Workpackage	Deliverable ID
WP5, System Behavior Layer design and interfaces	**D5.6 System behavior layer final report**

Summary

This deliverable provides description of System behaviour layer building blocks. Thus, it shows results of BB3 Robust condition monitoring and predictive diagnostics and BB6 Self-commissioning velocity and position control loops.

Since this deliverable is public, it contains mainly the description about the usage of individual components. It does not contain technical details which are protected and IP of the involved companies. BB3 components developed in the project are subdivided into hardware and software ones. There is one representative of hardware component in this deliverable, which is Smart Vibration Sensor (SVS). There are several software BB3 components which can be used for the computation of performance indicators, for the predictive maintenance and for conditionally executed tasks.

The automatic commissioning of motion control system was previously decomposed into four categories which reside in Layer 2 and Layer 3. The components are the trajectory generator blocks and routines for system identification and controller/filter tuning. There are also some examples of controller commissioning interfaces. These are rather fixed to planned pilots and uses cases. The data acquisition module is strongly dependent on target platform.

The implementation aspects of BB3 and BB6 components on relevant pilot applications, use cases and demonstrators are discussed as well but only on descriptive level.

BB3 hardware component (SVS) is equipped with the EtherCAT interface which should simplify its integration in greenfield and brownfield applications. Software components of BB3 and BB6 are located in separated folders with agreed structure which should simplify their inclusion in I-MECH platform. In that way the BBs here result is archived, but yet not part of the I-MECH platform to make applications ‘in a mouse click’ within I-MECH Centre.

The deliverable is organized as follows. Introduction describes the motivation of the work. Second section deals with System behaviour design and interfaces. This section is rather brief. It refers to deliverable D5.5 which has the same due date and where the interfaces are treated in detail. The third section deals with BB3 components. The fourth section is devoted to automatic commissioning of motion control system - BB6. The conclusion concludes the deliverable.

Author	Bohumil Klíma, Petr Blaha, Martin Doseděl

Keywords Condition monitoring, predictive diagnostics, electric drive, key performance indicator, maintenance, vibration modules, frequency response function, identification controller parameter tuning, control structure, biquadratic filter.

Coordinator Sioux CCM

Tel. 0031 (0)40.263.5000

E-Mail info@i-mech.eu

Internetwww.i-mech.eu

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

Table of contents

List of figures

List of tables

(Open) Issues & Actions

Open Issues (and related actions) that need central attention shall be part of a file called "IAL - Issues & Action List – Partners" which is can be found in the Goolge Drive Partner Zone.

ID	Description	Due date	Owner	IAL ID
	No issues were identified during document writing.			

Document Revision History

Revision	Status	Date	Author	Description of changes	IAL ID / Review ID
R01	Draft	22-JUL-19	Petr Blaha	Initial version of the document.	-
R02	Revision 2	18-SEP-19	all	Comments to the document template and content.	-
R03	Revision 3	5-OCT-19	Bohumil Klíma	Consolidated contribution of BB3 and BB6.	-

R04	Pre-final	25-NOV-19	Bohumil Klíma	Improvement of BB3 and BB6 descriptions.	-
R05	Final	1-DEC-19	Petr Blaha	Integrated comments from the internal reviewers.	-

Contributors

Revision	Affiliation	Contributor	Description of work
R01	BUT	Petr Blaha	Preparation of document structure, coordination of work.
R02	GEF, SIOUX CCM, BUT, EDI	Davide Colombo, Arend-Jan Beltman, Petr Blaha, Kaspars Ozols	Discussion on document structure, fixing the format for WP3, WP4 and WP5 final deliverables.
R03	BUT	Bohumil Klíma, Martin Doseděl, Luděk Buchta, Petr Blaha, Zdeněk Havránek, Pavel Václavek	Description of BB3 components.
	PHI	Harry Mansvelt	Links with Pilot 5, support with BBs integration, providing the data for initial tests.
	SIE PLM	Olivier Schmidt	Support with system modelling.
	UNIBS, GEF, ZAPUNI	Luca Simoni, Davide Colombo, Martin Goubej	Consolidated description of BB6 components.
R04	UNIBS	Luca Simoni, Antonio Visioli	Improved version of BB6 description.
	BUT	Bohumil Klíma, Luděk Buchta	Contribution in chapter 1 and 2, work on improvement of chapter 3.
	J&J, TNI, ITML	Sámus Hickey, Michael, Walsh, Dimitris Spiliotopoulos	Contribution to Demonstrator 1.
R05	BUT	Petr Blaha, Bohumil Klíma, Martin Doseděl	Integration of comments, finalization of deliverable.

Document control

	Status	Draft	Revision 2	Revision 3	Pre-final	Final
	Revision	R01	R02	R03	R04	R05
Reviewer Name	**Role**	**Selection**				
Arend-Jan Beltman	Coordinator (SIOUX CCM)	X				
Marc van Eert	Internal reviewer (TNL)					X
Olivier Schmidt	Internal reviewer (SIEMENS)					X
Petr Blaha	WP5 leader (BUT)		X		X	X

Pavel Václavek	WP5 co-leader (BUT)			X		

File Locations

Via URL with a name that is equal to the document ID, you shall introduce a link to the location (either in Partner Zone or CIRCABC)

URL	Filename	Date
https://www.i-mech.eu/publications/deliverables/	Partner zone > Project Breakdown > WP5 (BUT) > (Formal) Deliverables of WP5 > 19072201R05 D5_6 System behaviour layer final report.pdf	02-DEC-2019

Literature

[1] M. Giacomelli, D. Colombo, L. Simoni, G. Finzi, and A. Visioli, “A fast autotuning method for velocity control of mechatronic systems,” IFAC-PapersOnLine, vol. 51, no. 4, pp. 208–213, 2018.

[2] M. Giacomelli, D. Colombo, G. Finzi, V. Setka, L. Simoni and A. Visioli, “An autotuning procedure for motion control of oscillatory mechatronic systems,” ETFA-Conference 2019.

[3] H. Vold, J. Crowley and G.T. Rocklin “New ways of estimating frequency response functions”. Sound & Vibration, 18(11), 34–38.), 1984.

[4] B. Klima; L. Buchta; M. Dosedel; Z. Havranek; P. Blaha, “Prognosis and Health Management in electric drives applications implemented in existing systems with limited data rate”, ETFA-Conference 2019.

[5] M. Dosedel; Z. Havranek, “Design and performance evaluation of smart vibration sensor for industrial applications with built-in MEMS accelerometers”, 18th International Conference on Mechatronics - Mechatronika (ME), 2018.

[6] D. Colombo, “D5.3, Auto-tuning & Self-commissioning (BB6), I-MECH project Deliverable”, January 2019.

[7] Z. Havránek, B. Klíma, M. Doseděl, P. Blaha, L. Buchta, “D5.4, Module for robust condition monitoring and predictive diagnostics of electrical drives (BB 3)”, I-MECH project deliverable, January 2019.

[8] M. van Eert, H. Kuppens, L. Simon, P. Blaha, “D5.5: System behaviour tools, data processing and interfaces”, I-MECH project deliverable, November 2019.

[9] R. Pulles, T. Lembrechts, G. van der Veen, G. Collepalumbo, R. Pacios, E. Smeets, “D7.1: Definition of the pilots”, I-MECH project deliverable, April 2018.

[10] “ISO20186 – Mechanical vibration – measurement and evaluation of machine vibration”, 1st edition, Prague: Czech Office for Standards, Metrology and Testing, 2017.

Abbreviations & Definitions

Abbreviation	Description
BB	Building Block
CI	Condition Indicator
CNC	Computer Numerical Control
COTS	Commercial Off-The-Shelf
CSV	Comma Separated Values
FFT	Fast Fourier Transformation
FRF	Frequency Response Function
GUI	Graphical User Interface
HDL	High Definition Language
HIL	Hardware In the Loop
I/O	Input / Output
ISO	International Organization for Standardization
LSM	Linear Synchronous Motor

MLS	Maximum Length Sequence
MIL	Model In the Loop
PID	Proportional Integral Derivative
PIL	Processor In the Loop
OPC UA	Open Platform Communication Unified Architecture
PLC	Programable Logic Controller
PMSM	Permanent Magnet Synchronous Motor
POMS	Phase Optimal-harmonic Signal
RMS	Root Mean Squares
RPM	Revolutions Per Minute
RPMS	Random Phase Multi-harmonic Signal
RUL	Remaining Useful Lifetime
SMS	Schroeder Multi-harmonic Signal
SVS	Smart Vibration Sensors
TF	Transfer Function
UC	Use Case
WP	Work Package

Definition	Description

1 Introduction

The I-MECH target is to provide augmented intelligence for wide range of cyber-physical systems having actively controlled moving elements, hence support development of smarter mechatronic systems. They face increasing demands on size, motion speed, precision, adaptability, self-diagnostic, connectivity, new cognitive features, etc. The fulfilment of these requirements is essential for building smart, safe and reliable production systems. This asks for completely new demands also on bottom layers of employed motion control system which cannot be routinely handled by available commercial products. On the ground of this, the main mission of this project is to bring novel intelligence into Instrumentation and Control Layers mainly by bridging the gap between latest research results and industrial practice in related model-based engineering fields. I-MECH delivers new interfaces and diagnostic data quality for System Behaviour Layer. It strives to provide a cutting-edge reference motion control platform for non-standard applications where the control speed, precision, optimal performance, easy reconfigurability and traceability are crucial.

A three-layer generalized form is defined for the mechatronics systems (Instrumentation Layer, Control Layer, System Behaviour Layer). The proposed functionalities are provided in the building blocks distributed in these layers This deliverable discusses two building blocks (out of eleven):

BB3 Robust condition monitoring and predictive diagnostics,
BB6 Self-commissioning velocity and position control loops.

Building blocks can be sorted onto hardware and software building blocks. The hardware building blocks represent unified platforms for mechatronics system control and other hardware modules like amplifiers and sensors. The software building blocks implement functionalities for mechatronic systems commissioning, control, diagnostics and monitoring. The usage of the building blocks is expected in existing mechatronics systems (brownfield systems) and in unified new hardware greenfield systems as well.

This document provides a description of individual building block components developed mainly in Task 5.3 and Task 5.4. Task 5.3 dealt with BB3 Robust condition monitoring and predictive diagnostics and Task 5.4 dealt with BB6 Self-commissioning velocity and position control loops. The description tries to be as comprehensive as possible. The exceptions are the parts dealing with proprietary solutions and the parts linked with partner's IP which are described only shallowly. This fact is caused with public nature of this deliverable. The missing information is sometimes supported with the bibliography link which unfortunately directs the reader to confidential deliverables which were already prepared during the course of I-MECH project.

2 System Behaviour Design and Interfaces

The procedures, system core functionalities and entities behaviour can be found in I-MECH deliverable D5.5. It deals with system behaviour tools, data processing in Layer 3 and with interfaces. The aim of this deliverable is not to redefine these approaches but to use them as a basis for the realization of BB3 and BB6 components. These two building blocks are special (comparing with the ones realized in WP3 and WP4) as they usually span over several layers and thus, they have strong connection to the system where they are implemented and used. Despite this fact, BB3 and BB6 components are realized as much target independent as possible.

2.1 Typical building block component

Typical BB component is shown in Figure 1. Input data, output data and parameters are assumed to structures. The input data is a data stream which is regularly passed for its analysis. Parameters enable to modify the behaviour of the block, but they are communicated only when they need to be changed, i.e. asynchronously. The output data are pre-processed data, computed condition indicators or the results of the controller design.

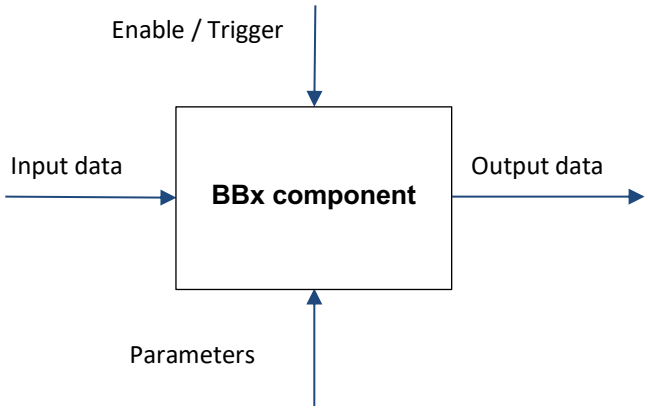

Figure 1. Typical software building block component.

The input Enable/Trigger enables to run the BB conditionally. Only when some specific conditions are met. This enables the evaluation of BBx component only in case when data are valid and useable. In the perspective of BB3, it enables to compute the condition indicator under the equal scenarios to be able to compare the results obtained in different time and to see the evolution of CI in time for predictive diagnostics. In the perspective of BB6 it is about identifying the parameters when the system is well excited, to design the controller when the estimated model is stable, etc.

2.2 How to operate with building block components

The building block components are designed like isolated functional MATLAB/Simulink blocks entities. These components have defined their inputs, outputs and parameters like single signals in basic form. The blocks represent essential functionalities which can be simply integrated into various platforms:

- The blocks can be translated into C/C++ functions or PLC standard function using Simulink coders.
- The I/O signals can be encapsulated into structures using Simulink Bus required by individual systems and implementations. The user can modify the I/O structure in Simulink according to implementation needs.
- The form of SW components is as general as possible to make them useable in greenfield and brownfield implementations. In case of brownfield implementations, it is recommended to use the existing data communication and existing protocols for passing the data between layers.
- Implementation user's guide is available for every SW BB component. The user can easily become familiar with its functionality, typical interconnection and other necessary implementation aspects as well as with simple demo implementation using MIL, PIL or HIL simulation.

Facts mentioned above support the interoperability requirement of building blocks.

2.3 Communication interfaces

EtherCAT communication interface is supported as the communication interface between Layer 1 and Layer 2, as well as OPC UA for the upper layer access (Layer 2 to Layer 3 and vice versa). The BBs define the communication interface parameters but not the interface itself, thus they are capable to operate with all standard interfaces implemented in the target platform.

3 BB3 - Robust control condition monitoring and predictive diagnostics of electrical drive

3.1 Functionalities

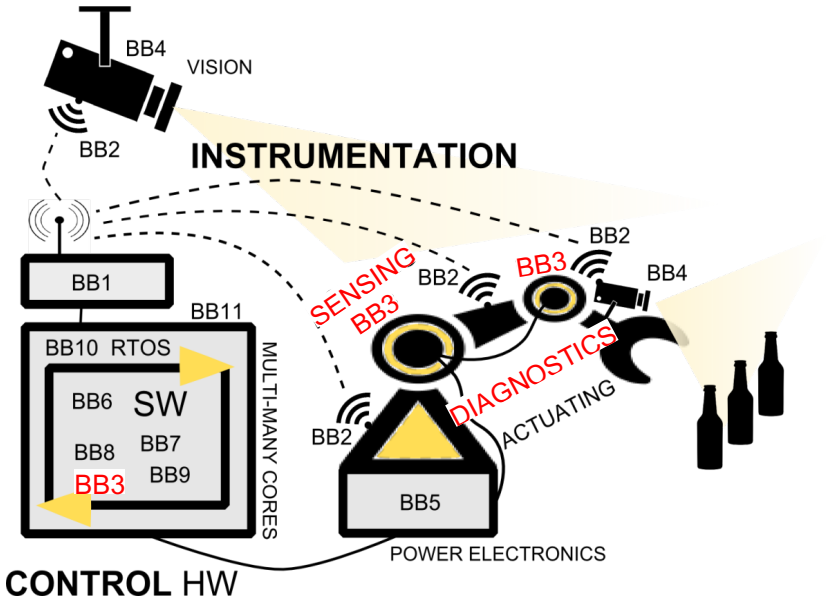

Figure 2. BB3 in I-MECH platform.

Condition monitoring in electric drives under I-MECH project (see Figure 2) is based on analysis of process quantities and variables. Data for condition monitoring come from inverter controller sensors which are traditionally installed in electric drives, e.g.; phase current sensors. The other data sources are control process variables, e.g. controller outputs. None of them can directly point on faulty situation. The reason is that the monitored signals can generally reach various values, have frequency bandwidth limited information or limited resolution for analysis pointing on specific faults. However, there exist some approaches how get condition indicators:

- Mutual relation of more signals monitoring (under specific conditions or dynamic states)
- Specific signals repeating sequence monitoring or performing periodic self-test under defined conditions
- Installing appropriate sensors with sufficient sampling and resolution – vibration sensors typically.

The placement strategy of the BB3 blocks can be seen in Figure 3. The block's source codes are developed in MATLAB and/or MATLAB Simulink and their implementation into final application depends on the individual capabilities of the target devices in general.

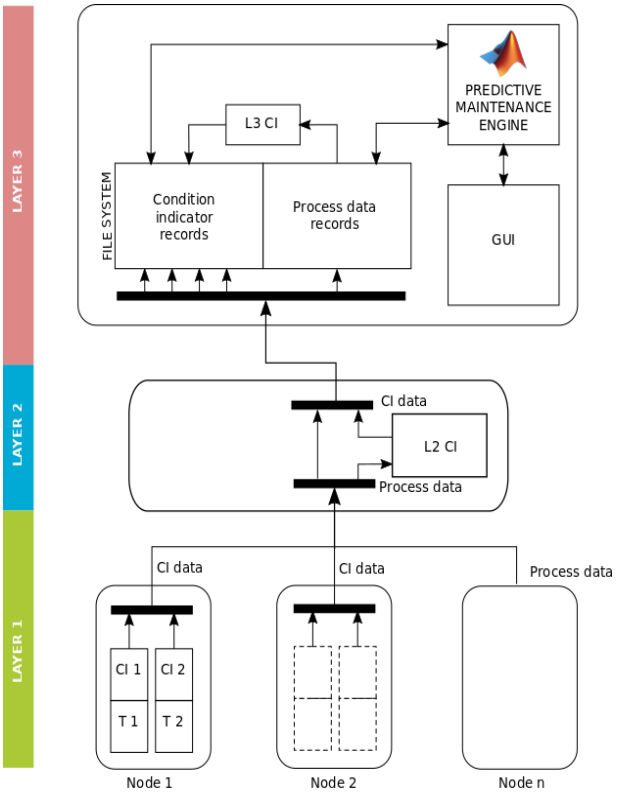

Figure 3. CI placement in I-MECH platform.

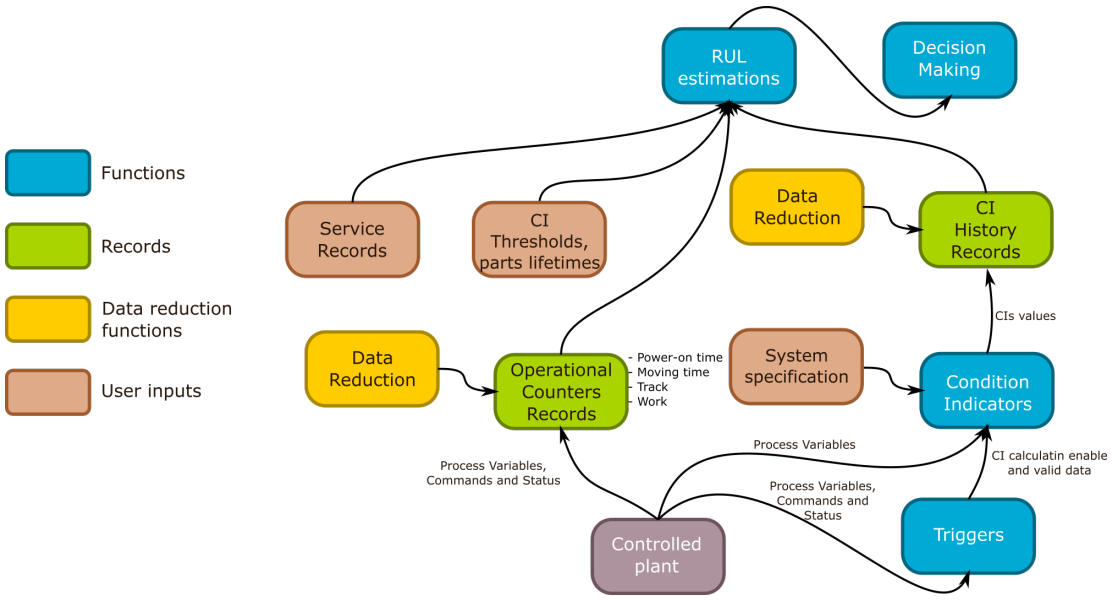

Figure 4. I-MECH condition monitoring and predictive maintenance data processing flow chart.

3.2 Data processing chain

The Figure 4 shows I-MECH condition monitoring and predictive maintenance functionalities and data flows. The structure is displayed in the Layers-independent perspective.

The i2t condition indicator described in following subsection is a type of monitoring of specific repeating action. Equal or very similar load condition are supposed to be used for the individual scenario repetitions. Propagating faults can affect condition indicator output in that case.

Various repeating scenarios can be monitored using appropriate triggers for enabling the condition indicators computation. The speedRampTrigger serves for enabling a condition indicator for drive acceleration/deceleration from starting to ending speed. Both blocks are shown and described in following subsections.

3.3 I2t condition indicator

The i2t algorithm is usually used for motor winding protection in electric drives field. The method is implemented in common drive control firmware like a sensor-less thermal protection.

3.3.1 General description - algorithm theory

The ability to dissipate the energy in a motor winding is proportional to the square of the current flowing through it. The winding is designed to sustain the nominal current in the continuous manner. The produced heat is in balance with the heat dissipated to the surrounding and winding doesn't exceed the operating temperature. The meaning of I2t method is the computation of the integral of the heat energy above its nominal value. The winding is able to withstand specific short thermal energy peaks above the nominal operation. The excess of the thermal energy is called "I-squared-t" i2t. It is expressed in [A^2·s] and can be calculated according to following formula:

$$i^2t = \int (i^2\int t - i_{nom}^2)\, dt$$

The bottom limit of the integral is limited to zero. The operation under nominal current causes decreasing of the i2t value to zero. The discrete time formula for i2t calculation in digital control systems is as follows:

$$(i^2t)_k = (i^2t)_{k-1} + T_s(i^2{}_k - i_{nom}^2)$$

Electric drives have i2t protection implemented in the control algorithm for preventing the motor damage during over-current operation. Protection is activated when i2t value reaches maximum allowed value defined by specific parameter for a protected motor. The controller current limit is reduced to nominal current level and then the i2t value cannot increase more. The protection is deactivated when i2t value falls to zero during drive operation under nominal current.

The I2t method can be adopted for condition monitoring purposes. It is assumed that the drive is used in specific application. If repeating actions are executed in the drive application, then specific current consumption is assumed for each action - typical action current. This current can be subtracted in the algorithm instead of nominal current as it was described previously. The advantage of i2t algorithm for repeating action monitoring is that it subtracts squares of actual motor current and typical current. This way of motor current processing provides good sensitivity for observing system parameter time changes and for fault propagation monitoring. Small changes in system parameters can affect i2t indicator value significantly due to the process of integration.

Typical monitored faults can be increased friction or mechanical load.

3.3.2 Implementation in MATLAB Simulink

I2t condition indicator block is developed in MATLAB Simulink. Direct conversion into C, C++, PLC language or VHDL is possible using Simulink Coder, PLC coder or HDL Coder is possible.

The implementation is done at the basic level, block contains individual inputs, parameters and outputs. The model can be encapsulated into the superior block with inputs outputs and parameters grouped in structures/busses for appropriate modification of data passing. This solution enables wider utilization of the block since the structures are

not always supported in the target application (Gefran drive (UseCase 1.1) did not accept the structures in Structured text which was automatically generated from the Simulink scheme with buses).

The internal structure of this block is shown in the Figure 5. The scheme contains three subsystems.

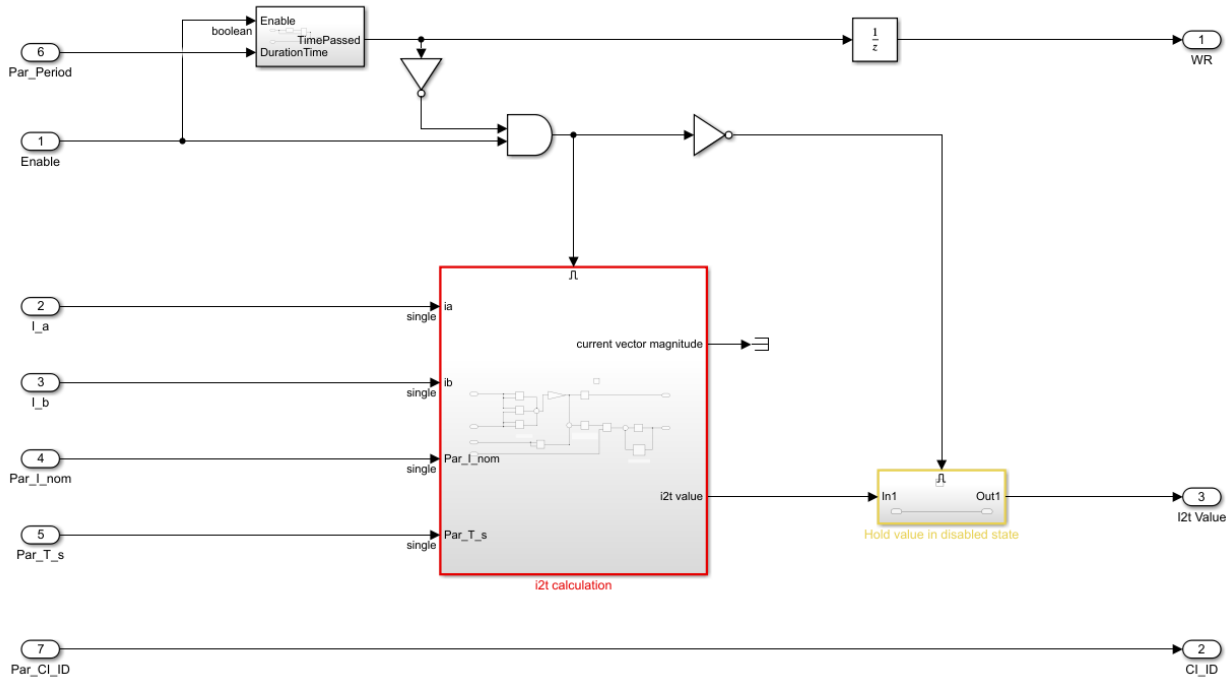

Figure 5. I2t condition indicator implementation.

The i2t calculation subsystem is shown in the Figure 6. The calculation is based on phase currents of three phase motor. Two phase currents are measured in the motor controller typically.

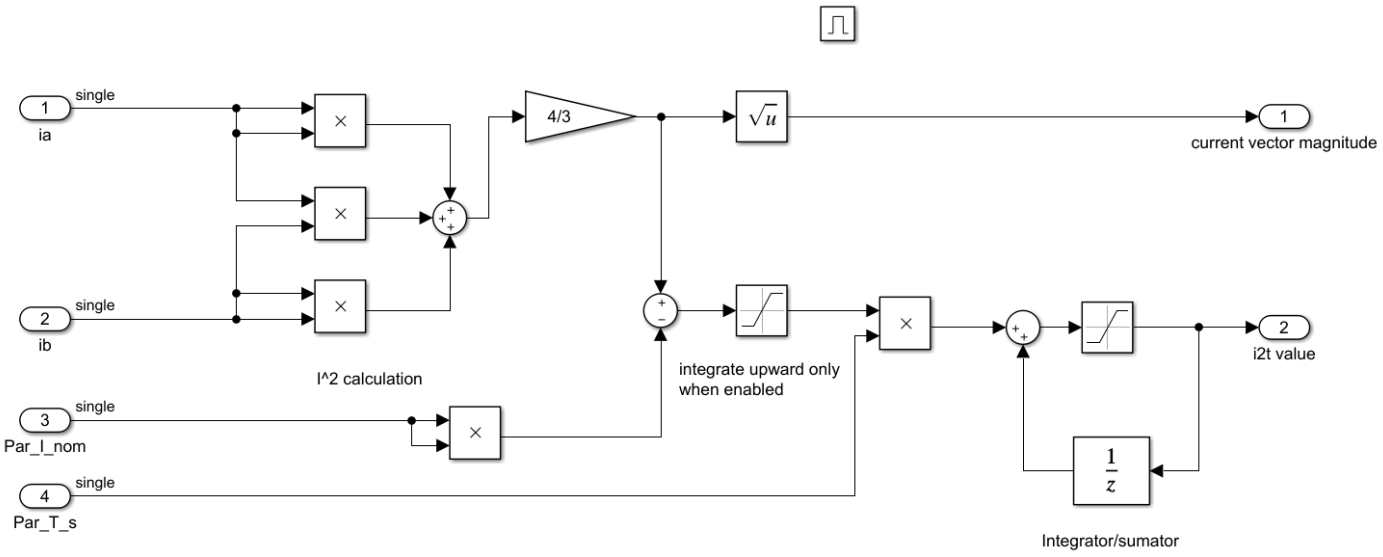

Figure 6. I2t condition indicator implementation – i2t equation implementation based on phase currents of the three-phase motor.

The scheme contains two saturation blocks. The bottom limit of the integrator/summator is set to zero. It is the one which is implemented in general i2t method. The second one has the bottom limit set to zero value as well. It allows upward integration only during monitored repeating action of the drive. It implements peak detection capability of i2t value for monitored action.

Figure 7 shows realization of pre-trigger condition measurement scheme. True value of enable input starts time measurement. Falling edge resets the integrator and stops the integration. The value of the integral is compared with value predefined by the second input. The block thus generates the pulse with predefined duration if it is not terminated with the enable signal drop.

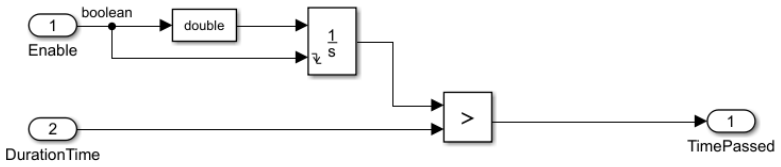

Figure 7. I2t condition indicator implementation – calculation period measurement subblock.

The third subsystem just holds the value of the integration if the measurement period was not terminated.

3.3.3 Inputs, outputs and parameters

Simulink block of i2t condition indicator is in the Figure 8. The inputs, outputs and parameter signals are visible in the figure. All the signals are described in the Table 1.

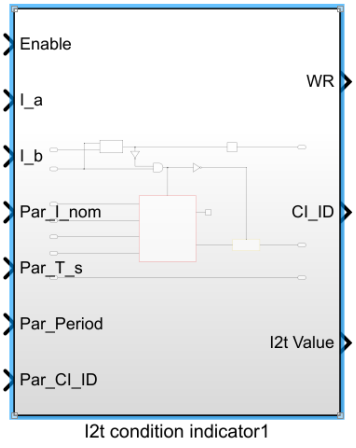

Figure 8. I2t condition indicator Simulink block.

Table 1: i2t condition indicator.

I2t condition indicator	
Inputs	
Name	**Description**
ENABLE	Enable input. True value enables integration of I2t and starts measurement of calculation period. Falling edge before elapsing the calculation period resets the i2t integral and doesn't generate WR signal because the output is not valid. Falling edge provides valid i2t value

	output and generates WR signal rising edge for confirming valid data when calculation period is elapsed.
I_a	Phase A current of the three-phase motor measured in the motor controller
I_b	Phase B current of the three-phase motor measured in the motor controller
Outputs	
Name	**Description**
I2t_value	I2t condition indicator value. The value is valid on WR rising edge. The enable input must be active for the whole calculation period.
WR	Positive edge of WR (write) signals confirms the valid i2t indicator value. The signal is used for asynchronous condition indicator data storage to its record files (write for transmitting)
CI_ID	Condition indicator ID identifies specific condition indicator and its history records. It has to be a part of the message when asynchronous communication is used to identify origin of the data and file where it should be recorded. For this reason, it is provided at the block output.
Parameter	
Name	**Description**
Par_I_nom	Nominal effective current of monitored repeating action. It should be set in a manner to get reasonably small i2t condition indicator output for monitored repeating action when system is healthy.
Par_T_s	Repeating period of i2t indicator calculation
Par_Period	Monitoring time
Par_CI_ID	Condition indicator ID input sets unique condition indicator ID and its history records. It has to be a part of message when asynchronous communication is used to identify origin of the data and file where has to be recorded. For this reason it is passed at the block output.

3.4 Speed ramp trigger block

3.4.1 Functional description

The purpose of this block is to enable the calculation of a condition indicator during specific scenario of the drive. The ramp with defined start and final value and defined acceleration of the drive is the scenario here to follow. The output of this block is used as enable for the block which computes the condition indication. If the speed is inside of defined limits, then the enable is kept active. If the speed gets out of defined limits then the enable signal goes down and terminates the comutation of condition indicator.

A steady state period of monitored speed can be required at the beginning. The pre-trigger condition is defined by top and bottom limits (*StartValMax* and *StartValMin* parameters) and by required duration period (*StartValDuration*).

Enable signal is issued when drive accelerates within limits specified by slope (*SlopeMin, SlopeMax*) parameters to final value (defined by *EndValMin* and *EndValMax* parameters). If the monitored signals exceed the limits, Enable signal goes to false value.

Individual parameters and Enable signal behaviour are shown Figure 9.

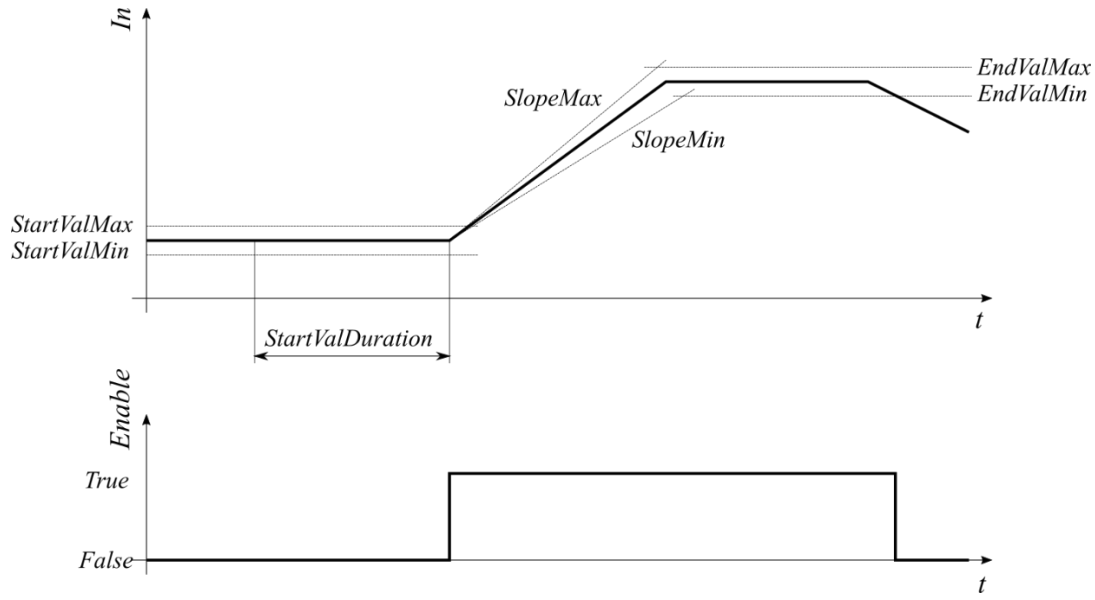

Figure 9. Speed ramp trigger parameter definition.

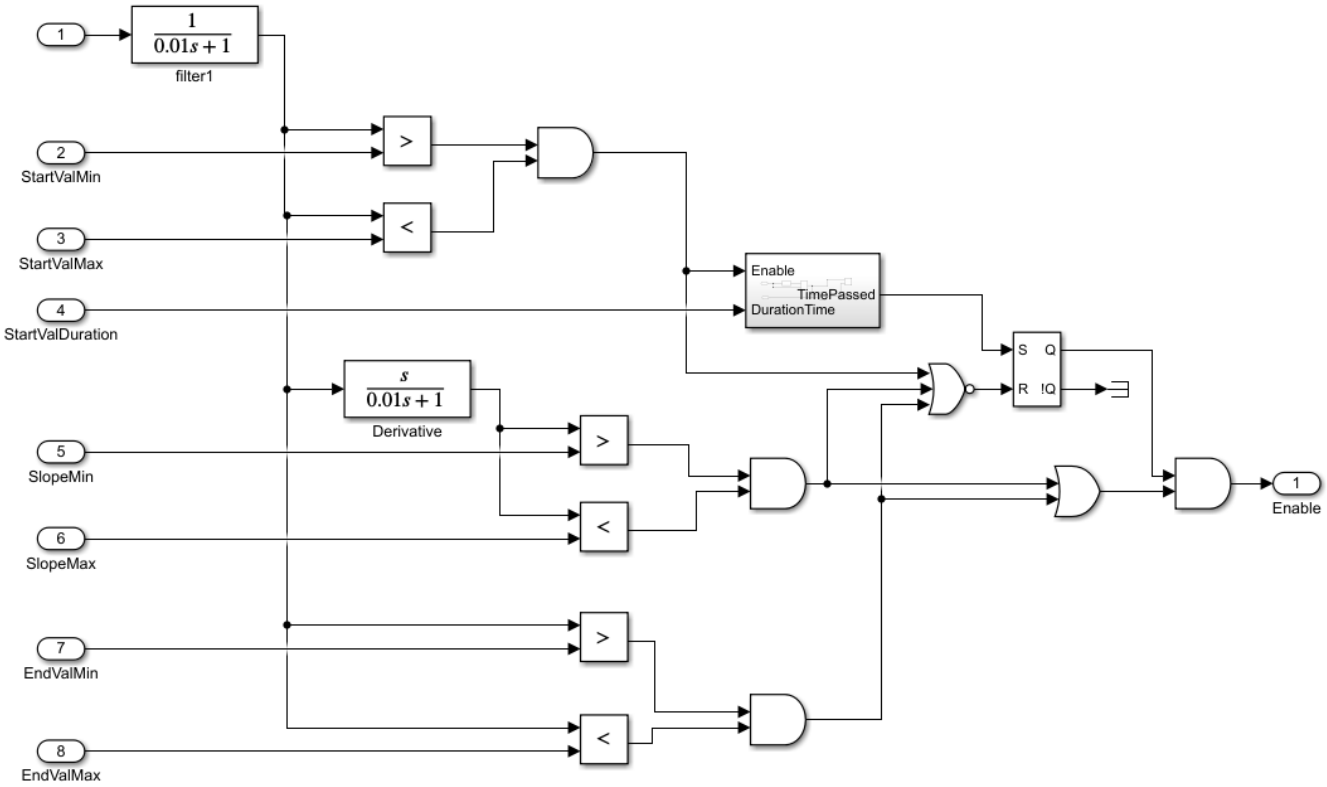

Figure 10. Speed ramp trigger – internal block scheme.

3.4.2 Implementation in MATLAB Simulink

I2t condition indicator source is developed in MATLAB Simulink. Direct conversion into C, C++, PLC languages is possible using Simulink Coder or PLC coder.

The implementation is done at the basic level, block contains individual inputs and outputs. The internal scheme of the speed ramp trigger is shown in Figure 10 and the Simulink block can be seen in the Figure 11.

3.4.3 Inputs, outputs, parameters description

Simulink block of i2t condition indicator is in the Figure 10. The inputs, outputs and parameter signals are visible in the figure. All the signals are described in the Table 2.

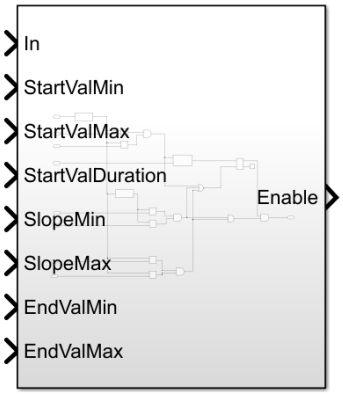

Figure 11. Speed ramp trigger Simulink block.

Table 2: Speed ramp trigger.

Speed ramp trigger	
Inputs	
Name	**Description**
In	Triggering input signal. Desired or measured speed can be typically used
Outputs	
Name	**Description**
Enable	Enable signal is true if triggering input goes in specified limits. Pre-trigger condition had been fulfilled and signal ramps operates within specified limits.
Parameters	
Name	**Description**
StartValMin	Bottom limit of input signal steady state value during pre-triggering phase
StartValMax	Upper limit of input signal steady state value during pre-triggering phase
StartValDuration	Duration of pre-triggering phase
SlopeMin	Minimum slope of input signal ramp
SlopeMax	Maximum slope of input signal ramp

EndValMin	Bottom limit of steady state input after the ramp
EndValMax	Upper limit of steady state input after the ramp

3.4.4 Validation in MIL

Figure 12. shows MIL testing scheme of the i2t condition indicator block and speed ramp trigger block. The MIL testing is done on a model of PMSM drive control. It has speed and torque setpoints as the inputs. The tested blocks are implemented at the bottom part of the scheme. The triggering variable is the desired speed of the drive. The trigger provides enable input for condition indicator. The I2t CI condition indicator uses phase currents for computing i2t value for the scenario which is given by the input parameters of RampTrigger.

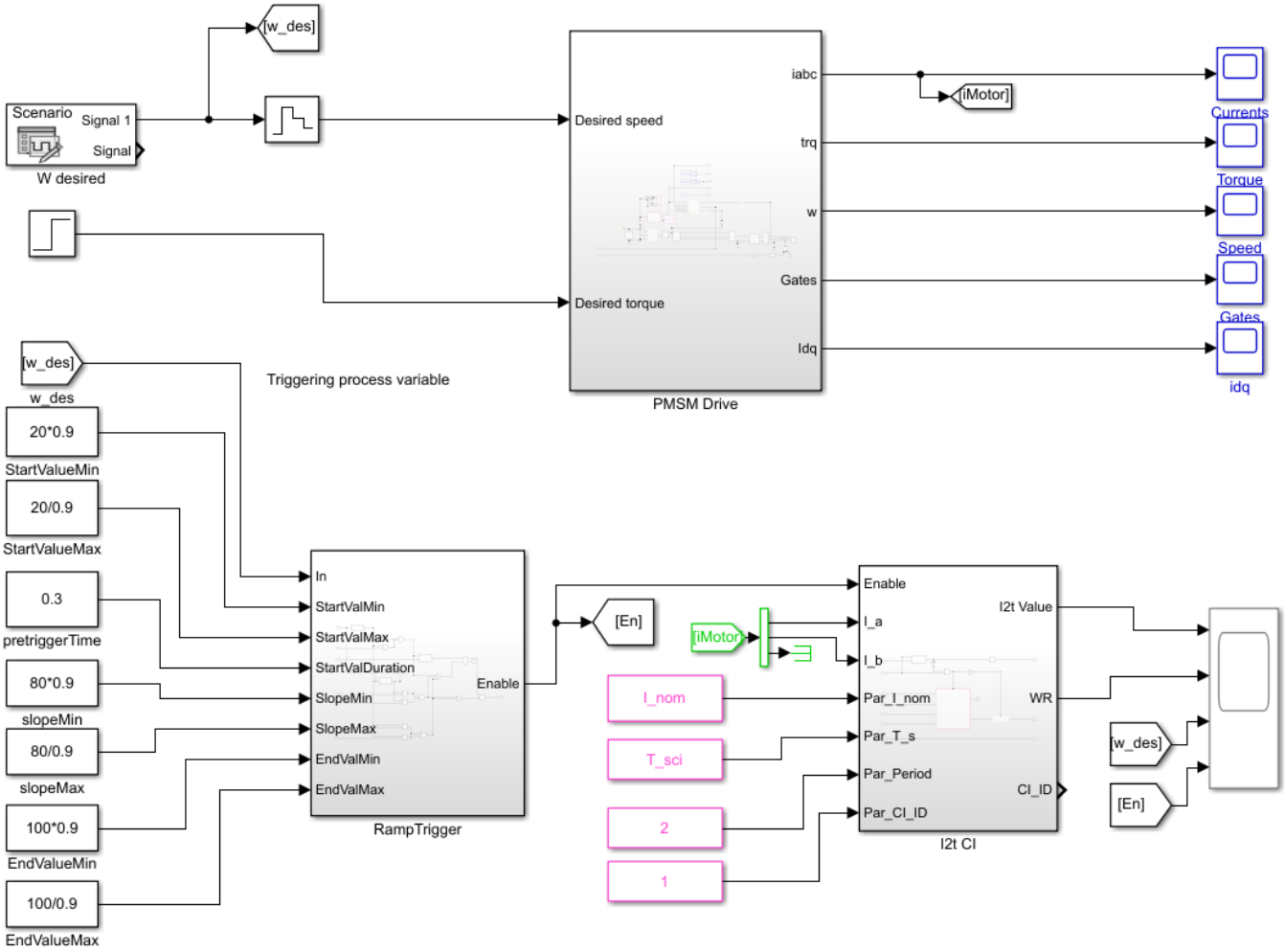

Figure 12. BB3 i2t condition indicator and speed ramp trigger blocks in MIL validation model.

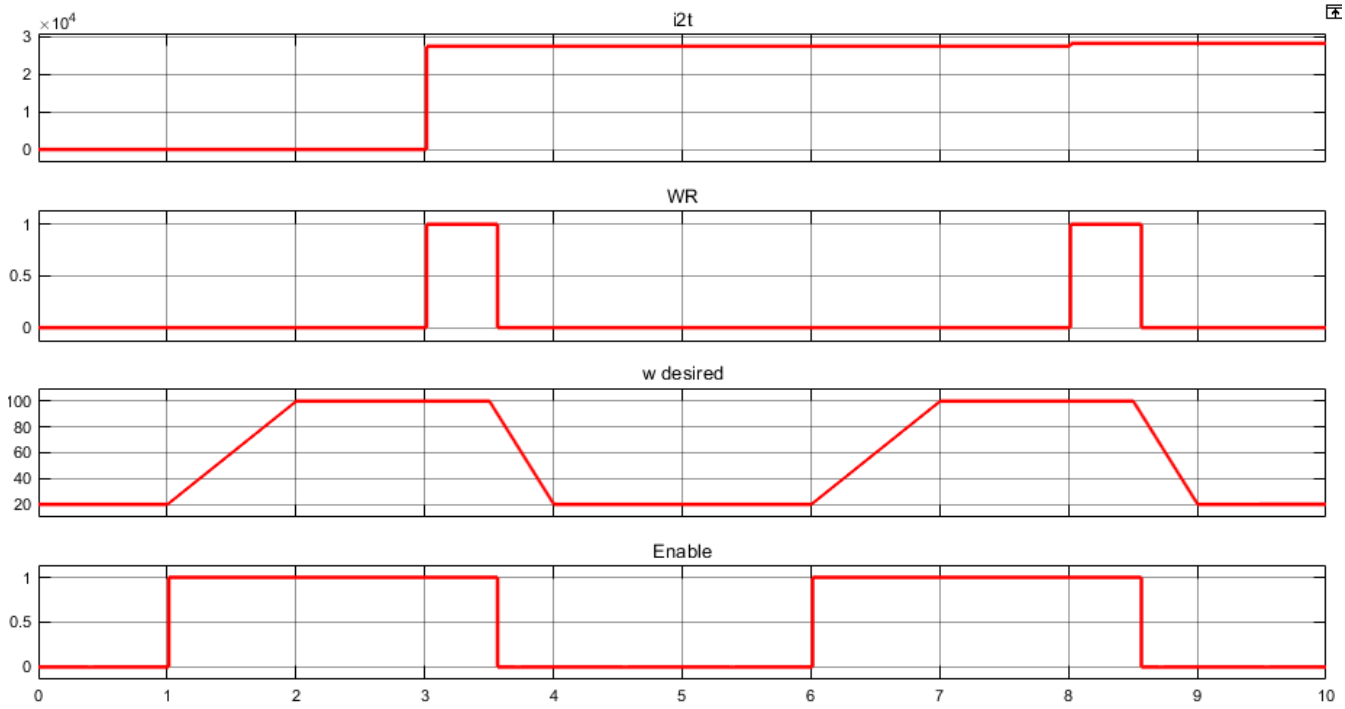

Figure 13.BB3 i2t condition indicator and speed ramp trigger blocks in MIL validation scheme.

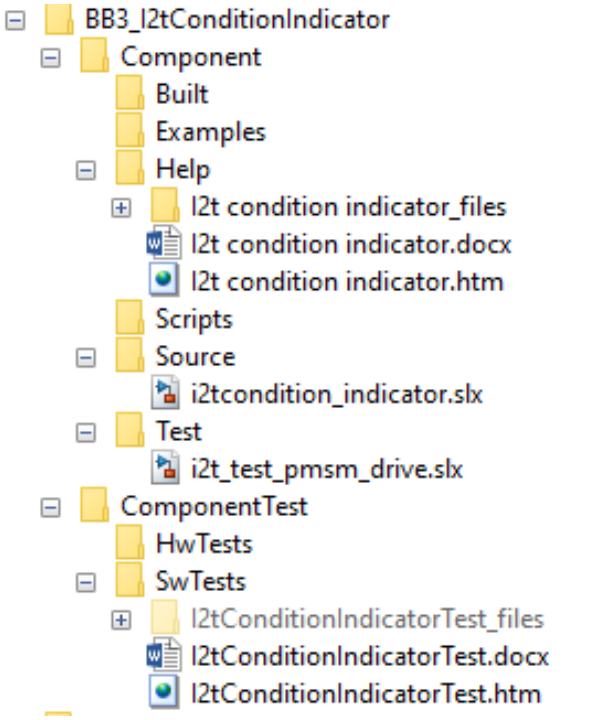

Figure 14. File structure for BB3 i2t condition indicator.

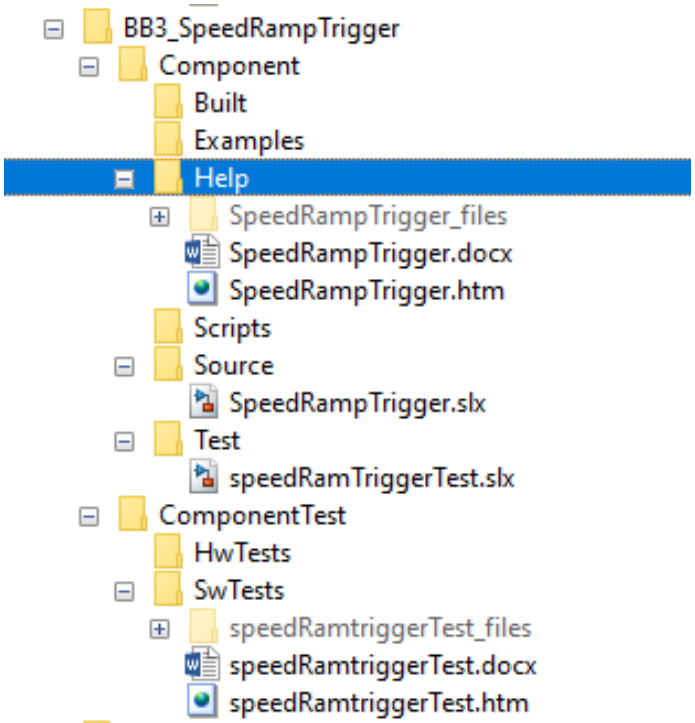

Figure 15. File structure for BB3 speed ramp trigger.

The MIL validation test in Figure 13 represents two times repeated acceleration and deceleration of the drive from 20 to 100 rpm. The first course shows condition indicator value. Zero value means that the output was not initialized at the beginning and no WR signal edge was generated. The output is updated after monitored scenario has been passed. The second course shows WR signal. The WR signal rising edge is generated on a next computing step after the output has been updated. The WR signal becomes inactive when Enable condition is ended. Third course represents desired speed used as triggering signal. There are two equal repeating accelerations and decelerations speed commands. The last course shows the Enable signal.

3.4.5 Location of files

The folder and file structures are prepared according to the proposal in D6.1 for BB3 components. Figure 14 shows Folder and file structure for i2t condition indicator. Figure 15 shows folder and file structure for BB3 speed ramp trigger.

3.5 Smart Vibration Sensor (SVS)

3.5.1 Overview of the Smart Vibration Sensor

The Smart Vibration Sensor (SVS) is a part of the BB3 serving mainly for the condition monitoring and the predictive diagnostic purposes of the electric drives. It evaluates a machine health based on the measurement of mechanical movements of a surface. The vibration is generated by a standard machine operation (in the normal machine state) or as a result of some incoming failure (in the e.g. starting failure state). SVS measures the vibrations of the machine, processes the signals and provides the information into the upper layers of I-MECH topology. The theory of vibration-based machine diagnostics and the parameters of the SVS were described in detail in I-MECH Deliverable D5.4 [7].

SVS is a part of the complete system consisting of the sensor itself, National Instrument cRIO with processing cards (input/output card with developed RS485 interface and EtherCAT communication card) serving as EtherCAT slave and second cRIO serving as the EtherCAT master. All the devices are perfectly in-line with the I-MECH topology [7] – see Figure 16.

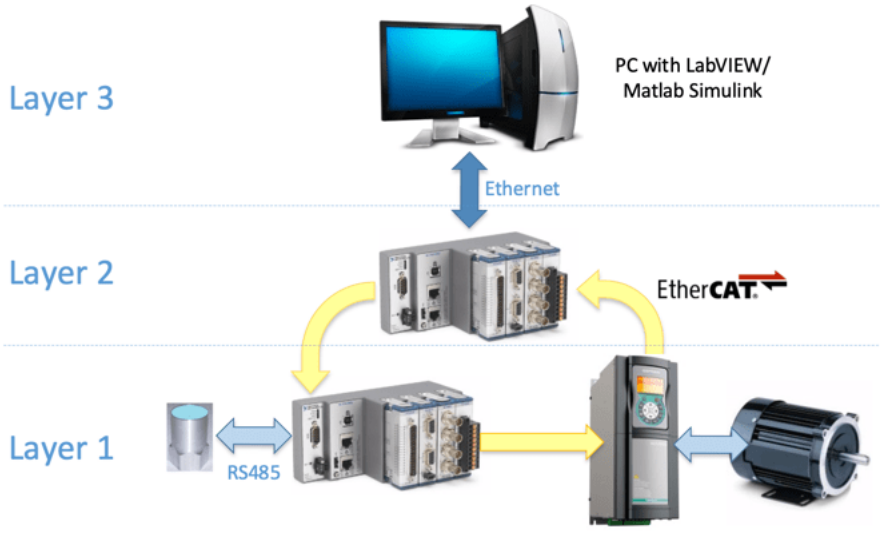

Figure 16: EtherCAT data exchange between diagnostic module with the SVS and the GEFRAN drive [7].

SVS is responsible for the conversion of the mechanical vibrations to the electrical signals using two low-noise MEMS accelerometers. Thanks to the use of the two sensors, it is possible to measure the vibration signals in three perpendicular axes, in the different frequency ranges and also to do a comparison of the elements so as to prevent the failure and to increase the operational safety of the whole system. Sensor provides the upper system with the real-

time vibration waveform samples as well as the temperature information and the other system information. Sensor is equipped with the RS485 communication interface and the data transfer is done at the speed of 3.686 MHz. Proprietary protocol for the communication is used. The sensor is in the present time used only as the transmitter – it sends the data immediately after power on sequence. The data is sent in the 8 bursts of the 4 bytes size each and the frame sequence (32 bytes) starts with the leading character 0x0011h. The communication protocol can be seen in Table 3 and Figure 17.

Table 3: SVS communication protocol.

	1B	2B	3B	4B	Note:
1. packet	START	ID	1002_H	1002_L	Start packet
2. packet	1002_H	1002_L	355_X2	355_X1	ADXL1002/ADXL355(X) vibration
	1002_H	1002_L	355_X0	STATUS	ADXL1002/ADXL355(X) vibration + status of the SVS
4. packet	1002_H	1002_L	355_Y2	355_Y1	ADXL1002/ADXL355(Y) vibration signal
5. packet	1002_H	1002_L	355_Y0	REF	ADXL1002/ADXL355(Y) vibration + voltage reference value
6. packet	1002_H	1002_L	355_Z2	355_Z1	ADXL1002/ADXL355(Z) vibration
7. packet	1002_H	1002_L	355_Z0	VOLTAGE	ADXL1002/ADXL355(Z) vibration
8. packet	1002_H	1002_L	355_T1	355_T0	ADXL1002/temperature signal

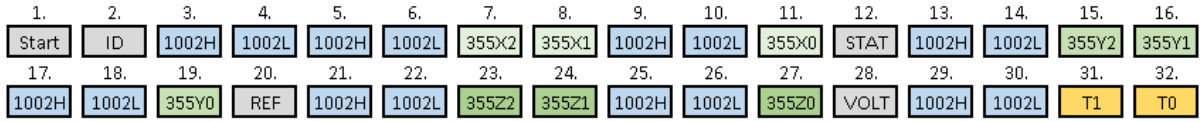

Figure 17: SVS communication protocol (32 bytes).

The communication is not confirmed by the receiver side, the sensor transmits the data to the RS485 bus and the receiver needs to synchronize the communication itself. The example of the packet timing can be seen in the

Figure 18.

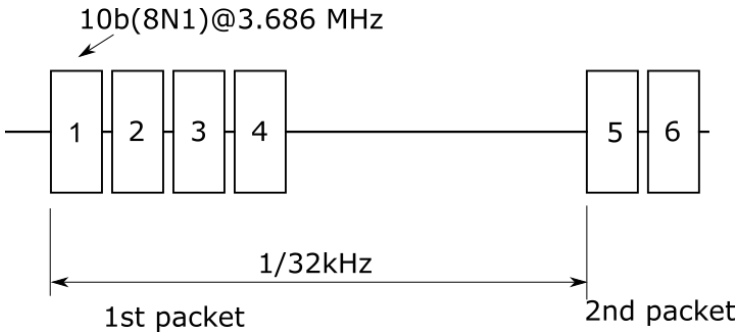

Figure 18: Time frame of the data packets communication of the SVS.

The receiver (cRIO in our case) must catch the data sent by the sensor and restore the data stream to the vibration time signal. It also has to calculate the condition indicators (CI) based on the received vibration values. The basic CIs are:

- ISO band RMS value (effective vibration velocity in the frequency range of 10 Hz up to 1 kHz define by the international standard ISO20816 [10].

- Overall vibration acceleration.
- Frequency spectrum in acceleration/velocity for individual mechanical faults detection.
- Fundamental harmonic value.
- Second and other harmonic values.
- Bearing fault diagnostics based on e.g. fault frequencies detection, envelope analysis calculation etc.

3.5.2 Implementation of the CI calculation blocks

Each CI calculation block is implemented in LabVIEW environment. SVS output signal (vibration time waveform) is always the input to each block. This signal is necessary, because it keeps the key information about monitored machine. Sampling rate, format of the number as well as the other parameters of the waveform are strictly defined by the system architecture and cannot be changed.

Another necessary input for each particular block is a vector with the parameters of the powertrain, especially with the main parts description (e.g. revolutions of the system, gearbox ratios, bearing dimensions etc.). The complete idea of the part of such system in shown in Figure 19.

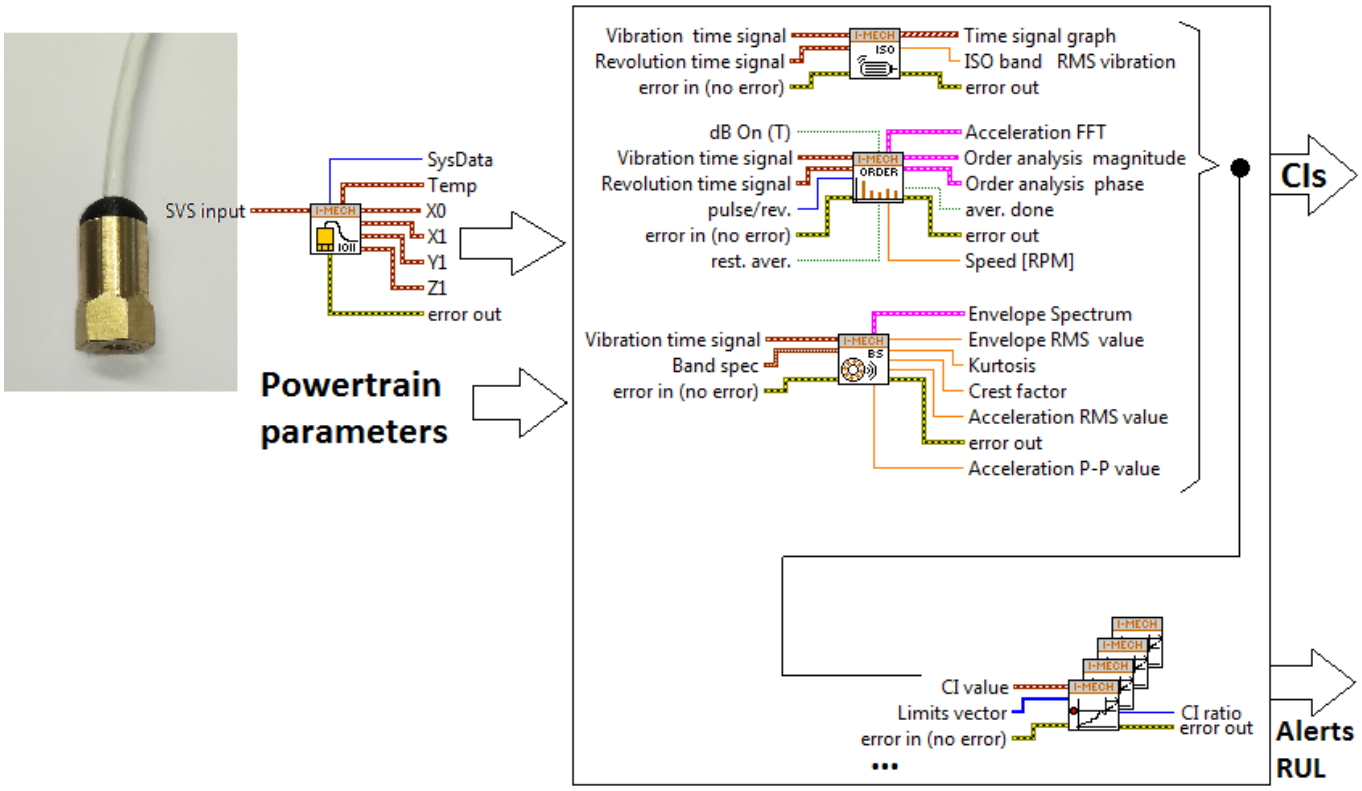

Figure 19: Block scheme of the predictive maintenance system.

A most important block of the whole system is the SVS communication interface block. This one ensures the data capture from the sensor, formatting the RS485 stream to the correct vibration, temperature and process data and providing the following blocks with the important input stream. The description of the inputs and outputs of this block can be found in the Table 6.

The example of the condition indicator block calculating overall ISO vibration velocity implemented in LabVIEW can be seen in Figure 20.

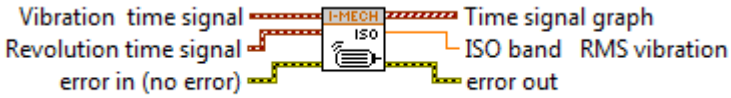

Figure 20: ISO vibrations block.

The inputs/outputs configuration of the CI calculation block can be found in the Table 4.

Table 4: ISO vibration block.

ISO vibration block		
Inputs:		
Vibration time signal	LabVIEW waveform	Vibration acceleration input time samples
Revolution time signal	LabVIEW waveform	Revolution input time samples
Error in	LabVIEW error wire	Internal LabVIEW signal
Outputs:		
Time signal graph	Array of LabVIEW waveforms	Integrated and filtered vibrations time signal (in the band 10 Hz … 1 kHz) + revolutions time signal
ISO band RMS vibration	Double	RMS ISO velocity of the input vibrations
Error out	LabVIEW error wire	Internal LabVIEW signal

Calculated Cis values are processed in the CI evaluation block. Its inputs, outputs and parameters can be seen in the Table 5. The calculated CI value is the input of this block, the Limit vector is used as the parameter. It consists of:

- exact limit value, which is critical, and a serious failure can occur in case of exceeding this value,
- time parameter, expressing the time, when CI must be higher, then exact limit. This prevents the false trigger caused by the random peaks and contributes to the robustness of the whole system.

CI ratio is the output of the block. It shows the ratio of the measured CI compared to the limit from the Limit vector. The meaning of this output is to express how serious the state of the particular component is.

Table 5: CI evaluation block.

CI evaluation block		
Inputs:		
CI value	LabVIEW waveform	CI samples
Error in	LabVIEW error wire	Internal LabVIEW signal
Outputs:		
CI ratio	Double (15 digits precision)	The severity of the CI compared to the limit value.
Parameters:		
Limits vector	LabVIEW cluster	Cluster of absolute value (showing the limit of the particular CI) and corresponding time delay (time, when CI must be higher then set limit).

Error out	LabVIEW error wire	Internal LabVIEW signal

Following table shows the set of the most important condition indicators blocks with description of the input signals, parameters, output information as well as a brief description of the block functionality. The details can be seen in the Table 6 to Table 10.

Table 6: Inputs, outputs and parameters of the SVS processing block description.

SVS processing		
The blockThis block ensures communication with the sensor, decoding and processing raw data and transforming the data to the output data stream suitable for following blocks.		
Inputs:		
SVS input	RS485 bus	Data stream from the sensor using proprietary communication format
Error in	LabVIEW error wire	Internal LabVIEW signal
Outputs:		
X0, X1, Y1, Z1	LabVIEW waveform	Vibration time signal from both elements in three perpendicular axes – are used as the inputs of the following blocks.
Temp	LabVIEW waveform	Temperature data
SysData	Array of double	System data (reference voltage, inner states of the sensor etc).
Error out	LabVIEW error wire	Internal LabVIEW signal

Table 7: Order analysis CI processing block description.

Order analysis CI		
The block providing the user with the frequency spectrum of the input rough vibration signal as well as complete (magnitude + phase) order spectrum depending on the speed of the machine.		
Inputs:		
Vibration time signal	LabVIEW waveform	Vibration acceleration input time samples
Revolution time signal	LabVIEW waveform	Revolution input time samples
Pulses/rev	Integer	Number of pulses per revolution used for tacho probe
dB On rest. aver	Boolean	Chart scaling in decibels, restart averaging signal
Error in	LabVIEW error wire	Internal LabVIEW signal
Outputs:		

Acceleration FFT	LabVIEW cluster	Cluster of the frequency spectrum of the rough acceleration data consisting of f0, df and array of magnitudes.
Order analysis magnitude	LabVIEW cluster	Cluster of magnitudes of the order frequency spectrum of the rough acceleration data consisting of f0, df and array of magnitudes.
Order analysis phase	LabVIEW cluster	Cluster of phases of the order frequency spectrum of the rough acceleration data consisting of f0, df and array of phases.
Aver. done	Boolean	Information that averaging process of the spectrum calculation is finished.
Speed [RPM]	Double	Number expressing calculated RPMs from the revolution's waveform and the number of pulses.
Error out	LabVIEW error wire	Internal LabVIEW signal
Parameters:		
Machine revolutions	Double (15 digits precision)	If the revolution signal is not available, it is possible to provide the block with the information about the speed of the machine from the system as a single number.

Table 8: Bearing state CI processing block description.

Bearing state CI		
Inputs:		
Vibration time signal	LabVIEW waveform	Vibration acceleration input time samples
Band spec	LabVIEW cluster	Cluster of two double-type variables: *Central frequency* and *frequency span*. These values define the band specification, where envelope analysis is performed.
Error in	LabVIEW error wire	Internal LabVIEW signal
Outputs:		
Envelope spectrum	LabVIEW cluster	Cluster of magnitudes of the enveloped frequency spectrum of the rough acceleration data consisting of *f0*, *df* and array of *magnitudes*.
Envelope RMS value	Double (15 digit precision)	RMS value of the enveloped signal
Kurtosis	Double (15 digit precision)	Kurtosis of the signal
Crest Factor	Double (15 digit precision)	Crest factor

Acceleration RMS value	Double (15 digit precision)	RMS value of the input signal
Acceleration P-P value	Double (15 digit precision)	P-P value of the input signal
Error out	LabVIEW error wire	Internal LabVIEW signal

Table 9: Unbalance CI processing block description.

Unbalance CI		
Inputs:		
Vibration time signal	LabVIEW waveform	Vibration acceleration input time samples
Revolution time signal	LabVIEW waveform	Revolution input time samples
Outputs:		
Unbalance harmonics	Double (15 digits precision)	Value of the harmonic component, which represents unbalance of the machine
Unbalance severity	Double (15 digits precision)	In case of historical data, this number shows ratio of increasing of the unbalance
Parameters:		
Machine revolutions	Double (15 digits precision)	If the revolution signal is not available, it is possible to provide the information about the speed of the machine from the system as a single number.

Table 10: Bearing fault CI processing block description.

Bearing fault CI		
Inputs:		
Vibration time signal	LabVIEW waveform	Vibration acceleration input time samples
Revolution time signal	LabVIEW waveform	Revolution input time samples
Outputs:		
Bearing fault	LabVIEW cluster	Cluster of the four calculated bearing fault frequencies and four acceleration values.
Bearing frequencies severity	LabVIEW cluster	In case of historical data, this cluster shows the ratio of increasing of the bearing fault frequencies data.
Parameters:		
Machine revolutions	Double (15 digits precision)	If the revolution signal is not available, it is possible to provide the information about

		the speed of the machine from the system as a single number.
Bearing dimensions	LabVIEW cluster	Cluster consisting of the main dimensions of the bearing necessary for the exact calculation of the frequencies.

3.5.3 General outlook and conclusion

The system also contains the blocks for trends storing and decision making as well. Its main purpose is to create the historical database for later comparison with the current values and to check the real rise of the particular value. Specific type of reduction of the stored data is applied; its details are described in the I-MECH deliverable D5.4 [7].

Currently, the major part of the predictive algorithms is processed in the LabVIEW running on the cRIO platform (which is together with SVS considered as Layer 1 device). The connection with the upper layer (Layer 2 device) is done through EtherCAT bus. The device in Layer 2 is currently considered only as an interface translator (EtherCAT to OPC UA) and the device in Layer 3 is showing the resulting parameters to the user.

In the future, most of the algorithms will be implemented into the sensor's hardware and it will provide the user (or the supervised system) with the calculated condition indicators instead of the raw vibration data stream.

3.6 Park's vector pattern module

The Park's vector pattern module is designed for processing the process data and system diagnostics on Layer 3. The module contains several functions, which include the conversion of process data into a form suitable for further processing, Park's vector calculation, averaging of the Park's vector and calculation of Park's vector pattern, total error calculation and prediction of the development of the total error.

3.6.1 General description

The Park's vector method is based on the calculation of complex vectors of the measured currents in $\alpha\beta$- coordinates. The phase currents should be measured in a steady state at least during one mechanical revolution of the rotor. Subsequently, the phase currents are transformed into $\alpha\beta$ - coordinates. Where i_α and i_β represent real and imaginary components of the Park's complex current vector. The details can be found in the I-MECH deliverable D5.4 [7].

3.6.2 Implementation in MATLAB

The function convert_to_ialphabeta is used to convert the process data to a suitable data structure for calculation of the Park's vector patterns.

- The user can set the conditions by parameters of this function under which the Park's vectors are calculated. The input signal mode flag (input_mode) indicates one of the possible input process data configurations. This parameter can range from 0 to 2. If one option is selected, e.g. 0, only measured phase currents enter the function. In this case, the other input parameters marked as (option) are not be used.
- The user selects a suitable area (the defined operation condition of the system) for Park's vector calculation by setting parameters (th_max, th_min, th_size). The parameters (th_max, th_min) indicate the maximum and minimum size of the limits within the reference value of the iq- current must be presented. The parameter (th_size) represents the minimum size of the output vectors which is necessary to calculate Park's vector pattern.
- The outputs of the function are vectors (ialpha, ibeta) in $\alpha\beta$ - coordinates that represent the real and imaginary part of the Park's current vector. The output vectors are suitable for further processing and the calculation of the Park's vector patterns.

Table 11. Function for the conversion of the process data.

Convert process data	
This function is used to convert process data to a suitable data structure for the calculation of the Park's vector patterns.	
Syntax: [ialpha, ibeta, real_time] = convert_to_ialphabeta(input_mode, ia_val, ib_val, ic_val, id_val, iq_val, pos_act, ialpha_val, ibeta_val, iset_val, th_max, th_min, th_size, vel_dem, real_time_val)	
Inputs, parameters and constants:	
Parameter	**Description**
input_mode	Input signals mode flag – 0 – phase current (ia_val, ib_val, ic_val) or only two current (ia_val, ib_val) third is computed. – 1 – for current in dq-coordinates (id_val, iq_val) and measured electric rotor position (pos_act). – 2 – for current in ab-coordinates (ialpha_val, ibeta_val).
ia_val	Vector of the measured phase current ia [A], *input_mode = 0*
ib_val	Vector of the measured phase current ib [A], *input_mode = 0*
ic_val	(optional) Vector of the measured phase current ic [A]. If not used, it is calculated according to the equation ia + ib +ic = 0. *input_mode = 0*
id_val	(optional) Vector of the current id in dq-axis coordinates [A], *input_mode = 1*
iq_val	(optional) Vector of the current iq in dq-axis coordinates [A], *input_mode = 1*
pos_act	(optional) Vector of the measured electric rotor position [rad], *input_mode = 1*
ialpha_val	(optional) Vector of the current ialpha in alphabeta-axis coordinates [A], *input_mode = 2*
ibeta_val	(optional) Vector of the current ibeta in alphabeta-axis coordinates [A], *input_mode = 2*
th_max	The maximum of the threshold of the iq- current reference.
th_min	The minimum of the threshold of the iq- current reference.
th_size	The size of the measured process data pack under the defined operation condition of the system and the minimum size of the output vectors.
iset_val	Vector of the demanded value of the current iq [A]
vel_dem	Vector of the demanded motor velocity [rad/s]
real_time_val	Real-time vector, time must be in [ms] or the MATLAB defined format
Outputs	
Parameter	**Description**
ialpha	Vector of the calculated current ialpha, the real part of the Park's current vector
ibeta	Vector of the calculated current ibeta, the imaginary part of the Park's current vector
real_time	Output real time vector

The function parks_pattern_average is used to calculate the Park's vector pattern.

- Input vectors *ialpha* and *ibeta* represents the real and imaginary part of the complex Park's current vector under user-defined operation condition of the system.

- The variable angle_step defines the step of the angle range at which module of the individual Park's vectors will be averaged. This angle corresponds to the electrical position. The averaging starts at 0 degrees and ends at 360 degrees. The minimum size of the averaging range is 1 degree.
- The output of the function is two vectors that represent the real and imaginary part of the resulting Park's vector pattern. The length of the vectors is given by the ratio 360/*angle_step*. The time stamp corresponding to the first sample from the input vector *real_time.*

Table 12. Park's vector patterns.

Park's vector patterns	
Syntax: [ialpha_sum, ibeta_sum, time_stamp] = pakrs_pattrern_average(ialpha, ibeta, angle_step, real_time)	
Inputs, parameters and constants	
Parameter	**Description**
ialpha	Input vector of the calculated current ialpha, the real part of the Park's current vector.
ibeta	Input vector of the calculated current ibeta, the imaginary part of the Park's current vector.
angle_step	Angle range for averaging individual Park's vectors in degrees (range 0-360 deg).
real_time	Real-time vector corresponding to the input current vector (MATLAB defined format).
Outputs	
Parameter	**Description**
lalpha_sum	The real part of the Park's current vector pattern. The averaging over the defined degree range by parameter angle_step.
ibeta_sum	The imaginary part of the Park's current vector pattern. The averaging over the defined degree range by parameter angle_step.
time_stamp	The time stamp corresponding to the first sample from the input vector *real_time.*

The function *pakrs_pattrern_error* is used to calculate the error between actual averaged Park's vector pattern and the correct Park's current vector pattern which is archived for given operating conditions.

- Matrix of the correct Park's current vector patter containing two column vectors (*ialpha, ibeta*). The real and imaginary parts must be averaged over the same degree range by parameter *angle_step* as current Park's vector pattern.
- The current vectors (*ialpha_sum, ibeta_sum*) are obtained as output from function *pakrs_pattrern_average* under same operation condition as correct Park's current vector pattern.
- The parameter *crit* is used to select the criterion quadratic or average for calculation of the Park's vector pattern error.
- The *pakrs_pattrern_error* function output is the sum of the errors of the individual Park's vector patterns.

Table 13. Total Park's vector pattern error calculation.

Total Park's vector pattern error calculation
Syntax: [parks_error] = pakrs_pattrern_error(pattern, ialpha_sum, ibeta_sum, crit, time_stamp)
Inputs, parameters and constants

Parameter	Description
pattern	Matrix containing two column vectors (*ialpha, ibeta*) of the correct Park's current vector pattern. The real and imaginary parts must be averaged over the same degree range by parameter angle_step as current Park's vector pattern.
ialpha_sum	The real part of the Park's current vector pattern.
ibeta_sum	The imaginary part of the Park's current vector pattern.
crit	The mode flag for criterion that can be used. Criterion – 0 – quadratic, – 1– average.
time_stamp	The time stamp of the Park's vector pattern (MATLAB defined format).
Outputs	
Parameter	**Description**
parks_error	The sum of the errors of the individual Park's vector patterns.

The function *pakrs_pattrern_error_extrapolation* is used to extrapolation the error.

- The input vector *parks_error* contains the total Park's vector pattern errors and input vector *x_vector* contains the corresponding values (e. g. time vector with timestamps), *x_vector (parks_error)*. Vector *exrap_points* contains the coordinates of the query points. The input vectors *parks_error* and *x_vector* must be the same length.
- The function returns extrapolated points. The length of output vector depending on the *exrap_points.*

Table 14. Extrapolation of the total Park's vector pattern error.

Extrapolation of the total Park's vector pattern error	
Syntax: [extrap] = pakrs_pattrern_error_extrapolation(parks_error, x_vector,exrap_points)	
Inputs, parameters and constants	
Parameter	**Description**
parks_error	The input vector of the Total Park's vector pattern errors.
x_vector	The timestamps vector for the Park's vector pattern errors vector (MATLAB defined format of the timestamps).
exrap_points	The vector of the coordinates of the query points.
Outputs	
Parameter	**Description**
extrap	The output vector of the extrapolation value of the error.

3.6.3 Validation in MIL

Verification of the Park's vector pattern approach was performed in MIL. The model of the Pilot 5 test bench was created in MATLAB/Simulink using the library Simscape. The pilot 5 test bench parameters from Philips have been implemented in the model. The Simscape model that is controlled by the proposed vector control algorithm is shown in Figure 21.

This model allows to simulate mechanical faults:

- Bearing fault
- Deformation of the gearbox teeth

The Park's vector pattern approach was tested with a mechanical fault in the system, which is represented by a change in PMSM friction. Subsequently, the measured data is analysed and the indicator KPI for this mechanical fault is calculated.

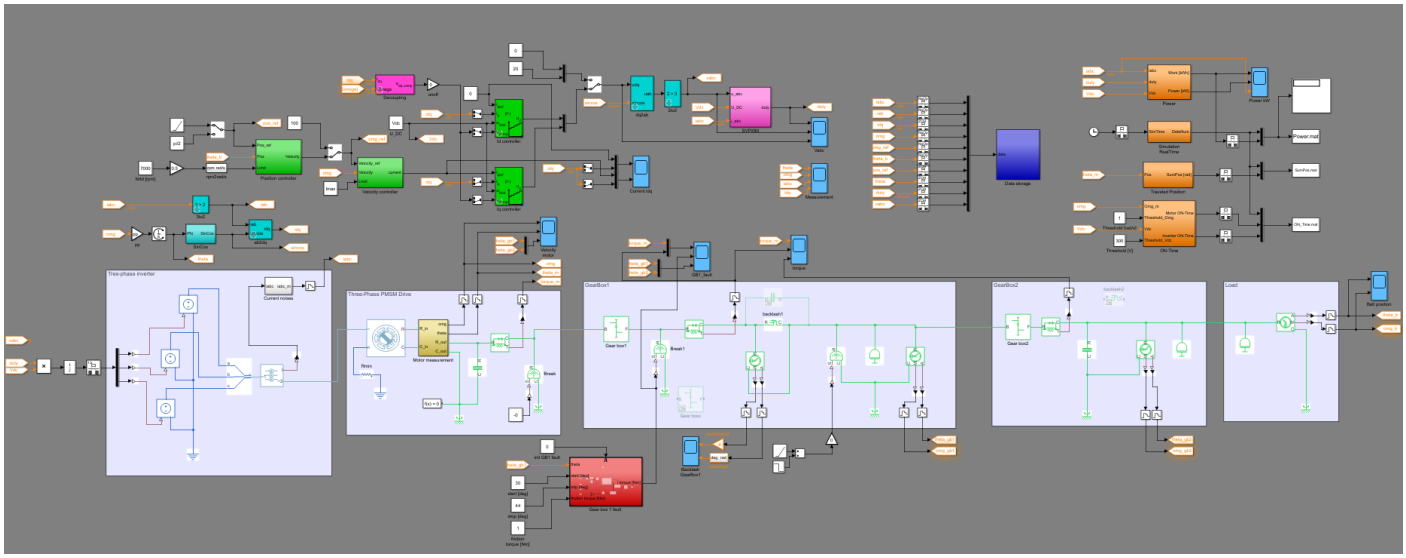

Figure 21. MATLAB/Simulink Simscape model of the Pilot 5 test bench.

3.6.3.1 Validation of the Park's vector pattern approach - Bearing friction fault

The Park's vector pattern method was verified by detection of the change of parameters Coulomb friction torque and Breakaway friction torque in block Rotational Friction. This fault represents a change in bearings friction of the PMSM, which may be caused by the worn bearing.

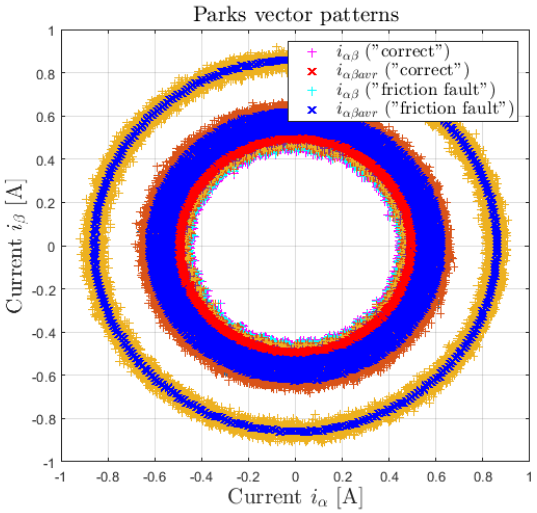

Figure 22. Park's current vector and Park's vector pattern.

Change the parameters in block Rotational Friction:

Coulomb friction torque - Cfr = [0.050 0.052 0.055 0.058 0.064 0.071 0.077 0.085 0.095 0.18]; [Nm]

Breakaway friction torque - Bfr = [0.050 0.052 0.055 0.058 0.064 0.071 0.077 0.085 0.095 0.18]; [Nm]

The first value in the vectors *Cfr* and *Bfr* represent the correct state value of the bearing friction. Other friction values in vectors *Cfr* and *Bfr* represent gradual wear of the bearing over time. The measured process data is processed. The phase currents under defined operating conditions are transformed to αβ- coordination and individually Park's current vectors are calculated. The Park's vectors for each system state (one correct state and nine fault states) are shown in Figure 22. Subsequently, the function *parks_pattern_average* is used to calculate the Park's vector pattern for each state of the system. The parameter *angle_step* is set to a minimum size of the averaging range is 1 degree. The output Park's vector patterns are shown in Figure 22. The vectors marking in Figure 22 are as follows: Symbol "+" indicates the endpoint of the Park's current vectors and symbol "x" indicates the endpoint of the vectors from Park's vector pattern. The red color is used for the correct Park's vector pattern and blue color is used for Park's vector patterns with fault.

Figure 23 shows Park's vector patterns depending on increasing coulomb and breakaway friction. The function *pakrs_pattrern_error* is used to calculate the errors between correct Park's vector pattern and the faults Park's vector patterns. The output of the function is the total error that is the sum of the errors of the individual Park's vector patterns. The courses of the total errors for two criteria (quadratic, average) are shown in Figure 24. Subsequently, the function *pakrs_pattrern_error_extrapolation* is used to extrapolation of the following three points as is shown in Figure 24.

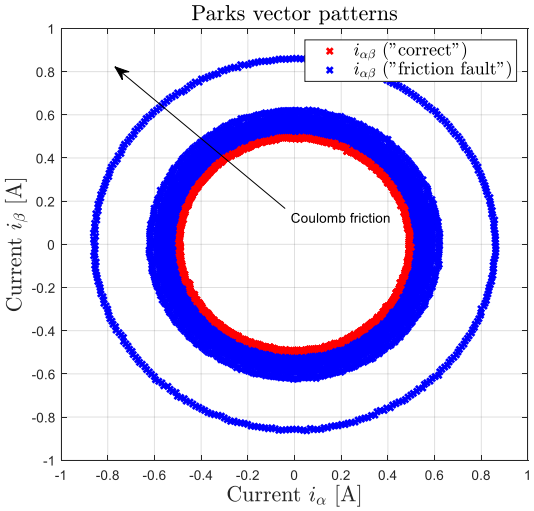

Figure 23: Park's vector pattern for correct and fault states of the system.

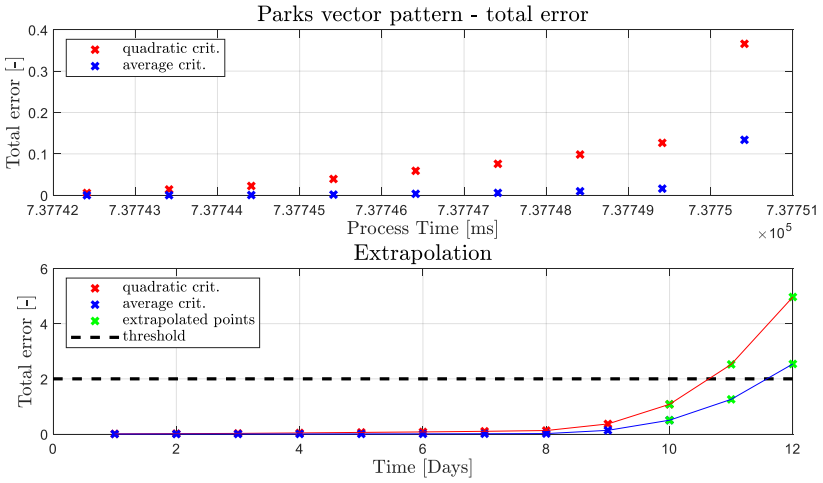

Figure 24. Extrapolation of the Park's vector pattern total errors.

3.7 Implementation aspects

3.7.1 Pilot 5

Validation of Parks vector pattern method was done by using experimental data acquired from Pilot 5 provided by Philips. These data were measured either on a real system or on a HIL simulator. From the perspective of implementation, the algorithms realized in MATLAB or in Simulink will be used either directly or as the automatically generated C or C++ code. The utilization of BB3 components is only possible in Layer 3.

3.7.2 Use Case 1.1

The i2t condition indicator and speed ramp trigger has been implemented in to UC1.1. MATLAB blocks of these components has been translated to Structured text using MATLAB PLC Coder. The final functions have been integrated into Gefran ADV 200 drive PLC program.

The Smart Vibration Sensor has been integrated to the testbench as well. EtherCAT connection has been established and complete communication framework has been tested between Layer 1 EtherCAT slaves (SVS and ADV200 drive) and Layer 2 EtherCAT master (demonstrated on CompactRIO system).

The validation tests have been done in the BUT laboratory dynamometer. The test bench setup is shown in the Figure 25.

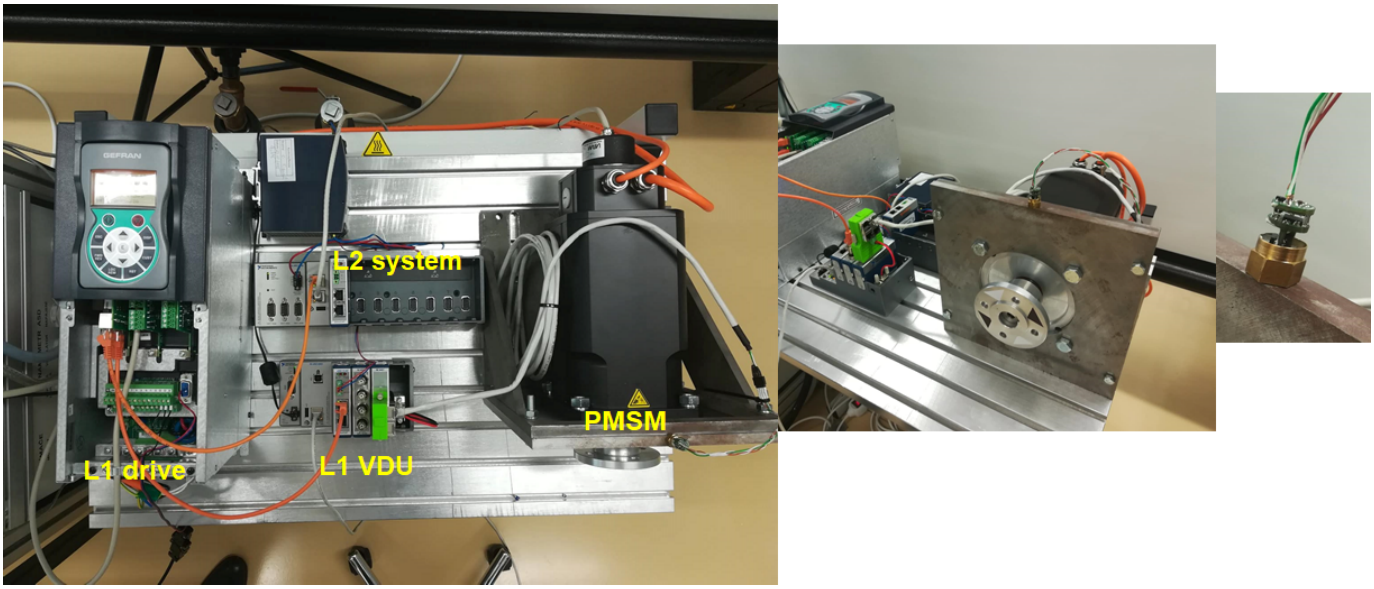

Figure 25. BB3 components integration into UC1.1 testbench at BUT site.

3.7.3 Demonstrator 1

As part of Deliverable 7.7 Demonstrators and Use Cases in industrial relevant environments, J&J Vision Care, with partners TNI and ITML, have implemented a condition monitoring algorithm in the Layer 3 Microsoft Azure cloud services platform. This BB3 element is supported by inputs from wireless sensors with edge computing capability developed as part of BB1 and BB2.

Demonstrator 1 incorporates BB3 condition monitoring to monitor and ultimately predict the health of the material transfer layer (Magnemotion Quickstick Linear Synchronous Motor (LSM)) used in the manufacturing of contact lenses, the product carrier/pallet and the product itself (contact lenses).
As per the functional requirements listed in D7.1 for the J&J Vision Care material transfer layer, the following parameters were identified as critical for monitoring the condition of the lens transfer system, Magnemotion Quickstick LSM:

- Vibration of the carrier/pallet,
- Magnetic field strength of the LSM,
- Temperature of the lens curing tunnel,
- Light intensity in the lens curing tunnel.

The condition monitoring algorithm, developed in partnership with ITML, is deployed with the Microsoft Azure cloud services platform. The parametric data from the wireless sensors deployed on the lens carriers and the Magnemotion Quickstick PLC used by the condition monitoring algorithm in Azure is communicated via parallel paths, as seen in Figure 26.
Sensor data from the lens carrying pallet is communicated from Layer 1 to Layer 3 directly via an edge gateway with a secure J&J approved image to securely transfer the data in an efficient manner. The PLC data follows OPC UA /Automation ML protocols to communicate data from Layer 2 to Layer 3 through the J&J firewall, streaming the data to Azure storage for use in contextualising the wireless sensor data and input to the condition monitoring algorithm developed as part of BB3.

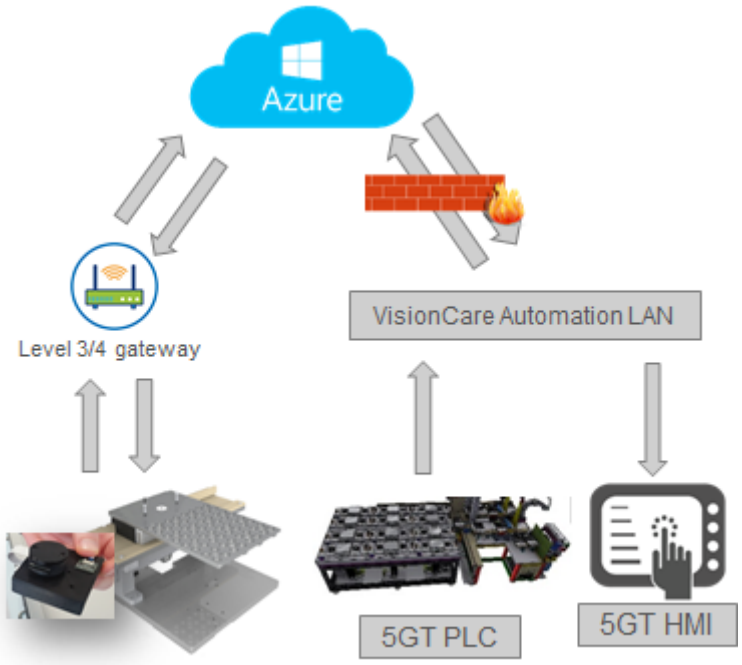

Figure 26. Sensor and PLC data communication to Layer 3 cloud services.

4 BB6 - Automatic commissioning of motion control system

Building Block 6 (BB6) is related to the self-commissioning of the velocity and position control loops of a mechatronic system. BB6 can be decomposed in 4 functional blocks located in layer 2 and layer 3 of the I-MECH - Figure 27.

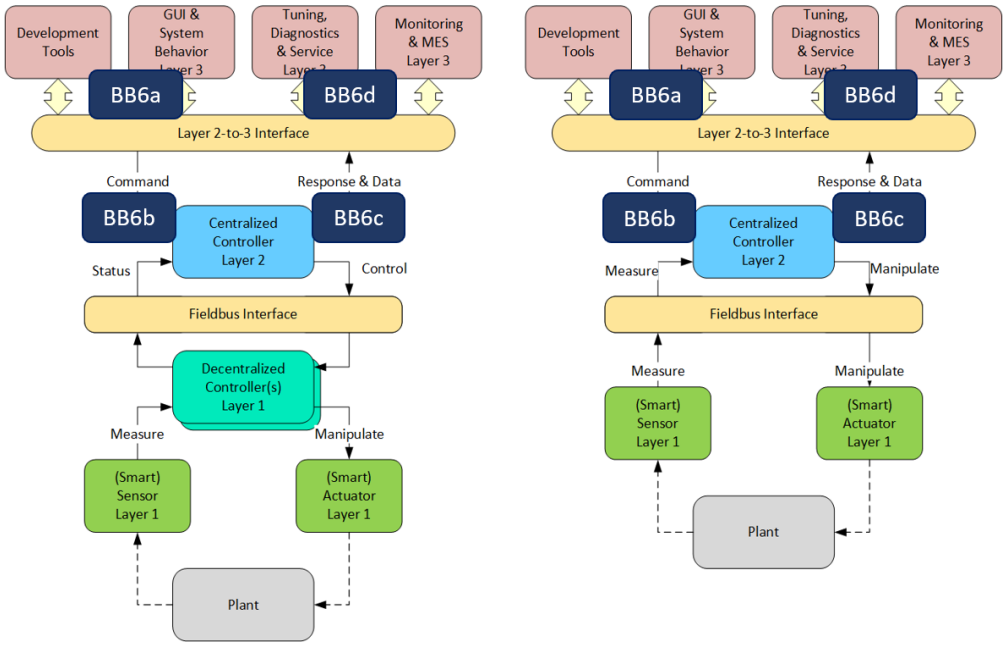

Figure 27: BB6 decomposition into the I-MECH structure.

The four functional blocks have been defined as follows (see Deliverable 5.3 [6] for further details):

- **BB6a** is the automatic controller commissioning interface (Layer 3);
- **BB6b** is the trajectory manager block (Layer 2);
- **BB6c** is the data acquisition module (Layer 2);
- **BB6d** is the system identification and tuning manager module (Layer 3).

Two different approaches have been adopted from the developer partners (UniBS/GEF and ZAPUNI) in order to split the work.

4.1 UniBS/GEF Approach Functionalities

The BB6 approach proposed by UNIBS consists of a series of procedures to determine the values of the controller parameters in order to obtain the required performance of a mechatronic system.

The methodology is based on a system identification strategy in the frequency domain. The data used for the system identification comes from an excitation trajectory computed in order to respect some predefined constraints on position, velocity and torque. Once the Frequency Response Function (FRF) coming from the system identification procedure is obtained, the transfer function is estimated by minimizing the error between the identified FRF and the estimated transfer function.

The tuning of the controller parameters is then performed based on the transfer function estimation. To tune the controller parameters a trade-off between performance and robustness is considered. The approach is flexible because it allows to consider only the velocity control, only the position control or the cascade position/velocity control structure. In case resonances and/or anti-resonances are present in the transfer function, biquadratic filters are used to remove

their effect from the system output. The use of biquadratic filters is therefore used to reduce the oscillations of elastic systems.

The functionalities developed from UniBS/GEF to create BB6 are shown hereafter.

4.1.1 BB6a - automatic controller commissioning interface

4.1.1.1 General description - algorithm theory

The automatic controller commissioning interface represents the information exchange point between the BB6 and the user. Inside the interface the user can find the information to set in order to set the automatic tuning preferences.

4.1.1.2 Implementation

An example of automatic controller commissioning interface implemented in MATLAB/Simulink has been tested on the GEF HIL setup at UNIBS that is built with the same drive as the Use Case 1.1.

In Figure 28 to Figure 31, examples of the BB6a automatic controller commissioning interface built in collaboration with GEF for the Use Case 1.1 are shown.

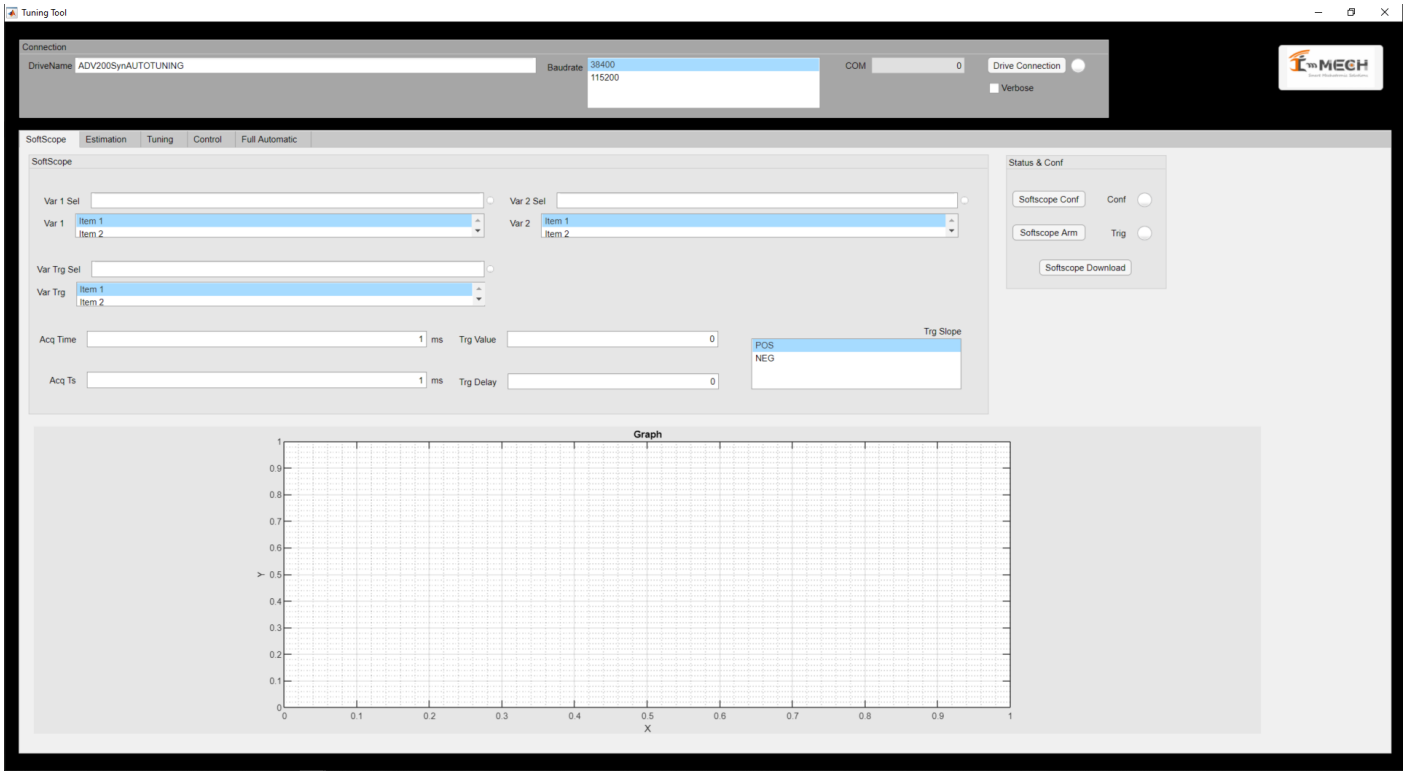

Figure 28: Example of BB6a automatic controller commissioning interface created by UNIBS/GEF.

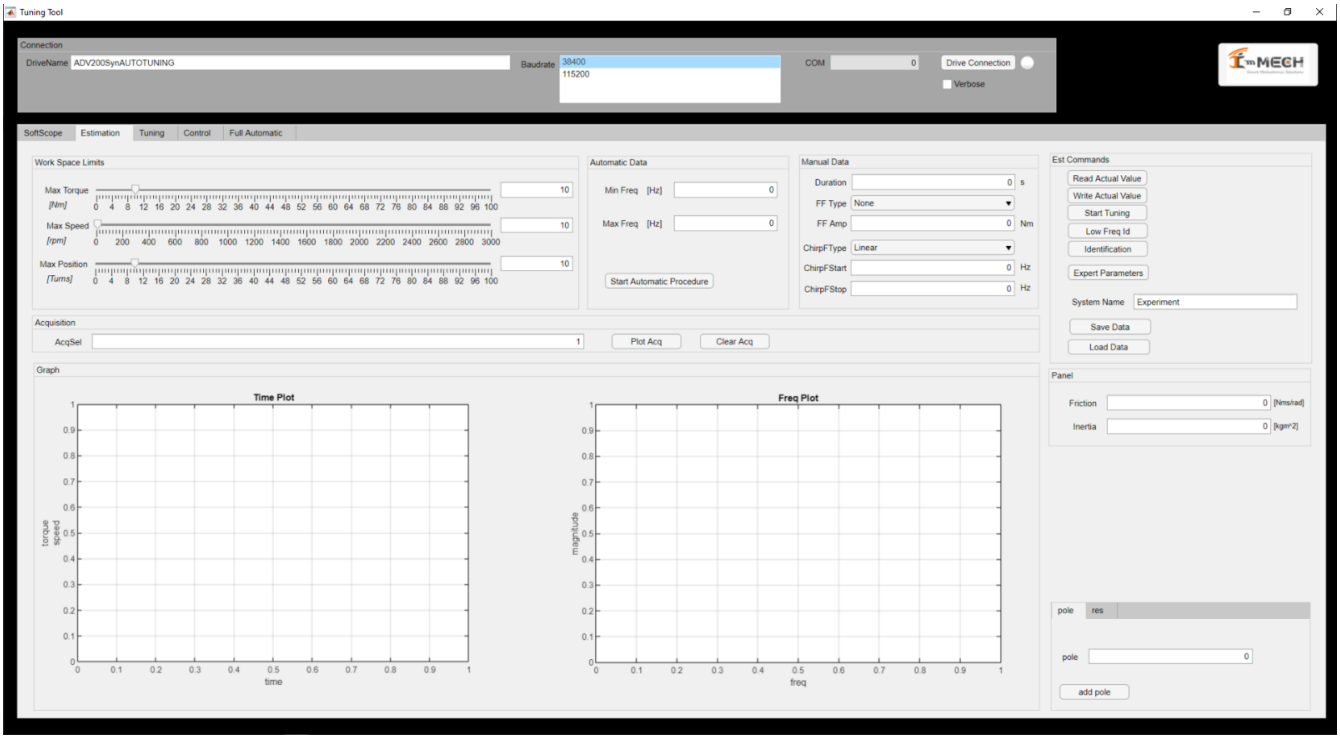

Figure 29: Example of BB6a automatic controller commissioning interface created by UNIBS/GEF.

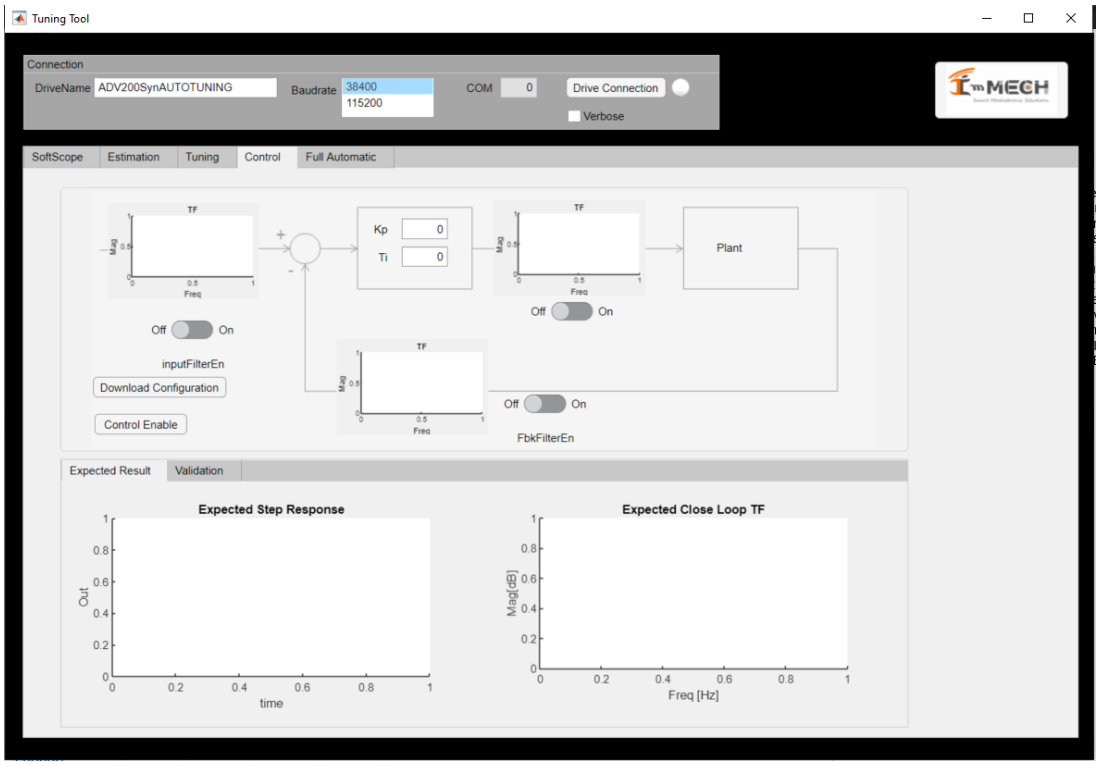

Figure 30: Example of BB6a automatic controller commissioning interface created by UNIBS/GEF.

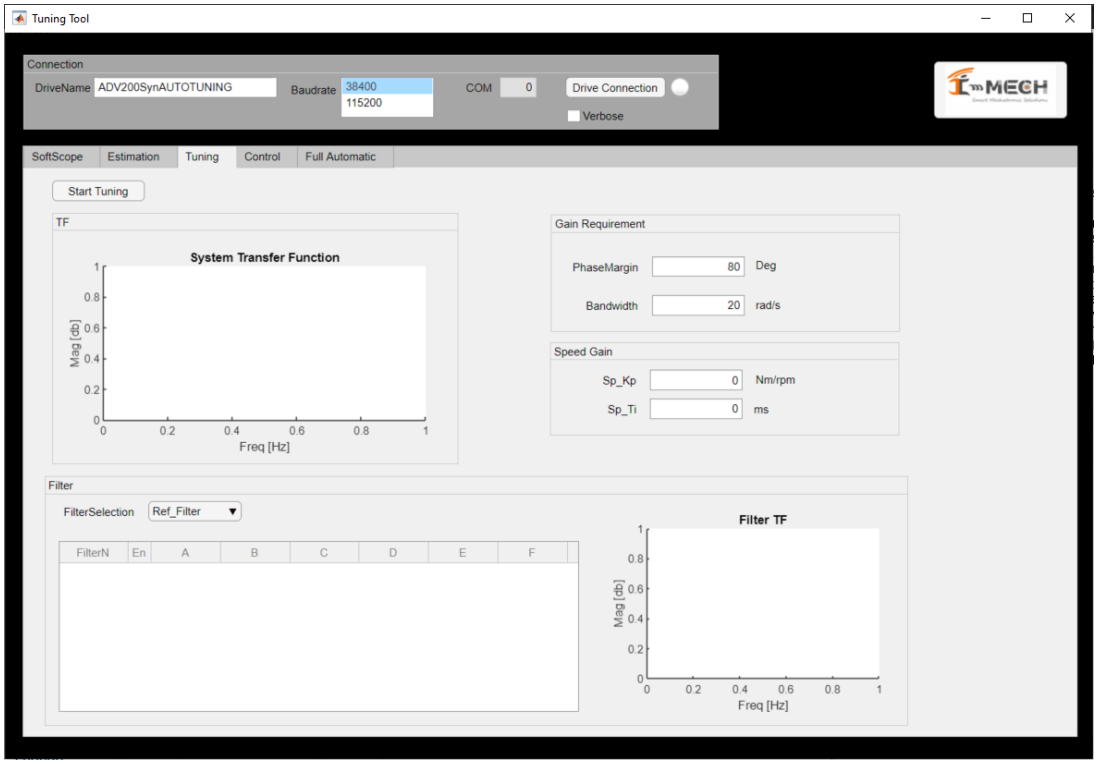

Figure 31: Example of BB6a automatic controller commissioning interface created by UNIBS/GEF.

4.1.1.3 Input output parameters, constants

The information that the standard user can set and modify are shown in Table 15.

Table 15: Automatic controller commissioning interface.

Automatic controller commissioning interface	
Inputs, parameters and constants	
Parameter	**Description**
START	Autotuning procedure enable
STOP	Autotuning procedure stop
AT_TYPE	Autotuning type (open-loop or closed-loop)
POS_LOOP	Desired position loop enable
VEL_LOOP	Desired velocity loop enable
Trj_type	Set point trajectory type
Trj_ff_type	Feedforward type
max_pos	Maximum reachable position
max_vel	Maximum reachable velocity
max_trq	Maximum reachable torque/force

ff_amplitude	Feedforward torque/force amplitude value

It is important to notice that the parameters shown in Table 15 represent a minimum set of possible settable BB6 parameters. In fact, depending on the plant where BB6 has to be used, the automatic controller commissioning interface can change.

It is therefore suggested that the automatic controller commissioning interface has to be developed from the owner of the mechatronic system.

4.1.1.4 Validation in MIL and HIL

The validation of the minimum set of automatic controller commissioning interface configurable parameters has been done in MIL (MATLAB/Simulink) and in HIL (Use Case 1.1).

4.1.1.5 Location of files

The BB6a automatic controller commissioning interface example developed in the MATLAB/Simulink environment for the MIL test is available.

4.1.2 BB6b - trajectory manager block

4.1.2.1 General description - algorithm theory

The trajectory manager block consists in the set-point and feedforward generation for the system identification procedure.

Different set-points and feedforward generators have been developed and implemented in order to guarantee the best excitation and the respect of the imposed physical constraints (maximum torque/force, maximum position and maximum velocity).

4.1.2.2 Implementation

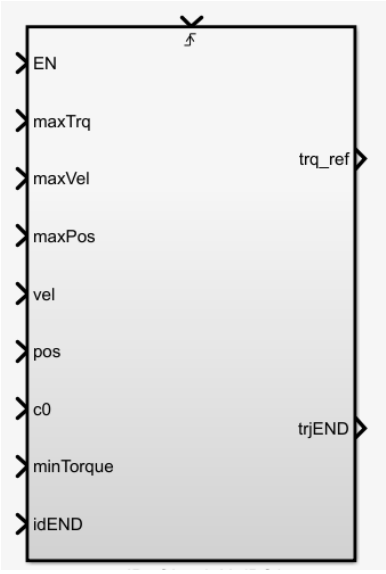

Figure 32: UniBS open loop trajectory generator block.

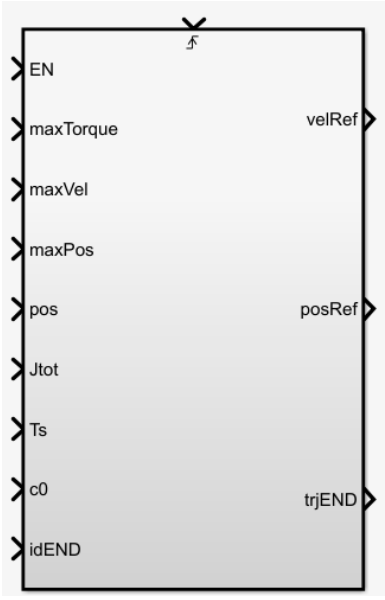

Figure 33: UniBS closed loop trajectory generator block.

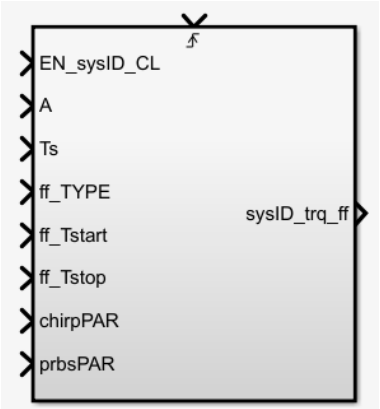

Figure 34: UniBS feedforward generator block.

The implementation of the trajectory manager block has been done in Simulink. Three separated blocks can be used depending on the considered autotuning type (open-loop or closed-loop).

If the autotuning type is set as open-loop, just the open-loop trajectory generator block can be used. In the framework of the I-MECH project, the UniBS open loop trajectory generator block Figure 32 has been developed in order to guarantee the respect of the imposed position, velocity and force/torque constraints.

If the autotuning type is set as closed-loop, the closed-loop trajectory generator block can be used in combination with the feedforward generator block. In the framework of the I-MECH project, the UniBS closed loop trajectory generator block (Figure 33) has been developed in order to guarantee the respect of the imposed position, velocity and force/torque constraints, while the feedforward generator block has been implemented to provide the standard feedforward signals for the system identification such as chirp, noise or PRBS (Figure 34).

An example of the use of the trajectory generator blocks is shown in Figure 35. As it is possible to see, the inputs, outputs and parameters of the trajectory generator block consists of a subset of the ones described in Section 4.1.3.3.

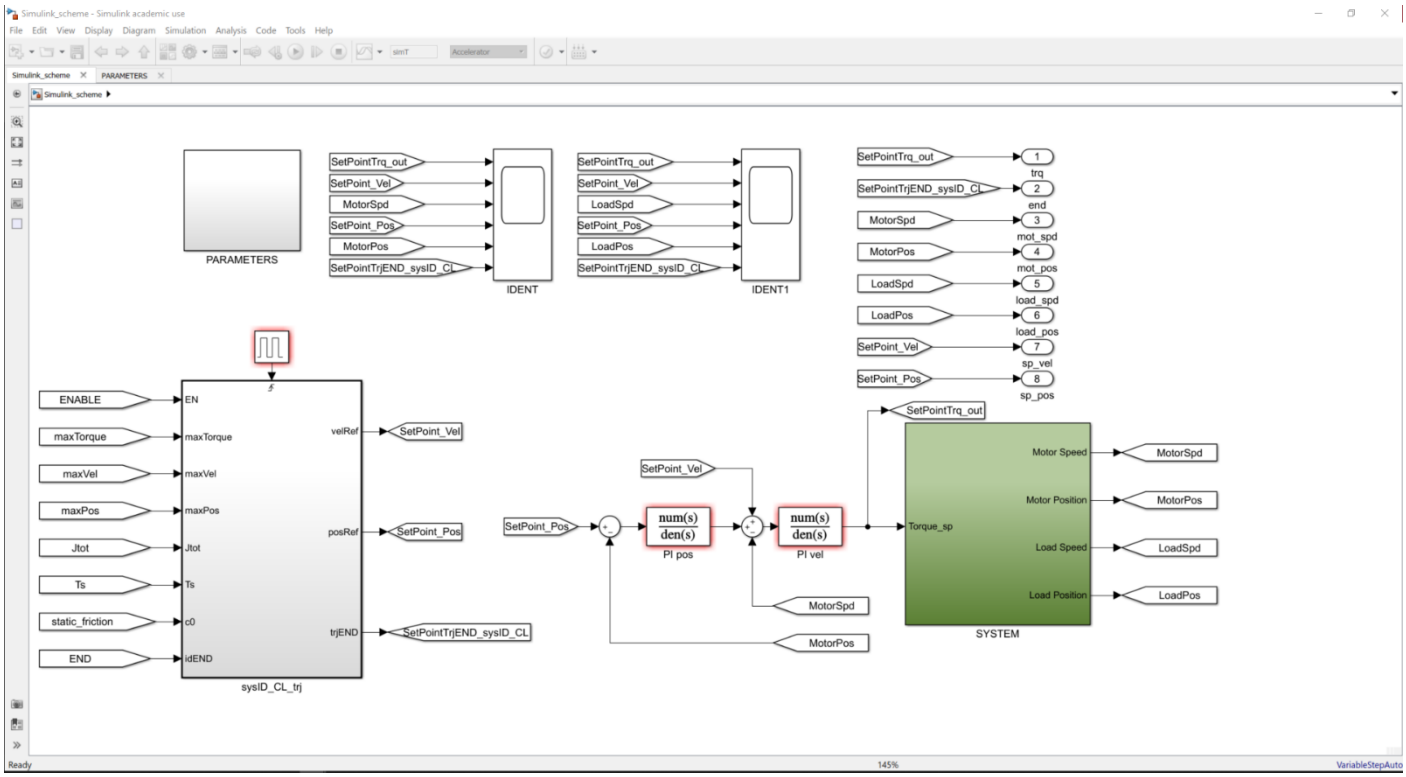

Figure 35: Example of the use of the CL trajectory generator made by UNIBS/GEF.

4.1.2.3 Input output parameters, constants

Input, output, parameters and constants of the trajectory generator modules proposed before are shown in Table 16 to Table 18.

Table 16: UniBS open loop trajectory generator block.

UniBS open loop trajectory generator block	
Inputs, parameters and constants	
Parameter	**Description**

ENABLE	Trajectory generation enable
Max_trq	Maximum torque/force
Max_vel	Maximum velocity
Max_pos	Maximum position
Vel	Actual velocity
Pos	Actual position
c0	Estimated static friction coefficient
Min_trq	Minimum torque
END	STOP
Outputs	
Parameter	**Description**
Trq_ref	Torque/force reference
END	STOP

Table 17: UniBS closed loop trajectory generator block.

UniBS closed loop trajectory generator block	
Inputs, parameters and constants	
Parameter	**Description**
ENABLE	Autotuning procedure enable
Max_trq	Maximum torque/force
Max_vel	Maximum velocity
Max_pos	Maximum position
Pos	Actual position
Jtot	Estimated total inertia
Ts	Sampling time
c0	Estimated static friction coefficient
END	STOP
Outputs	
Parameter	**Description**
Vel_ref	Velocity reference
Pos_ref	Position reference
END	STOP

Table 18: UniBS feedforward generator block

UniBS feedforward generator block	
Inputs, parameters and constants	
Parameter	**Description**
ENABLE	Autotuning procedure enable
A	Feedforward amplitude
Ts	Sampling time
FF_type	Feedforward type (Noise, Chirp, PRBS)
FF_Tstart	Starting feedforward time
FF_Tstop	Stop feedforward time
Chirp_PAR	Chirp parameters vector
PRBS_PAR	PRBS parameters vector
Outputs	
Parameter	**Description**
Trq_ff	Torque/force feedforward

4.1.2.4 Validation in MIL and HIL

The validation of the trajectory generator modules has been done in MIL (MATLAB/Simulink) and in HIL written in IEC61131.3 ST standard language (Use Case 1.1).

4.1.3 BB6c - data acquisition module

4.1.3.1 General description - algorithm theory

The data acquisition module consists in a memory area where the data can be stored during the trajectory generation.

4.1.3.2 Implementation

No implementation has been done in this case. In fact, the data acquisition module is strictly dependent on the Layer 2 specific products used to control the system. Each product has its own memory structure, it is important that the stored data are made available, but the way to access to the specific data is not a topic of BB6.

4.1.3.3 Input output parameters, constants

The minimum set of input, outputs and parameters to store each sampling period to perform a system identification procedure are shown in Table 19.

Table 19: UniBS Store/acquire.

UniBS Store/acquire	
Inputs, outputs, parameters and constants to store	
Parameter	**Description**

Trq	Input force/torque to the system
Vel	Actual velocity
Pos	Actual position

4.1.3.4 Validation in MIL

The MIL validation of data acquisition module was not necessary because MATLAB/Simulink provides a build in data acquisition process.

4.1.4 BB6d - system identification and tuning manager module

4.1.4.1 General description - algorithm theory

The system identification and tuning manager module is composed of 3 different parts:

- The FRF estimation unit,
- The system identification unit,
- The tuning manager unit.

The FRF estimation unit permits the estimation of the Frequency Response Function (FRF) of the system by using the H_v method proposed in [3]. For better FRF estimation from the system input it is suggested to remove the static friction and the weight components that are usually nonlinear. While from the output position it is suggested to consider that the system motion starts at position equal to zero. It is important to remark that the FRF estimation method works in both cases of measured output position or measured output velocity in the same way.

The system identification unit provides the estimation of the analytical formulation of the Transfer Function (TF) of the system. The system identification unit is able to change the TF formulation depending on the feedback side used to control the system (it means on where the output sensor is mounted), motor-side feedback or load-side feedback.

The tuning manager unit allows the tuning of the position PID controller, the velocity PID controller and up to three BI-QUADRATIC filters able to remove the effects of possible resonances and/or anti-resonances.

In case just PI controllers are needed, or if the Maximum Sensitivity value is not required explicitly, pole placement strategies are used to tune the controller parameters. On the contrary, if the Maximum Sensitivity value is explicitly required (together with the cut-off frequency) a minimization strategy is used to compute the PID controller parameters to obtain the best possible trade-off between these competing objectives [1,2].

4.1.4.2 Implementation

An example on how to use the FRF estimation unit, the system identification unit and the tuning manager unit can be found in the function help.

Further details on inputs, outputs and parameters of the FRF estimation unit, the system identification unit and the tuning manager unit can be found in Section 4.1.4.3.

4.1.4.3 Input output parameters, constants

The minimum set of input and output, parameters and constants of the system identification and tuning manager module is shown in Table 20, Table 21 and

Table 22.

Please, refer to the help of each function for further details.

Table 20: FRF estimation unit.

FRF estimation unit	
Inputs	
Parameter	**Description**
fcn_in.input	System torque/force input vector
fcn_in.output	System velocity or position output vector
fcn_in.sampling_time	Acquisition sampling time
Parameters	
Parameter	**Description**
fcn_par	Structure that defines the parameters (please, refer to the function help)
Outputs	
Parameter	**Description**
fcn_out.amplitude	FRF amplitude
fcn_out.amplitude_dB	FRF amplitude in dB
fcn_out.phase_deg	FRF phase in degrees
fcn_out.freq_rads	FRF frequencies (in rad/s) where amplitude and phase has been computed

Table 21: System Identification unit.

System Identification unit	
Inputs	
Parameter	**Description**
fcn_in.amplitude	FRF amplitude
fcn_in.amplitude_dB	FRF amplitude in dB
fcn_in.phase_deg	FRF phase in degrees
fcn_in.freq_rads	FRF frequencies (in rad/s) where amplitude and phase has been computed
Parameters	
Parameter	**Description**
fcn_par	Structure that defines the parameters (please, refer to the function help)
Outputs	
Parameter	**Description**
fcn_out.sys_vel_num	numerator of the torque/force-velocity TF
fcn_out.sys_vel_den	denominator of the torque/force-velocity TF
fcn_out.sys_vel_est	TF of the torque/force-velocity

fcn_out.sys_pos_num	numerator of the torque/force-position TF
fcn_out.sys_pos_den	denominator of the torque/force-position TF
fcn_out.sys_pos_est	TF of the torque/force-position

Table 22: Tuning Manager unit.

Tuning Manager unit	
Inputs	
Parameter	**Description**
fcn_in.phi_m_v	Desired phase margin for the velocity loop
fcn_in.Wc_v	Desired bandwidth for the velocity loop
fcn_in.phi_m_p	Desired phase margin for the position loop
fcn_in.Wc_p	Desired bandwidth for the position loop
Parameters	
Parameter	**Description**
fcn_par	Structure that defines the parameters (please, refer to the function help)
Outputs	
Parameter	**Description**
fcn_out.PIDv_par	Structure containing the computed velocity loop PID parameters
fcn_out.PIDp_par	Structure containing the computed position loop PID parameters
fcn_out.BQ1_par	Structure containing the computed first biquadratic filter parameters
fcn_out.BQ2_par	Structure containing the computed second biquadratic filter parameters
fcn_out.BQ3_par	Structure containing the computed third biquadratic filter parameters
fcn_out.BQv_par	Structure containing the computed biquadratic filter parameters on the output velocity
fcn_out.BQp_par	Structure containing the computed biquadratic filter parameters on the output position
fcn_out. obtained_phi_m_v	Obtained phase margin for the velocity loop
fcn_out. obtained_Wc_v	Obtained bandwidth for the velocity loop
fcn_out. obtained_phi_m_p	Obtained phase margin for the position loop
fcn_out. obtained_Wc_p	Obtained bandwidth for the position loop

4.1.5 Validation in MIL

The validation of the system identification and tuning manager module has been done in MIL (on MATLAB/Simulink simulated data) and on HIL data coming from Use Case 1.1.

4.2 UniBS/GEF Implementation aspects

The implementation of the UNIBS/GEF approach of BB6 in Pilots and Use Cases will be described in WP6-WP7 deliverables. However, some aspects of the specific implementation are shown hereafter.

4.2.1 Pilot 1

In Pilot 1, the BB6a automatic controller commissioning interface has not been fully implemented. In fact, it has been **integrated** in the already existing GUI.

The BB6b trajectory generator block has been **added** to the existing trajectory generator block.

The BB6d system identification and tuning manager block has been **added** to the Pilot 1 Layer 3 functionalities.

4.2.2 Pilot 2

In Pilot 2 the BB6 integration will follow the Pilot 1 integration due to the fact that they share the same AxChange Architecture.

4.2.3 Pilot 5

In Pilot 5 the BB6a automatic controller commissioning interface has not been fully implemented. In fact, it has been **integrated** in the already existing GUI.

The BB6b trajectory generator block has not been used, in fact Pilot 5 has its own trajectory generator.

The BB6d system identification and tuning manager block has been **added** to the Pilot 5 Layer 3 functionalities.

4.2.4 Use Case 1.1

The implementation of the UNIBS/GEF approach of BB6 in Use Case 1.1 has been tested on the GEF HIL setup at UNIBS that uses the same control architecture as the Use Case 1.1.

In HIL at UNIBS the BB6a automatic controller commissioning interface has been fully developed and tested.

The BB6b trajectory generator block has been **added** to the existing trajectory generator blocks present in the Gefran ADV200s Drives (blocks have been converted in standard IEC61131.3 Structured Text language).

The BB6d system identification and tuning manager block has been **added** to the HIL Layer 3 functionalities.

4.3 ZAPUNI Functionalities

4.3.1 BB6a - automatic controller commissioning interface

The controller commissioning interface offers three basic ways of interaction with the user. The first one is a simple command-line access via a set of library functions responsible for the individual functionalities of excitation trajectory generators (BB6b), system identification and controller tuning routines (BB6d). The software algorithms are available in the form of compiled C-MEX functions which can be run in the MATLAB environment. This allows simple utilization of the SW by means of the envisioned I-MECH centre. The second option to interface the SW is the MATLAB GUI developed by UNIBS which may call the respective functions of the SW package. This part corresponds to the Layer 3 of the reference I-MECH structure. The third possible form of user interface is a stand-alone application which can be generated from the library functions using the MATLAB Compiler SDK. The resulting SW can run on a PC under Windows OS providing the same functionality as the remote connection via the I-MECH Portal which is described below. The standalone application is not supported in the current version of BB6 and is assumed to be a feature for further development beyond the scope of the I-MECH project.

The Layer 2 part of the BB6 involves the controller algorithm, functions for the excitation trajectory generation and data acquisition. Those parts are always vendor-specific and are assumed to be realized by user based on a particular SW and HW platform of the control system. The BB6 can provide the functionality of automatic C-code generation for the control algorithm execution, when needed. It uses the Simulink Coder by Mathworks to deliver this functionality with

the control structure corresponding to the BB6 reference architecture, basically the conventional PID cascade with additional shaping filters (see deliverable D5.3 [6] for detailed description).

The typical workflow for a remote user accessing the I-MECH Portal is assumed as follows:

- The user chooses appropriate **excitation trajectory** and invokes the Trajectory manager block **BB6b** which synthesizes the testing signal to be used for the identification of the plant model either under open- or closed-loop setup. Various parameters affect important characteristics of the testing signal such as its shape, bandwidth and power density in the relevant frequency band.
- The output of the generator is a testing sequence stored in a CSV file which can be downloaded from the *I-MECH Portal*. The **user employs the testing sequence** on his machine/setup and collects the relevant data for the system identification.
- The user uploads the data from the experiment and launches the **System identification and tuning** manager module **BB6d.**
- The **system identification part** delivers relevant plant model based on the experimental data. Validity of the model can be checked using the outputs of the identification methods (e.g. the uncertainty bounds on the model estimates), validation signal can also be provided by the user to check the model fidelity. If the model does not correspond well with the real system, some corrective means are suggested to take, usually by modifying experimental conditions and/or excitation signal. Multiple iterations may be needed before delivering a suitable model.
- The **controller tuning part** receives information about the plant model and uses it for the synthesis of the controller based on the formulated design requirements. This step results in a parameterized control structure corresponding to the BB6 reference scheme. Usually a whole set of the controllers is given allowing a fine-tuning on the real plant using a user-friendly parameter such as bandwidth and damping.
- The user either uses the **parameters** of the controllers in his own control system implementation or uses the code generation functionality to export the whole controller algorithm. He employs the proposed controller and validates the results.

4.3.2 BB6b - trajectory manager block

4.3.2.1 General description - algorithm theory

The trajectory manager block offers various ways of synthesis of proper excitation signals suitable for the task of plant model acquisition. Two types of wide-band excitation are supported: maximum length sequences (MLS) belonging to the class of pseudo-random binary signals and multi-harmonic excitation signals including the variants of random-phase (RPMS), Schroeder (SMS) and phase-optimal (POMS) signals. Each type of signal offers different possibilities for customization of its waveform and spectrum (see the I-MECH deliverable D5.3 [6] for more details).

4.3.2.2 Implementation

The implementation of the trajectory manager blocks was done using the MATLAB language. Library of C-MEX functions is available for MATLAB environment, export of the C(++) code is also possible when needed using the MATLAB Coder allowing to embed the generator code in an arbitrary target environment.

4.3.2.3 Input output parameters, constants

Table 23: Maximum length PRBS generator block.

Maximum length PRBS generator block	
Syntax: [y_gen,kTs,bw_act,it_act]=gen_prbs(bw,nbit,amp.nit,Ts)	
Inputs, parameters and constants	
Parameter	**Description**
bw	Desired bandwidth of the generated signal in Hzs

nbit	Number of bits in the PRBS sequence determining the overall length of the generated signal
amp	Maximum amplitude of the generated binary sequence
nit	Number of iterations for periodic excitations
Ts	Sampling period of the generator in seconds
Outputs	
Parameter	**Description**
y_gen	Generated excitation signal
kTs	Number of equal samples in the generated sequence
bw_act	Actual bandwidth of the excitation signal with respect to the actual chosen sampling period [Hz]
it_act	Number of actual iterations corresponding to output y_gen

Table 24: Wide-band multi-sine generator block.

Wide-band multi-sine generator block	
Syntax: [y_gen, it_act]=gen_msine(f0,fmin,fmax,ampu,nit,Ts,stype)	
Inputs, Parameters and Constants	
Parameter	**Description**
f0	Base frequency of the multi-sine sequence determining the lowest power line and at the same time the resolution of the excitation signal power spectrum
fmin	Lower bound of frequencies contained in the excitation signal [Hz]
fmax	Upper bound of frequencies contained in the excitation signal [Hz]
ampu	(optional) Vector of amplitudes at the frequency lines between fmin and fmax
nit	Number of iterations for periodic excitations
Ts	Sampling period of the generator in seconds
stype	Signal type determining phase distribution: 1-random phase multi-sine, 2 – crest factor optimized multi-sine, 3 – Schroeder multi-sine
Outputs	
Parameter	**Description**
y_gen	Generated excitation signal
it_act	Number of actual iterations corresponding to output y_gen

4.3.2.4 Validation in MIL and HIL

MIL validation done in the MATLAB/Simulink environment, HIL validation using ZAPUNI motion stage and use-cases UC1.3, 2.2.

4.3.3 BB6d - system identification and tuning manager module

4.3.3.1 General description - algorithm theory

The system identification module processes the provided experimental input/output data and employs a two-stage procedure for the derivation of mathematical model of the controlled plant. The sampled time domain data are first transformed to the frequency domain by computing an estimate of the nonparametric frequency response function model including uncertainty bounds. The FRF data are subsequently approximated by a best linear approximation in the form of transfer function model. Its structure can be explicitly chosen by the user or automatically derived from the data.

The derived plant model is used in the tuning manager module responsible for derivation of proper structure and parameters of the velocity/position controller. The resulting controller can be exported as a Simulink model, generated C-code or a list of parameters of the reference cascade PID+filters structure. The algorithms of modal control and H-infinity loop-shaping are embodied in the implemented routines. The details can be found in the I-MECH deliverable D5.3 [6] which also refers to relevant scientific publications exploiting new theoretical results in this field.

Several parameters of this module are optional to allow fine-tuning of the achieved results by an experienced user with strong background in control theory. Default values are set internally allowing utilization of the SW also by an inexperienced user minimizing the necessary number of inputs and design choices to be made.

The whole process of the model-based controller design can be performed step-by-step or as a single batch using a simple command interface.

4.3.3.2 Implementation

The implementation was done using the MATLAB language. Library of C-MEX functions is available for MATLAB environment, export of the C(++) code is also possible when needed using the MATLAB Coder allowing to embed the generated code in an arbitrary target environment.

4.3.3.3 Input output parameters, constants

Table 25: FRF identification module.

FRF identification module	
Syntax: [frf,f,cuy,Guu,Gyy,Guy,varu,vary,covaruy,coruy,varfrf,SNRu,uncfrf]=id_frf(u,y,uc,isper,per,Ts)	
Inputs, parameters and constants	
Parameter	**Description**
u	Excitation input [arbitrary units scaling]
y	Plant output, typically position/velocity [arbitrary units scaling]
uc	Total plant input under closed-loop setting (optional, used for closed-loop identification)
isper	Periodic excitation flag – 0 – nonperiodic signals, 1 – periodic signals
per	Period length for isper=1
Ts	Sampling period
Outputs	
Parameter	**Description**
frf	Vector of complex numbers – nonparametric FRF estimate
f	Vector of frequencies corresponding to FRF data [Hz]

cuy	Coherence function for the FRF estimates
Guu	Autospectrum of input u
Gyy	Autospectrum of output y
Guy	Cross-spectrum uy
varu	Input variance
vary	Output variance
covaruy	Input-output covariance
coruy	Input-output noise correlation
varfrf	FRF estimate variance
SNRu	Input signal-noise ratio
uncfrf	Uncertainty bounds for the FRF estimate – 98% confidence interval

Table 26: Best linear approximation of FRF data using transfer function model with manual/automatic model structure selection.

Best linear approximation of FRF data using transfer function model with manual/automatic model structure selection	
Syntax: tf_est=id_tf_bla(f,frf,fmin,fmax,wfit,not,uncfrf,ams,Ts,nb,na)	
Inputs, parameters and constants	
Parameter	**Description**
f	Vector of frequencies corresponding to the input FRF data
frf	Complex vector containing non-parametric FRF plant model
fmin	(optional) Lower bound of relevant frequencies for parametric model fitting
fmax	(optional) Upper bound of relevant frequencies for parametric model fitting
wfit	(optional) User defined weighting of the individual frequencies
nopt	Nonlinear optimization flag, 0 – disabled, 1 – enabled – requires more computational time but often improves overall model accuracy
uncfrf	(optional) Uncertainty bounds for the FRF estimates
ams	Automatic model structure flag – 0 – user specified transfer function structure (number of poles/zeros), 1 – automatic derivation of proper structure from experimental data
Ts	Sampling time of the resulting transfer function model, setting to 0 leads to a continuous-time transfer function model
nb	(optional) Number of plant zeros if ams=0
na	(optional) Number of plant poles if ams=0
Outputs	
Parameter	**Description**

tf_est	Estimated transfer function model in the zero-pole-gain format

Table 27: PID+filters structure tuning block.

PID+filters structure tuning block	
Syntax: [pid_std.F1,F2,F3]=tune_pidf(tf_est,uncfrf,nrsp,css,der,der_param,der_man_Td,der_man_N,F1type,F1bw,F2type, F2par,F3type,F3par,damp, pm,gm,Ms,Mt,Mtl)	
Inputs, parameters and constants	
Parameter	**Description**
tf_est	Plant model in the transfer function type object
uncfrf	Uncertainty bounds in the frequency domain for robust controller design
nrsp	Nominal/robust stability and performance flag, 0 – nominal stability and performance w/o assuming uncertainty model, 1 – robust design taking uncertainty model into consideration
css	Controller structure selection flag, 0 – cascaded velocity + position loop, 1 – single velocity/position loop
der	Derivative action flag, 0 – PI+filters type control, 1 – PID+filters control structure
der_param	(optional) Derivative action specification, 0- automatic selection by tuning algorithm, 1-manual adjustment of derivative part of the controller
der_man_Td	(optional) User-specified value of the derivative gain of the PID controller in the standard ISA form, relevant if der_param=1
der_man_N	(optional) User-specified value of the time constant of the filtered derivative part of the PID controller in the standard ISA form, relevant if der_param=1
F1type	(optional) Structure of the additional filter Nr. 1 with a low-pass characteristic to improve high-frequency roll-off, 0-no filter, 1-1st order low-pass, 2-2nd order Butterworth low-pass
F1bw	(optional) Bandwidth of the low-pass filter defined by F1type, 0 – automatic tuning of filter bandwidth requested (default), >0 – user-specified filter bandwidth
F2type	(optional) Structure of the additional filter Nr. 2 with a notch characteristic to supress high-frequency oscillations above closed-loop bandwidth, 0-no filter, 1-2nd order notch
F2par	(optional) Structure with user-specified parameters of the notch filter, automatic tuning to a specified oscillatory mode possible

F3type	(optional) Structure of the additional filter Nr. 3 with a notch characteristic to supress high-frequency oscillations above closed-loop bandwidth, 0-no filter, 1-2nd order notch
F3par	(optional) Structure with user-specified parameters of the notch filter, automatic tuning to a specified oscillatory mode possible
Damp	(optional) Basic parameter affecting desired closed-loop damping 0-mild, 1-medium, 2-strong. Higher damping level leads to smaller overshoots and more stable behaviour, usually at cost of achievable bandwidth
Pm	(optional) Desired phase margin in degrees
Gm	(optional) Desired gain margin
Ms	(optional) Maximum modulus of the closed-loop sensitivity function
Mt	(optional) Maximum modulus of the closed-loop complementary sensitivity function
Mtl	(optional) Maximum modulus of the load-side closed-loop complementary sensitivity function
Outputs	
Parameter	**Description**
pid_std_	Structure containing a set of admissible PI(D) controllers
F1,F2,F3	Structure containing additional filters
ctrl_mdl	Simulink file containing the whole proposed controller structure, suitable for simulations or automatic code generation

Table 28: Direct model-based controller design from the experimental data.

Direct model-based controller design from the experimental data	
Syntax: [pid_std.F1,F2,F3]=tune_pid_data(u,y,uc,isper,per,Ts, css,der,der_param,der_man_Td,der_man_N,F1type,F1bw,F2type,F2par,F3type,F3par,damp, pm,gm,Ms,Mt,Mtl)	
Inputs, Parameters and Constants	
Parameter	**Description**
u	Excitation input [arbitrary units scaling]
y	Plant output, typically position/velocity [arbitrary units scaling]
uc	Total plant input under closed-loop setting (optional, used for closed-loop identification)
isper	Periodic excitation flag – 0 – nonperiodic signals, 1 – periodic signals
per	Period length for isper=1
Ts	Sampling period

css	Controller structure selection flag, 0 – cascaded velocity + position loop, 1 – single velocity/position loop
der	Derivative action flag, 0 – PI+filters type control, 1 – PID+filters control structure
der_param	(optional) Derivative action specification, 0- automatic selection by tuning algorithm, 1-manual adjustment of derivative part of the controller
der_man_Td	(optional) User-specified value of the derivative gain of the PID controller in the standard ISA form, relevant if der_param=1
der_man_N	(optional) User-specified value of the time constant of the filtered derivative part of the PID controller in the standard ISA form, relevant if der_param=1
F1type	(optional) Structure of the additional filter Nr. 1 with a low-pass characteristics to improve high-frequency roll-off, 0-no filter, 1-1st order low-pass, 2-2nd order Butterworth low-pass
F1bw	(optional) Bandwidth of the low-pass filter defined by F1type, 0 – automatic tuning of filter bandwidth requested (default), >0 – user-specified filter bandwidth
F2type	(optional) Structure of the additional filter Nr. 2 with a notch characteristics to supress high-frequency oscillations above closed-loop bandwidth, 0-no filter, 1-2nd order notch
F2par	(optional) Structure with user-specified parameters of the notch filter, automatic tuning to a specified oscillatory mode possible
F3type	(optional) Structure of the additional filter Nr. 3 with a notch characteristics to supress high-frequency oscillations above closed-loop bandwidth, 0-no filter, 1-2nd order notch
F3par	(optional) Structure with user-specified parameters of the notch filter, automatic tuning to a specified oscillatory mode possible
damp	(optional) Basic parameter affecting desired closed-loop damping 0-mild, 1-medium, 2-strong. Higher damping level leads to smaller overshoots and more stable behaviour, usually at cost of achievable bandwidth
pm	(optional) Desired phase margin in degrees
gm	(optional) Desired gain margin
Ms	(optional) Maximum modulus of the closed-loop sensitivity function
Mt	(optional) Maximum modulus of the closed-loop complementary sensitivity function
Mtl	(optional) Maximum modulus of the load-side closed-loop complementary sensitivity function
Outputs	
Parameter	**Description**
pid_std_	Structure containing a set of admissible PI(D) controllers
F1,F2,F3	Structure containing additional filters
ctrl_mdl	Simulink file containing the whole proposed controller structure, suitable for simulations or automatic code generation

4.3.3.4 Validation in MIL and HIL

MIL validation done in MATLAB/Simulink environment. HIL validation done using ZAPUNI motion testbeds (flexible arm motion setup, CNC grinding machine).

4.4 ZAPUNI Implementation aspects

4.4.1 Pilot 5

Validation of system identification and controller design algorithms was done using experimental data acquired from Pilot 5 provided by Philips.

4.4.2 Use Case 1.3

Implementation of the algorithms in commercial PLC systems produced by TECO company is currently ongoing. Validation of the BB6 functionalities is to be demonstrated on the flexible production machine defined in UC 1.3. The goal is to prove that the auto-tuning methods can be used in conjunction with existing industrial COTS hardware and software equipment. Also, the integration with other building blocks of WP4, namely the vibration damping module of BB7 and visual feedback of BB4 is envisioned.

4.4.3 Use Case 2.2

Implementation and validation of the algorithms is currently ongoing. The goal is to demonstrate the BB6 functionalities on custom-made collaborative robotic platform designed by ZAPUNI. Collaboration with power electronics by Ingenia (BB5) will also be demonstrated via this use case.

5 Conclusion

5.1 General conclusion remarks

The aim of the I-MECH project is to provide smart functionalities for wide range of electric drives and automation systems in mechatronic fields. There are some (usually big) drive manufacturers providing various sets of similar functionalities in their systems. However, the functions are strictly bind to particular products and are not available for brownfield applications. I-MECH platform and building block components described in this document could simplify the life of small producers of mechatronic systems because these components can be simply integrated in their system and the functionality can be increased in much shorter time comparing with building their own solutions.

The question of interoperability is solved for the building block components which are realized in MATLAB Simulink environment by nature. The latest versions of this software support FMU import/export by FMI Toolbox. SVS is supported with the algorithms which are realized in LabView. From the perspective of sensor usage and its EtherCAT interface, the algorithms seem to be the part of the sensor. But it is possible that these algorithms will be converted to MATLAB Simulink during the integration phase which will increase their interoperability. Interoperability was also tested in Amesim from Siemens which enables the insertion and co-simulation of Simulink code. This part was already presented in previous deliverable D5.4.

The implementation aspects of BB3 and BB6 have been described in this deliverable. The theory of functions, implementation details, input output definitions and description of validation methods for individual BB components is provided in this document. The testing is already running in the course of WP6. Most of building block components were already tested and evaluated and the results of these activities were already summarized in D6.3. Further testing activities were provided to D6.5.

The implementation of building block components in Pilots, Use cases and Demonstrators is in progress. It will mainly continue in the rest of the project. It is expected that the building block components will be improved and enhanced based on the experiences with the implementation in diverse real applications. They will be also adapted to simplify their accommodation in I-MECH centre.

5.2 Contribution beyond the state of the art

The specific contribution beyond the state of the art of BB3 software blocks is a possibility to integrate condition monitoring and predictive maintenance into small and low-cost drives and applications and brown filed solutions with no necessity of additional hardware costs. Smart Vibration Sensor, which represents HW building block component allows vibration-based condition monitoring and PHM by integration of low-cost sensor into the existing communication and computational structures into brownfield application as well as in greenfield applications. Existence of such BBs allows implementation of condition monitoring to the applications for which the condition monitoring usage wasn't economically beneficial before.

Condition monitoring and predictive maintenance task are actually usually realized on a separate hardware. This system has its own sensors and its own communication channels which are then connected directly to Layer 3. The idea revealed here is different. There is an attempt to use the existing components in the system. Computational components are used for data pre-processing, data stream reduction, computation of condition indicators up to predictive diagnostics. Building block components are realized in such a way that they can be placed anywhere. Existing interfaces are used for passing the data between individual layers. This integration brings additional savings. Smart vibration sensor and associated algorithms share the information with the drive. Drive can immediately react to increased vibrations while the vibration processing algorithms can reuse the information from the position sensor in the inverter for the analysis of bearings faults.

Similar situation is in case of BB6. The auto-tuning and self-commissioning capability can be implemented in wide range of drives in the greenfield and brownfield systems. The strong contribution of BB6 is the ability to identify and tune the control loops for the multi-mass systems, i.e. the systems with flexible couplings. The developed BB6 components enable to design not only the parameters of the controllers but also the parameters of the accompanied biquadratic filters. The precision of filter parameter design is crucial for the behaviour improvement. Even some build

in auto-tuners do not provide satisfactory filter designs which leads to necessity of either time consuming manual tuning or to decreased control performance. Even thought that neither the proposed automatic design is always optimal, it gives the solution which is close to optimal and further manual tuning is then much less time demanding.

5.3 Dissemination and exploitation

The work on this deliverable brought together the results in dissemination activities. Three papers cited in the publication list were prepared and presented on international conferences. Two of them were presented on I-MECH special session organized during ETFA 2019 conference in Zaragoza in Spain. BB3 and BB6 leaders have prepared the video giving fast view on the work being realized in related tasks, namely in Task 5.3 and Task 5.4. This video was shown during ECSEL Symposium in June 2019. It is available on project www pages https://www.i-mech.eu/publications/videos/. BB3 and BB6 leaders, with the help of involved partners, are regularly contributing to I-MECH Newsletter, see https://www.i-mech.eu/publications/dissemination-material/.

The main part of the presented work was realized by the universities and research institutions. BUT, ZAPUNI, UNIBS and partly also UCC are already exploiting their results in their teaching activities. The goal is also to make described building block components available through I-MECH center and thus to make them more visible for potential customers. This work will be realized in the rest of the project run.

Acknowledgement

This project has received funding from the Electronic Component Systems for European Leadership Joint

Undertaking under grant agreement No 737453. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program and Netherlands, Czech Republic, Latvia, Spain, Greece, Portugal, Belgium, Italy, France, Ireland