Optimization of Industrial Conveyor based SCADA System using Machine Learning Techniques

Bob Chege Mugo
Department of Electrical and Electronics Engineering,
University of Nairobi,
Nairobi, Kenya

ABSTRACT

SCADA stands for Supervisory Control and Data Acquisition. This is a system that monitors and controls a plant or equipment in industries. Its application is in a wide range of industries such as power, telecommunication, water distribution, waste control, energy, oil refineries etc. A SCADA system comprises of data transfer between a central host computer and several other units. Those units are either RTUs(Remote Terminal Units) or PLCs(Programmable Logic Controllers), and operator terminals.

Many of the traditional SCADA systems utilized open loop control systems but most modern ones utilize closed loop systems where certain configurations are setup usually by a controller via a HMI(Human Machine Interface). The SCADA system maintains the actuators within certain parameters for relevant configured variables.

Because of the large amount of data provided by the SCADA systems on an almost real-time basis usually via a GUI(Graphical User Interface) or back-end provisioned sensors and controllers, they are prime candidates for utilization of Al and ML techniques.

Currently a lot of utilization of ML is done to identify alarms and even predictive analysis of the same. In addition, ML(Machine Learning) is utilized in the determination of cyber threats by analyzing incoming traffic patterns via IP and bandwidth for example in cases of DDOS(Distributed Denial of Service) attacks.

If these large systems can use the data to progress to a higher level of self-autonomy, this may release the human operators to more meaningful and interesting tasks and not the mundane.

This paper is interested in a study of how to use of data provided via a conveyor based SCADA system model to attempt self-error identification, self-correction, and self-healing functionalities using ML based control techniques.

To this end, because of the steep costs in setting up a fully-fledged system, this paper will utilize modeling tools that will simulate parts of an industrial production line managed via a SCADA system. In addition, this system will test a limited scope section of the model via an industrial logic controller namely Siemens S7-1200 PLC interfacing with a DC(direct current) motor actuator and with an STM32F4 Nucleo Board as a controller.

General Terms

Model Predictive Control, MPC, Nonlinear Auto-regressive Moving, Average-L2, NARMA-L2

Keywords

SCADA, Machine Learning, Artificial Intelligence, Model Predictive Control, MPC, NARMA-L2, Nonlinear Auto-regressive Moving Average-L2

1. INTRODUCTION

Modern Industrial systems are composed of a PLC utilizing PID (Proportional Integral Derivative) algorithms, RTUs such as motors, feedback sensors, data transmission protocols i.e. Modbus, ICCP (Inter-Control Center Communications Protocol) and a SCADA system. These systems act as controllers to large and complicated engineering systems such as electrical power grids, subway transportation systems, factory management systems and security systems such as for the Kenya Police Surveillance system.

The secure and proper operation of these systems is usually as a critical and important concern for various utility providers, private companies, and governments. Currently, in many of these setups, the structural features and variables are configured before commissioning and usually vary as required by a human operator. Successful system operation has always been dependent on constant analysis of the input and output values by an operator who then adjusts accordingly.

While SCADA systems have improved productivity and reduced labor requirements by adding of intelligence to the depiction and analysis of the network, they have remained static in terms of predictive analysis of the industrial manufacturing process and alarm identification and fault correction.

Some research has been expended towards detection of faults and sub optimal production. In this paper, various AI techniques will be utilized to take this further and develop the idea of a model of “super SCADA systems” that detect the prior mentioned issues and have capacity to control and re-calibrate without human operator intervention.

The SCADA system is part of an ICS (Industrial Control Systems)

and is composed of a central host computer and RTUs, operator views/terminals which interface with PLCs via the communication infrastructure.

SCADA Systems are very important and utilized for:

1. Monitoring of Electrical and Mechanical equipment.
2. Alerts and Events management.
3. Controlling and Orchestration of Industrial Processes.

Most SCADA systems generate alerts and events based on certain prior configured conditions. While these are essential for on-site operations and maintenance teams, this information is usually

after the fact. In addition, there are also several times where the alarm systems have been turned off or ignored due to many generated alarms. Most of these alarms are carefully reviewed after a component or machine malfunctions.

In addition, there is no predictive analysis of component wear and tear and this results in down-times during fault resolution. Where such component information exists, it is a prior configured global variable that is not always representative of operating environment. In addition, capacity production parameters are generally fixed by an operator usually periodically and at the operator discretion. This leads to over/under utilization of production levels and resultant costs.

For example, training of industrial actuators such as robots is usually a costly tedious affair and a slight re-purposing would involve software rewrites even where there is no need to modify the relevant hardware. Also, because of a lack of interconnection between Industrial Systems in real time, a fault or a more efficient process cannot be transmitted to similar nodes located remotely and would require the vendors intervention for resolution.

The following are the specific objectives that this paper hopes to achieve.

1. To present a developed training model for sample industrial components using AI.
2. To show results of an investigation performance modeling and thus determine normal performance parameters as compared to utilization of ML.

Al is a broad area. There are many various areas in the broader ecosystem of Al[2], ranging from text mining to deep learning and recommendation engines. AI use cases have been finding their way into industrial processes which are managed by SCADA processes.

In the paper [4], Takagi and Sugeno use a SCADA system with the boiler system model and Fuzzy logic to control PID parameters and to increase stability of control system.

The integration of industrial systems with IoT will provide real time sharing of component information over the Internet and this will further increase production by reducing down-times.

A major contributing factor to the operators' problem is that there are just too many interacting complex processes with too many variables and accompanying issues to watch and control. This is an unintended consequence of installing powerful SCADA systems and the many instruments available[5].

This research topic will enable an analysis into utilization of ML models in industrial automation processes as follows:

1. Evaluate viability of ML models to enable more widespread adoption in industrial automation processes.
2. To encourage data sharing and lessons learnt in terms of machine adoption and usage and processes used among industrial nodes.
3. Evaluate on training models for industrial operations repurposing.

The scope of this will involve research into the AI techniques that will be utilized in a sample SCADA production line and their implementation. These techniques will be utilized in a sample actuator process to check on their efficacy. For the simulation of the industrial processes, MATLAB will be utilized while to develop the AI algorithms. Several AI models including MPC and NARMA-L2 will also be utilized to develop comparative models.

In addition, to program the STM32F446RE PIC, C Language will be utilized. Ladder logic will be utilized to program the Siemens S7-1200 PLC and HMI via TIA portal software.

Thereafter an analysis of the following will be done:

1. Analysis of production performance of ML algorithm on industrial process.
2. Analysis of ease of re-purposing of an RTU or actuator.
3. Predictive analysis of capacity management and alarm event handling.

2. SYSTEM IDENTIFICATION MODEL DERIVATION

In this section of the paper, there is need to determine the plant of the system. Since the consideration involves looking at a conveyor belt as this could be considered ubiquitous and available in most production lines, this papers task is to determine what to model. Since all conveyors are driven by a motor in practise, it seems most practical to consider the motor as the plant of the system.

The motor determines the speed and position control of the conveyor and all items placed on them, it can be regarded as the system plant. It will be controlled by a Siemens S7-1200 PLC for start/stop since this is a normal system available in normal factory plants. It is usually supported by a speed controller namely a VFD(Variable Frequency Drive) that is coupled directly to the induction motor.

An STM32F446RE microcontroller will be utilised instead and it will be coupled to a 12V rated DC(Direct Current) motor due to the highly prohibitive costs for using an actual VFD coupled to an induction motor. Additionally, utilizing an induction motor and VFD requires three phase AC(Alternating Current) power which introduces safety and availability concerns as well.

Below is a basic electrical circuit diagram of the conveyor motor model highlighting the most important sections of the motor including the resistance, inductance, the actuator as well as the free body diagram of motor rotor as shown in figure 1.

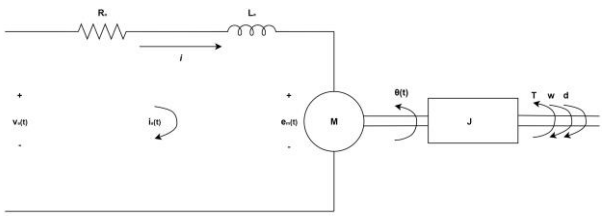

Fig 1. Motor State Space Model Circuit of DC Motor indicating electrical parameters of armature resistance R_a, inductance L_a, current I_a and Voltage V_a and mechanical properties such as moment of Inertia J, Torque T, w, d.

Also included in the table 1 below is a listing of all the relevant elements that will be utilized in the modelling process. Table 1. Motor Parameters and Constants.

Symbol	Unit	Definition
v	Volts (V)	Input voltage
i	Ampere (A)	Armature current
em	Volts (V)	Back EMF
R	Ohm (Ω)	Armature Winding Resistance
L	Henry (H)	Inductance of Armature Winding
K_e	Nm	emf constant

T	Nm	Mechanical Torque
w	Nm	Disturbance torque
d	Nm	Damping torque
	Radians	Shaft angular position
θ	Radians/s	Angular speed
Ω		
α	Radians/s^2	Angular acceleration
J	kg m	Moment of Inertia
b	Nms	Viscous Frictional Constant
K_t	Nm	Torque constant

2.1 DC Motor angular/ rotational System model control problem considering disturbances.

To calculate the moment of inertia for the conveyor motor, the following expressions are used.

$$J\frac{d\theta^2(t)}{dt^2} = T - w - d \quad (1)$$

$$J\ddot{\theta} = T - w - d \quad (2)$$

$$J\ddot{\theta} = T - w - d \quad (3)$$

From the motor law, it is determined that:

$$T = K_t i \quad (4)$$

This means that the generated torque is proportional to the armature current. The damping torque can be expressed as follows:

$$d = b\dot{\theta} \quad (5)$$

$$d = b\frac{d\theta(t)}{dt} \quad (6)$$

where b is a damping coefficient. By substituting the last two equations in equation 1 describing the motor mechanical dynamics,the following equation was obtained:

$$J\ddot{\theta} + b\dot{\theta} = K_t i - w \quad (7)$$

$$J\frac{d\theta^2(t)}{dt^2} + b\frac{d\theta(t)}{dt} = K_t i - w \quad (8)$$

The armature current has its own dynamics. From figure 1 , the following equation 9 can be written:

$$L\dot{i} + Ri = v - e_m \quad (9)$$

where e_m is given by the following equation:

$$e_m = K_e\dot{\theta} \quad (10)$$

$$e_m = K_e\frac{d\theta(t)}{dt} \quad (11)$$

By substituting the equation 11 in the equation 9, the following is obtained:

$$L\dot{i} + Ri = v - K_e\dot{\theta} \quad (12)$$

$$L\frac{di(t)}{dt} + Ri = v - K_e\frac{d\theta(t)}{dt} \quad (13)$$

The equations 8 and 13 govern the dynamics of the DC motor as indicated below:

$$J\ddot{\theta} + b\dot{\theta} = K_t i - w \quad (14)$$

$$J\frac{d\theta^2(t)}{dt^2} + b\frac{d\theta(t)}{dt} = K_t i - w \quad (15)$$

$$L\dot{i} + Ri = v - K_e\dot{\theta} \quad (16)$$

$$L\frac{di(t)}{dt} + Ri = v - K_e\frac{d\theta(t)}{dt} \quad (17)$$

The voltage v is an external voltage used to control the motor.

To derive the state space model, the equations 14 and 16 are used in a state-space form which is generally in the form below:

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu + Wdy = C\mathbf{x}f \quad (18)$$

$$\frac{dx(t)}{dt} = Ax + Bu + Wd \quad (19)$$

$$y = Cx \quad (20)$$

where x is a state vector, u is the control input vector, y is the system output (a scalar), and A,B,C and W are the system matrices.

The state-space variables, control input variables, and the output variable are now introduced as follows:

$$x_1 = \theta \quad (21)$$

$$x_2 = \frac{d\theta(t)}{dt} \quad (22)$$

$$x_3 = i \quad (23)$$

$$y = x_2 \quad (24)$$

$$u = v \quad (25)$$

Three state-space variables are necessary due to the overall system order being equal to the same number. The overall system order is the sum of the orders of two differential equations. The order of the first differential equation 14 where the highest derivative appearing the differential equation is 2, and the order of the second differential equation 16 is 1.

The output of the system should be based on observed measured output. For this case, the output variable is the angular velocity $\dot{\theta}$. The control input u is the variable that controls the system external voltage v. From the equations 21, 22 and 23, the following can be derived:

$$\dot{x}_1 = x_2 \quad (26)$$

$$\dot{x}_2 = \ddot{\theta} = \frac{K_T}{J}x_3 - \frac{1}{J}d - \frac{b}{J}x_2 \quad (27)$$

$$\dot{x}_3 = \dot{i} = \frac{1}{L}u - \frac{R}{L}x_3 - \frac{K_e}{L}x_2 \quad (28)$$

$$y = x_2 \quad (29)$$

The state-space block can also be determined in a Simulink model[3] for a nonlinear and linear state-space models and this will be encountered shortly.

2.2 DC Motor angular/ rotational System model control problem WITHOUT considering disturbances.

As indicated above, it is possible to model a conveyor motor so as to obtain the physical and system identification system models in form of a state-space representation. Note that it is also possible to obtain a transfer function of the plant.

Therefore, it is possible to determine the state-space model of the DC motor model using mechanical and electrical methods[1]. The torque (T) generated by the DC motor is

$$T = K_t i \tag{30}$$

where K_t is the constant factor, i is the armature current.

The back-EMF is determined as shown below.

$$e = K_e \dot{\theta} \tag{31}$$

The torque constant and back-EMF constant have the same unit as $K_t = K_e$. The motor can be modeled then by using Newton's second law and Kirchhoff voltage law as follows:

$$J\ddot{\theta} + b\dot{\theta} = K_t i - w \tag{32}$$

$$J\frac{d\theta^2(t)}{dt^2} + b\frac{d\theta(t)}{dt} = K_t i - w \tag{33}$$

$$L\frac{di(t)}{dt} + Ri = v - K_e\frac{d\theta(t)}{dt} \tag{34}$$

The DC motor model can be written in the state-space form utilising the general form as indicated below.

$$\dot{x} = Ax + Bu \tag{35}$$

$$y = Cx \tag{36}$$

This results in more or less the same equation as 26, 27, 28, 29 albeit simpler due to the lack of consideration of disturbance.

3. SIMULATION, RESULTS AND COMPARISONS

For this paper, the following design parameters can be considered.

Table 2. General design criteria objectives.

Settling time	≤	2 seconds
Overshoot	≤	5%
Steady-state error	≤	1%

3.1 Open Loop Simulation

To build out the sub-block from a main model (fig 3.1) and create the various components of the state space model, one can use discrete blocks such as integrator and gain modules as indicated below.

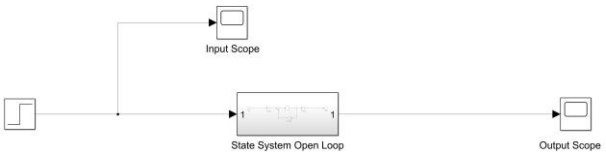

Fig 2. Open loop implementation with step response with input and output scope in Simulink.

The sub-model utilized in the main model is indicated in the figure 3 below.

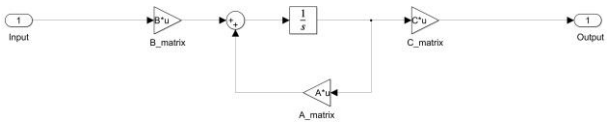

Fig 3. Detailed Simulink sub-block included in the state system open loop block.

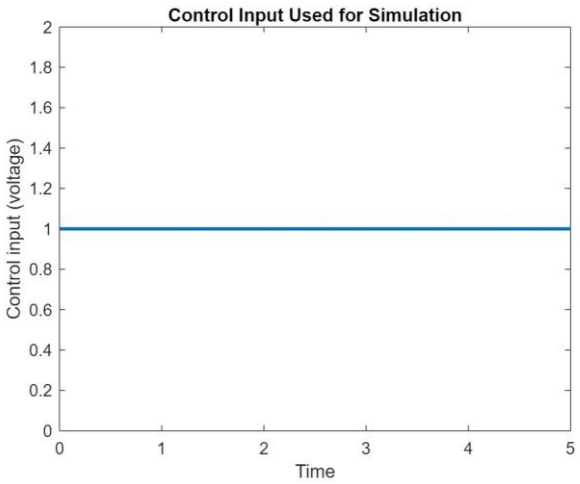

Fig 4. Sample control step input used for the MATLAB simulation.

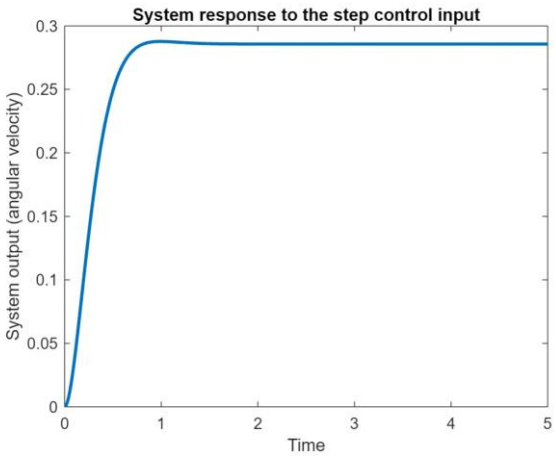

Fig 5. Simulated system response output due to discrete step control input.

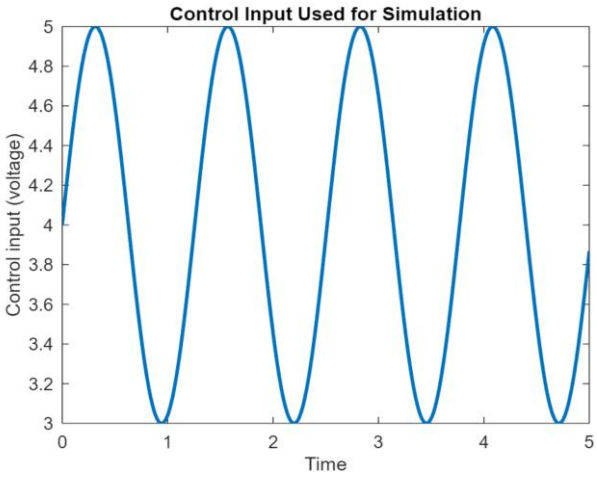

Fig 6. Simulated sample sinusoidal control signal input used for simulation.

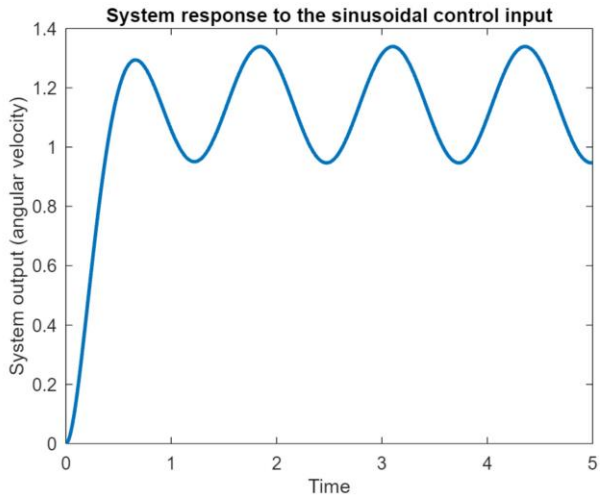

Fig 7. Simulated system response output due to sinusoidal control input.

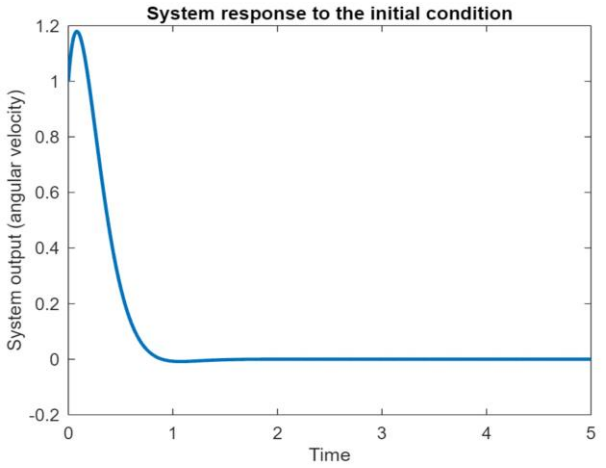

Fig 8. Simulated system response output due to non-zero control input.

3.2 MPC Feedback Simulation

Running the controller as show in fig 3.1 against the simulated plant shows good performance with good step input of 10 while having output which achieves steady state after less than 2s with no overshoot as per requirements on the probes.

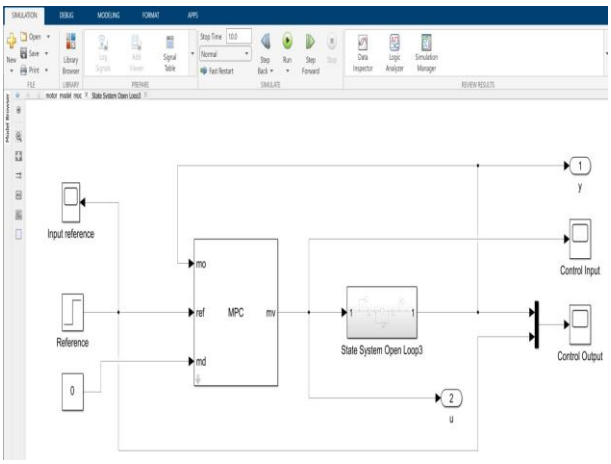

Fig 9. Simulation setup in Simulink with step reference input, MPC controller, open loop system state model with feedback control.

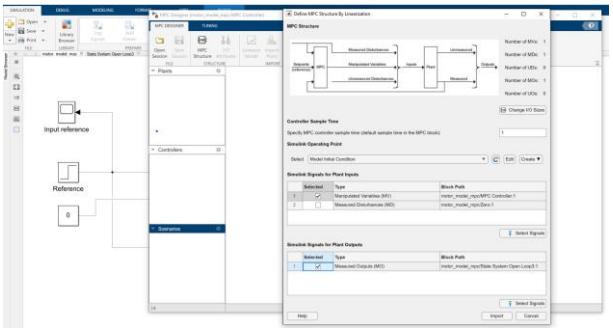

Fig 10. MPC plant configuration with definition of input manipulated variables(MV) and output measured disturbances(MD).

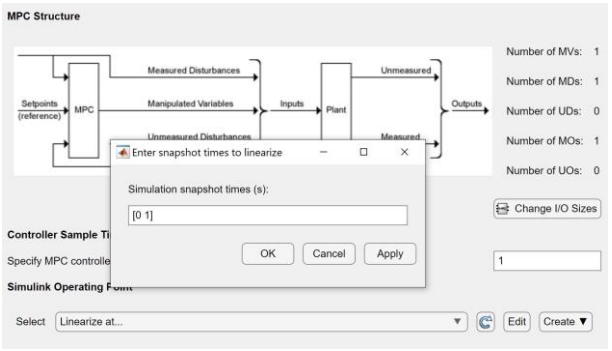

Fig 11. Linearize attempt of MPC with simulation snapshot times defined of [0 1].

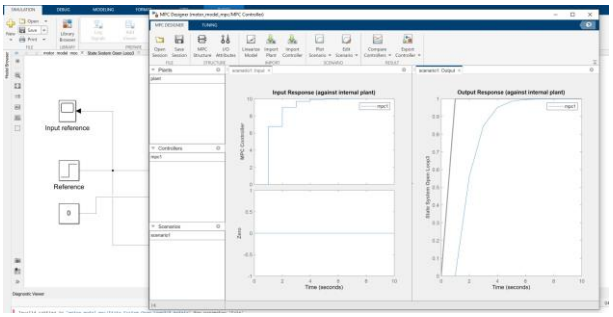

Fig 12. MPC designer output indicating preliminary results showing varied stepped input response(7-10v) and 7s to achieve steady state for output response.

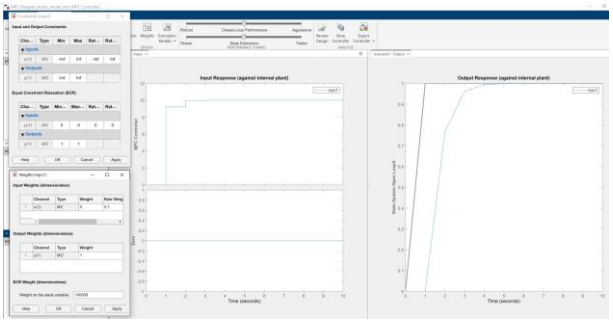

Fig 13. MPC tuning of input weights to 0.1 as well as output weights to 1 showing better stepped input response with improved output response of 5s.

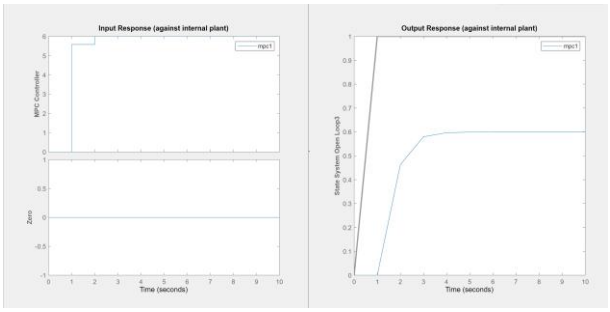

Fig 14. Analysing MPC system response to adjusting input control voltage to 6v showing improved input response and steady state output response achieved in 3s without overshoot.

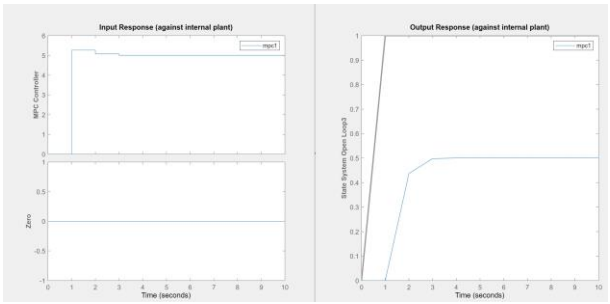

Fig 15. Analysing impact on MPC system response to setting output constraint input to 0.5 showing overshoot in input response of about 0.2v with however smoother output response with steady state achieved in 3s.

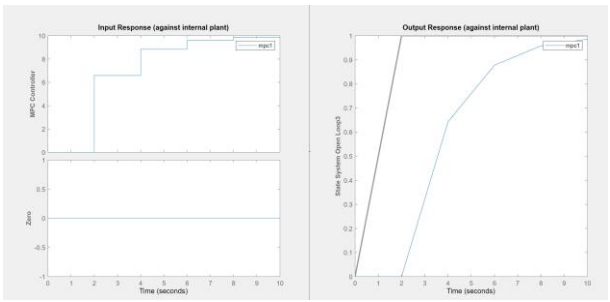

Fig 16. Analysing impact MPC system response to increasing sample time to 25s showing stepped input response to 10v and reduced output response performance with steady state achieved after 10s.

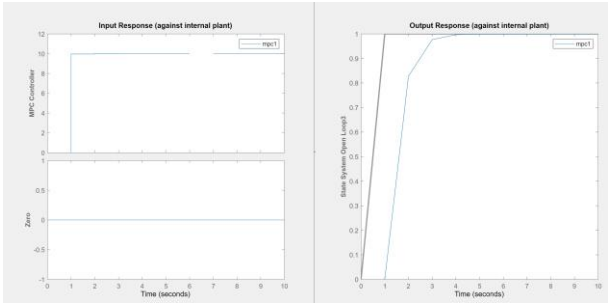

Fig 17. Impact of analysing MPC impact of increasing prediction horizon to 200 showing excellent input step response to 10v and achieving steady state output response in 4s.

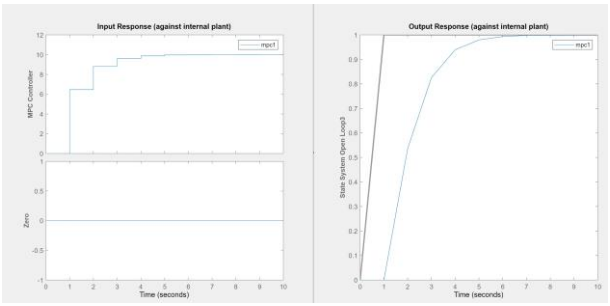

Fig 18. Decreasing control horizon to 100 shows decreased MPC system response with stepped input response and output response with steady state achieved is 7s.

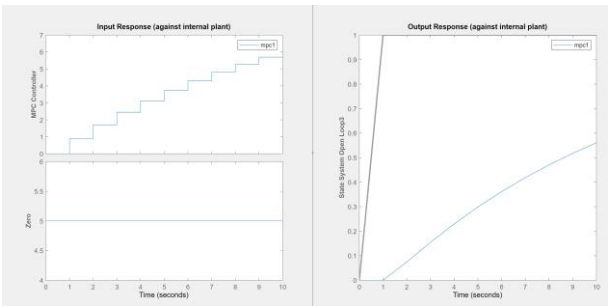

Fig 19. Increasing input weights to 1 shows much reduced MPC system response performance with multiple stepped input values and output response unable to achieve steady state.

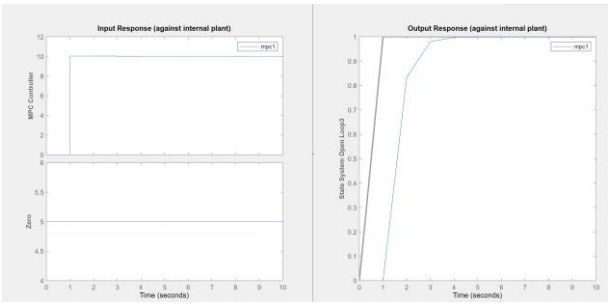

Fig 20. Output of the final chosen parameters shows excellent input step response as well as prompt steady state response within 3s for the final MPC exported controller.

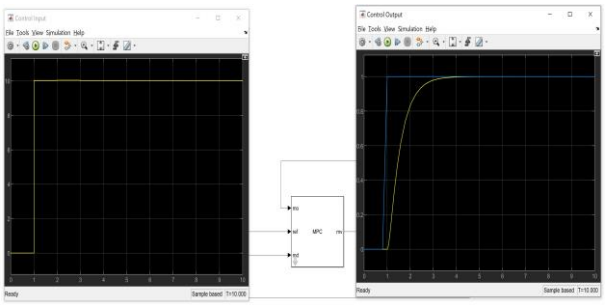

Fig 21. Implementation of the exported MPC controller on step input in Simulink shows good system response.

Table 3. MPC controller simulation results summary.

Parameters	rise time(sec)	peak overshoot	steady state error
Desired Specs	≤ 2 seconds	≤ 5%	≤ 1%
MPC Controller	1˜ second	≤ None	None

3.3NARMA-L2 Feedback Simulation

In this section, it is considered to simulate a NARMA-L2 controller with a step input. In this case the simulated controller is used to effect a plant model of a simulated motor control system as indicated in figure 22. The subsystem utilized here is basically the same plant model as before in figure 3. A detailed view of the simulink block diagram is shown in figure 23.

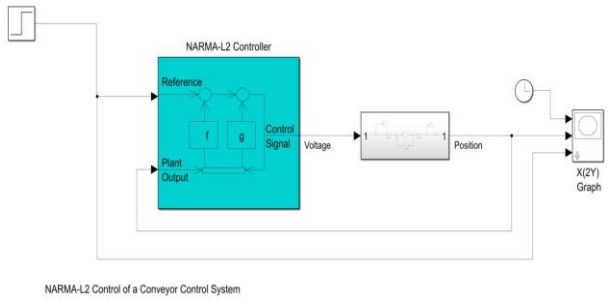

Fig 22. Detailed Narma-L2 Simulink feedback model with step input response and open loop system block. The NARMA-L2 controller has the reference input and plant output as input and computed voltage as output.

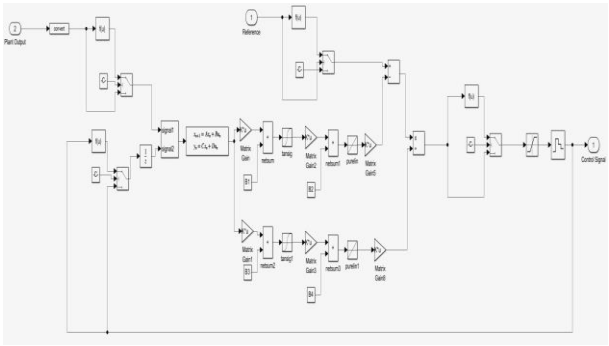

Fig 23. Narma-L2 Simulink block diagram with detailed nodes and the state space model included as well as the feedback paths.

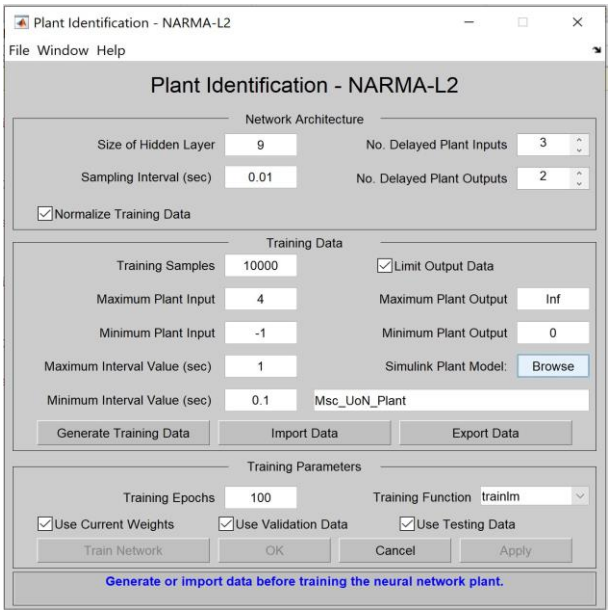

Fig 24. Configured NARMA-L2 training model detailing the number of layers 9, Sampling interval 0.01, Training samples 4 and training epochs 100.

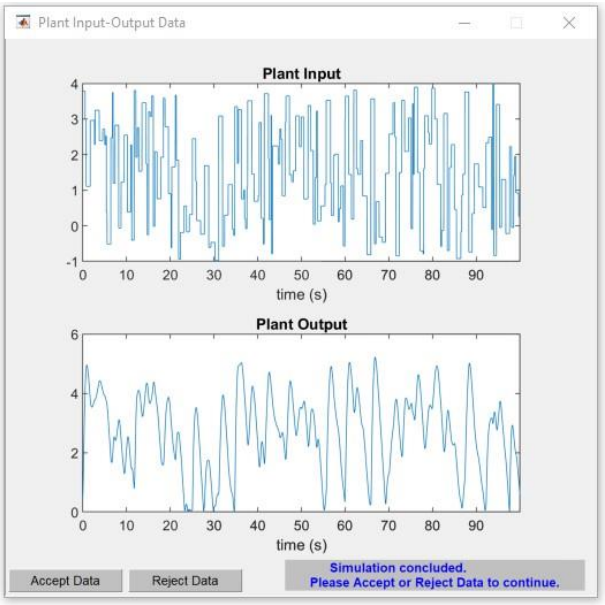

Fig 25. Plant input and output data due to run of NARMA-L2 simulation training showing that there is some fidelity between input and output data. The closer these two graphs look indicates the better training of the model.

Fig 26. NARMA-L2 neural network training progress details implementing the Levenberg-Marquadrt Algorithm.

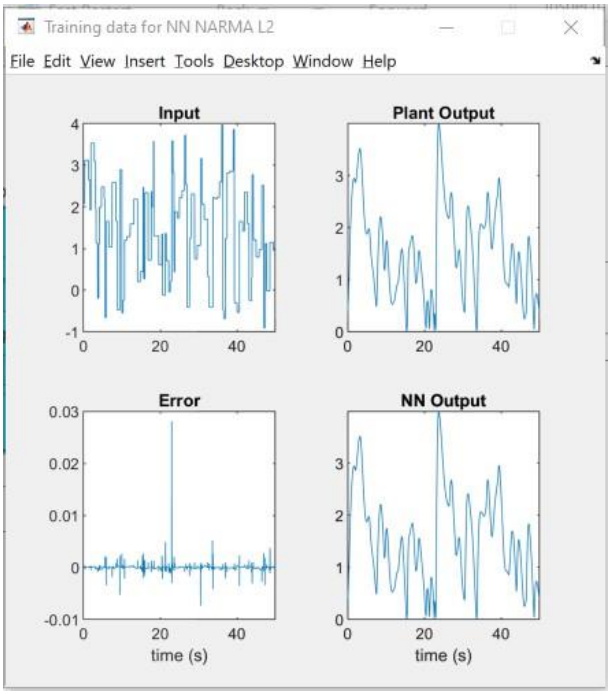

Fig 27. Several graphs indicating input, plant output, error and NN output. During training it is seen output trying to track input and minimize error while indicating the NN output.

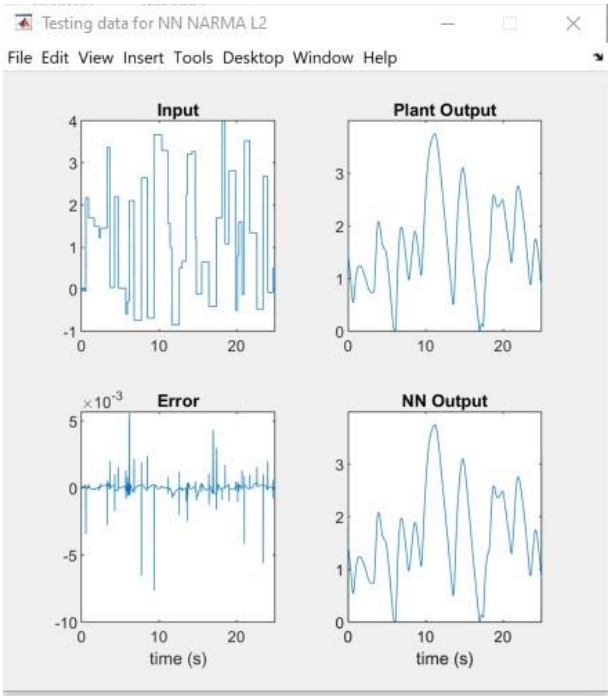

Fig 28. The graphs indicate the final results after subjecting the trained controller to the testing data and deriving the plant output due to test data input as well as error.

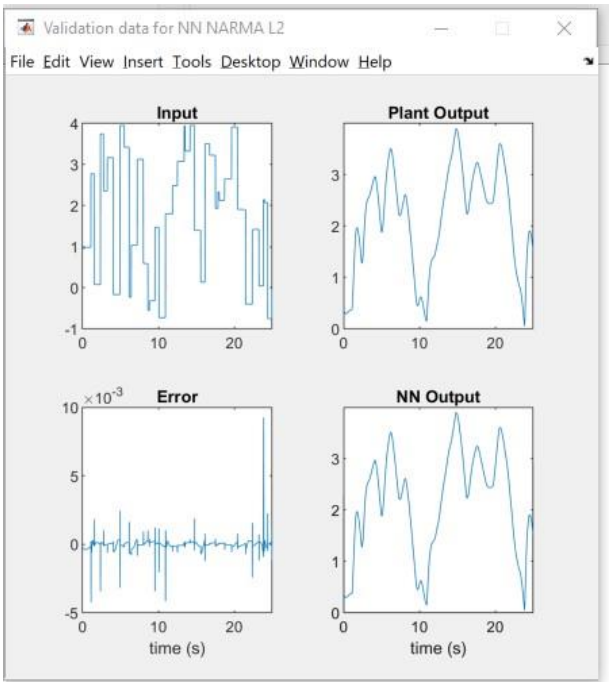

Fig 29. The graphs indicate the results of running NARMA-L2 validation data including input, output, error and NN output.

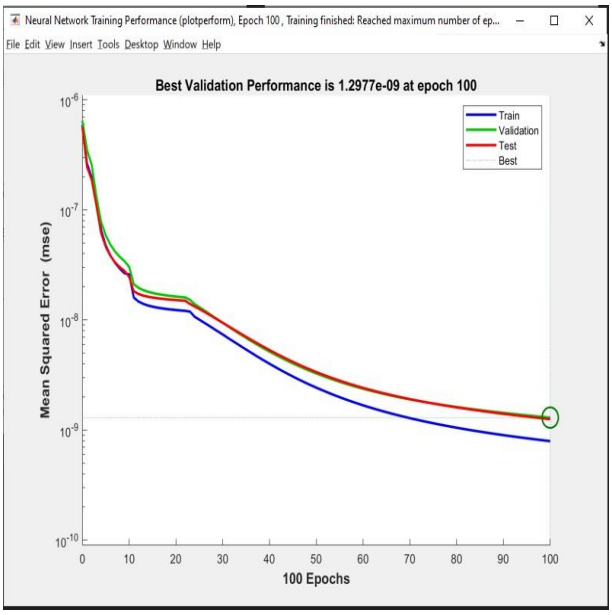

Fig 30. Summary of validation performance graphs showing Training, Validation and test data vs Mean Squared Error(mse).

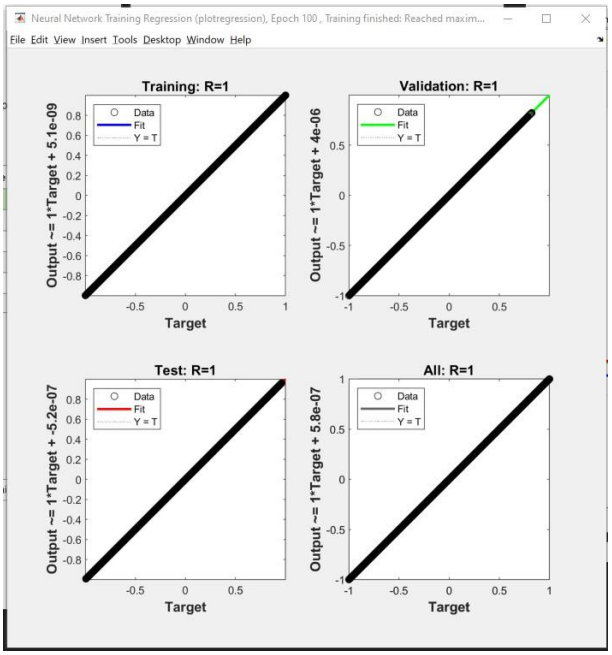

Fig 31. Neural network regression information after training is completed.

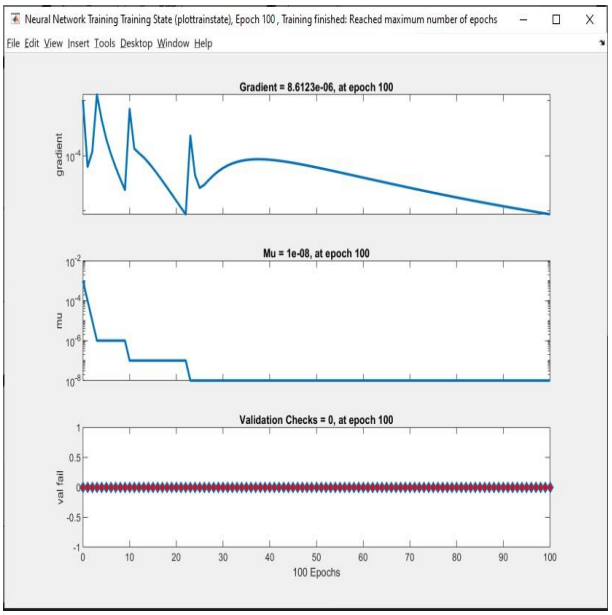

Fig 32. Neural network training state graph showing that the validation criteria has been achieved.

Table 4. NARMA-L2 controller simulation results obtained.

Parameters	rise time(sec)	peak overshoot	steady state error
Desired Specs	≤ 2 seconds	≤ 5%	≤ 1%
NARMA-L2	1.15˜ seconds	≤ 1%	None

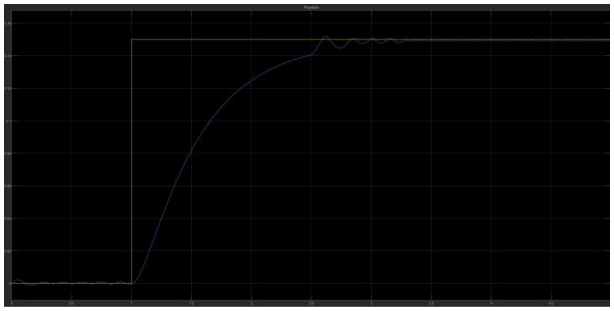

Fig 33. Output of the NARMA-L2 final controller run showing how the output is tracking the input step input. It shows that steady state is achieved in 3.5s with very little overshoot and undershoot.

Implementing the generated controller on the plant results in the following system graphs which shows the input responding to the step output though with some residual steady state error oscillation from figure 25 to figure 33.

4.RUNNING PHYSICAL SYSTEM

In the figures below from figure 34 to figure 57 , it is shown the major components that are implemented in this paper to implement and test the various feedback control protocols.

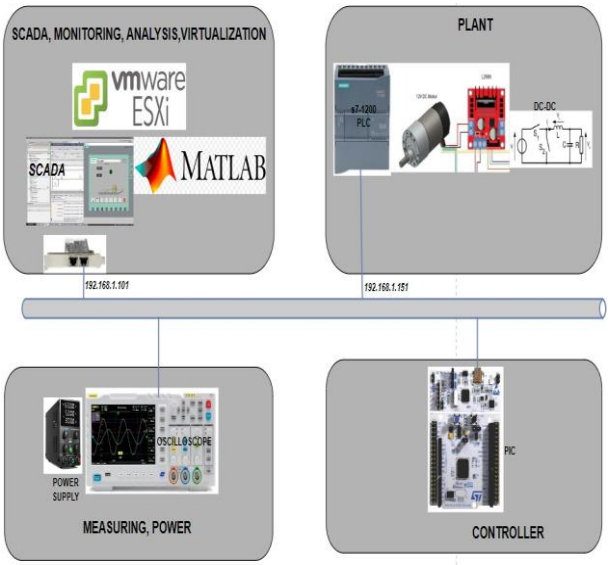

Fig 34. Physical architecture diagram indicating SCADA systems, physical plant circuitry, measuring setup and microchip controller.

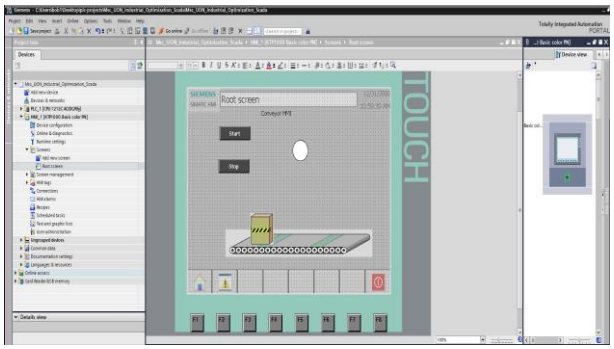

Fig 35. Siemens TIA HMI view indicating the main component modules namely motor conveyor belt, start/stop button as well as running indicator and its integration to S7-1200 PLC.

Algorithm 1 Plant Controller Algorithm

Require: $PulsesPerRev \geq 260$
Require: $pulseCount \geq 0$
Require: $rpmToRadians \geq 0.10471975512$
Require: $radToDeg \geq 57.29578$
Require: $rpm \geq 0$
Ensure:
$GPIOMotor \leftarrow Initialized$
$GPIOMotorStartStopNBI \leftarrow Initialized$
$GPIOMotorStartStopSBI \leftarrow Initialized$
$GPIOAPeripheralClockControl \leftarrow Initialized$
$GPIOCPeripheralClockControl \leftarrow Initialized$
$GPIOMotorEncoder \leftarrow Initialized$
$InterruptConfigEXTI15-10 \leftarrow Enabled$
$InterruptConfigEXTI0 \leftarrow Enabled$
$Usart2IRQInterruptConfig \leftarrow Enabled$
$GPIOMotorStartStopNBI \leftarrow Started$ ▷ This resets the motor by setting ENA to 0
InitializeUsart2TxRx() ▷ This function initializes UART2 on PA2 and PA3 for Tx and Rx
UART2WriteENA() ▷ Initialize USART buffer
SystickConfig(16000000); ▷ This function enables timer to every second
while 1 **do** ▷ This is the background process to run the controller
end while
EXTI15-10IRQHandler() ▷ Function interrupt for PC10
EXTI0IRQHandler() ▷ Function interrupt for PA0
EXTI1IRQHandler() ▷ Function interrupt for PA1
Usart2IRQHandler() ▷ Handle the USART PA2 and PA3

Fig 36. Summary algorithm implementation on STM32F4 showing function calls, interrupts/interrupt-handlers and code flow.

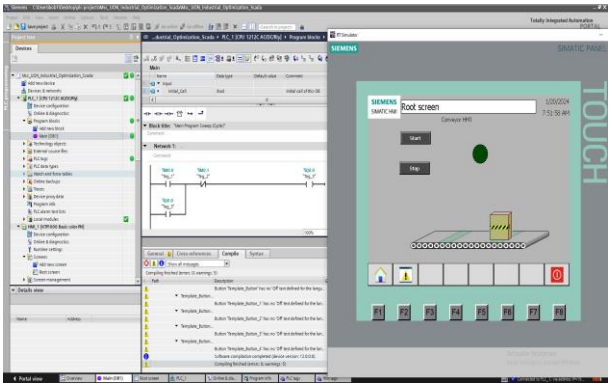

Fig 37. Running ladder logic in TIA HMI when the start button has been activated(Green).

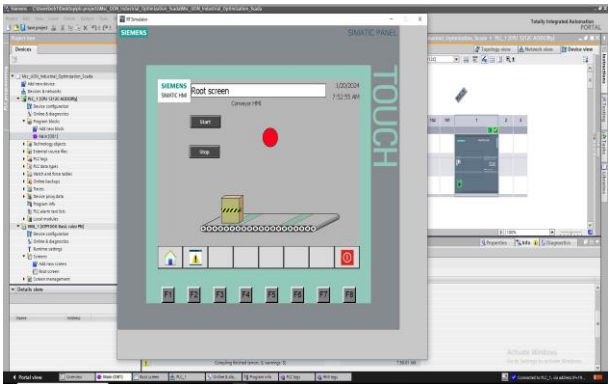

Fig 38. Running ladder logic in TIA HMI when the start button has been de-activated(Red)

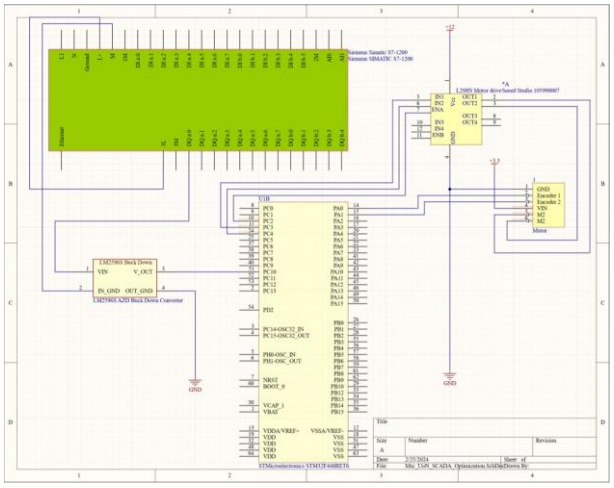

Fig 39. Electrical schematic design between plant components. This includes the microcontroller, Siemens -7-1200 PLC, Buck-Down converter and DC motor.

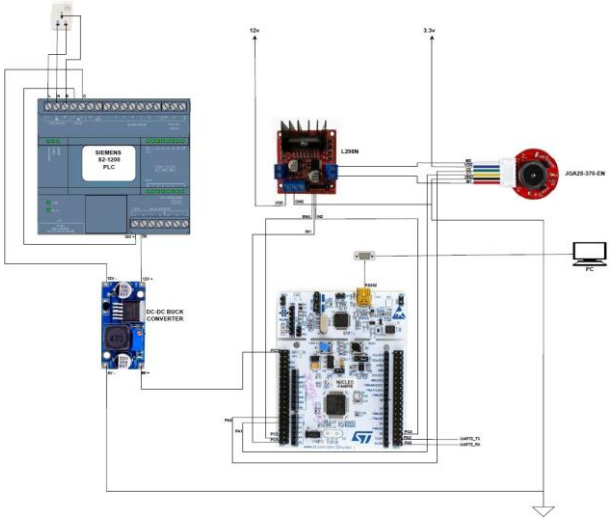

Fig 40. Logical cabling connection showing the connections between various components in the plant.

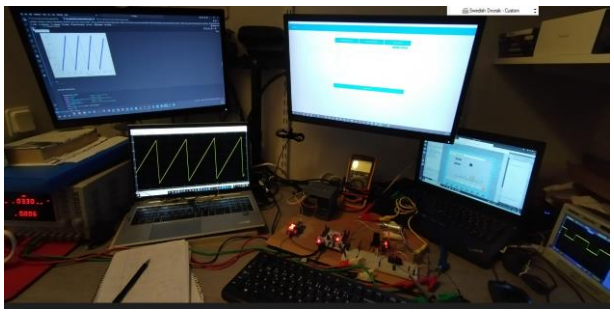

Fig 41. end-to-end physical running setup with all the components including SCADA HMI, microcontroller plant system and measuring oscilloscope included.

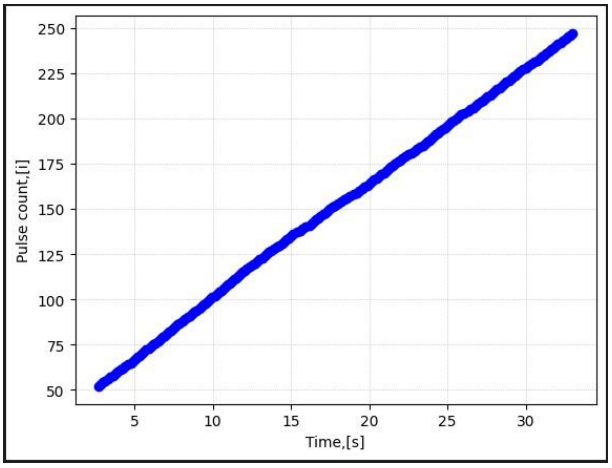

Fig 42. Linear pulse count vs time output of the motor obtained in the STM32F4 microcontroller.

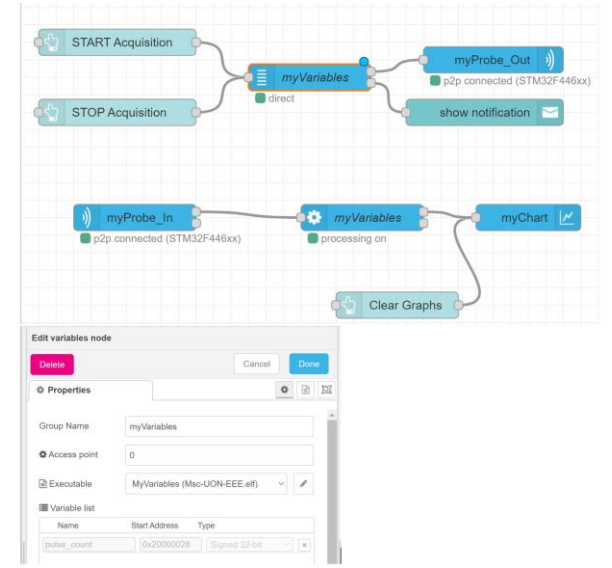

Fig 43. Pulse counter variable configuration in the STM32 Cube Monitor showing myVariables probe that reads the memory store in the S7-1200 PLC.

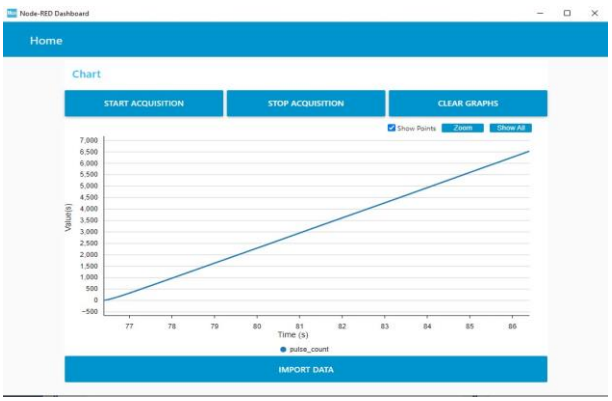

Fig 44. Output of the pulse counter showing the pulse count increasing in linear regression list.

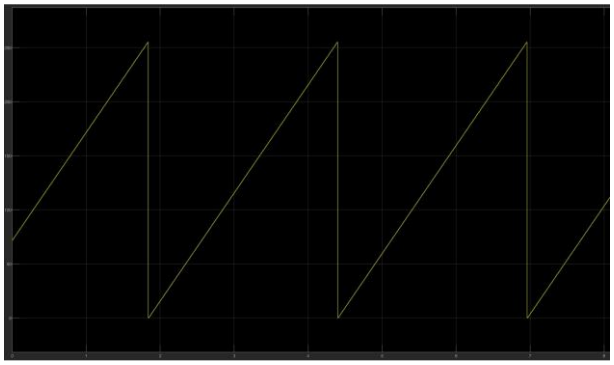

Fig 45. MATLAB pulse counter scope showing that the pulse increases linearly but with periodic resets resulting in a saw-tooth graph.

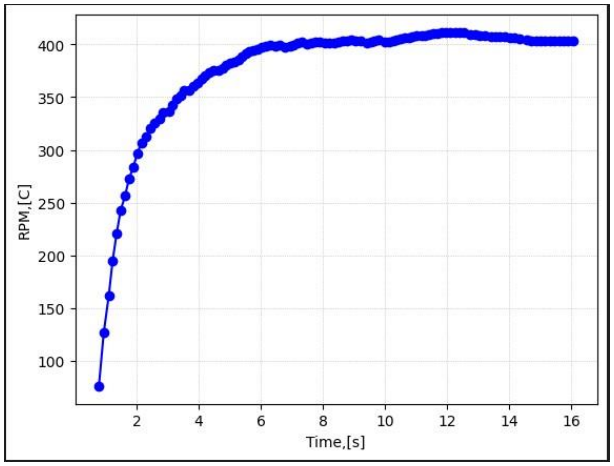

Fig 46. Calculation of RPM via python visualization shows that the value increases rapidly and achieves steady state in 6s. There is however slight overshot and undershoot of about 10RPM for the next 10s.

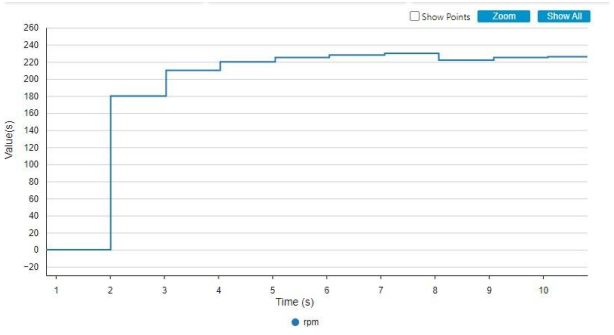

Fig 47. The same RPM calculation viewed cube monitor which is a discrete output indicating the same time 6s to achieve steady state within 10RPM.

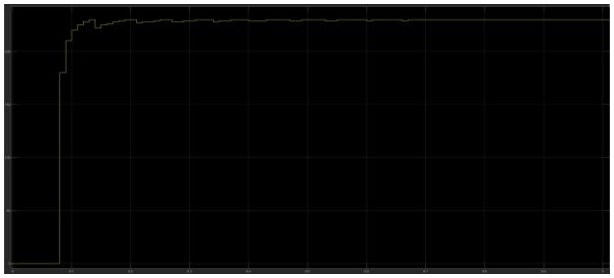

Fig 48. The same RPM calculation viewed in MATLAB scope which is a also discrete indicating the same time 6s to achieve steady state within 10RPM

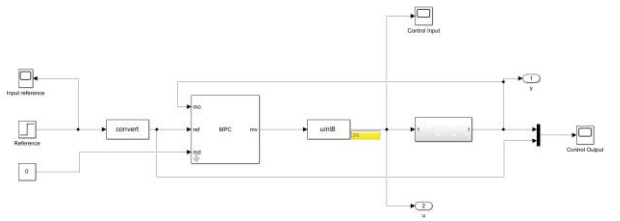

Fig 49. The physical main MPC controller integrated with the physical system via USART in the open loop block. The input data type is cast/converted to uint8 for processing required by the microcontroller code. A step input is invoked from the Simulink GUI.

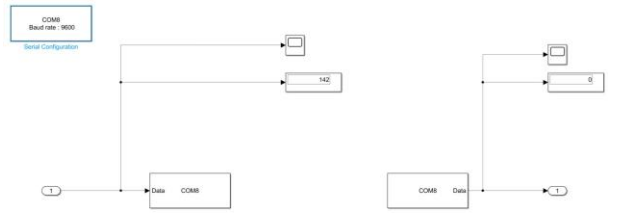

Fig 50. The implementation of USART integration with the attributes defined to microcontroller physical plant configuration namely via COM8, Baud-Rate 9600.

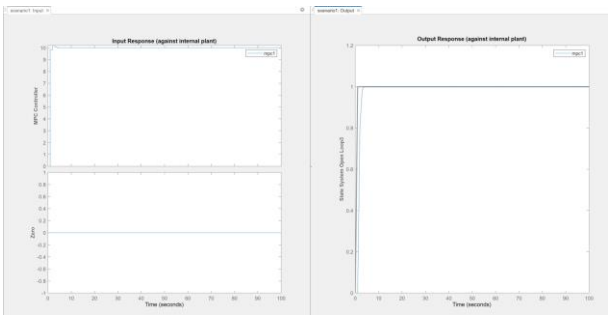

Fig 51. Input and Output response of the MPC Controller that will be integrated to the physical plant showing good input characteristics with slight but acceptable input while having excellent output characteristics.

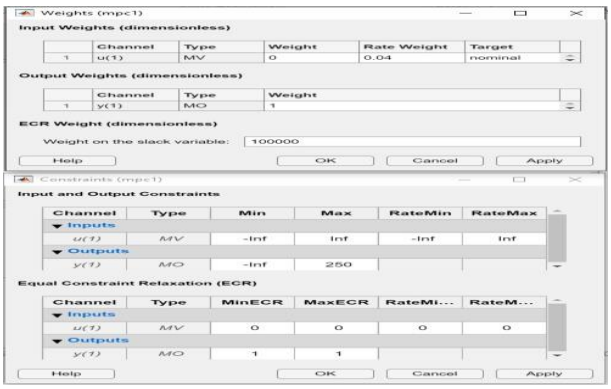

Fig 52. The definition of the actual input and output weights and constraints for MV and MO.

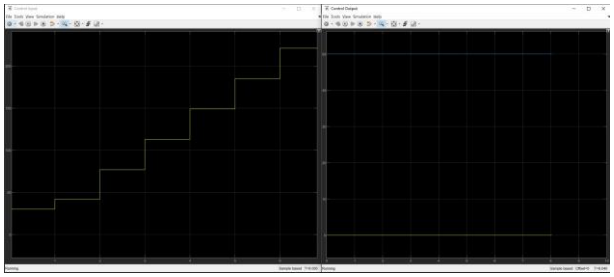

Fig 53. MPC sanity check performed indicating that output increases due to tracking high reference as compared to control input as nominal behaviour of controller.

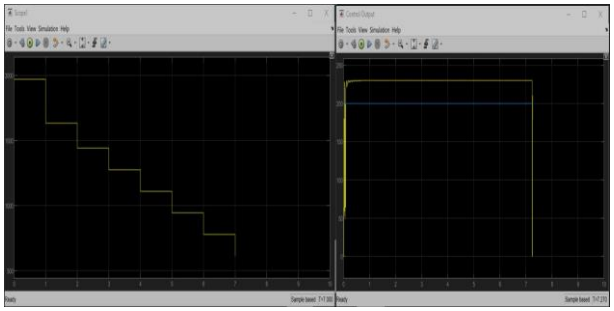

Fig 54. MPC sanity check performed output decreases due to tracking low reference as compared to control input as nominal behaviour of controller.

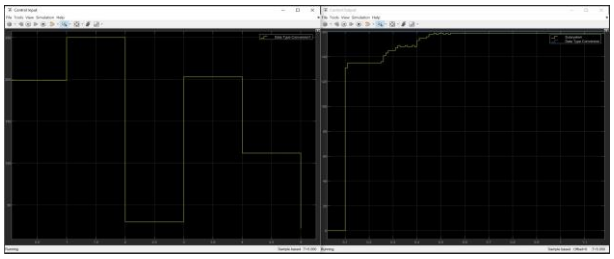

Fig 55. MPC run on physical system indicating that actions of the MPC controller to achieve steady state. Graph on the left shows the controller increasing/decreasing input to the physical system to achieve steady state while graph on the right shows the output system response.

It is noted that the MPC controller and the NARMA-L2 controllers are able to successfully achieve the desired parameters as indicated in table 5 and 6.

Table 5. MPC Physical Run Results.

Parameters	rise time(sec)	peak overshoot	steady state error
Desired Specs	≤ 2 seconds	≤ 5%	≤ 1%
MPC Controller	0.8˜ seconds	0%	1%

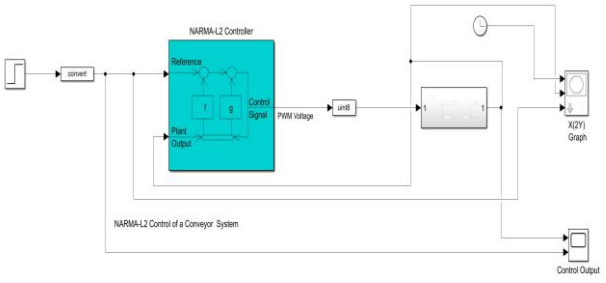

Fig 56. Integration of NARMA-L2 controller with physical plant system implemented on microcontroller system via USART. A step input/output is introduced to the controller via cast/conversion block to uint8.

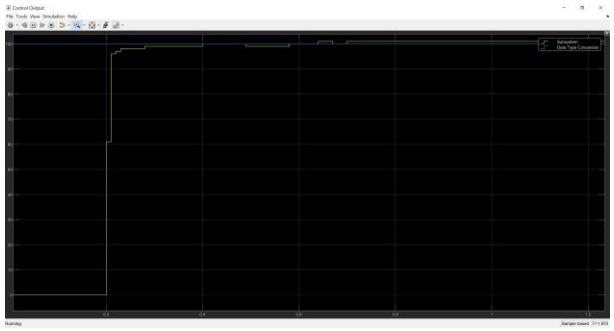

Fig 57. System response output due to NARMA-L2 control showing good response within 1s. The response is noted to have better system response.

Table 6. NARMA-L2 Physical Run Results.

Parameters	rise time(sec)	peak overshoot	steady state error
Desired Specs	≤ 5 seconds	No Overshoot	≤ 1%
NARMA-L2 Controller	0.5˜ seconds	0.5%	1%

5.CONCLUSIONS AND FUTURE WORK

This paper attempted speed control of a SCADA based conveyor system utilizing both simulated and physical test system. This tests were done utilizing ML techniques namely MPC and NARMAL2 as the feedback controller. In the first phase, simulation of the entire system was implemented on various virtual systems and it was able to successfully control the speed output within the desired parameters.

In the simulated setup, it is noted that it was it is much simpler to manipulate various parameters and quickly analyse the output.

In the second phase the speed control was also attempted utilizing a physical industrial PLC control as well as the on a physical motor and PIC. It was noted that it is relatively more difficult to implement this end to end due to the need for knowledge of more technologies.

Additionally, it is more difficult to meet the desired parameters such as steady state error and performance due to the issues of interacting with real electrical connections.Therefore this objective was seemingly successfully achieved.

The next phase of these works would be to implement embedded controller within the PLC. The current implementation uses an inline controller and this takes substantially more time to execute in real time even though on the simulated time is quite brief.

This is because the simulation depends on the computer resources such as CPUs and memory required to execute the software. To have the embedded system working in real life, it would be better to develop the code and load it in the PLC directly. This would be possible via a direct implementation of an optimization principle and a linearization algorithm or by use of open source software available.

6. REFERENCES

[1] Control of dc motor using integral state feedback and comparison with pid: Simulation and arduino implementation. Available at https://www.researchgate.net/publication/348272348_Control_of_DC_Motor_Using_Integral_State_Feedback_and_Comparison_with_PID_Simulation_and_Arduino_Implementation (2023/05/22).

[2] Outlook on artificial intelligence in the enterprise 2016.pdf. Available at http://www.datascienceassn.org/sites/default/files/OutlookonArtificialIntelligenceintheEnterprise2016. pdf (2023/05/22).

[3] State space control systems-matlab simulink approach. Available at https://www.morganclaypoolpublishers.com/catalog_Orig/samples/9781681739793_sample.pdf (2023/05/22).

[4] Jiri Kocian, Stepan Ozana, and Jiri Koziorek. An approach to optimization of takagi-sugeno type fuzzy regulator parameters by genetic algorithm from mamdani regulation surface. *Applied Mechanics and Materials*, 248:545–550, 12 2012.

[5] Tony Lange and Johann Botha. An overview of intelligent control systems in cement plants: shop floor to boardroom. *Instrumentation Measurement Magazine, IEEE*, 7:20 – 25, 01 2005.