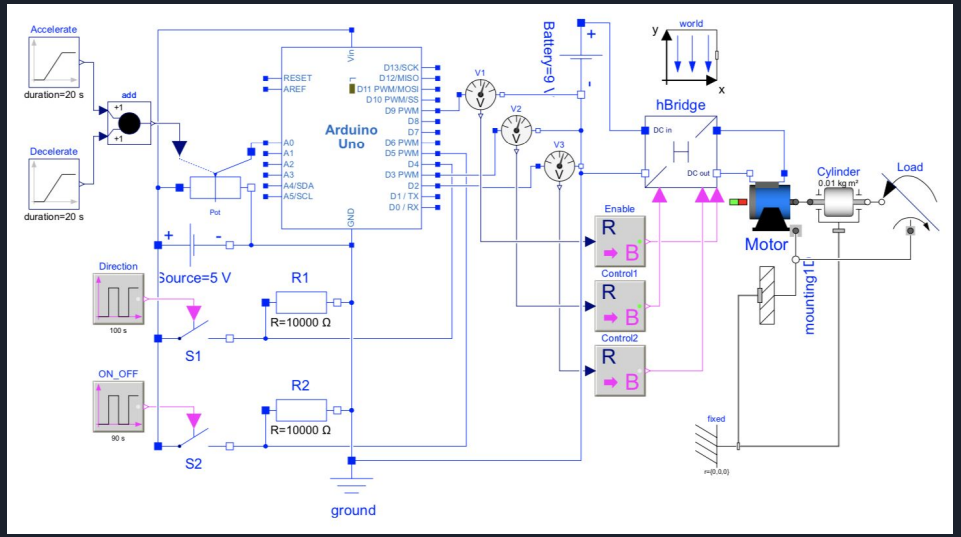
Test Run of Lab 01 Assignments

Full Course Taught at KAUST Summer 2019

Zoetrope Project



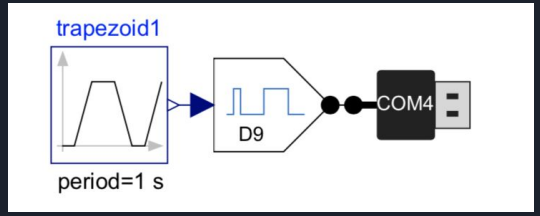


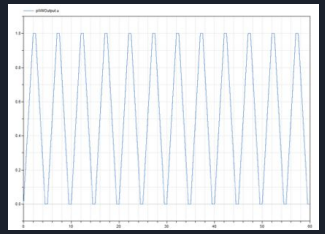


Development of Lab. Sessions

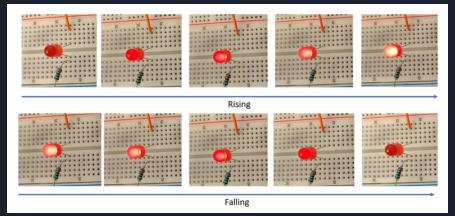
Lab. 2: Real-Time Hardware-in-the-Loop Simulation

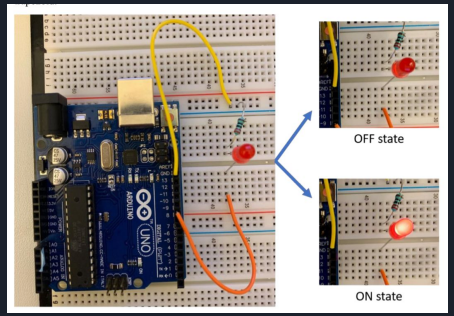
Led Dimming Exercise





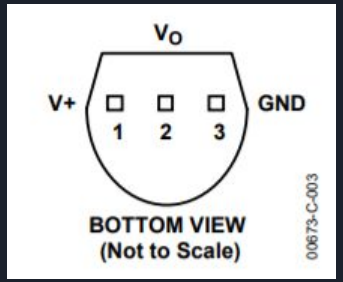
- ***Learning Objectives:***
  - Apply knowledge on real-time simulation through a hands-on hardware-in-the-loop example.
  - Learn to interface a model (control) to external hardware via serial communications.
    - Learn to use libraries for integrating HW with software.
    - Learn to configure communications between the simulator (computer) and the hardware.
  - Learn about issues with real-time simulation:
    - Solver selection, step size selection, noise, etc.
- ***Tasks:***
  - Build the HW prototype.
  - Interface the model (of controls/logic), with the external cyber-physical system.
  - Acquire data to analyze performance of the controls.
  - Extend the controls for additional actions.

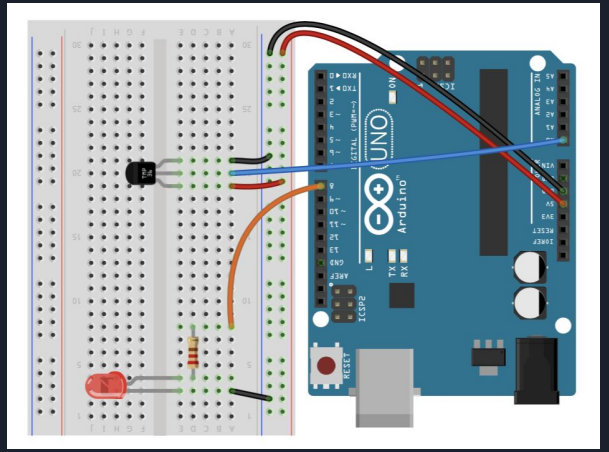




Lab. 2: Real-Time Hardware-in-the-Loop Simulation

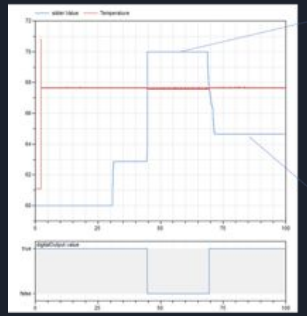
Temperature Monitor Exercise

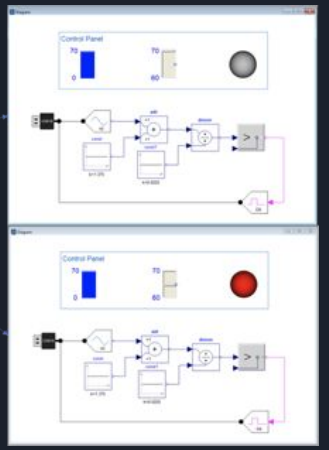




The lab is organized as follows:

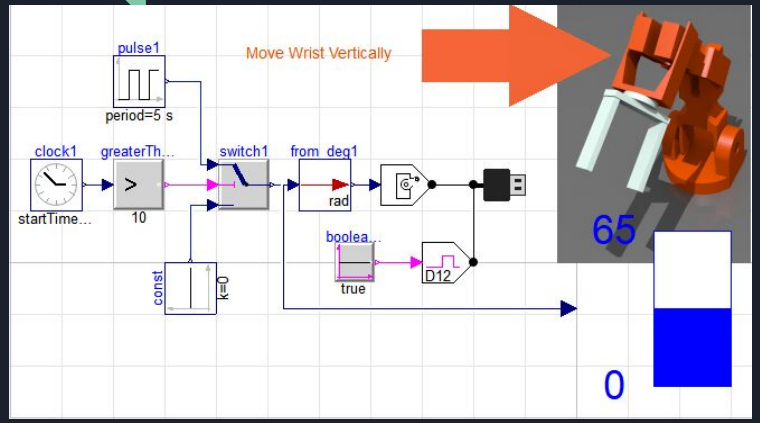
- ***Before the lab:*** The students are asked to install the Modelica Arduino library and configure the Arduino software "Firmata" protocol for data exchange between the computer and the hardware.
- ***During the lab - Tutorial /Portion:***
  - A tutorial using three examples is given:
    - Dimming a LED Example
    - Temperature monitor/alarm example.
    - Robotic Arm control
  - Simulation logs and results are analyzed.
- ***After the lab - Independent Lab/Portion:***
  - The students prepare a lab report.

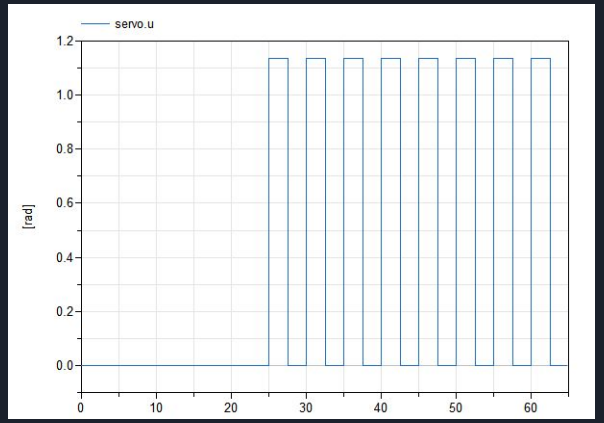


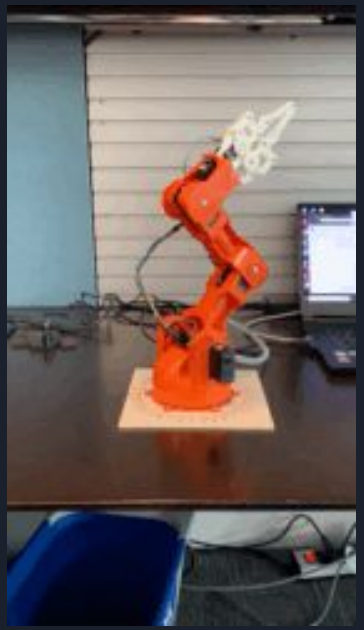


Lab. 2:
Real-Time Hardware-in-the-Loop Simulation

Robotic Arm Exercise







Lab. 2:
Real-Time Hardware-in-the-Loop Simulation

Robotic Arm Exercise

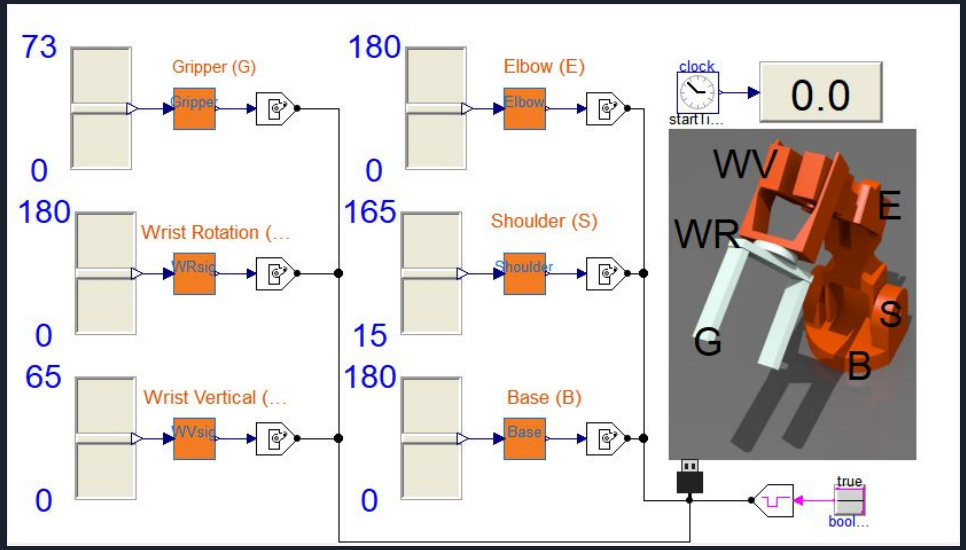


Table – Behavior of the arm by the input range.

	Gripper (GR)	Wrist Rotation (WR)	Wrist Vertical (WV)	Elbow (Elb)	Shoulder (S)	Base
0-10	Shaking	Shaking	Shaking	NR/Shaking	N/A	NR/Shaking
20-30	Responsive	Shaking	Responsive	NR/Shaking	NR/Shaking	NR/Shaking
30-40	Responsive	Responsive/ Shaking	Responsive	NR/Shaking	NR/Shaking	NR/Shaking
40-50	Responsive	Responsive	Responsive	Responsive/ Shaking	Responsive/ Shaking	NR/Shaking
50-60	Responsive	Responsive	Responsive	Responsive/ Shaking	Responsive/ Shaking	NR/Shaking
60-70	Shaking	Responsive	Responsive	Responsive/ Shaking	Responsive/ Shaking	NR/Shaking
70-80	Shaking	Responsive	N/A	Responsive/ Shaking	Responsive/ Shaking	NR/Shaking
80-90	N/A	Responsive	N/A	Responsive/ Shaking	NR/Shaking	NR/Shaking
90-100	N/A	Responsive	N/A	Responsive/ Shaking	NR/Shaking	NR/Shaking
110-120	N/A	Responsive	N/A	Responsive/ Shaking	NR/Shaking	NR/Shaking
120-130	N/A	Responsive	N/A	Responsive/ Shaking	NR/Shaking	NR/Shaking
130-140	N/A	Responsive	N/A	Shaking	NR/Shaking	NR/Shaking
140-150	N/A	Responsive/ Shaking	N/A	Shaking	NR/Shaking	NR/Shaking
150-160	N/A	Responsive/ Shaking	N/A	NR/Shaking	NR/Shaking	NR/Shaking
160-170	N/A	Shaking	N/A	NR/Shaking	N/A	NR/Shaking
170-180	N/A	Shaking	N/A	NR/Shaking	N/A	NR/Shaking

NR = Not responsive; N/A = not applicable (more than the maximum permitted); Shaking = the arm looks that it is trying to get in a position but comes back, and repeats this movement in a loop.

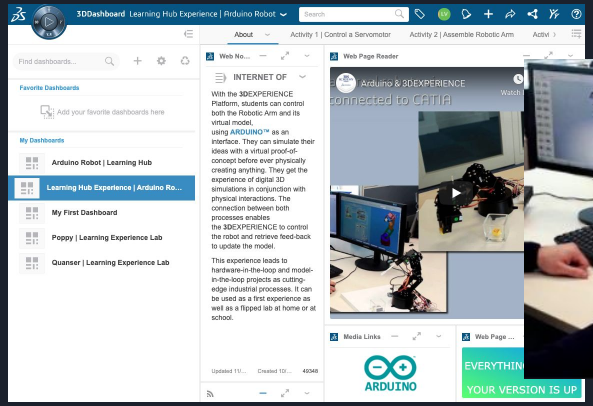
M&S *for* CPS

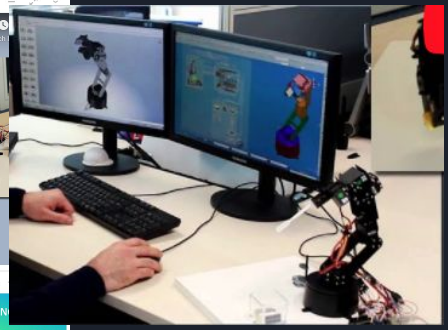
Conclusions and Future Work

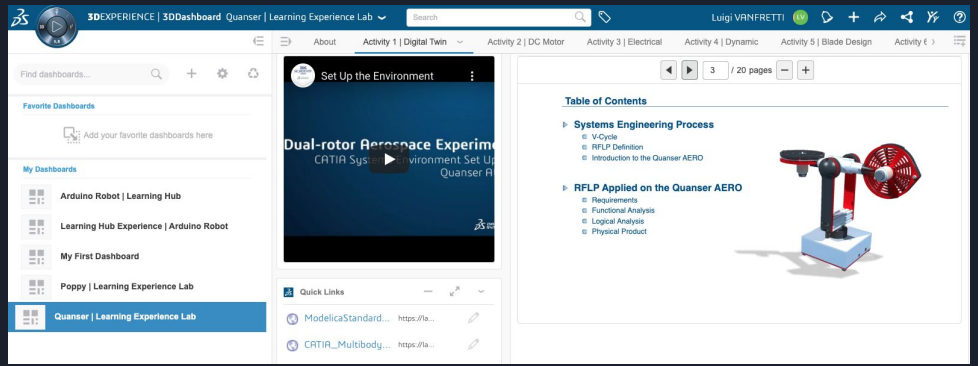
Next Steps:

Ongoing and Future Course Dev. Activities

- VR-Based Early Design Interaction for UAV's
  - Student course project from Fall '19!
  - Need to provide access to Modelica Visualization Library licenses to students (to be evaluated)
- 3DS Experience
- Ongoing discussions with Dassault to have access for the 3DS platform for use in the course
- Goal:
  - Expose the students to state-of-the-art collaborative engineering tools and methods
  - Introduction to Model-Based System Engineering
  - e.g. coupling of 3D/CAD Model coupling with system engineering studies
- Required efforts:
  - Solve access to 3DS licenses
  - Develop course lectures, HW's, etc.
  - Test Run before introducing to the course







Conclusions and Future Work

- The course has been offered in three occasions:
  - F'18, F'19 (RPI), Summer '19 at KAUST.
  - Student feedback is encouraging and good reception.
- Hands-on Labs
  - Lab 1: student feedback is that they are surprised on how difficult it is to develop models that match reality, even for simple systems.
  - Lab 2: first trial showed that the arm has to be built before the lab, 2nd trial showed the students really enjoy hands-on real time simulation.
  - Introduction of lab activities was appreciated by the students, easier to see the utility of models
- Future work:
  - Engaging with industry to develop course projects and provide guidance to students!
  - Please reach out if you want to help!

It is a very interesting and modern course. I really enjoyed my learning about the topic. For being the first year of this course I am satisfied with it and I hope it will be given in the next years.

It was a very fun course and I learned a lot since all of the homeworks were very informative and brought us through every aspect of the software. It should be continued to be offered in the future :)

The course was generally well-defined in terms of the learning objectives. The students were exposed to the (nearly) full range of capabilities of Modelica as a modeling and simulation tool. The biweekly workshops, although a hassle at first, turned out to be essential for learning Modelica at an accelerated pace.

ALSETLab

Thank you!

Prof. Luigi Vanfretti

ALSETLab, ECSE, RPI

http://ALSETLab.com vanfrl@rpi.edu