# Remote Laboratory for Advanced Motion Control Experiments

http://dx.doi.org/10.3991/ijoe.v10i5.3835

S.D. Gadzhanov, A. Nafalski and Z. Nedic
University of South Australia, Mawson Lakes, Australia

***Abstract* — Lately, a number of remote laboratories for motion control has been developed worldwide as a valuable tool for teaching mechatronics students basic control strategies. However, there is a lack of remote laboratories that can be used in teaching advanced motion control. In this paper a remote laboratory for advanced motion control developed at the University of South Australia is described. The real time control algorithms, including H-infinity control, are successfully implemented in an FPGA environment. This paper also gives a detailed description of the graphical user interface developed for this laboratory, which enables effective remote experimentation.**

***Index Terms*—remote laboratory, NI LabVIEW, FPGA, H-infinity control, motion control, brushless DC motor**.

## I. INTRODUCTION

Remote laboratories for advanced motion control are an important tool for teaching students in mechatronics courses of control systems engineering and experimentation with a wide range of feedback devices. An already developed prototype of a remote laboratory for motion control has been presented in [1, 2]. It utilises a 12 inch linear stage, coupled with a Brushless DC motor and controlled by a high-performance motion controller. The linear position feedback is provided by a precision optical linear scale and the rotary position feedback is delivered by a resolver and a resolver-to-digital converter. All advanced control algorithms and remote laboratory functionalities are implemented in an industrial grade hard real-time controller with a Field-Programmable Gate Array[1] (FPGA) core from National Instruments.

The Graphical User Interface (GUI) is the interface which connects the sometimes geographically distanced students to the experimental workbench and delivers the means of interaction with the software layer controlling the equipment. On one hand, this interface is supposed to be flexible enough to provide a variety of choices for setting up and performing control experiments. On the other hand, it must deliver reliable experimental results in terms of graphical plots and measurement data files, and create a sense for reality in the distant user.

While the prototype for this project needs utilisation of some additional requisites such as: students' account registration facility, web-based materials for theoretical preparation, enhancements of reality such as web-cameras, chat-rooms, etc., it provides all the functionality needed for remote execution of the control experiments and paves the way for further upgrades and improvements.

In this paper, the hardware structure and the architecture of the remote laboratory based on the prototype are first explained. Then, the GUI, comprising of three front panels for setup, execution and diagnostics of the control experiment, is described. Functionality of the LabVIEW code standing behind this GUI and performing all the tasks is also discussed. This code is utilised in two major applications – the host application running in the real time target, and the slave application running in the FPGA target.

Further, the modeling and the model validation, and the implementation of the advanced control algorithms are also presented.

## II. HARDWARE WORKBENCH

The universal hardware workbench, shown in Figure 1, consists of a BLDC motor, an intelligent drive, a linear stage and a linear scale. The power stage is comprised of a high-performance motion controller (utilising a sinusoidal field-oriented control strategy), a signal processing module (containing opto-isolation and instrumentation amplifiers/filters, and bridge drivers) and a full bridge 3-phase inverter with current sensors. The feedback devices for linear/angular position, velocity and commutation are: an optical linear scale, a rotor-excited resolver, a magnetic encoder and Hall-sensors.

In order to accomplish the linear/angular position control and to monitor the important signals of interest, two different hardware solutions for data acquisition and control have been implemented, namely: PCI/USB based data acquisition board and real-time controller with a reconfigurable FPGA core. The former uses the PC Windows OS for software based monitoring and control in LabVIEW environment. Opposite, the latter performs high-speed

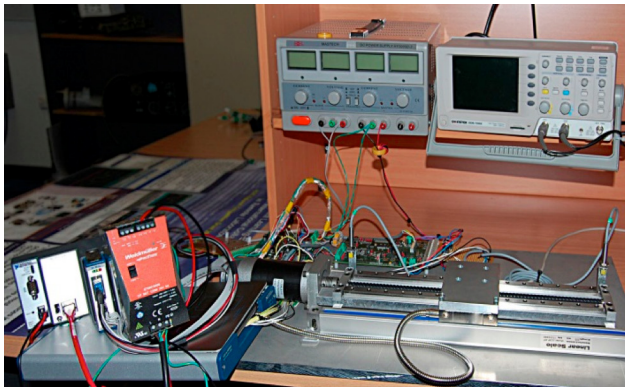

Figure 1. Hardware Workbench

[1] Field-Programmable Gate Array (FPGA) – a type of a reconfigurable integrated circuit (IC) where the programmable logic is described by Hardware Description Language (HDL)

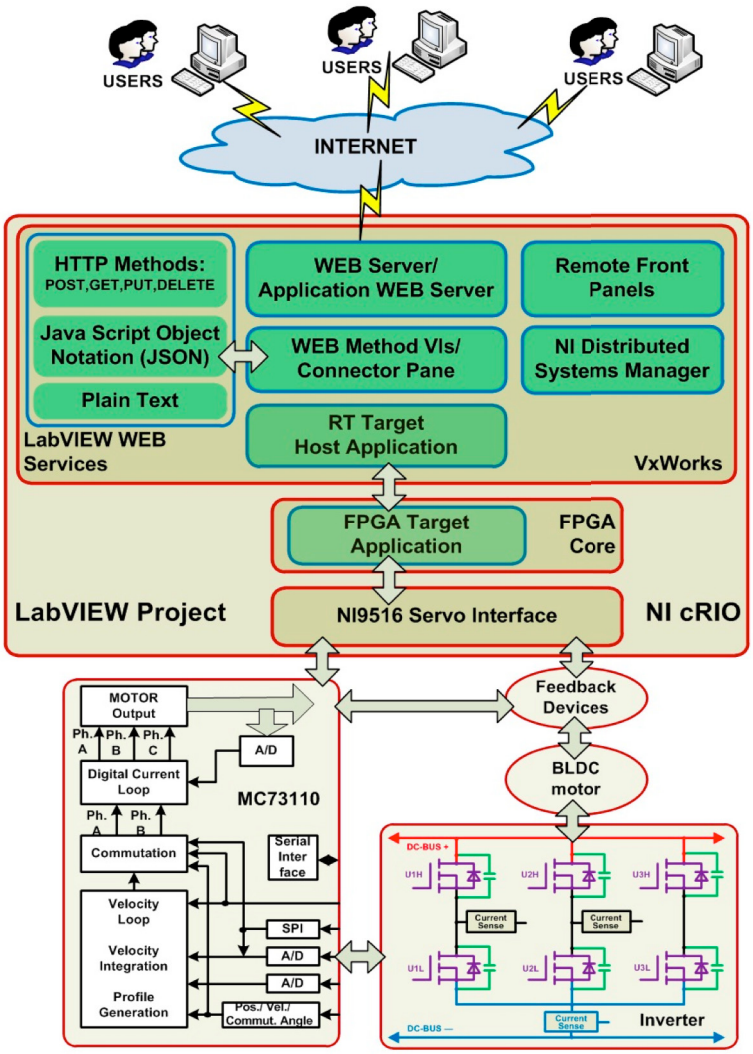


Figure 1. Architecture of the Remote Laboratory for Motion Control

hardware/software control with some signal monitoring capabilities using a robust real-time OS and specialised LabVIEW application for system control, which also provides the software user interface.

## III. ARCHITECTURE OF THE REMOTE LABORATORY

A diagram which shows the architecture of the remote laboratory and the technology for a remote connection of the web-based clients is depicted in Figure 2. In more details, the FPGA approach in the remote laboratory control has been presented in [3]. Typically, with the utilisation of a LabVIEW Application/ Web Server, two technologies for remote connection and control are possible – namely, LabVIEW Web Services and Remote Front Panels:

*LabVIEW Web services* [4] - use *Web Method Virtual Instruments* to exchange data with the following Web-clients: HTTP clients/forms, Web browsers or customised web-based applications. This data could be in XML, HTML, Java Script Object Notation (JSON) or plain text formats. The following standard HTTP methods are supported: Post, Get, Put and Delete [5]. Furthermore, the LabVIEW application exchanges data with the *Web Method VIs* by means of *Connector Panes*[2]. Thus it is actually deployed as a Web Service (within the LabVIEW project) to the LabVIEW Application Web Server.

*Remote Front Panels* [6] - the host application is run on the Web server and the corresponding front panel can be sent to multiple web-based clients simultaneously. The remote clients can be LabVIEW or Web-browser environments (the latter case requires installation of LabVIEW Run-Time Engine). The remote client can be granted "observer" or "full" types of control. In either case the server manages the clients' queue requests and remote control time limitations.

Being a vital part of the remote laboratory for motion control, the hardware workbench can be distinguished with its functionality, flexibility and cost-effectiveness.

## IV. USER INTERFACE

The LabVIEW programming environment provides all the tools for building a multi-purpose graphical user interface. As an integral part of the comprehensive RT/FPGA

[2] Connector Pane - a specific LabVIEW programming tool providing connection between the front panel indicators and the application data. For that purpose it utilises terminals with *Connector Pane Pattern*.

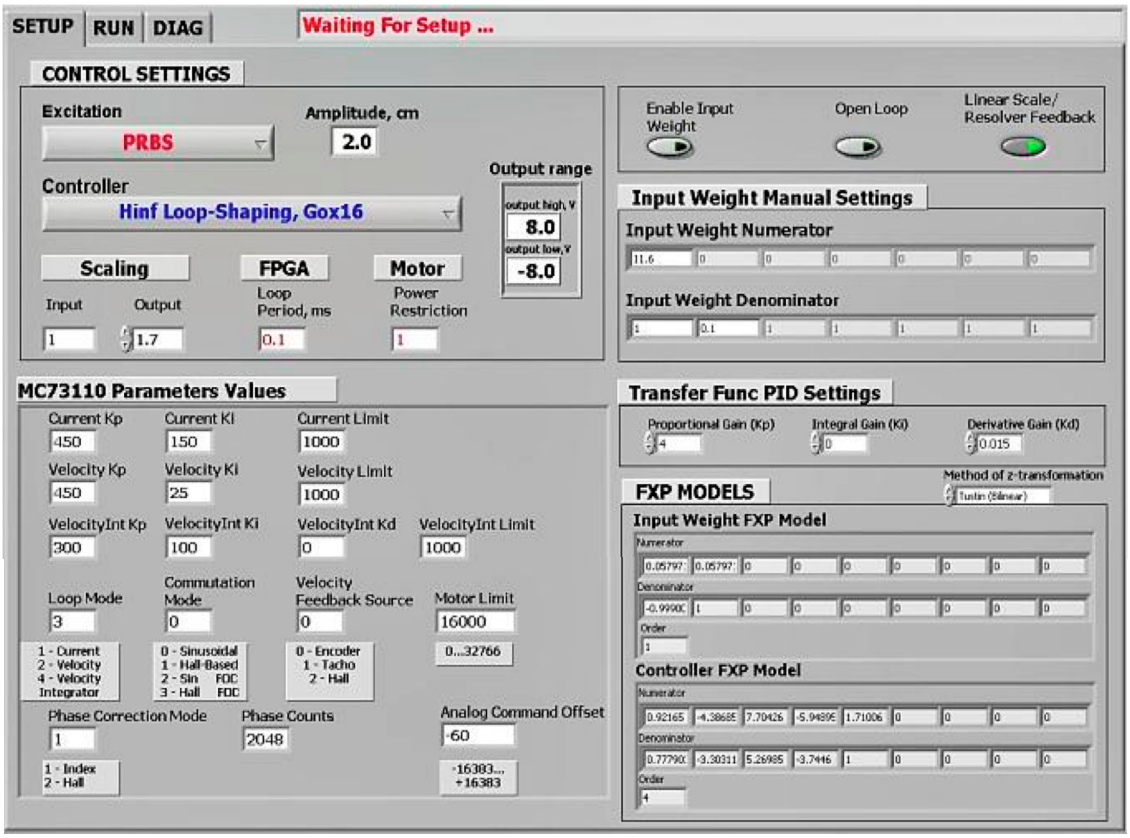


Figure 2. LabVIEW FPGA GUI for Setup of the Remote Experiment

control GUI, three switchable front panels are developed. They are designated for setting-up, execution and diagnostics of the control experiment. A number of different input controls, indicators, graphing tools and more complex structures have been utilised to accomplish the tasks for successful interaction between the user and the workbench.

### A. The Setup GUI

This the most important stage of the control experiment since it determines all the control parameters. Distant users are supposed to be well prepared prior to the experiment and are required to complete very carefully these settings. This would prevent the equipment from any damage caused by neglectful user actions even if there are enough measures already undertaken for protection. The following areas can be visibly distinguished – Figure 3:

**Current Action Information bar** (in red colour) – provides information about the current state of the remote control interface:

Waiting for Setup;

Writing a Waveform data into the FPGA memory;

Test running of the motorised linear stage;

Finishing and saving the measurement data.

**Control Experiments Settings area** – provides the main tools for setting-up of the control experiment:

*Excitation signals* (arbitrary waveforms) – Square Pulse, PRBS and Swept Sine;

*Amplitude of the excitation signals* – it is expressed in centimetres (cm) of linear stage displacement;

*Controller selection* – PID and a range of $\mathcal{H}_\infty$ controllers are available for experimentation;

*Controller output limitation* – it will limit the maximum value of the controller output to $\pm 10$V which corresponds to the maximum analogue command value of the MC73110 servo controller;

*Loop Period* – allows the FPGA loop period to be set/modified in-between 0.1-1ms;

*Power Restriction* – additional parameter complementing the MC73110 parameter *Motor Limit*, which allows the power output to the motor to be restricted for control or other purposes.

**MC73110 Parameter Values area** – allows the control/commutation loops and some other adjustments to be manually entered/ modified, as follows:

*Current Loop parameters* - $K_p/K_i$/Current limit;

*Velocity Loop parameters* - $K_p/K_i$/Velocity limit;

*Velocity Integrator Loop PID parameters* - $K_p/K_i/K_d$/VelocityInt limit;

*Loop Mode* – various loop combinations of current, velocity and velocity-integrator;

*Commutation Mode* – it can be set by the user in accordance with the BEMF of the motor – trapezoidal or sinusoidal, and in addition to switch to the more advanced FOC control;

*Velocity Feedback Source* – for best performance this should be left to the *Encoder* setting;

*Motor Limit* – by default the power output to the motor is limited to approximately 50% and can be further reduced for control purposes;

*Analogue Command Offset* – to prevent the motor to drift;

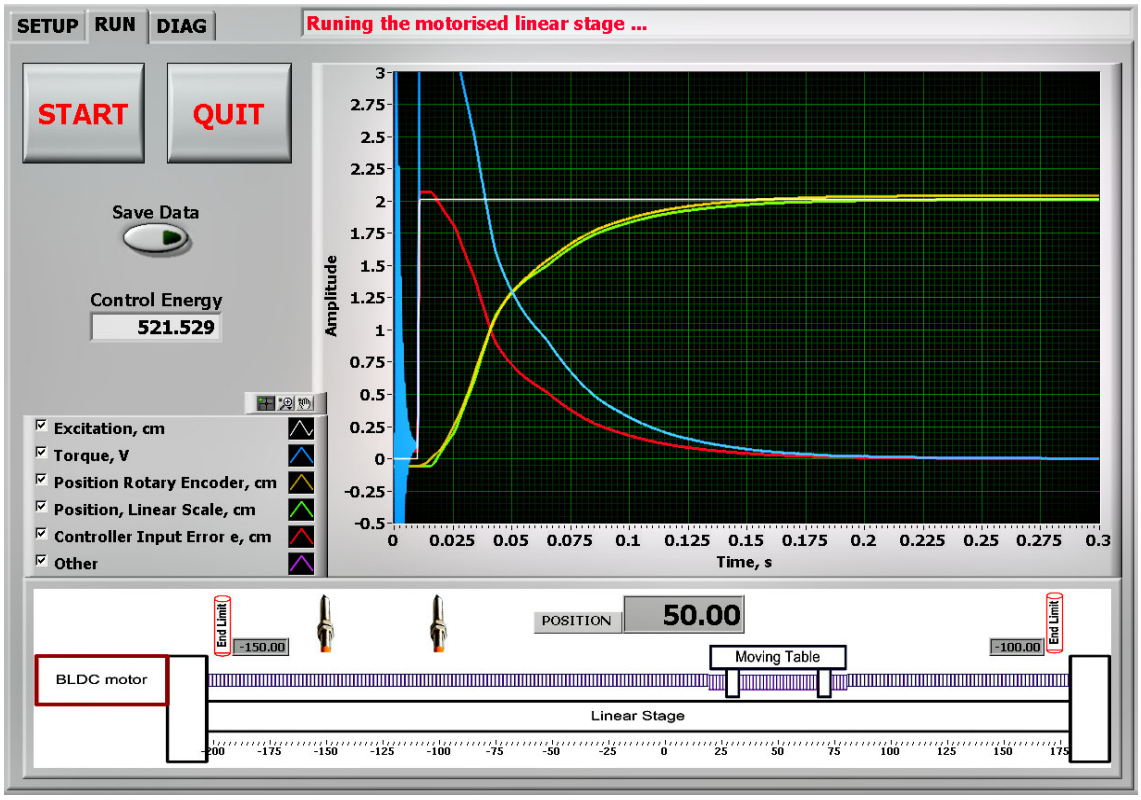


Figure 4. LabVIEW FPGA GUI for Execution of the Remote Experiment

*Phase Counts* and *Phase Correction Mode* – allows the number of the encoder counts per motor pole-pair to be readjusted if the resolution of the resolver-to-digital-converter is changed; allows the source for the MC73110 phase correction to be selected amongst the encoder zero index or Hall-effect sensors.

**Input Weight Manual Settings area** – allows the polynomial function coefficients of the numerator/ denominator (the *TF* pre-compensator from the FPGA control algorithm realisation), to be manually entered or modified. Three virtual buttons are seen above this area:

*Enable Input Weight* – allows the input weight to be enabled or bypassed;

*Open Loop* – allows the position feedback to be disabled (open loop operation);

*Linear Scale/Resolver Feedback* - allows the source of position feedback to be selected between the linear stage and the resolver (rotary encoder). Thus more control experiments and analysis can be conducted.

**Transfer Function of the PID Controller /Manual Settings area** – allows PID parameters $K_p/K_i/K_d$ to be manually entered/modified and the discrete Fixed-Point (FXP) TF model[3] to be automatically determined.

**FXP models area** – visualises information (numerator, denominator and TF order) about the discrete FXP transfer functions related to the chosen control method:

*Input Weight FXP model* – this is the *TF* pre-compensator from the FPGA control algorithm realisation;

*Controller FXP model* – this is the *TF* controller from the FPGA control algorithm realisation;

*Method of z-transformation* – if the transfer functions are presented in the s-domain they have to be transformed in the z-domain in accordance with the selected FPGA loop sampling time. Several methods of transformation are possible but for best results achieved so far are with the Tustin (Bilinear) and Pre-warp algorithms:

| *Tustin (Bilinear)* | *Pre-warp* |
|---|---|
| $s \rightarrow \frac{2(z-1)}{T(z+1)}$ | $s \rightarrow \frac{z\omega(z-1)}{2\tan\left(\omega T/2\right)(z+1)}$ |

## B. The Experiment Execution GUI

The interface, shown in Figure 4, allows distant users to execute the already arranged (in the Setup front panel) control experiment and collect the experimental data by means of FIFO tables and DMA channels. Furthermore, data collected from several selectable sources is plotted on the screen using a LabVIEW Waveform Graph Window and can be saved for further analysis with LabVIEW post-processing VI in the form of LabVIEW measurement data.

In addition, the amount of control energy (the command servo voltage for given period of time) output for the control experiment is calculated and shown on the screen. A virtual moving linear stage compliments the graphical interface and creates a feeling of reality of the remote experiment in the end user. In real time, it recreates the movements of the linear stage (the moving table) and

[3] Fixed Point (FXP) TF model – a specific representation of a proper transfer function (up to eight order) in LabVIEW which uses fixed-point simulation data type

shows its current position. The positions of the proximity/software end limit switches are also visualised.

The **Waveform Graph Window** displays the selected graphical data (signals) from the current experiment and provides functionality for a number of manipulations of the visualised graph so that areas of interest can be shown in a convenient way. Signals which can be selected and plotted on the graph are:

*Excitation* (the selected waveform from the setup window) – in cm of linear displacement;

*Torque* – a relative quantity expressed in volts on the command input of the servo amplifier;

*Rotary Encoder position* – the position of the resolver (via the RCD converter), transformed in cm of linear displacement for the purposes of comparative analysis;

*Linear Encoder Position* – the position of the linear scale, in cm of linear displacement;

*Controller input error* – the input error $e$ signal on the input of the controller (the difference between the excitation and real position signals, which *de facto* is the controller input signal);

*Other signals* – for the diagnostic purposes of the FPGA controller software program, this could be some other intermediate signal or the FPGA loop timing (in μs of the loop period).

### C. The Diagnostics GUI

Advanced users and support personnel, who have deeper innermost understanding of the functionality and programming of the NI cRIO FPGA system and its interrelation with the NI 9516 Servo Interface and MC73110 high performance servo controller, can gain from the additional tools provided for current state information and diagnostics of these modules. Several areas are distinguished, namely:

**NI 9516 Module Initialisation area** – it visualises the following information during the module initial parametrisation:

Module's actual state during the initialisation, configuration and activation stages;

Number of FPGA cycles needed for successful parametrisation;

Indicators for Module Ready; Scan method indication related to the FPGA Write data buffer current condition; Scan method indication related to the FPGA Read data buffer current condition.

**FPGA Current State area** – provides real time information about:

NI 9516 module operating state – Active, Initialisation, Configuration and Fault;

Raw position data for the Rotary Encoder (Resolver via the RDC converter) and for the Linear Encoder (Optical Linear Scale);

Current address of the FPGA memory reference;

Error status, codes and messages;

Indicators for NI 9516 module active state; FPGA maximum memory address indication (reached during the generation of the arbitrary waveform); FIFO table timed out alarm.

**NI 9516 Module Faults and Errors area** – provides the error status, codes and messages, and information for the following faults (obtained from the manufacturer status register):

Undervoltage/Overvoltage of the external power DC supply (Vsup) to the module;

Internal power supply failure;

Short Circuit of the Drive Enable output;

Timing Synchronisation RT system problems;

Interrupted Heartbeat (chassis) Communication;

Module internal processor Watchdog alarm.

**MC73110 Diagnostics** – opens an additional window for comprehensive diagnostics of the high performance servo controller. This separate diagnostics interface, presented in [2], provides a diagnostics information in the following areas: Activity status and limits; Signal sensing; Motor status; Serial communication; PWM adjustment; Events status; PID parameters for current, velocity and velocity-integrator control loops; Condition mask; Analogue signals status; Other status information.

## V. FUNCTIONAL DESCRIPTION

Two major applications are running in the Real Time target (VxWorks OS) and in the cRIO FPGA target, namely: the Host and the FPGA (Slave) Applications.

### A. Functional Description of the Host Application

At the start-up the initial parameters and resources for this application are read, the MC73110 default parameters are loaded, decoded and shown on the computer screen (Figure 5), and the main RT Host loop is started. Thus, several functional stages can be distinguished:

**Experiment Setup Loop** – this is the remote experiment setup loop with the corresponding GUI – Figure 3. It includes:

*Choice of a controller* – after selection the matching models are loaded from the database and converted into FXP models for further use. The input weight could be modified under some specific circumstances;

*MC73110 parameters modification and encoding* – the user is allowed to modify a number of parameters related to the servo control loops (Figure 3). Next, the MC73110 set of parameters is encoded in a specific format for further initialisation of the servo controller;

*Choice of Excitation and Other Parameters Setup* – data for the selected control waveform is loaded from the database and prepared for further use.

**cRIO FPGA Resources/Modules Initialisation** – this is a transitional stage needed for setting up of the cRIO controller. The following steps are distinguished:

*Opening of a FPGA Reference* – relates the already compiled FPGA code to the FPGA target;

*Transferring of parameters to the FPGA target* – this transfers important parameters to the FPGA target by means of Host-to-FPGA write method;

*NI 9516 dual-encoder interface Initialisation*;

*FPGA Memory Write* – the data for the selected excitation signal (control waveform) is transferred into the FPGA memory for RT execution.

**Experiment Execution** – after all needed parameters and data are sent to the FPGA target, it indicates that it is ready for execution, and the MC73110 controller is initial-

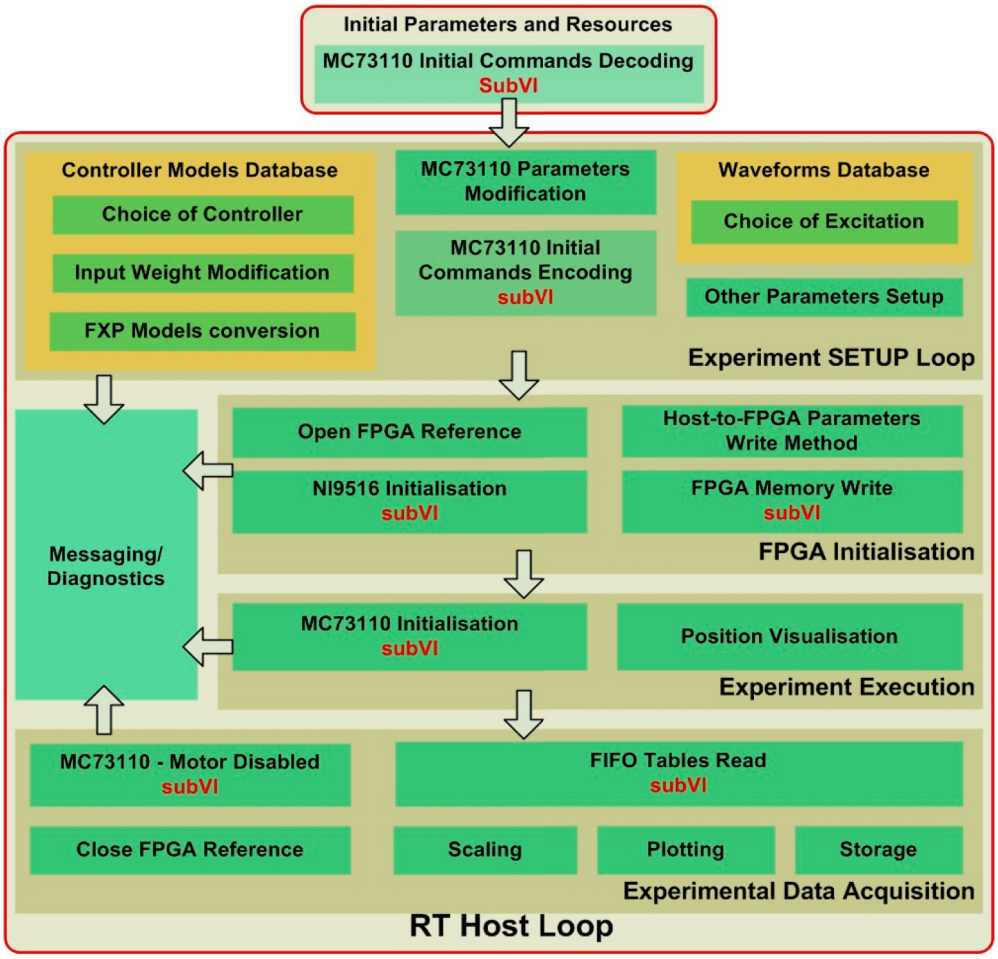


Figure 5. Functional Diagram of the RT Target /Host/

ized. Then the FPGA target is turned into an experiment execution mode.

**Experimental Data Acquisition** – this is the final stage of the control experiment, divided into the following steps:

*MC73110* – Motor Disabled – this disables the motion axes and the motor;

FIFO Tables Read – the FIFO tables containing the RT data, acquired from the FPGA target during the control experiment, are read;

Acquired Data Processing – the data is appropriately scaled and plotted on the user's screen. It is further saved by demand;

Close FPGA Reference – the FPGA target is disconnected from the reference.

**Messaging and Diagnostics** – during all stages of the control experiment different informative and diagnostic messages are displayed on the user's screen. A separate diagnostics GUI is dedicated to diagnostics.

### B. *Functional Description of the FPGA (slave) Application* - Figure 6

Similar to the host application, some initial parameters are read at the start-up. Two loops are running simultaneously on the FPGA target:

**Main Loop** – it initiates a number of tasks:

*NI 9516 Configuration* – performs a procedure of configuration of the NI 9516 servo interface module initiated by the host application;

*RT Host Read Method* – reads the parameters sent by the host application;

*NI 9516 Write/Scan/Read Methods* – constantly exchanges input/output parameters between the NI 9516 module and the currently running procedure into the FPGA target;

*Experiment Execution Cycle* – executes all the tasks related to the control experiment and initiated by the host application:

- *Read FPGA Memory* – reads the data for the control waveform from the FPGA memory and sends it to the FPGA Controller;
- *FPGA Controller* – performs in real time the actual positioning control of the linear stage in conjunction with the chosen FXP controller model and control waveform;
- *NI 9516 Position Data* – reads constantly the current position and sends it to the host application or to the FIFO Tables;
- *FIFO Tables Write* – writes the acquired data (excitation signal, torque, angular and linear position, and control error) into the FIFO tables by means of high-speed DMA channels;
- *Messaging* – during the experiment execution sends a number of information and error messages to the host application.

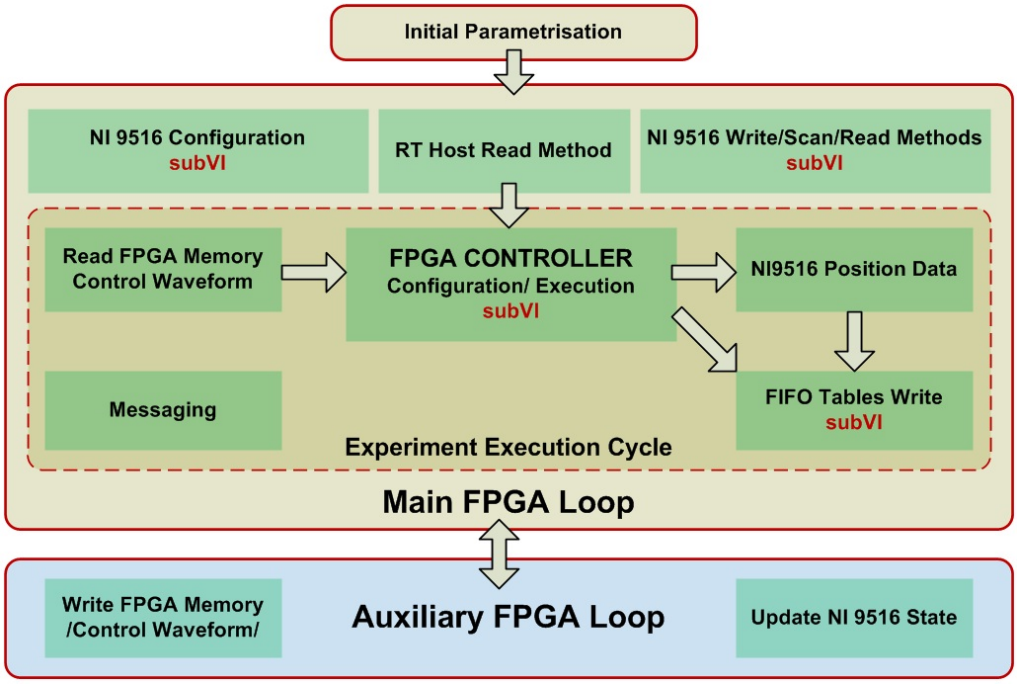


Figure 6. Functional Diagram of the FPGA Target

**Auxiliary Loop** – it performs the following supporting operations:

*Write FPGA Memory* – works together with the host application during the process of writing of the data of the excitation signal (control waveform) into the FPGA memory;

*Update NI 9516 state* – constantly updates the state of the NI 9516 module.

## VI. MODELLING OF THE SYSTEM

The system identification process is a turning point in the area of control systems engineering – the whole development of the control system depends on the proper model-upbuilding. It is based on the data obtained after observation of the system properties of interest - the *input signals (*controlling the position controller) and the *output response signals*– angular/linear position, torque, velocity, acceleration, etc. The following stages are distinguished:

### *A. Signal Acquisition*

#### *1) Excitation Signals*

For the purposes of the motorised linear stage system identification and control experimentations, a number of standard well-known excitation signals were applied: *Square pulse signal, Pseudo-Random Binary Sequence* (PRBS) and *Swept-sine* (periodic chirp) signals. They allow the moving table to be kept around a certain start point (usually the middle of the linear stage) and protect the mechanics from destruction, initiating a rapid wide range motion and exciting a wide range of frequencies. Thus, an excellent signal to noise ratio results could be achieved, especially in a case where a low resolution encoder has been used as a feedback position device.

#### *2) Signal Processing and Conditioning*

Without a proper data acquisition wrong results could be attained especially in the Frequency Domain. Two stages of data acquisition could be identified:

*Hardware stage* [7] – the signals from the current sensors are filtered by an anti-aliasing filter and then passed to isolation and instrumentation amplifiers. After that they are filtered by a Bessel low pass filter with a Sallen-Key topology. The advantage of this filter is the constant group delay of all signals.

*Software stage* – after some digital signal conditioning and pre-filtering, the signal is passed through a FIR filter (Savitzky-Golay), which applies a polynomial smoothing. It works by fitting a polynomial into a predefined frame of samples and minimises the least square error. Thus, it preserves the shape and height of the waveform peaks.

Finally the acquired signal is ready to participate further into the System Identification process – the modeling.

### *B. Box-Jenkins Model*

The theoretical model of the system, even after some simplifications, was of a very high order and additional model reduction would increase its uncertainties. The process of experimental parametric identification required some additional sensors and might not necessarily lead to satisfying results. That is why we decided to apply a procedure of a "Black box" *parametric model prediction.*

The general-linear model (Figure 7) is very hard to be realised in practice, and there is not guarantee of global convergence during the computations. A more precise specification of the properties of the dynamic system will lead to a simpler and a more realisable model. For systems where disturbances affect the output, a good model describing separately the system dynamics and the disturbance dynamics, is the Box-Jenkins Model. In the z-domain it is defined by the following equations [8, 9]:

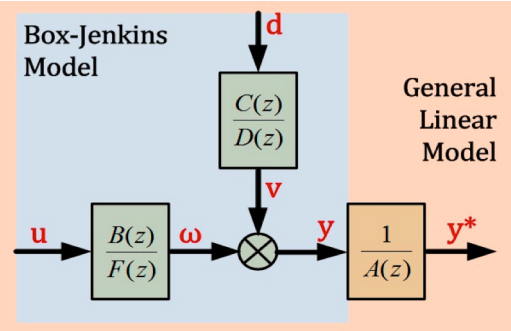


Figure 7. Block Diagram of the Box-Jenkins Model

$$y(k) = \frac{B(z)}{F(z)}u(k-n) + \frac{C(z)}{D(z)}d(k) \quad (2)$$

$$B(z) = b_0 + b_1 z^{-1} + \cdots + b_{k_b-1} z^{-(k_b-1)}$$

$$C(z) = 1 + c_1 z^{-1} + \cdots + c_{k_c} z^{-k_c}$$

$$D(z) = 1 + d_1 z^{-1} + \cdots + d_{k_d} z^{-k_d}$$

$$F(z) = 1 + f_1 z^{-1} + \cdots + f_{k_f} z^{-k_f}$$

$k_b$, $k_c$, $k_d$,$k_f$ - the polynomial orders of *B*, *C*, *D*, *F*.

### C. Model Order Estimation Procedure

The model prediction was a long process of trial and error until a satisfying result was achieved. Several steps were taken to assure that a proper model with the lowest possible order is predicted, and that it fits into the assessment and validation criteria:

**Step 1** – A simpler model (Output-error model) was deployed as a start point to determine approximately the orders of the B(z)/F(z) polynomials.

**Step 2** – The delivered orders from Step 1 were used to determine a constricted range for the Final Prediction Error Criterion (FPE) which was used to estimate the model-fitting error.

**Step 3** - A Box-Jenkins model prediction procedure was deployed using the estimated orders ($k_b$, $k_c$, $k_d$, $k_f$, optimal delay) from the previous step as a start point.

**Step 4** – A residual and model prediction error analysis were deployed to assess the predicted model.

**Step 5** – A correction of the delay/orders was done and the whole process started again from Step 3. At this stage some conclusions about model orders were made.

**Step 6** – After a model satisfying all the requirements was found, an attempt to reduce further the orders was initiated and steps 3,4 and 5 repeated again.

### D. Model Validation

Several ways to validate the model were considered in order to prove or disprove its applicability:

#### 1) Residual Analysis

Residuals represent the "missing part" of the model or what is does not describe of the real system (plant). They are delivered by calculating the difference between the predicted response (one step ahead) of the model and the measured response of the real system, to the same set of input data (excitation signal).

*Cross Correlation* (Independence Test Criteria) – tests the independence of the residuals or how they correlate with the past inputs. In order to check if the model needs revision, a high *confidence interval* of 99% (a region of the residual results which are considered statistically insignificant with some value of probability) was applied. The actual result shows how well the residuals of the model fit into the confidence interval - Figure 8.

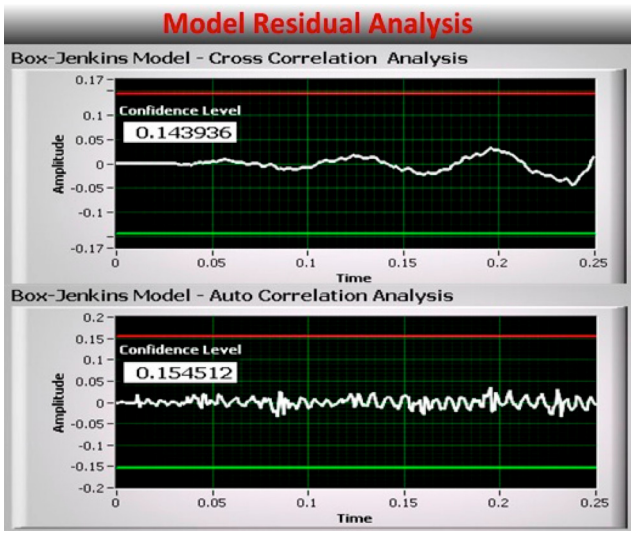


Figure 8. Model Residual Analysis

*Auto Correlation* (Whiteness Test Criteria) – investigates the residuals representing a zero-mean white noise. The actual result from the auto-correlation analysis of the model shows a very small value of non-white noise inside the confidence interval - Figure 8.

#### 2) Model Forecasting (Model Response Prediction)

Another way to test and validate the model was to check the system response using the already available input and output information from the previous steps (set of data different from the one used in the model prediction process). The prediction mean-square error (calculated as about 0.0015% of the maximum possible value of the signal) shows the difference between the predicted model response and the actual measured response of the real system - Figure 9. From the zoomed plot it can be seen that the curves of the predicted model response and the actual measured system (plant) response fit very well.

Thus, it was concluded that the process of system identification, including model prediction and validation, is successfully completed.

## VII. CONTROL ALGORITHMS IMPLEMENTATION

For the purposes of motion control experimentation, an application of several H-infinity ($\mathcal{H}_\infty$)[4] advanced control algorithms (namely S/KS sensitivity, 1DOF and 2DOF Glover-McFarlane loop-shaping procedures) and a reference PID controller were synthesised and implemented. From theory (particularly for motion control positioning application) it can be seen that the peak magnitudes of the sensitivity functions can be associated with the $\mathcal{H}_\infty$ norm and the emanating outcomes from these relations. Based on this, the S/KS $\mathcal{H}_\infty$ mixed sensitivity design was selected [10, 11].

[4] H-infinity ($\mathcal{H}_\infty$) – modern control methods in the $\mathcal{H}_\infty$ space of matrix-valued functions for mathematical optimisation and controller synthesis, which achieve guaranteed robust performance or stability

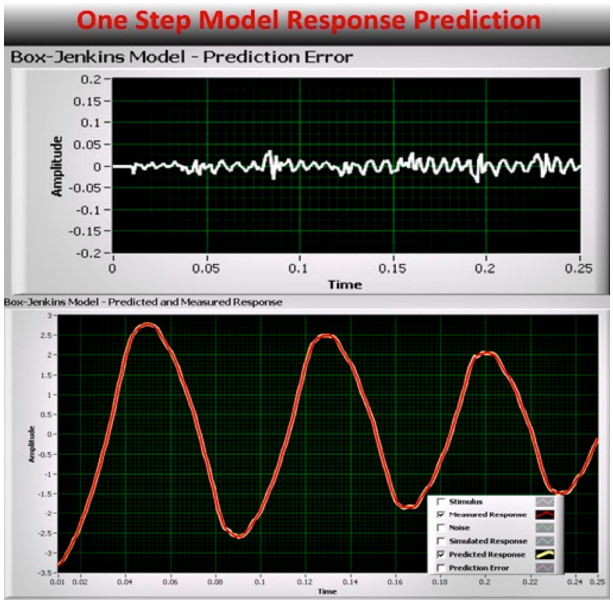


Figure 9. One Step Model Response Prediction

Further, the normalised coprime factor plays an important role in the robust stabilisation. The Glover-McFarlane loop-shaping procedure [12] was chosen, together with the choice of the pre-compensator and post-compensator weights of the augmented plant. That leads to the $\mathcal{H}_\infty$ optimal problem formulation, the maximisation of the stability and the one-step synthesis of the $\mathcal{H}_\infty$ controller. The two-degrees-of-freedom design [12, 13] comes as a natural continuation of the Glover-McFarlane loop-shaping procedure since it shows better outcomes when the reference signal is treated separately from the measured feedback signal.

PID controllers have been an industry standard for a long time. Therefore the comparison of the performance and stability between this type of controllers and the synthesised $\mathcal{H}_\infty$ controllers is inevitable. For this purpose, a PID optimal controller was obtained by the *Linear-Quadratic (LQ) cost-function optimisation* method [14].

The practical implementation of the abovementioned algorithms was a complex, daunting and time-consuming task. The FPGA environment in conjunction with the real-time target proved to be the only way of successful application of the high-speed advanced control. The *discrete fixed-point transfer function model* [15] was effectively applied in the FPGA core despite some limitations of the application. Based on this, a two transfer functions model, compatible with all the synthesised controllers, was proposed and successfully implemented.

## VIII. CONCLUSIONS AND FUTURE WORK

This paper describes in details the architecture and the user interface of a remote laboratory for advanced motion control developed at the University of South Australia. Given are examples of advanced control strategies, plant modeling and model verification results. Users can then develop control strategies and upload their algorithms into FPGA based environment for real time control of platform position, speed, acceleration, etc.

The remote laboratory enables remote connection of the web-client (representing the remote user) through Remote Front Panels. In order to support the implementation of advanced control algorithms the system is quite complex and a multi-panel graphical user interface (GUI) has been developed to enable remote settings of a large number of parameters. The GUI is developed in LabVIEW and consists of three front panels which provide the functionality for setting up, execution and diagnostics of the control experiments.

The development of the system has just been completed and tested, but it is yet to be implemented in teaching. For this to be realised development of the courseware is needed. Also, further improvements are planned, such as the implementation of a web camera to provide the remote users with a telepresence in the laboratory, chat-room for coordination of collaborative work and an interactive theoretical introduction for embedding the laboratory into the online delivery of advanced control courses.

## REFERENCES

[1] S. Gadzhanov, A. Nafalski, and Ö. Göl, "A Remote Laboratory for Motion Control and Feedback Devices," *Electrotechnical Institute Warsaw, Poland,* pp. 37-50, 24-27 Jun 2010.

[2] S. D. Gadzhanov, A. Nafalski, and Z. Nedic, "A Universal Workbench for Motion Control Experimentations in LabVIEW Environment," in *9th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, Bilbao, Spain, 2012, pp. 51-57.

[3] S. Gadzhanov, A. Nafalski, and Z. Nedic, "A FPGA Approach in a Motorised Linear Stage Remote Controlled Experiment," *International Journal of Online Engineering (iJOE),* vol. 9, pp. 55-63, April 2013.

[4] N. Instruments. (2010, 28th of July). *LabVIEW Web Services*. Available: http://zone.ni.com/reference /en-XX/help/371361G-01/lvconcepts/webservices/.

[5] N. Instruments. (2013, 28th of July). *Accessing a Deployed LabVIEW Application*. Available: http://zone.ni.com/reference/en-XX/help/371361G-01/lvconcepts/ interacting_vis_ws/.

[6] N. Instruments. (2010, 28th of July). *Viewing and Controlling Front Panels Remotely*. Available: http://zone.ni.com/reference/en-XX/help/371361G-01/lvconcepts/viewing_fp_remote/.

[7] PMD. (2013, 11th of Feb). *MC73110 Developer's Kit*. Available: http://www.pmdcorp.com/advanced-motion-control/mc73110.cfm

[8] L. Ljung, *System Identification: Theory for the User*, 2nd ed. ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999.

[9] N. Instruments. (2010, 16th of April). *Box-Jenkins Model Definitions (System Identification Toolkit)*. Available: http://zone.ni.com/reference/en-XX/help/372458C-01/lvsysidconcepts/modeldefinitionsbj/

[10] U. Mackenroth, *Robust Control Systems: Theory and Case Studies*. Berlin, Germany: Springer Verlag, 2004. http://dx.doi.org/10.1007/978-3-662-09775-5

[11] K. Zhou and J. Doyle, *Essentials of Robust Control*. New Jersey: Prentice Hall, 1998.

[12] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. New York: Wiley, 1996.

[13] D. Hoyle, R. Hyde, and D. Limebeer, "An $H_\infty$ Approach to Two Degree of Freedom Design," in *30th IEEE Conference on Decision and Control*, 1991, pp. 1581-1585 vol.2.

[14] N. Instruments. (2012, 30th of April). *SIM Optimal Design VI*. Available: http://zone.ni.com/reference/en-XX/help/371894G-01/lvsim/sim_optimal_design_vi/

[15] N. Instruments. (2013, 20th of February). *NI LabVIEW FPGA Module*. Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/11834.

## AUTHORS

**S.D. Gadzhanov, A. Nafalski** and **Z. Nedic** are with School of Engineering, University of South Australia, Mawson Lakes, Australia.

Submitted 03 May 2014. Published as resubmitted by the authors 13 September 2014.