EMULATION OF AEROSPACE ACTUATION SYSTEMS | SR DESIGN

Presented by Kori Eliaz, Jake Dorsett, & Dylan Gaub

Faculty Advisor: Dr. James Cale
Engineer in Residence: Matt Heath
Industry Partner: Woodward, Inc.

EAAS

AGENDA

01 OUR TEAM

02 PROJECT BACKGROUND

03 PROJECT GOALS

04 FALL PROGRESS

05 NEXT STEPS

EAAS

OUR TEAM

KORI ELIAZ

Electrical Engineering
Project Manager
System Modeling

JAKE DORSETT

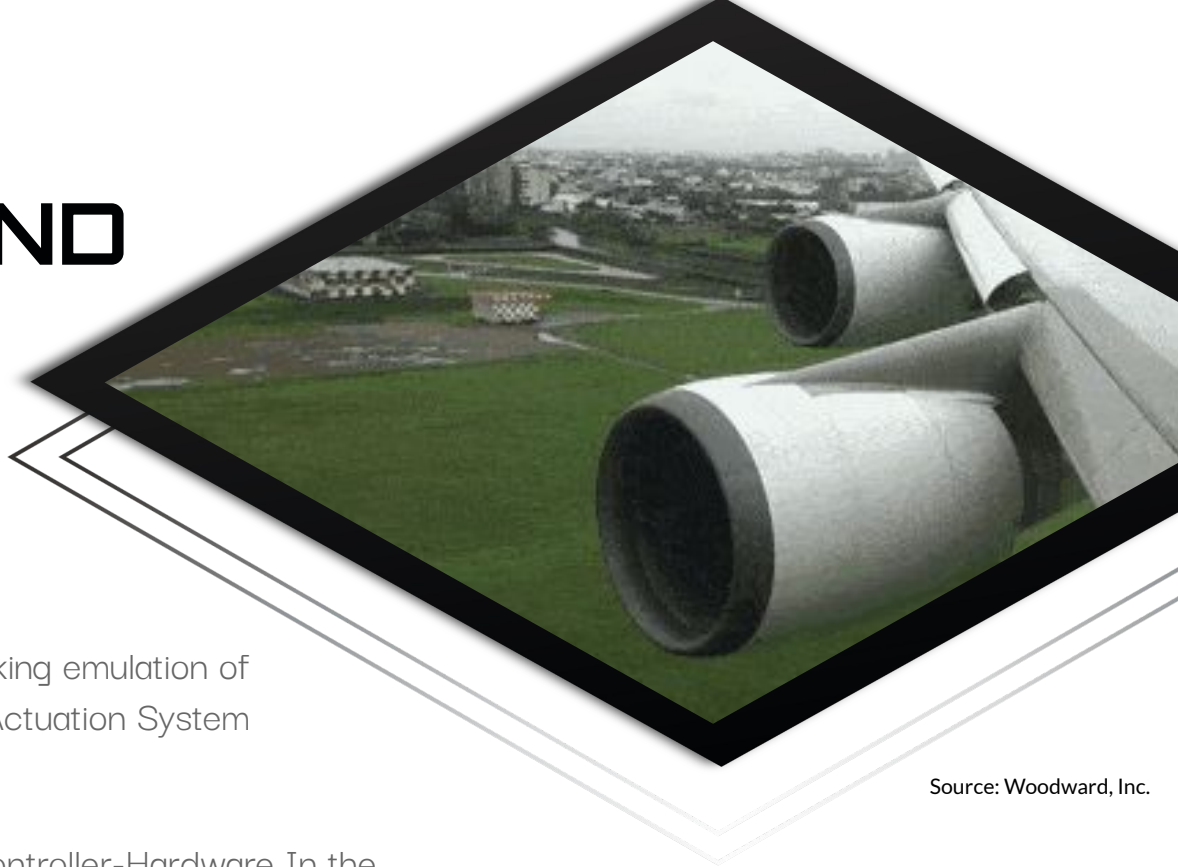
Electrical Engineering
Project Design
Hardware Interfacing

DYLAN GAUB

Computer Engineering
Project Design
Software Architecture

EAAS

PROJECT BACKGROUND



Source: Woodward, Inc.

Key Motivations:

✈ More Electric Aircraft (MEA)

- Reduced weight
- Reduced maintenance
- Better RT data analytics

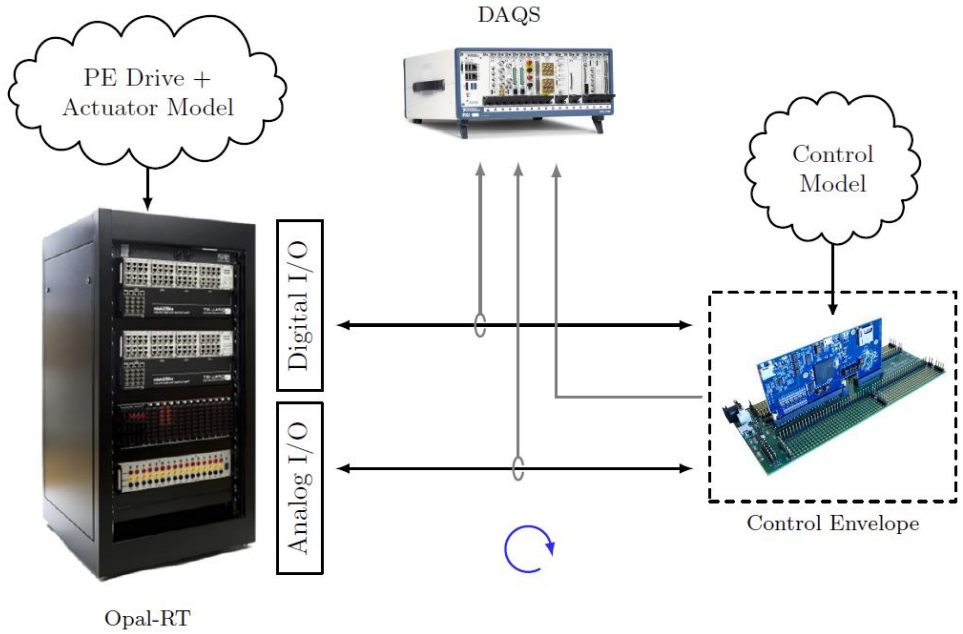
✈ Woodward's focus: Develop working emulation of Electromechanical Thrust Reverser Actuation System (EMTRAS)

✈ **Our focus:** Develop a working Controller-Hardware In the Loop (CHIL) Platform for use by Woodward, Inc. and CSU students/faculty

EAAS

WHAT IS CHIL?

Controller **H**ardware **I**n The **L**oop



FALL PROJECT GOALS

- Determine I/O requirements for TI and OPAL-RT ✅
- Develop C code for a CHIL experiment on a real-time DC machine system model ✅
- Execute CHIL experiment and collect measurement information ✅
- Include system requirements flow-through and design information using MBSE ✅

EAAS

TIMELINE

Introduction and research into the OPAL-RT and TI Microcontroller

Sep 2021

Oct 2021

MBSE Requirements & Diagrams

HIL with 2 Inputs and Outputs

Nov-Dec 2021

Jan 2022

Begin creation of User Interface for DAQ

Simulations with interactive UI and testing case scenarios

March 2022

Apr 2022

Final demonstration at E-Days

EAAS

BUDGET

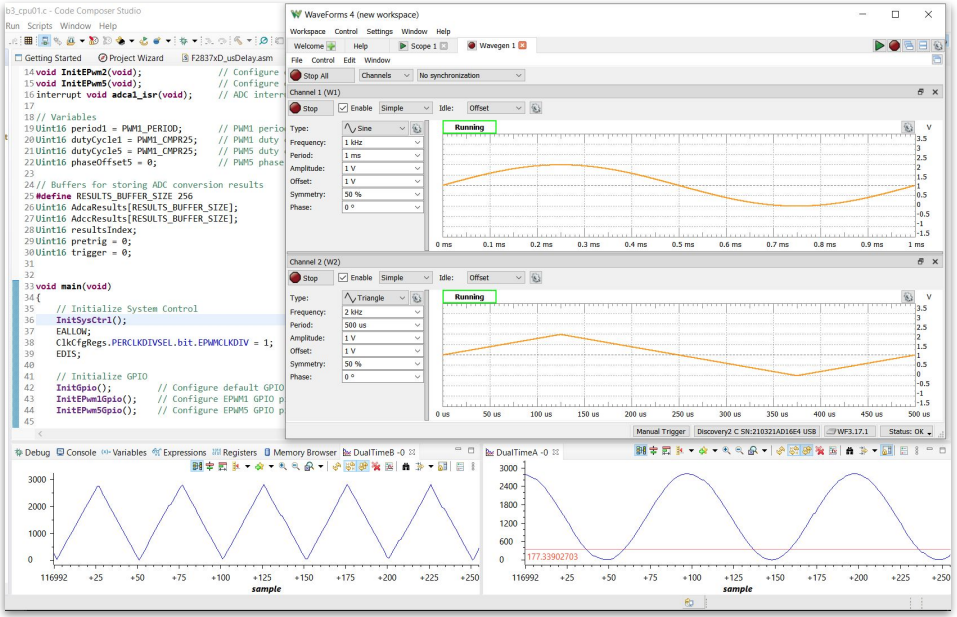
Expense	Amount	Qty	Status	Purchased	Reimbursed
BNC to MiniBNC Connector	$11.57	1	Returned	9/21/2021	9/28/2021
Replacement Connector	$7.63	1	Arrived	9/27/2021	9/28/2021
Additional TI Controller	$250.00	1	Pending	1/2022	TBD
E-Days Materials	$50.00	1	Pending	4/2022	TBD

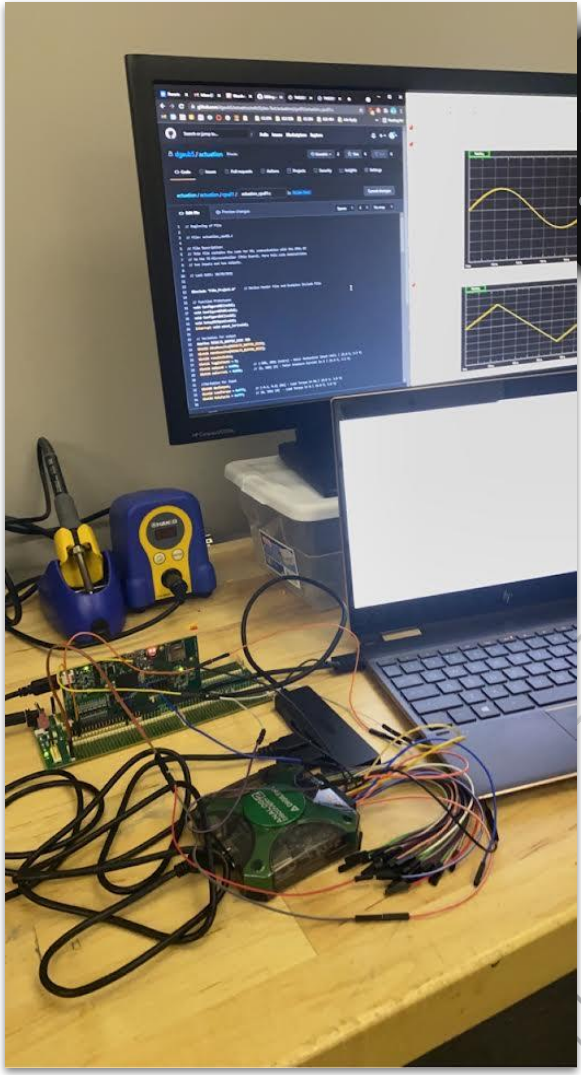
Starting Balance	**$600.00**
Total Spent	**$(319.20)**
Total Remaining	**$280.80**

EAAS

FALL PROGRESS

- Read documentation for the TI Microcontroller
- Explored TI libraries and example labs for digital and analog I/O
- Used Analog Discovery and Waveforms

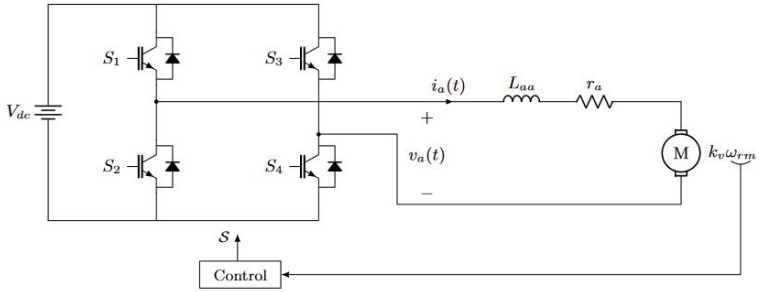


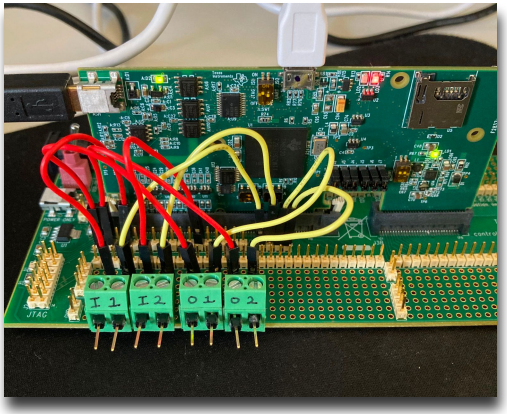


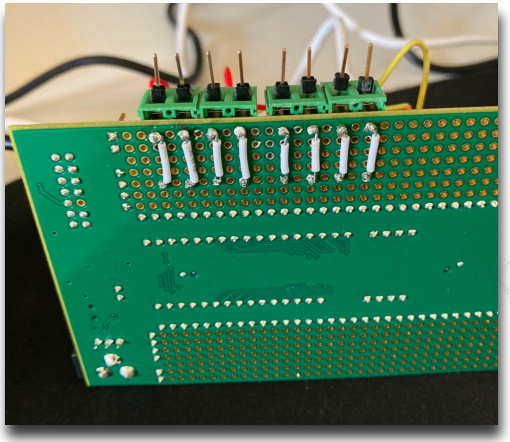
EAAS

FALL PROGRESS

- Simplified EMTRAS Simulink Model
- Loaded Model onto OPAL-RT
- Interfaced TI Microcontroller with OPAL System





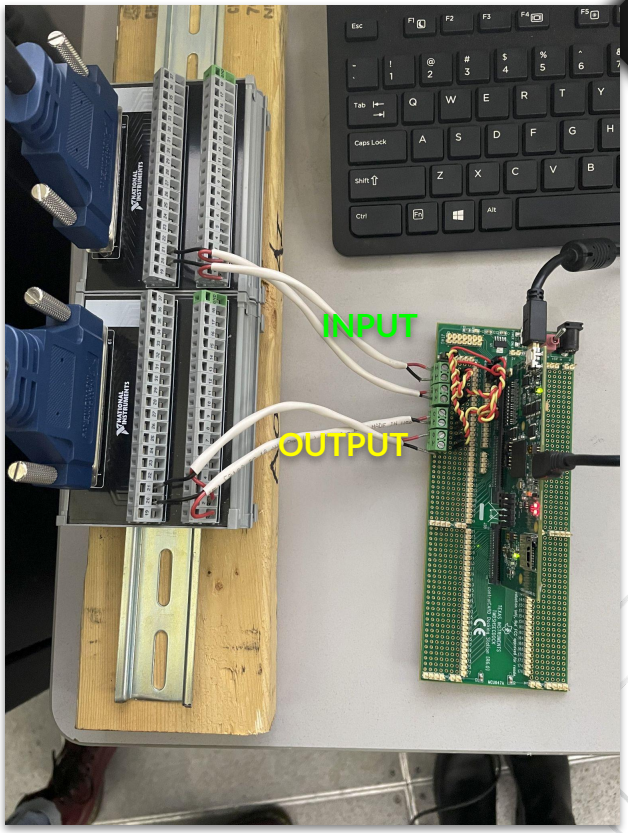


FALL PROGRESS

- Two DB-37 Cables with 4 wires on each cable, interconnection between OPAL and TI

- Receiving two signals from the OPAL-RT:
  1. Motor Speed (-600 to +600 rad/s +/- 3%)
  2. Motor Armature Current (-2.5 to 2.5 A +/- 3%)

- Send two signals from the microcontroller:
  1. Load Torque (-0.2 Nm to +0.2 Nm +/- 3%)
  2. Control Level Duty Cycle (0-100% +/- 3%)



EAAS

FALL PROGRESS

- Inputs are as follows:

```
12  #include "F28x_Project.h"           // Device Header File and Examples Include File
13
14  // Variables for output
15  Uint16 resultsIndex;
16  Uint16 ToggleCount = 0;
17  Uint16 mmSpeed = 0x000;         // {-600, 600} [rad/s] - Motor Mechanical Speed rad/s | {0.0 V, 3.3 V}
18  Uint16 maCurrent = 0x000;       // {0, 100} [A] - Motor Armature Current in A | {0.0 V, 3.3 V}
19
20  //Variables for input
21  Uint16 dacOutput;
22  Uint16 LoadTorque = 0.1;      // {-0.2, 0.2} [Nm] - Load Torque in Nm | {0.0 V, 3.0 V}
23  Uint16 DutyCycle = 95;       // {0, 100} [%]  - Duty Cycle in % | {0.0 V, 3.0 V}
24
25
26  // Definitions for PWM generation
27  #define PWM1_PERIOD 0xC350          // PWM1 frequency = 2kHz
28  #define PWM1_CMPR25 PWM1_PERIOD>>2  // PWM1 initial duty cycle = 25%
29
30  // Function Prototypes
31  void ConfigureADC(void);
32  void ConfigureDAC(void);
33  void ConfigureEPWM(void);
34  void SetupADCEpwm(void);
35  void InitEPwm1(void);               // Configure ePWM module 1
36  void InitEPwm2(void);               // Configure ePWM module 2
37  void InitEPwm5(void);               // Configure ePWM module 5
38  interrupt void adca1_isr(void);     // ADC interrupt service routine
```

```
123         DutyCycle = (DutyCycle/100)*4095;          // convert Duty Cycle to bit representation
124         LoadTorque = (10237.5)*(LoadTorque)+2047.5; // convert Load Torque to bit representation
```

EAAS

FALL PROGRESS

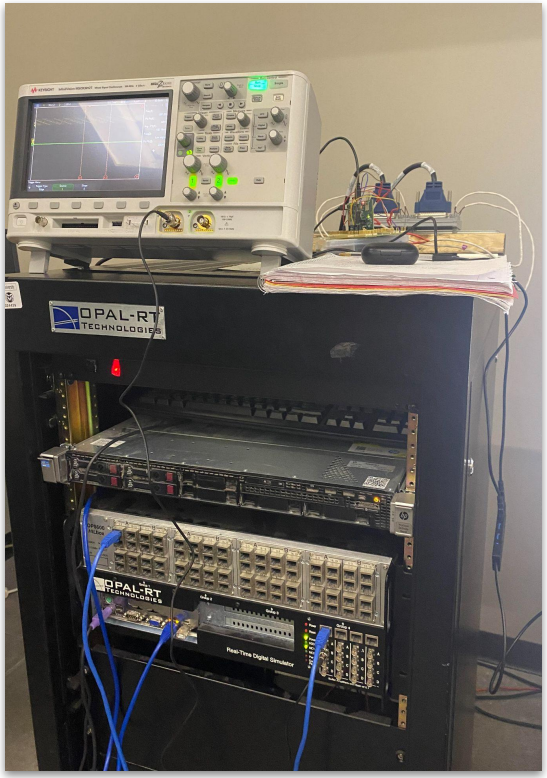
- Setting I/O pins inside of Code Composer Studio for TI

```
134 void ConfigureDAC(void)
135 {
136     EALLOW;
137     DacaRegs.DACCTL.bit.DACREFSEL = 1;          // Use ADC references (HSEC Pin 09)
138     DacaRegs.DACCTL.bit.LOADMODE = 0;           // Load on next SYSCLK
139     DacaRegs.DACOUTEN.bit.DACOUTEN = 1;         // Enable DAC
140     DacbRegs.DACCTL.bit.DACREFSEL = 1;          // Use ADC references (HSEC Pin 11)
141     DacbRegs.DACCTL.bit.LOADMODE = 0;           // Load on next SYSCLK
142     DacbRegs.DACOUTEN.bit.DACOUTEN = 1;         // Enable DAC
143     EDIS;
144 }
```

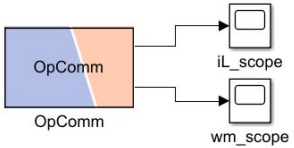
```
169
170 void SetupADCEpwm(void)
171 {
172     // Select the channels to convert and end of conversion flag
173     EALLOW;
174     AdcaRegs.ADCSOC0CTL.bit.CHSEL = 2;          // SOC0 will convert pin A0 (HSEC Pin 15)
175     AdcaRegs.ADCSOC0CTL.bit.ACQPS = 14;         // Sample window is 100 SYSCLK cycles
176     AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 7;        // Trigger on ePWM2 SOCA/C
177     AdcaRegs.ADCINTSEL1N2.bit.INT1SEL = 0;      // End of SOC0 will set INT1 flag
178     AdcaRegs.ADCINTSEL1N2.bit.INT1E = 1;        // Enable INT1 flag
179     AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;      // Make sure INT1 flag is cleared
180
181     // Also setup ADC-C3 in this example
182     AdccRegs.ADCSOC0CTL.bit.CHSEL = 3;          // SOC0 will convert pin C3 (HSEC Pin 33)
183     AdccRegs.ADCSOC0CTL.bit.ACQPS = 14;         // Sample window is 100 SYSCLK cycles
184     AdccRegs.ADCSOC0CTL.bit.TRIGSEL = 7;        // Trigger on ePWM2 SOCA/C
185     EDIS;
```

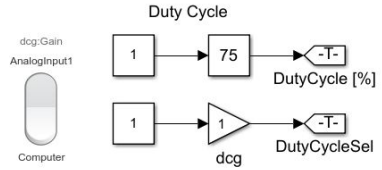
EAAS

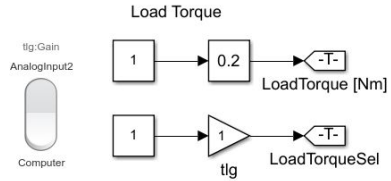
FALL PROGRESS

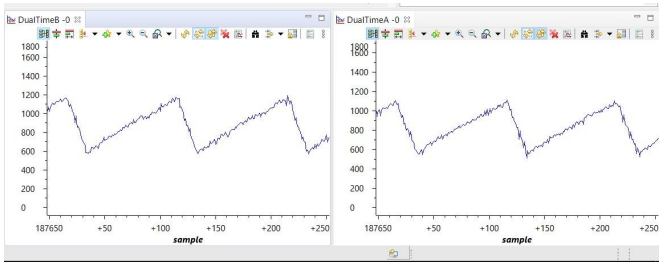


Automatically generated by RT-LAB during compilation.







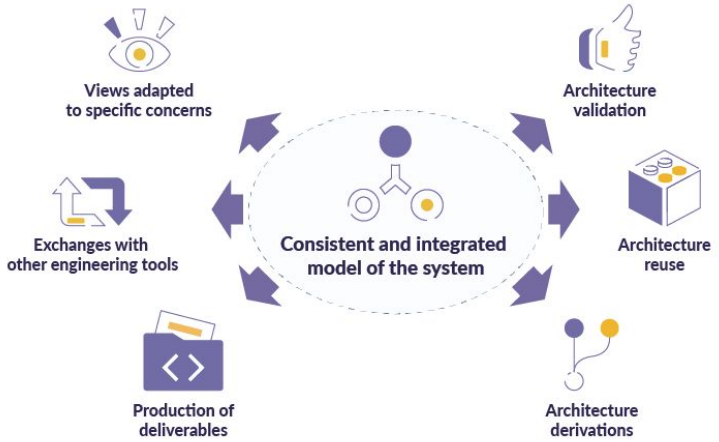


EAAS

WHAT IS MBSE?

Model **B**ased **S**ystems **E**ngineering

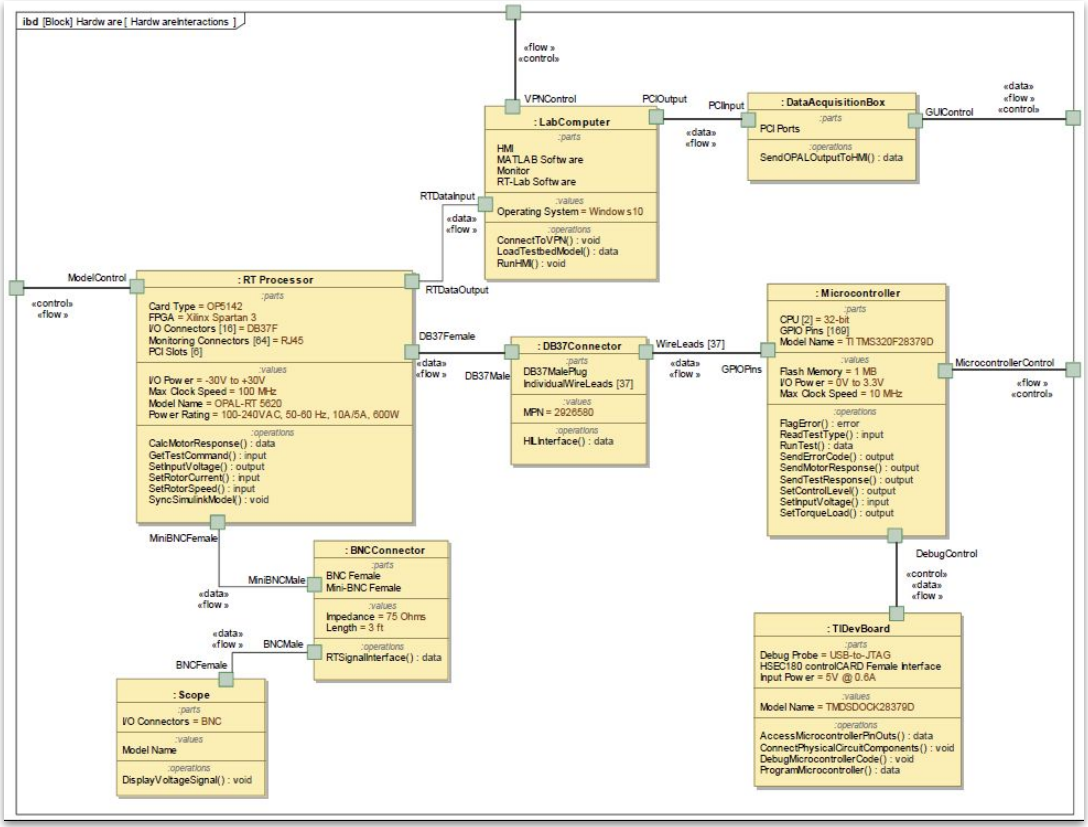
- **Formalized methodology** used to support requirements, design, analysis, verification, and validation of complex systems
- Puts **models** (rather than documents) at the **center of system design**
- Provides a **central repository of information** that describes the system
- **Highly relevant topic in modern systems engineering** and pertains directly to **current industry practices**



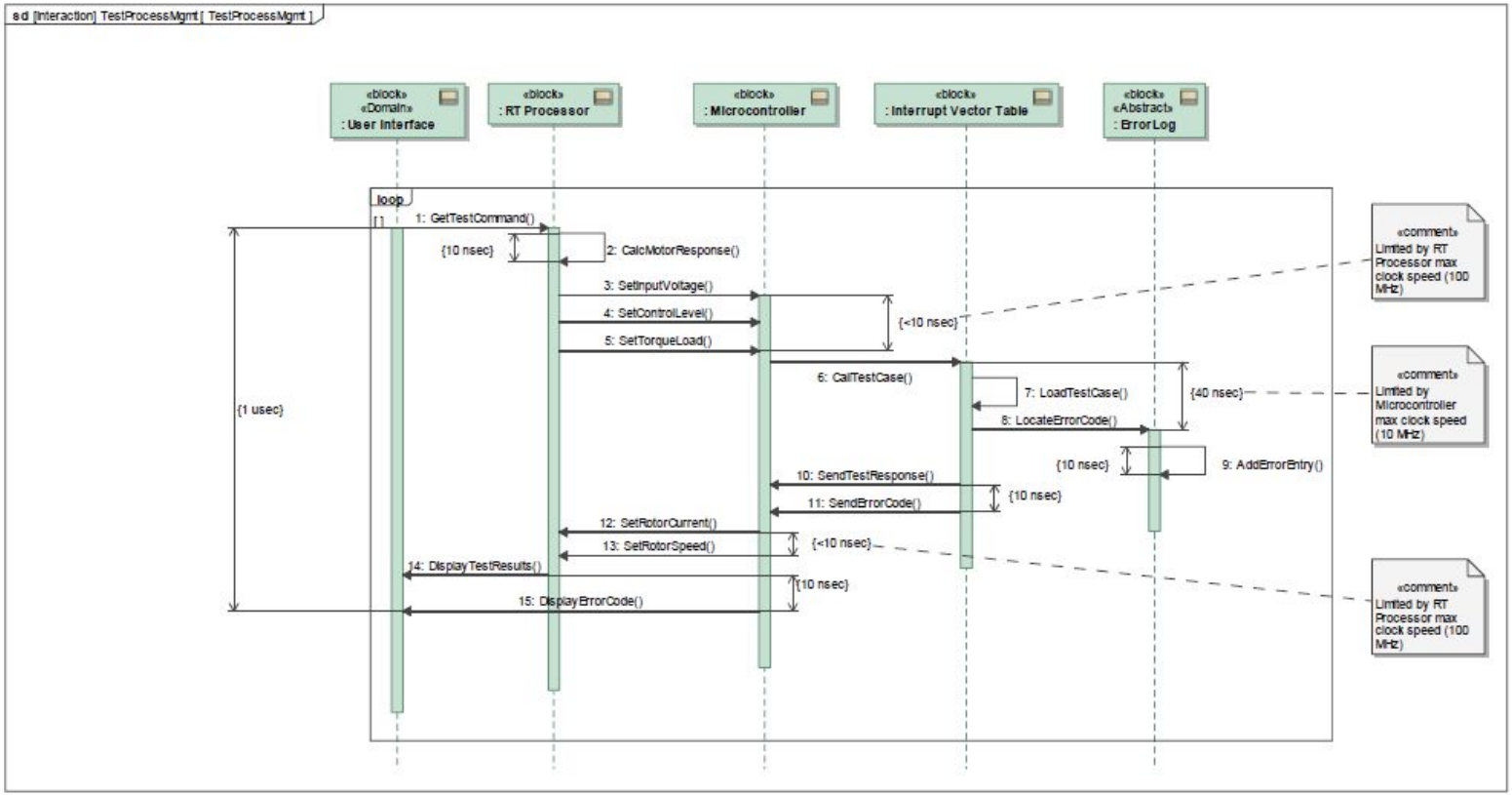
Source: eclipse.org

EAAS

INTERFACE MODELING



TIMING REQUIREMENTS



EAAS

NEXT STEPS FOR SPRING '22

- Develop GUI using LabView for the CRIO and then the PXI DAQ Systems
- Extract data from the OPAL-RT and process in MATLAB using FFT to identify source of output noise
- Replace hard-coded inputs with variables that can be adjusted via the GUI
- Run a documentation tool on the code to extract an HTML report of all documentation and create a comprehensive user guide
- Support conference paper on CHIL, MBSE, and testbed emulation of EMTRAS

EAAS

THANK YOU!

Questions?

Visit Our Website!

Contact Us:

kori.eliaz@colostate.edu
jake.dorsett@gmail.com
dgaub@rams.colostate.edu

EAAS