2020 7th International Conference on Control, Decision and
Information Technologies (CoDIT'20) | Prague, Czech Republic / June 29 - July 2, 2020

Model-Based Design Streamlines for STM32 Motor Control Embedded Software System*

Loubna BELHAMEL[1], Arturo BUSCARINO[2], Antonio CUCUCCIO[3], Luigi FORTUNA[2], Gaetano RASCONA'[3]

***Abstract*— This paper presents a Model-Based Design (MBD) methodology as a promising approach for the rapid and secure development of Embedded System applications, including those involving digital controllers. Based on existing hardware and software application-oriented tools by STMicroelectronics, a new modeling technique has been implemented to move from a traditional design workflow to an MBD one by using Mathworks® software platform. For a practical application of the proposed approach, a case study is reported to design a Permanent Magnet Synchronous Motor (PMSM) drive based on the STM32 MCU family. The main contribution of this paper is to make focus on concretely building a unique model architecture of a complex software/hardware system able to implement all the major aspects of MBD methodology such as executable requirements, Normal and Processor-In-the-Loop (PIL) simulation modes, continuous test and verification, automatic code generation. To achieve this purpose, a new Simulink® blockset dedicated to the STM32 Motor Control ecosystem was developed. It includes Simulink blocks for maths, algorithms, and IPs, electronic circuitries, MCU peripherals, speed sensors. Simulink® Embedded Coder tool has been used to automatically generate code for a specific STM32 MCU target to overcome the time consuming and error-prone problems of the handwritten coding.**

I. INTRODUCTION

Many industries need to reduce time-consume and errors when they start the development of a new product. Working efficiently is indispensable to success in a globalized market, especially for high-tech industries such as automotive, aerospace, and communications. Embedded systems and electronic controls are vital parts of each new modern product [1].

Model-Based Design (MBD) methodologies have been introduced for time-saving, cost-effective, design reuse [2] [3]. These methods and tools may permit to obtain the structural response under relevant loads and improve the structural capacity to withstand extreme loads by a specific design solution.

*This work was supported by STMicroelectronics organization

[1]L. Belhamel is with the department of Electrical, Electronic and Computer Engineering, University of Catania, ,95125 Catania, Italy loubna.belhamel@unict.it

[2]L. Fortuna and A. Buscarino are with the department of Electrical, Electronic and Computer Engineering, University of Catania, 95125 Catania, Italy and with CNR-IASI, Italian National Research Council-Institute for Systems Analysis and Computer Science, A.Ruberti, Rome,Italy. luigi.fortuna@unict.it ,arturo.buscarino@unict.it

[3]A. Cucuccio and G. Rascona' are with the STMicroelectronics, Catania site, 95121 Catania, Italy. antonio.cucuccio@st.com ,gaetano.rascona@st.com

The MBD approach is widely considered to be the most important design flow method as it allows to transform the classical design methods from the lab and handwritten to the desktop [4]. It provides an efficient methodology that includes six key steps in the system development process [5]:

- Modeling: system modeling activities involve creating a mathematical and behavioral representation of the system under consideration.
- Simulation: a method for implementing a model and behaviors in executable software.
- Rapid prototyping: provides a fast and cost-effective way for control and signal processing engineers to verify designs at an early stage and evaluate design trade-offs.
- Embedded deployment: a software process that converts the controller model to detailed executable software code.
- In the loop testing: combining target hardware and production code into model-based testing, one can compare dynamic outputs of models with data collected through software-in-the-loop and processor-in-the-loop test or with data measured in the test lab, using data inspectors or logging tools.
- Integral activities: most of the MBD environments automate the generation of documentation from models. In this case, the documentation is in template form allowing users to specify the content of each documentation section.

Simulink from Mathworks is a block diagram environment for a multi-domain simulation that can be used for Model-Based Design [5] that are suited for embedded systems. Beyond Simulink, the required toolboxes for the presented use case example are Simscape Electrical, Stateflow, Embedded Coder.

In this paper, the advances in platforms and tools are examined allowing the system developers to have a unique architecture both for simulation and for code generation model. As a real application example of an embedded system, a complex motor control drive based on the STM32 MCU is presented with a dedicated Simulink model. The main contributions of this paper will regard the following points:

- A new Simulink toolbox that allows integrating the MBD approach to the existing STM32 ecosystem ded-

978-1-7281-5953-9/20/$31.00 ©2020 IEEE

icated to advanced motor control has been developed. Its blocks provide models of many system components such as hardware devices, extensive Math and Motor control functions, embedded peripherals that are used for both function emulations and code generation. Each block of Math and MC function library has been derived from the STM32 MC library legacy code. In this way, the simulation precision replicates, at the bit level, the calculations performed by the target MCU;

- a FOC motor control system model with an encoder sensor allowing to perform normal mode simulation, PIL mode simulation, and automatic code generation for STM32F3 MCU has been carried out. In particular, MBD methodologies have been used to move from the traditional workflow design to the modeling in the Simulink platform. The provided model can deal with the same drive parameters used in the previous workflow.

The paper is organized as follows. In Section II a brief overview of MBD methodology definitions and which aspects of product development cycles will be discussed. In Section III the hardware and software for modeling the PMSM motor drive are presented. In Section IV the implementation of the MBD approach for the STM32 motor drive ecosystem will be examined. In Section V the simulation and code generation results are presented. Conclusions are made in Section VI.

II. MODEL BASED DESIGN METHODOLOGY

In this section, the basics of the model-based design approach are presented.

A. MBD workflow

MBD modifies traditional methods adopted in model development process and introduces a better way to implement the workflow as shown in Fig. 1. For each stage of the

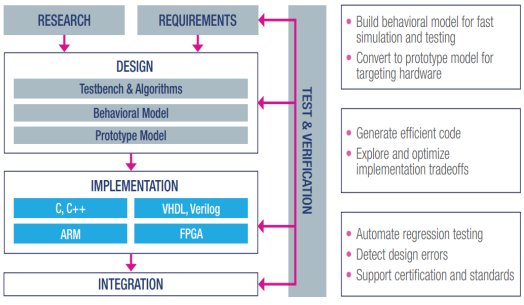

Fig. 1. MBD workflow

development cycle, MBD introduces several benefits:

- the requirements stage allows to early validation and test development, clear specification, simulate whole system including environment, tight link to requirements;
- the design stage permits to fast design exploration, design optimization, find flaws before implementation;
- the implementation stage allows to eliminate hand-coding, eliminate hand-code errors, promote hardware target portability and better testability;
- the test and the verification stage help to detect errors earlier, reduce the use of physical prototypes, reuse tests throughout development stages.

B. V-Model

As depicted in Fig. 2, the MBD supports the V-model development cycle that is illustrated in the following steps:

- executable specifications;
- design with simulation;
- continuous test and verification;
- automatic code generation.

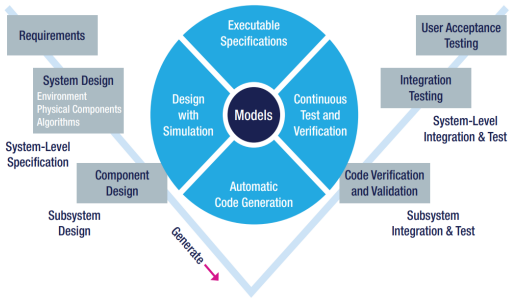

Fig. 2. V-model with MBD

Based on the steps presented in this section, a model in Simulink will be developed to implement the MBD workflow.

III. MODELING THE PMSM MOTOR DRIVE SYSTEM

In this section, the STM32 ecosystem both as regard the hardware and the software is described. The target drive system is a PMSM motor with encoder position feedback connected to a 3-phase ac power inverter with three shunt resistors for phase current feedbacks. The drive control algorithm is implemented on a STM32 MCU that includes peripherals dedicated to advanced motor control techniques as shown in Fig. 3.

The drive system hardware consists of an X-Nucleo-IHM07M1 power board, a NucleoF302R8 control board, and a 3-PH PMSM motor with encoder feedback. The control board is based on a MCU of the STM32F302x6/8 family. These MCUs are perfectly suited for Motor Control application as they feature a fast 12-bit ADC (5 Msps), three comparators, one operational amplifier, one advanced-timer dedicated to motor control. The STM32F302 MCU executes the drive control algorithm [6] based on the commonly used Field Oriented Control (FOC), whose basic scheme is shown in Fig. 4.

Motor currents and rotor position feedback signal acquisitions are synchronized to the PWM switching frequency using the processor interrupt mechanism. When a new acquisition is completed, the FOC control algorithm executes.

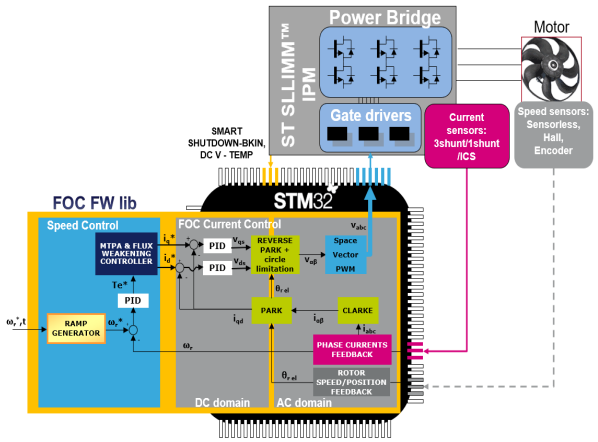

Fig. 3. Drive system scheme

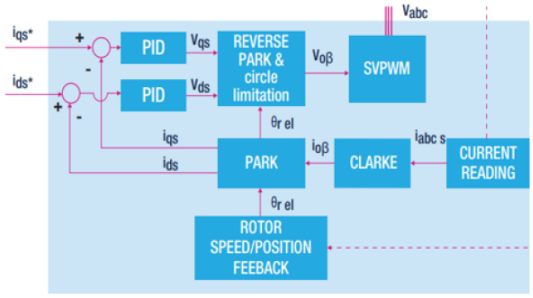

Fig. 4. FOC scheme

In FOC algorithm, sensed motor currents are math transformation performed by Clarke and Park blocks to get two time-invariant components i_{ds} and i_{qs} respectively in phase and quadrature with rotor flux. In this way, it is possible to offer electromagnetic torque T_e regulation by acting on component i_{qs} and using a PID regulator and, to some extent, to operate a flux weakening, by acting on component i_{ds} and using a second PID. For modeling purposes, the system has three main components: the power inverter and motor (plant), the device interfaces for the control and feedback signals and the digital control unit. The plant model uses Simulink Simscape components to simulate the power inverter electrical circuit and the motor electromechanical elements in the continuous-time domain. The feedback circuit models take care of the gain and data type conversions between the controller and motor drive models. The models for the embedded signal interfaces replicate the MCU peripherals by function emulation. They include conversion functions because the ADC converter, the MC dedicated PWM Timer, the encoder timer have 16-bit or 32-bit fixed-point output data registers.

The motor drive system performs several functions beyond the ones dedicated to the motor control algorithm. The embedded software architecture is modular to match both usual demanding requirements for platform flexibility and to make easier the development. These modules typically are devoted to system initialization, communications interface, application tasks, motor control interface, and motor control algorithm as illustrated in Fig. 5.

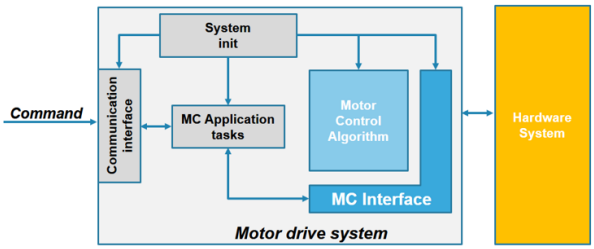

Fig. 5. Motor drive system tasks

The MC interface module manages the signal data flow between the motor drive hardware and the control algorithm. The code is specific to the drive circuit connections and the MC peripheral configurations needed to provide the appropriate feedback signals for the control algorithm [7].

The system model is partitioned into the logical blocks [8] shown in Table I. Each block is divided into sub-blocks.

TABLE I
MODEL PARTITIONING

Block	Modeling/Code Generation
Electromechanical System	Inverter/Motor/Mech-System
Sensors and Interfaces	Function peripherals models
Processor	Peripherals/Code algorithm
Driver Circuits	Function models

In the next section, two dedicated Simulink toolboxes, respectively dedicated to Motor Control Algorithm and STM32 peripheral driver will be described and used to implement MBD workflow.

IV. IMPLEMENTATION DETAILS

A PMSM Motor drive model has been realized to show the application of MBD concepts to design an embedded control system. This model is used for both drive simulation and code generation for target device. In order to exploit all the potentialities offered by MBD approach, the designer should be given the chance to continue using existing application-dedicated tools offered by ST when start using new tools of MATLAB/Simulink environment. For this reason, the ST Motor Control Workbench (STMCWB) tool is used to generate a system configuration file (*.ioc* file) and it is passed as an input to the Simulink environment through *MC_Config* block to set model parameters.

Maths and IPs blocks dedicated to FOC algorithm has been strictly built on STM32 Motor Control library and they allow accurate simulations (in normal, SIL and PIL modes) replicating the same behavior of the application running in the real STM32 hardware system. The application designer can configure all the system parameters using the most appropriate method, conceive his project completely in the desktop, exploit all the powerful toolboxes available from Mathworks to test and validate the results, analyze waveforms, and so on. Finally, numerical approximations

generally introduced by microcontroller-based calculations (due to fixed-point mathematics, for instance) are also taken in consideration at this stage.

Once the simulation results match the project requirements, the second important aspect of MBD methodology is the automatic code generation for target device. By leveraging on STM32 Motor Control SDK and the Embedded Coder tool from Simulink, the application model will generate embedded code (in C or C++ language) in order to create the Application Software layer represented by *MC_App* in Fig. 6. The software project can be organized into several C modules for code readability and easy debugging.

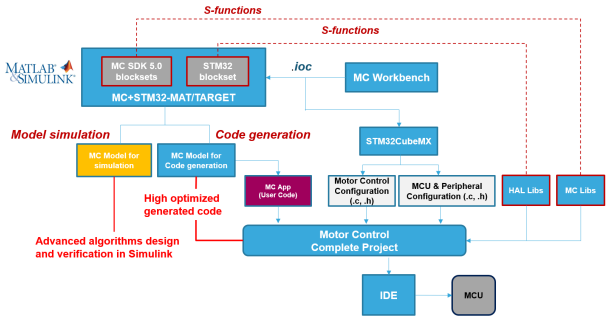

Fig. 6. Modeling and code generation scheme

A detailed explanation of the STM32 MC toolbox is given below.

A. Simulink toolbox for STM32 motor control

A Simulink toolbox, STM32-MAT/MC, has been developed for providing a set of motor control libraries that supports maths, algorithms, interface devices, hardware IPs for FOC motor control design as seen in Fig. 7.

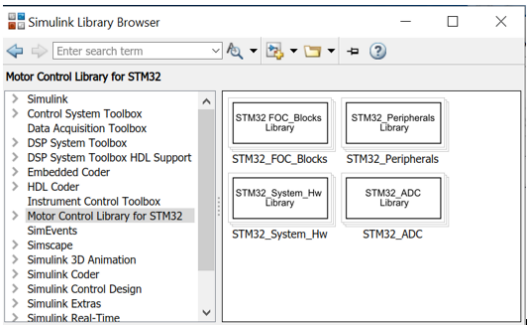

Fig. 7. Motor control library for STM32

- STM32 FOC Blocks: It provides Simulink blocks for mathematical transformations and a PID controller normally used to design FOC Motor Control algorithm. Each block is an S-Function and it has been coded in C-MEX language by using same function prototypes of C libraries of STM32 Motor Control SDK [7]. A block target file (TLC) has been developed for each block to inline the S-function and to perform an automatic code generation. The Fig. 8 shows the mathematical blocks currently available in the *STM32_FOC_blocks* library:

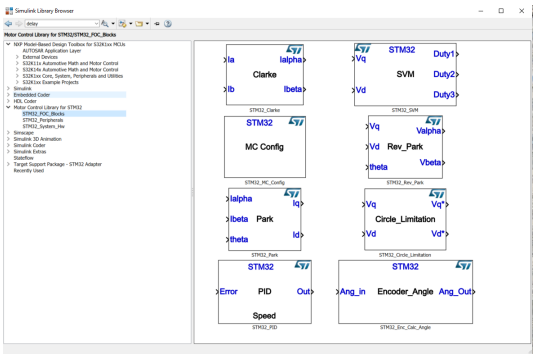

Fig. 8. STM32 FOC blocks library

- STM32 Peripherals: The *STM32_Peripherals_blocks* library as seen in Fig. 9 provides three blocks replicating basic operations of STM32 peripherals: an Advanced PWM Timer, an ADC converter and an Encoder Timer. The functional peripherals were implemented in order to behave, in simulation, like the real ones do. The Timer block emulates the operations of an STM32 Advanced PWM timer by providing three complementary PWM outputs at a specified frequency. The ADC block emulates the operations of an STM32 ADC 12-bit converter which converts input signals under external trigger event occurrence; when a conversion cycle is completed, an external event is generated to run the FOC algorithm. The Encoder timer emulates the operations of an STM32 Timer configured in Encoder mode.

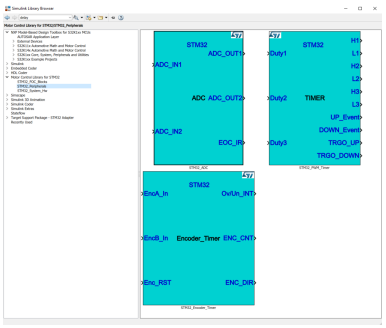

Fig. 9. STM32 peripherals library

- STM32 System Hw: The *STM32_System_HW* library as seen in Fig. 10 provides two basic blocks to model a 3-PH inverter with shunt resistors and an incremental encoder. They can be considered just as off-the-shelf utility blocks to allow the MC designer to use the same hardware devices found in the majority of the motor control kit available from ST.

B. Model configuration parameters

STM32 STMCWB software GUI contains several sections to configure motor drive operations and parameters: PMSM

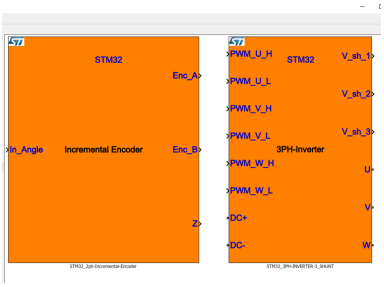

Fig. 10. STM32 HW library

Motor, Power Stage, Drive Management and Control stage just to cite some. Outputs from this tool lead to H files generation that can be included in the MC project. To import configuration parameters set by means STMCWB, a *MC_Config* block is available from STM32-MAT/MC toolbox; it allows configuring each block of model for simulation in a rapid and seamless manner.

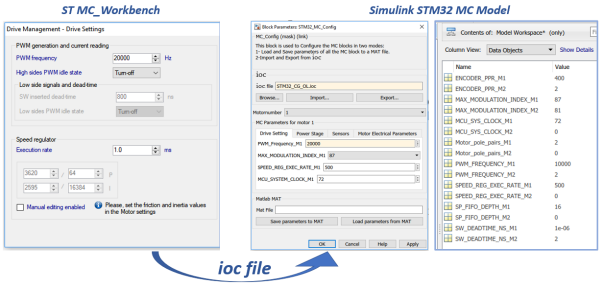

Fig. 11. Motor control workbench vs *MC_Config*

C. FOC engine algorithm

The FOC control scheme is shown in Fig. 12. It consists of blocks from the *STM32_FOC_blocks* library. Each block contains specific parameters that can be set automatically by referring to *MC_Config* block and by specifying the motor drive ID (integer number).

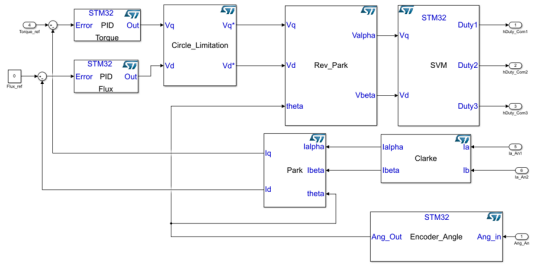

Fig. 12. FOC engine

D. Model Architecture

The model architecture presented in Fig. 13 follows the model partitioning scheme described in Table I:

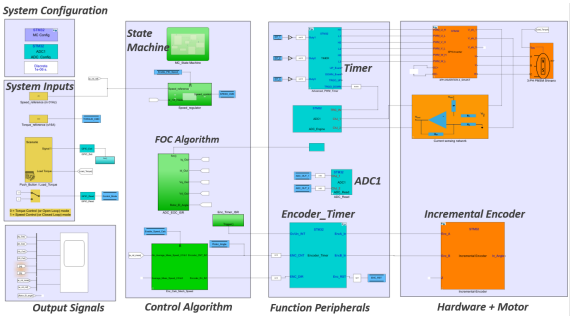

Fig. 13. Model architecture

This architecture has been conceived to easily move from normal mode simulation, entirely executed on the host PC, to PIL mode simulation, where math's and IPs blocks (and subsystem) are executed in the target STM32 MCU. By using a specific configuration block, this model can be used for both normal simulation and automatic code generation. The only difference between the two configurations consists of the replacement of functional peripheral blocks into the correspondent ones from STM32-MAT/TARGET [9] peripheral driver library.

E. Code generation

As indicated in Fig. 14, FOC C code is generated from the motor control algorithm part of the model. The settings for the code generation are in the Simulink Configuration parameters → code generation → System Target File, where the target file *stm32.tlc* must be specified. In STM32 Options, the paths for STM32CubeMX and STM32-MAT/TARGET must be specified as well.

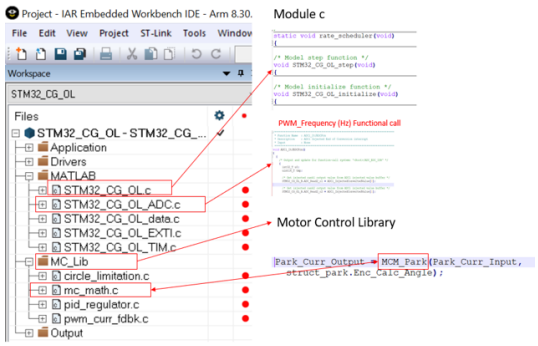

Fig. 14. Code module organization

The code is organized in a modular fashion that makes integration of application-specific functions easier. High priority tasks, such as the FOC motor control algorithm, are executed in the Interrupt Service Routines at peripheral event occurences. Application-level tasks are called, as scheduled tasks, from a basic scheduler kernel. Simulink Embedded Coder and STM32-MAT/TARGET toolboxes take care of all these aspects automatically.

V. SYSTEM RESULTS

A. Simulation results

By simulating the model shown in Fig. 15, relevant signal waveforms can be viewed in the scope Simulink block. For example, Torque Mode was chosen. The scope screen in Fig. 14 has been divided into three subscreens: the first one is the i_q quadrature current, the second one is the motor phase current and the third one represents the three inverter duty cycles computed by SVPWM block. The shapes of waveforms show that the motor runs well.

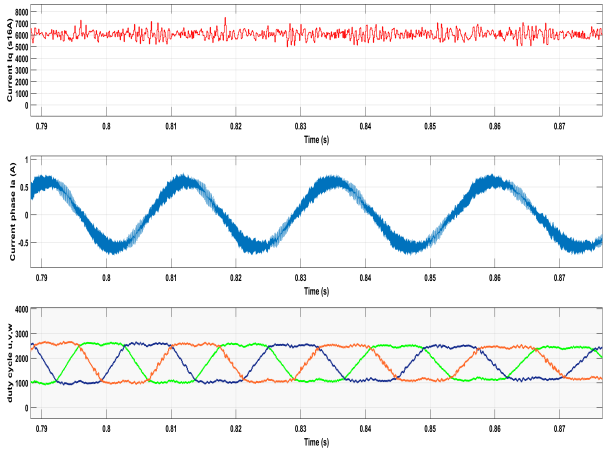

Fig. 15. Simulation results

B. Code generation results

The experimental set-up consists of interfacing the STM32-Nucleo board (connected to the host PC via USB cable) with the inverter board and the power supply and the PMSM motor as shown in section 3. After the project generation from Simulink Embedded Coder, the binary application code is built in the user IDE and downloaded in the STM32F302R8 MCU. Then, the DC link is supplied with 24V DC. The Fig. 16 shows the system. By pushing

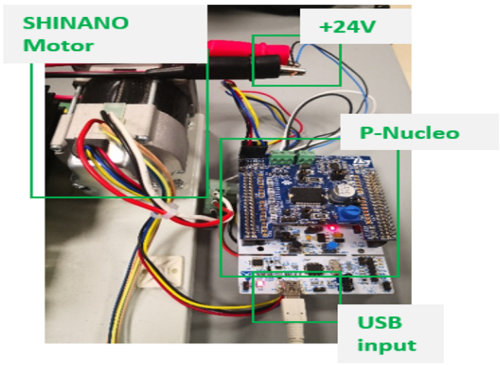

Fig. 16. The hardware system

the user button on the control board the motor shaft starts spinning. Two channels of an oscilloscope were connected to the Nucleo CN10 connector pins to read two PWM outputs. A third scope channel was connected to read the motor phase current. The Fig. 17 shows a screenshot from the actual oscilloscope: as can be easily seen, the motor waveforms

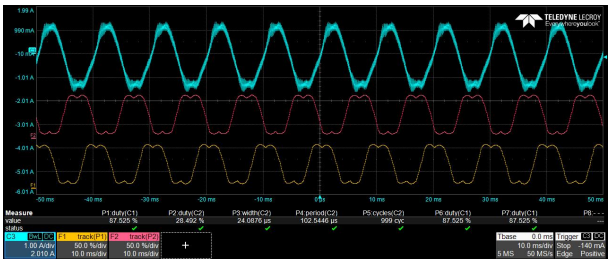

Fig. 17. The motor wave-forms, the motor phase current (cyan), the duty cycles (red and yellow)

are quite similar to the ones got in simulation. Absolute values for phase current and PWMs are different because torque load conditions, applied to the motor shaft, used in simulation model and in the real system were different.

VI. CONCLUSIONS

In this paper, a new Simulink toolbox, STM32-MAT/MC, for STM32 Motor Control design has been presented and applied to develop a PMSM motor drive model exploiting model based-design methodologies. Then, the embedded C code for the STM32F3 target has been automatically generated through the Embedded Coder and the STM32-MAT/TARGET toolboxes. The implemented MBD approach has been conceived as a complementary design path to the traditional one and it provides blocks IP that integrates on existing tools dedicated to the STM32 Motor Control ecosystem. The signal waveforms from the simulation model and the real system are quite comparable, showing that the MBD approach and developed toolboxes are useful tools to speed up and improve the design development cycle. The developed toolbox aims to support ST teams and customers to move from the traditional workflow to a new platform control design approach.

REFERENCES

[1] P. Andriani, F. Conti, L. Fortuna, M. Frasca, G. Passiante, A. Rizzo, Innovation systems by nonlinear networks, Nonlinear dynamics 44 (2006) 263–268.
[2] J. C. Jensen, D. H. Chang, E. A. Lee, A model-based design methodology for cyber-physical systems, Proc. 7th Int. Wireless Commun. Mobile Comput. Conf. (2011) 1666–1671.
[3] J. Reedy, S. Lunzman, Model based design accelerates the development of mechanical locomotive controls, SAE Technocal Paper 2010-01-1999 (2010).
[4] MathWorks, Simulink® Getting Started Guide (R2019b), 10th Edition, www.mathworks.com (2019).
[5] MathWorks, Managing Model-Based Design, www.mathworks.com (2015).
[6] STMicroelectronics, STM32F PMSM single/dual FOC SDK v4.3, User manual UM1052, www.st.com (2016).
[7] STMicroelectronics, Getting started with STM32 motor control SDK v5.4, User manual UM2374, www.st.com (2019).
[8] D. O'Sullivan, J. Sorensen, A. Murray, Model-Based Design Streamlines Embedded Motor Control System Development, www.analog.com (2015).
[9] STMicroelectronics, STM32-MAT/TARGET Hands On Rev 2.4, www.st.com (2019).