DEVELOPMENT OF MODEL BASED DESIGN APPROACH: MODIFICATION OF DC MOTOR POSITION CONTROL

Donghyun Kim
Alan Levin Department of Mechanical and Nuclear Engineering
Kansas State University, Kansas 66506

ABSTRACT

Model Based Design (MBD) has been transforming how the engineers and scientists around the world develop, test, and implement their ideas into reality. Many companies and researchers have seen an increase in design and development efficiency all the while saving costs and development time [1]. As more and more companies are adopting MBD approach, it is critical to help our students understand fast growing technology. This paper explores into the basic methods of modifying the existing microcontroller-based DC motor position controller to be able to autogenerate source code for embedded deployment while minimizing time and human error introduced when manually coded. The resulting control system demonstrates the time it takes to deploy the position controller and its performance compared to the existing DC motor apparatus.

Keywords: Model Based Design, DC motor, embedded system, microcontroller, position control

1. INTRODUCTION

The Department of Mechanical and Nuclear Engineering's curriculum at Kansas State University offers an introductory control of mechanical systems course. ME570, along with its laboratory section, focuses on the basic modeling and analysis of feedback control utilizing DC motor controller [2]. The DC motor controller is based on the STM32 microcontroller and has been used by numerous students who graduated from the school. This is made possible since the system is easy to use based on the visual component of Graphical User Interface (GUI) developed specifically for the current apparatus.

One of the shortcomings of utilizing the easy to use system, as observed by many instructors who taught the course, is the pattern of lack of motivation in working with the electronics side of the control theory and interpretation of diagrams. Because the device is setup in a such way that masks various electronic components from students, many were able to follow instructions and complete the course without understanding how the mechanical control system is implemented in embedded C/C++ code. To mitigate the issue and to increase the student's involvement and understanding of electromechanical system at the minimum level, interactive diagram building via the MBD approach could potentially attract more eyes and attention during the student's lab session or homework while reinforcing the knowledge of interpreting diagrams.

Development for the MBD approach of the current control system is motivated by the efficiency in time it takes for students to develop, test, and deploy the entire DC motor apparatus for position control application due to its automated process. The less time spent compiling and debugging manually written code means more time working with diagrams and playing with the device. This time efficiency in the MBD process arises from the reduction of the complexity inherent to coding with C/C++ especially for students with no coding background. The course already exposes students in the idea of how to interpret block diagrams used to describe the mechanical system which then can be applied directly to build a basic model for position control of the DC motor. Some students might have trouble understanding how the block diagrams capture the physical system, but this can also be further resolved by continuously working with block diagrams in the MBD. The MBD can eventually lead students to understanding how the source code is developed and deployed without going into depth of compilers and other C/C++ related knowledge. Writing every source code for driving the peripherals can be too much learning curve in an introductory course. Incorporated along with the introduction to basics of compilers and C/C++ language, the MBD can help students to be prepared to be well rounded controls engineer in dealing with both theoretical and practical use of the control system.

The MBD implementation proposed in this paper is made possible with STMicroelectronics' software development tools and MathWorks products, MATLAB and Simulink, coupled with the DC motor apparatus. Utilization of pre-existing program within the course can eliminate the process which involves students learning a new program along with added benefits in saving cost associated with developing a new device. Students understanding the basic syntax of MATLAB and the basic knowledge of working with Simulink diagrams can pick up the MBD and develop, deploy, and test the control system within a single lab exercise.

And for the last part of the paper, limitations at the current state of the development is identified and the list of future works to be completed to investigate the further capabilities of the apparatus will be included.

2. MATERIALS AND METHODS

This section will cover the theoretical model, drivers for each component used in the motor lab apparatus, and the programs used in development of the MBD. All constants and dynamic system characteristics specific to the motor are listed in Table 1 [3].

	Variable	Units	Value
Manufacturer			Shinano Kenshi
Rated Power		W	40
Rated Voltage		VDC	24
Rated Speed		RPM	3,000
Rated Torque		N-cm	12.7
Rated Current		A	2.5
Torque Constant	k_t	N-cm/A	5.0
Back EMF Constant	k_b	V/kRPM	5.2
Phases Resistant	R	Ohm	1.18
Phase Inductance	L	mH	4.4
Instantaneous Peak Torque		N-cm	38.2
Max Speed		RPM	5,000
Rotor Inertia	J	g-cm²	110
Power Rate		kW/s	1.48
Mechanical Time Constant		ms	5.2
Electrical Time Constant		ms	3.7
Mass		kg	0.6
Friction Coefficient	b_{est}	N-m-s/rad	3e-5

TABLE 1: MOTOR DYNAMIC SPECS

2.1 Simulink Model

The course, ME 570, first introduces Simulink modeling of position control in Lab 5. The basic closed loop block diagram and its relation to the electro-mechanical system is shown below in Fig. 1 [4]. All constants not specified can be found in Table 1 above. The position control of the device uses unity feedback and 'fast' amplifier.

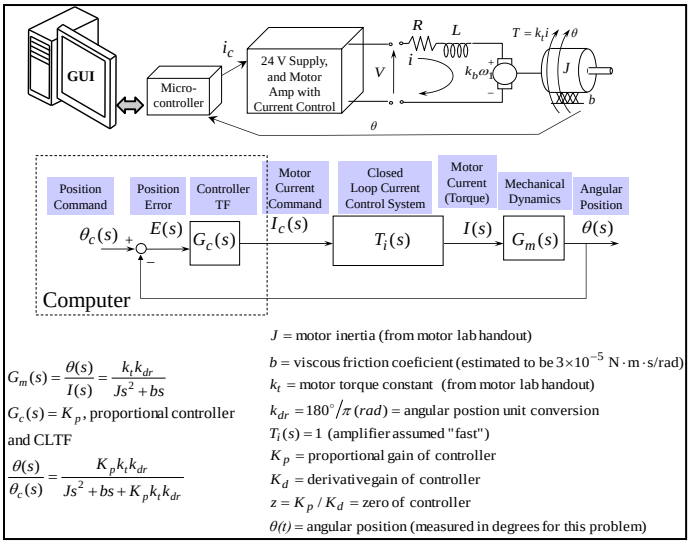

FIGURE 1: BLOCK DIAGRAM OF POSITION CONTROLLER

The lab requires students in understanding the theoretical development of the closed loop system of position controller in Simulink environment. From the Fig. 1, students are then asked to complete a Simulink diagram to simulate the position control by inserting missing blocks such as the saturation block to limit the current feeding into the controller. The completed Simulink diagram is shown below in Fig. 2. As part of the lab, students must learn how to interpret the diagram in order to place the saturation block to specifically limit the current. This exercise can help students realize the order of the blocks is important as well as how the signal propagates within the diagram.

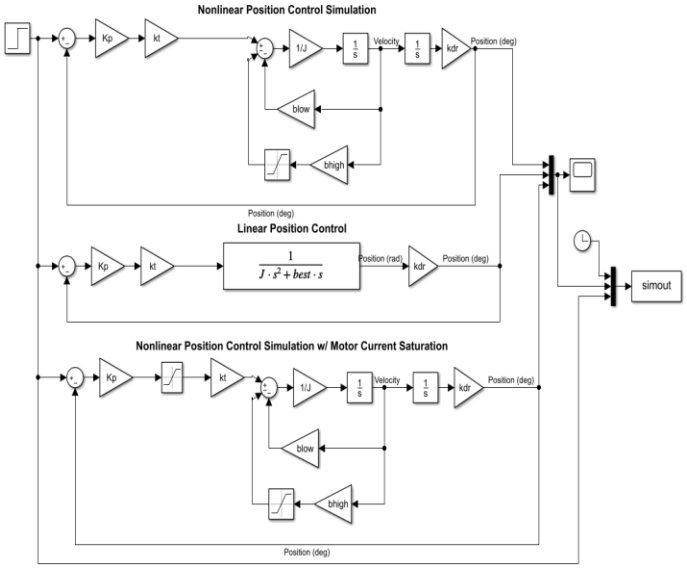

FIGURE 2: SIMULINK DIAGRAM FROM LAB 5 OF ME570

Prior to transition from the lab 5 exercise above into the MBD exercise that can be deployed to and debugged with the microcontroller, the host PC must be installed with the following programs first: STM32-MAT/TARGET, STM32CubeIDE, and STM32CubeMX from STMicroelectronics, and MathWorks' additional products such as Simulink Coder and MATLAB Coder. STM32-MAT/TARGET is used within Simulink to compile and export the diagram to various Integrated Development Environment (IDE) projects. The program also provides end users with various blocks that can be utilized to configure the settings of the target microcontroller board and to access the peripherals in use. In the process of configuring the target board within the Simulink diagram, STM32CubeMX program is launched to assist in selecting the microcontroller and its available peripherals. The program will then autogenerate the base project code along with Hardware Abstract Layer (HAL) library to control the peripherals. STM32CubeMX program was developed with one of the same benefits of the MBD in mind by STMicroelectronics. It is a graphical tool for configuring and generating initialization C code for the processor of the target board to minimize human errors from manual coding. The program is shown in Fig. 3 below.

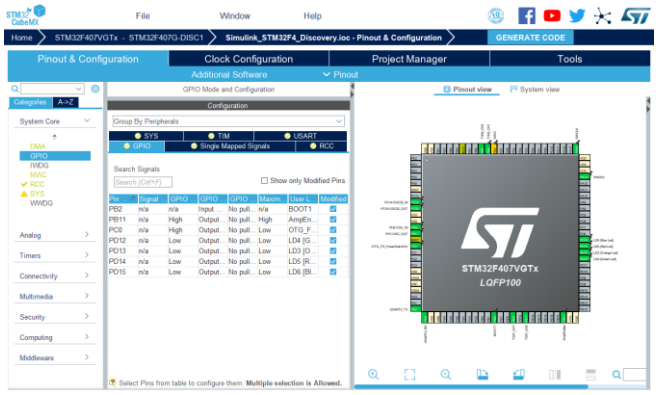

FIGURE 3: STM32CubeMX GUI

When the Simulink diagram autogenerates and exports the project code, any supported IDE's can be used to compile and deploy to the target board for debugging purposes. In this paper, STM32CubeIDE is used simply because it is based on Eclipse and is free to use.

2.2 Simulink Blocks

As captured in Fig. 4, the appearance of the MBD of position control is as simple as the block diagram shown in Fig. 1. Achievement of simplicity in appearance can be a great deal to many students as in not being discouraged when a complex diagram is presented during the lab. Simplicity of the diagram is made possible with the peripheral driver blocks working under the hood to autogenerate project code. The peripheral driver blocks, named DigitalOut and UART2_TX, were developed to control the amplifier and to transmit the encoder signal back to the PC using UART. Translation between the driver code and the visual blocks can be done utilizing Simulink's built in MATLAB System block [5]. The structure of the system block first defines the number of inputs and outputs, any associated header files to be called, and manipulating the signal by calling peripheral specific C functions. For example, UART2_TX system block will take the encoder signal as an input and transmit the signal in human readable ASCII format to the serial monitor on the host PC. Any serial monitor programs such as Arduino and PuTTY can be used for reading and writing the serial data.

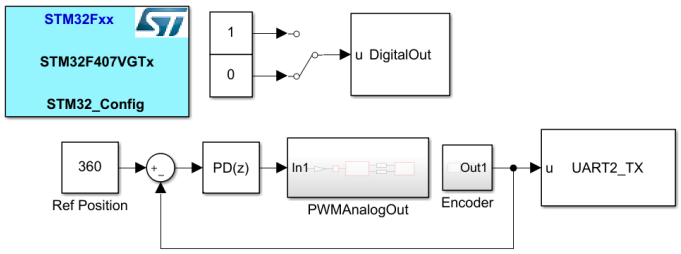

FIGURE 4: SIMULINK MBD OF POSITION CONTROL

In comparison to the knowledge of C/C++ required to drive the peripherals on the target board with manually written source files, the students can simply automate the process and save significant amount of time and human errors by utilizing the premade driver blocks.

Inside PWMAnalogOut and Encoder blocks are the driver blocks provided from STM32-MAT/TARGET along with the built in Simulink blocks as shown in Fig. 5 and 6. The blocks named TIM1 and TIM3 are the peripheral driver blocks from STMicroelectronics. TIM3 block is associated with generating Pulse Width Modulation (PWM) to drive the motor. The output signal from Proportional-Derivative (PD) controller block in Fig. 4 is used to determine the direction and value of duty cycle to input to TIM3 block. The block named dutyCycle will take the saturated signal and convert into percentage output to feed into TIM3 PWM block.

FIGURE 5: INSIDE THE PWMAnalogOut BLOCK

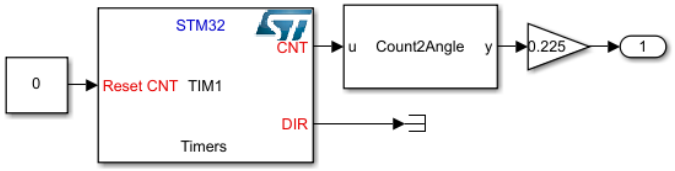

FIGURE 6: INSIDE THE Encoder BLOCK

TIM1 block from Fig. 6 above is essentially the encoder block. It keeps track of the direction and counts from the motor encoder to convert it into the angular position in degrees. Count2Angle block in conjunction with the gain block are used for unit conversion according to the encoder's hardware documentation. Encoder block does not have an input port as it is the feedback sensor. The output port is connected to UART2_TX block to send the data back to the host PC for the verification of data integrity. Before building the model within IDE, make sure to set the linker flag "-u_printf_float" in order to pass on float values in UART2_TX block. TIM1 and TIM3 blocks needs to be configured according to the board manufacturer's data sheet and user manual to conform to the available hardware clock speed and other necessary register values.

2.3 Model Deployment to Microcontroller

Once the Simulink diagram is built, STM32_Config block will compile and link all autogenerated code to export as a project specific filetype to preselected IDE. An IDE then can be executed in order to build and deploy the project to the target microcontroller board.

At this stage of the MBD, the board is ready for debugging using an IDE which is the most useful tool in identifying any bugs within the system. In the process of debugging, developers of source driver blocks can step through the execution of the board to recognize and assure the correct behavior of any custom blocks driving the peripherals. When the debugging procedure is

completed the blocks are ready to be used in the MBD. One of the benefits of custom blocks developed and tested to be used in the MBD is the reusability. When a custom block is tested to drive the peripherals as required, it can be then used in any of the MBD projects which requires to drive the same peripherals. As an example, UART2_TX block can be used to create a communication device using any STM32F4 Discovery boards.

The entire process of compiling and exporting C/C++ projects from the Simulink diagram to deploying to the target board takes less than two minutes depending on the host PC's hardware specification, mainly the Core Processing Unit (CPU) and Random-Access Memory (RAM). With the current machine in use, equipped with four CPU cores at 2.60GHz and 16GB of RAM, Simulink takes less than a minute to export and STM32CubeIDE takes roughly seven seconds to build as shown in Fig. 7 and Fig. 8 respectively.

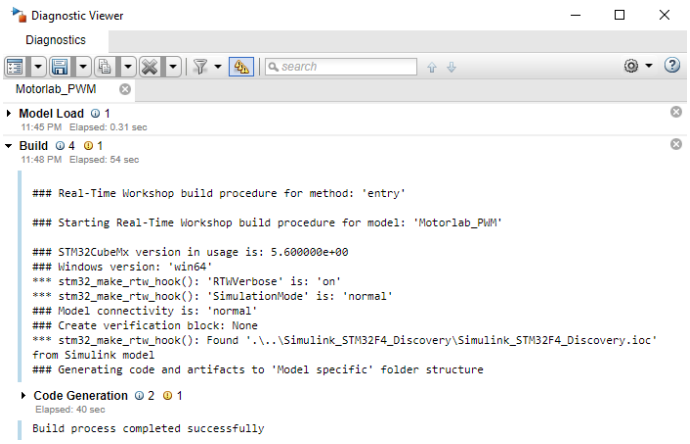

FIGURE 7: BUILD PROCESSING TIME

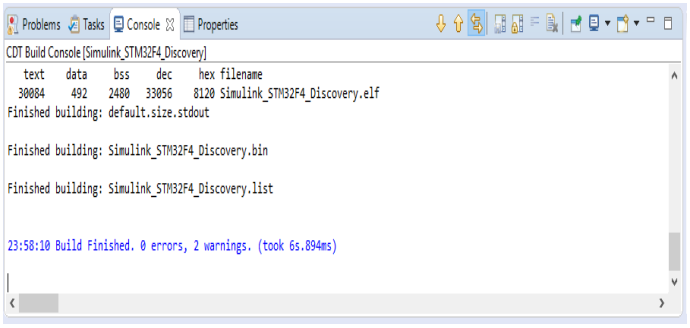

FIGURE 8: IDE'S PROJECT BUILD TIME

The output from the position controller built using the MBD was verified to be sampling the encoder data and running the control loop at 10kHz. The plot of the output data using a MATLAB script is shown in Fig. 9. The initial plan for the result section was to include the current device's position control output as well in order to compare the two results but due to the fact that the lab was inaccessible, this will be included in the next paper.

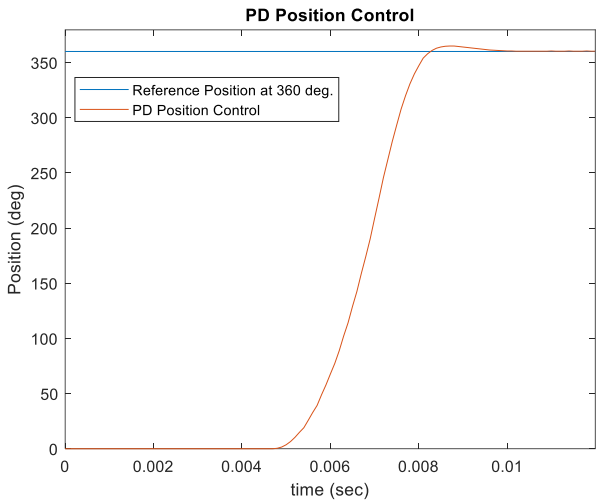

FIGURE 9: STEP RESPONSE OF PD CONTROLLER

3. CONCLUSION

This paper has shown the implementation of the MBD to an existing DC motor control apparatus for executing a simple position control algorithm. At the current state of the development for the MBD implementation, the device is only limited to position control with no communication protocol established. This results in incapability of modifying parameters such as the target reference input position or controller gain values. When the Simulink diagram builds and exports the project code to an IDE, everything is hardcoded into the target board and it only transmits positional data back to the host PC. With the current limitation of the board's functionality, further development in the driver blocks are required in order to create more user-friendly interface for the students to use in the lab environment. Main custom driver blocks to be developed are the following: UART2_RX and Serial_RX.

UART2_RX block to receive all parameter changes such as the controller gain values and the target reference position. This block will allow the target board to act as a transceiver mimicking the current device used in the lab.

Serial_RX block to read and import the position data into the workspace using a new Simulink diagram. This diagram will be separate from the target board's diagram and will explore the possibility of enabling the real-time data analysis such as the plotting incoming data from the device. If the overhead of plotting function within Simulink is minimal, students could plot the data and observe the incoming response data in real-time.

ACKNOWLEDGEMENTS

The modification of DC motor apparatus is made possible by taking advantage of Dr. Schinstock's source code for the original motor lab device.

REFERENCES

[1] “Adopting Model-Based Design”, https://www.mathworks.com/campaigns/portals/adopting-model-based-design.html.
[2] “ME 570 – Control of Mechanical Systems I”, https://catalog.k-state.edu/preview_course_nopop.php?catoid=16&coid=82554.
[3] “Motorolab”, https://k-state.instructure.com/courses/78873/files/folder/Info%20For%20First%20Day?preview=9837536.
[4] “Laboratory #5”, https://k-state.instructure.com/courses/86477/files/folder/Labs/Lab5?preview=12788796.
[5] “MATLAB System Block”, https://www.mathworks.com/help/simulink/ug/what-is-matlab-system-block.html.