# Complex Modeling for Feed Forward Control System for Furnace Temperature Optimization

**Thomas Prica, Sawyer Poel, and Aleksandr Sergeyev**

**Mechatronics/Electrical and Robotics Engineering Technology/College of Computing**
**Michigan Technological University, Houghton, Michigan**

## Abstract

Simulink PLC Coder extension for Mathworks Simulink modeling software provides a novel approach for designing and exporting complex controls systems for PLCs and PACs. Our senior design team utilized the software to better optimize furnace controls in a feed forward system for our industrial sponsor. This report will reflect on the process of using PLC coder to model and create IDE specific structured text code for the furnace control system. The coder extension allows the user to build a complex system model in Simulink, including mock-ups of the controls and plant. Simulink can verify operation through simulation and built in diagnostics. After the model is verified, the controls system can be isolated and PLC Coder can generate PLC code from it for multiple target IDEs. The resulting code will be commented and tagged to assist in the ease of integration within an existing system. This process will greatly reduce the time for delivery of complex controls systems for industrial PLC applications.

## 1. Introduction

As modern industry evolves to meet increased demands for safety, speed, and quality from customers so must control systems. Increasingly complex industrial processes require more robust control systems to operate. An extension of Mathwork's MATLAB software, Simulink, is a program designed to model and verify real word systems in a simulated environment. Simulink is an invaluable tool for control designer as it can be used to observe processes and take data in a cost-effective way. MathWorks took the practicality of the Simulink software one step further in 2016 with the release of PLC Coder, an extension designed to take complex control systems modeled in Simulink and convert them to PLC code for multiple IDEs [1]. Not only does PLC Coder develop the PLC code text, it can also thoroughly comment the code, generate testbenches, and troubleshoot the Simulink model to be best adapted for PLC.

This paper will cover the process for designing a steel furnace model using Simulink, how to properly design the system to work with PLC Coder, and how to convert the code to PLC. The data and furnace model used in this paper were pulled from a senior design project the authors have completed. The purpose of the paper is to introduce the power of PLC Coder and how it can be used in the creation of complex control system.

## 2. Similar Work

Festo AG a robotics company used PLC Coder in the pursuit of creating an industrial robot which could interact directly with operators. The robotic arm deemed, the Bionic Handling Assistant, operated with 11 degrees of freedom, 12 pneumatic chambers, 13 actuators, and 12 position sensors [2]. In order to save time and money, Festo AG opted to model the controls in Simulink instead of using a traditional hardware prototyping approach. From the Simulink model Festo AG was able to generate IEC 61131 structured text code for PLC integration. Vintecc, a Belgian consulting firm, used PLC Coder to help design the controllers for a 780hp custom harvesting machine. The new machine was designed to haul 100 tons of produce at a time and operated on multiple axes. By implanting a model-based design through Simulink Vintecc was able to complete the project months ahead of schedule since 90 percent of the design was verified before hardware was introduced [3]. Once again, the PLC Coder was used and generated IEC 61131 structured text for PLC, specifically CODESYS compliant structured text. Engel, a global leader in injection molding machines, services customers in the automotive, packaging, and medical industries. When designing the injection molding machines, they used to have teams of engineers create structured text for the control systems by hand. Once created the controls would be verified using a mix of simulated and hardware tests [4]. By introducing PLC Coder, Engel was able to reduce testing and coding time by design control systems within the Simulink environment.

## 3. Modeling Systems in Simulink

Before Simulink is a powerful tool for modeling and testing real world systems; it uses a visual block diagram system which the user connects routes between function blocks to create models. To better understand how to create PLC Coder-friendly models in Simulink, the user must be able to identify four main components in the system being modeled.

- Inputs: Every system will have some kind of input, these inputs can range from sensor readings that can be counting, testing level, taking temperature, and many more actions. Identifying the inputs before modeling will help the designer better understand how to integrate the created PLC code into a real-world system.
- Control Model: Once inputs are identified a control system which takes and manipulates the data from the inputs can be created. The control system is designed to control a determined process, this means the controls can be as simple as a greater than or less than function or as complex as a PID controller with feedback.
- Plant Model: In order to properly model a control system, the plant must also be modeled in Simulink. The plant should be created to mimic the real-world process being controlled. In the case of this paper, the plant model will mimic a furnace for heating strips of steel to a predetermined temperature.
- Outputs: The inputs are the parameters which drive the control system, outputs on the other hand are used to drive the plant operation. The outputs will be taken from the control model and are used to control the real-world plant once converted to PLC code.

For the purpose of this paper, a model of a steel furnace is used as an example of a system modeled in Simulink. The example shown in Figure 1 is labeled to help identify the four main components (Figure 1): one for inputs, two for control model, three for plant model, and four for outputs.

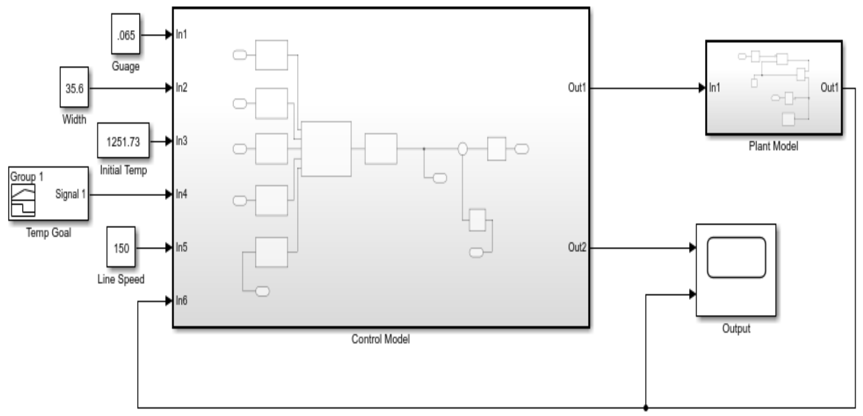


Figure1. Example of control system in Simulink

## 4. Preparing the Model for Conversions

For the PLC Coder to effectively convert the Simulink model for the controls into structured text usable by a PLC, a few alterations need to be made. First, the control and plant models need to be separated into individual subsystems. Next, all functions in the control subsystem must be changed to respond discretely. Finally, the control subsystem must be changed to an atomic subsystem.

### A. Isolating the Controls from the Plant

To properly transfer the Simulink model to PLC code, the controls in the model and plant model need to be separated into individual subsystems. This is done to isolate the controls, since they are the portion of the model that is to be converted to PLC code. Figure 2 shows an example of a combined furnace model; the line represents where the model would be separated in order the isolate the controls from the plant.

### B. Convert to Discrete Operation

Next, all function blocks within the control subsystem must operate discretely. This is done so the system is optimized to run on the weaker hardware found in PLCs. If run in continuous mode, the function blocks will operate as though they are receiving and sending analog signals; this takes much more computing power to operate quickly. In discrete mode, the function blocks will act as though they are operating on digital signals.

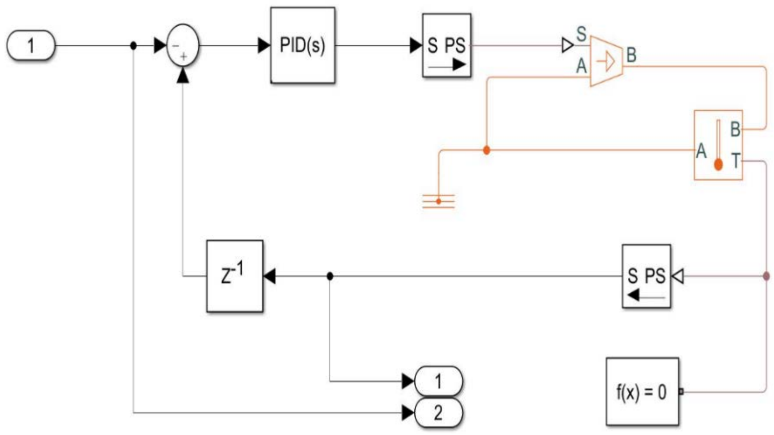


Figure 2. Isolating the control from the plant

C. Treat Subsystem as Atomic Unit

Finally, the controls subsystem must be converted to be treated as an atomic unit as shown in Figure 3. The subsystem is treated as an atomic unit in order to prevent execution order errors among the function blocks within the subsystem. It ensures that the execution order also will not have conflicts with other subsystems created on the same level as the control subsystem [5].

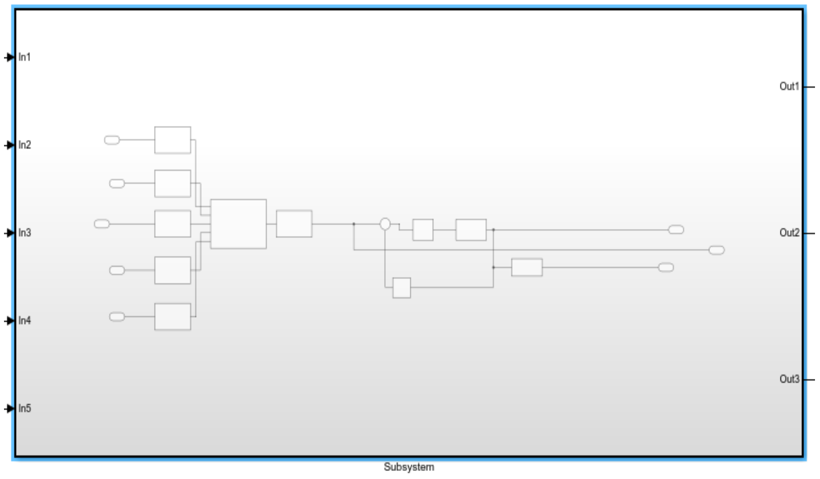


Figure 3. Atomic subsystem

## 5. Converting to PLC

The first step to converting the Simulink model to PLC is verifying that the model meets the requirements for PLC Coder. This is done by right clicking the model and navigating to the "Check Subsystem Compatibility" tab as shown in Figure 4. Once the compatibility check is run, any errors will appear in a separate window which will identify them in the model.

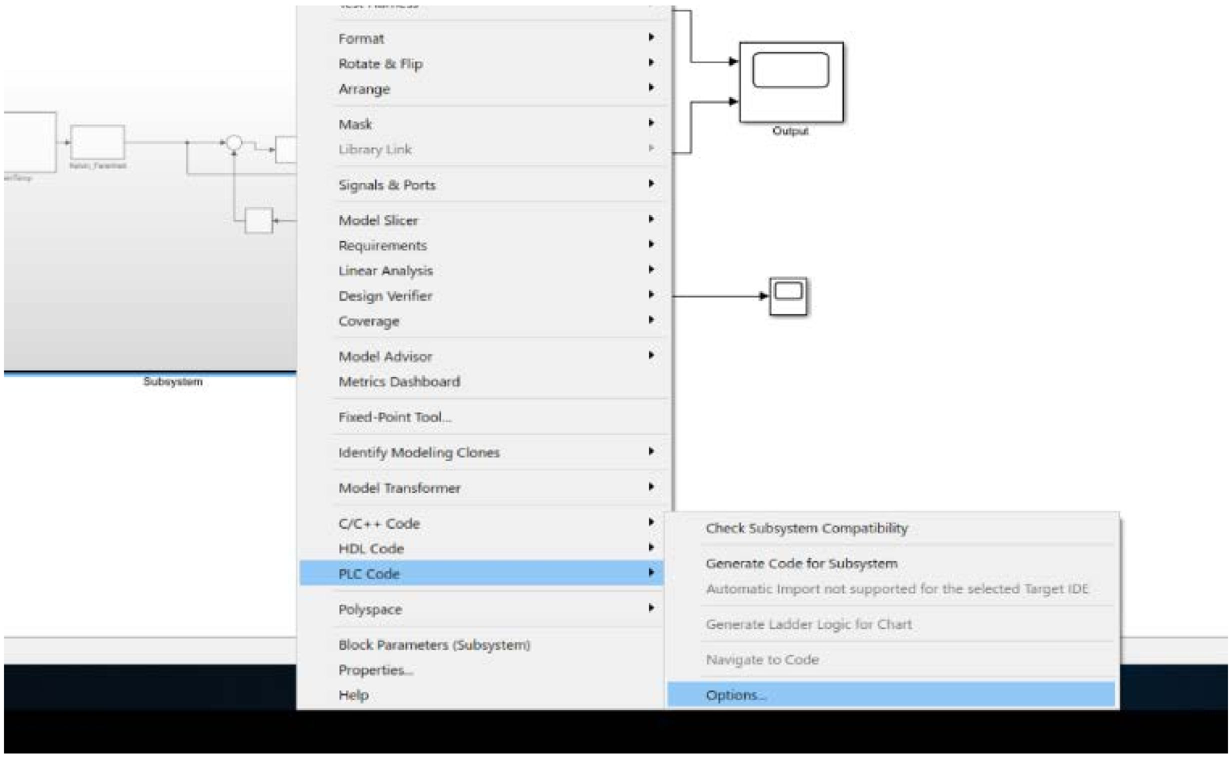


Figure 4. Check for system compatibility

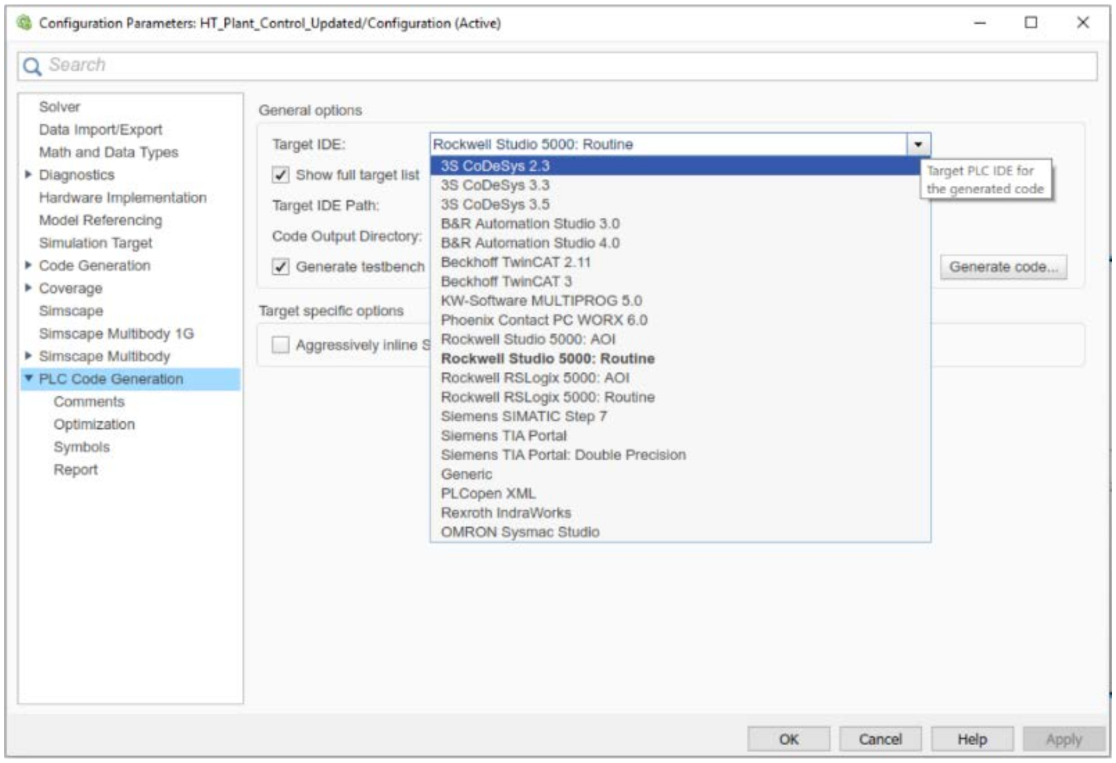
Next, the user will navigate back to the PLC Code selection and press the options tab. This will bring up a selection screen which allows a few actions to take place. A target IDE can be selected from a drop-down menu, shown in Figure 5. Some IDEs include are

- CoDeSys
- B&R Automation Studio 3.0 & 4.0
- Rockwell Studio & RSLogix 5000
- Siemens Simatic & TIA
- OMRON Sysmac Studio

The final step before generating the code is to decide whether or not to include a test bench with the code. The testbench is generated to verify the operation of the code once it is downloaded onto the PLC and can be a useful troubleshooting tool. The code generated from PLC Coder will be IEC 61131 structured text.

## 6. PLC Output

The final step of the conversion process is understanding the output from the PLC Coder and is shown in Figure 6. The code generated is IEC 61131 structured text, and in the example for this paper the code was generated to accommodate Rockwell Studio 5000. When the code is generated There are a few additions made to help the designer. First, all the input and outputs are turned into tags for the PLC, this allows the designer to quickly plug the program into existing systems as a subroutine. Next, the code is heavily commented to help the designer identify what different parts of the code are in relation to the Simulink model. Overall, the generated code is created to be quickly identified and implemented into a controls system.



Configuration Parameters: HT_Plant_Control_Updated/Configuration (Active)

Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Coverage
Simscape
Simscape Multibody 1G
Simscape Multibody
PLC Code Generation
Comments
Optimization
Symbols
Report

General options

Target IDE: Rockwell Studio 5000: Routine

✓ Show full target list

Target IDE Path:

Code Output Directory:

✓ Generate testbench

Target specific options

Aggressively inline S

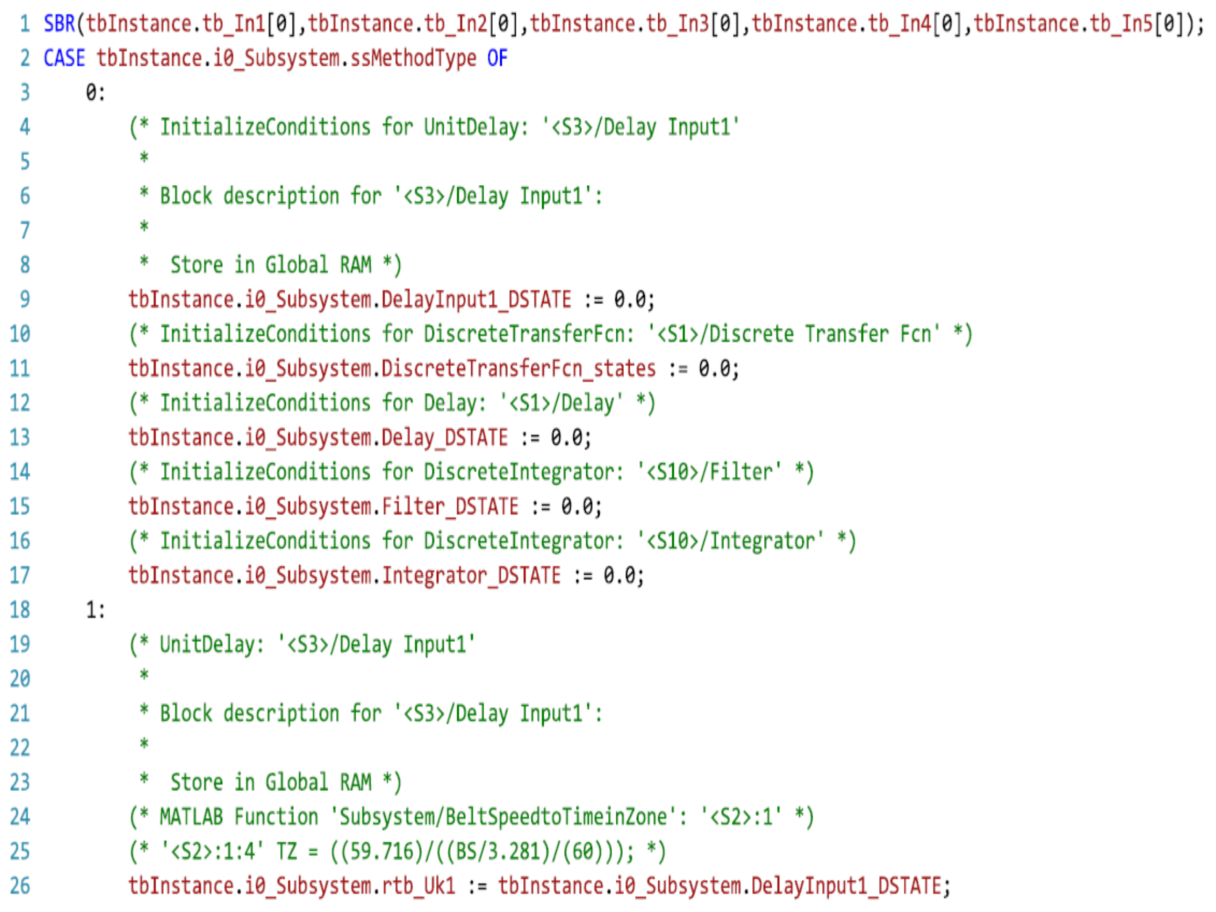
3S CoDeSys 2.3
3S CoDeSys 3.3
3S CoDeSys 3.5
B&R Automation Studio 3.0
B&R Automation Studio 4.0
Beckhoff TwinCAT 2.11
Beckhoff TwinCAT 3
KW-Software MULTIPROG 5.0
Phoenix Contact PC WORX 6.0
Rockwell Studio 5000: AOI
**Rockwell Studio 5000: Routine**
Rockwell RSLogix 5000: AOI
Rockwell RSLogix 5000: Routine
Siemens SIMATIC Step 7
Siemens TIA Portal
Siemens TIA Portal: Double Precision
Generic
PLCopen XML
Rexroth IndraWorks
OMRON Sysmac Studio

Target PLC IDE for the generated code

Figure 5. IDE selection & code generation

## 7. Conclusion

The PLC Coder extension for MathWorks Simulink will help revolutionize how industry designs, tests, and implements complex control systems. Using the steel furnace design as an example, using PLC Coder the system was designed, verified, and programmed in an afternoon without any expense other than the cost of the software. Simulink PLC Coder is an invaluable tool for designers creating PLC code for complex control systems and represents a cost-effective and time-saving approach to model-based design.



```
SBR(tbInstance.tb_In1[0],tbInstance.tb_In2[0],tbInstance.tb_In3[0],tbInstance.tb_In4[0],tbInstance.tb_In5[0]);
CASE tbInstance.i0_Subsystem.ssMethodType OF
    0:
        (* InitializeConditions for UnitDelay: '<S3>/Delay Input1'
         *
         * Block description for '<S3>/Delay Input1':
         *
         *  Store in Global RAM *)
        tbInstance.i0_Subsystem.DelayInput1_DSTATE := 0.0;
        (* InitializeConditions for DiscreteTransferFcn: '<S1>/Discrete Transfer Fcn' *)
        tbInstance.i0_Subsystem.DiscreteTransferFcn_states := 0.0;
        (* InitializeConditions for Delay: '<S1>/Delay' *)
        tbInstance.i0_Subsystem.Delay_DSTATE := 0.0;
        (* InitializeConditions for DiscreteIntegrator: '<S10>/Filter' *)
        tbInstance.i0_Subsystem.Filter_DSTATE := 0.0;
        (* InitializeConditions for DiscreteIntegrator: '<S10>/Integrator' *)
        tbInstance.i0_Subsystem.Integrator_DSTATE := 0.0;
    1:
        (* UnitDelay: '<S3>/Delay Input1'
         *
         * Block description for '<S3>/Delay Input1':
         *
         *  Store in Global RAM *)
        (* MATLAB Function 'Subsystem/BeltSpeedtoTimeinZone': '<S2>:1' *)
        (* '<S2>:1:4' TZ = ((59.716)/((BS/3.281)/(60))); *)
        tbInstance.i0_Subsystem.rtb_Uk1 := tbInstance.i0_Subsystem.DelayInput1_DSTATE;
```

Figure 6. Structured text

## 8. Future Work

The furnace design model represented in this paper was created in MATLAB version r2018a, which only supports structured text conversion using PLC Coder. With the recent release of MATLAB version r2019a, PLC Coder has been updated to generate ladder logic alongside structured text. This will pander to a wider audience whose experience is in designing control systems for PLC in ladder logic. On top of generating ladder logic for PLC, the new update will allow designers to import existing ladder logic programs directly to Simulink. This function will greatly change the cost of prototyping and testing code in industry. Since control systems can be verified in a simulated environment, it will greatly reduce cost and time to deliver for projects.

## References

[1] "Simulink PLC Coder," MATLAB & Simulink. [Online]. Available: https://www.mathworks.com/products/sl-plc-coder.html. [Accessed: 01-May-2019].

[2] "Festo Develops Innovative Robotic Arm Using Model-Based Design," MATLAB & Simulink. [Online]. Available: https://www.mathworks.com/company/user_stories/festo-develops-innovative-robotic-arm-using-model-based-design.html. [Accessed: 01-May-2019].

[3] "Vintecc Develops PLC System for Multi-Axle Harvesting Machine Using Model-Based Design," MATLAB & Simulink. [Online]. Available: https://www.mathworks.com/company/user_stories/vintecc-develops-plc-system-for-multi-axle-harvesting-machine-using-model-based-design.html. [Accessed: 01-May-2019].

[4] "ENGEL Speeds Development of Injection Molding Machine Controllers," MATLAB & Simulink. [Online]. Available: https://www.mathworks.com/company/user_stories/engel-speeds-development-of-injection-molding-machine-controllers.html. [Accessed: 01-May-2019].

[5] "Enabled Subsystem," Group blocks to create model hierarchy - Simulink. [Online]. Available: https://www.mathworks.com/help/simulink/slref/codereusesubsystem.html. [Accessed: 01-May-2019].

## Biographical Information

**THOMAS PRICA** is an undergraduate student in Electrical Engineering Technology, School of Technology at Michigan Tech

**SAWEYER POEL** is an undergraduate student in Electrical Engineering Technology, School of Technology at Michigan Tech

**Dr. ALEKSANDR SERGEYEV** is a professor of MERET program and a director of Mechatronics Graduate Program in the College of Computing at Michigan Tech. He has a strong record publishing in prestigious journals and conference proceedings such as measurements science and technology, adaptive optics, sensors and materials, the *Technology Interface International Journal*, ASEE, IEEE, and SPIE. Dr. Sergeyev is a PI and co-PI on several NSF and DOL awards and multiple significant industry awards.