Incremental Quadrature Encoder-Decoder Design, Simulation and Implementation in Angular Position Servo Control System

Asma Eswehli
ameswehli@elmergib.edu.ly

Izziddien Alsogkier
iaalsogkier@elmergib.edu.ly

Elmergib University/Department of Electrical and Computer Engineering \ Alkohms, Libya

Article information

Key words

Quadrature encoder-decoder Sensor; angular position servo control system; real time control system.

*Received 23 **02** 2025,*
*Accepted **07 0** 2025,*
*Available online **09 04** 2025*

Abstract

In this paper, a simulation model for an incremental relative quadrature encoder is designed based on sine function and developed to generate standard quadrature signals usually called phases A and B, the model is programed or built up by using Matlab-Simulink environment. Consequently, 1x and 4x resolution decoder algorithms are designed and developed to decode the incremental signals generated by the encoder simulator. Together, the encoder and the decoder algorithms are combined to construct and emulate a realistic rotary incremental angular position sensor. The sensor emulator is used in closing the loop of feedback angular position servo control systems, to test the impact of incremental sensors on the performance of the feedback servo control systems, and to assess the effect of the sensor resolution on the error between the desired value and the actual output of the system not the measured one. Finally, the developed algorithms in the form of Matlab-Simulink models are compiled and implemented successfully to control a simple real angular position servo control system with a very low resolution sensor in real time.

I. Introduction

Linear position sensors and rotational angular position sensors are very important in realization of servo control systems with a wide range of industrial as well as domestic applications. Specially, angular position sensors are needed in measurement and control of rotational machines, dc and ac motors, robotic servo systems, also in automobile applications, for example, anti-lock braking systems, traction control systems, electronic stability control systems, internal combustion engine control, for example, crankshaft position sensor, etc. [1].

There are many categories to classify the measuring processes of the angular position, for example, absolute and relative. Absolute gives away the absolute position directly, while the relative gives indication on how many and quick steps are taken usually encoded as pulses which need to be decoded and counted to reveal the position. There are many ways the incremental encoders are constructed and realized, some of them for example, generate their pulses through electrical contacts as in the case of the simple rotary switch, others sense magnetic or reluctance changes, but most of them are based on optical sensors [2, 3, 4, 5].

Incremental encoders and their decoders are the most frequently used sensors in position measurement, also its derivatives, velocity and acceleration measurements, and they are implemented in position and velocity servo control systems [6, 7, 8, 9]. Hence, it is important to study and understand their impact of implementation in closing the loop of feedback servo control systems, in order to see, how much they will affect the performance of the closed loop, particularly, in low resolution range.

Therefore, in this work, a simple model is designed and developed for a general rotational relative incremental quadrature encoder as well as its counterpart decoder algorithms. Together combined, the encoder and the decoder, they emulate a real sensor behavior that can be intensively tested and studied in simulation as well as in real time hardware implementation, particularly, by using the Matlab-Simulink environment and its real time control libraries to control a simple hardware test platform of angular position servo control system.

The structure of the paper is as following, in the next section II, an incremental rotary encoder is designed based on sine function and modeled using Matlab-Simulink, while in section III, simple 1x and 4x decoders algorithms are designed to decode the incremental position signals from the encoder. In section IV, the developed encoder and decoder algorithms are combined to emulate a realistic incremental angular position sensor to test their impact on the feedback closed loop of servo control systems. Furthermore, in section V, the developed decoder algorithms are also implemented in real time measurement and control of a simple angular position servo control system using Arduino mega board. Finally, conclusions are given in section VI, and a list of references is given at the end.

II. INCREMENTAL QUADRATURE ENCODER SIMULATOR

It is very important to design, develop and evaluate algorithms that can be used to decode the position signals generated from an incremental quadrature encoder.

The typical relative quadrature encoder signal is a train of pulses coding incremental relative changes in position, for a rotational system, it has a specific number of pulses per revolution (Np), which determines the sensor resolution degree per pulse ratio or rather the periodic pulse space or cycle in degrees (Ps = 360/Np).

Moreover, to encode the direction of the rotation, a second train of pulses is needed which is called phase B signal usually 90 degrees out phase to the main phase A signal, see figure 1. For more flexibility, a Starting Phase Shift (SPS) can also be introduced that represents an arbitrary initial starting position.

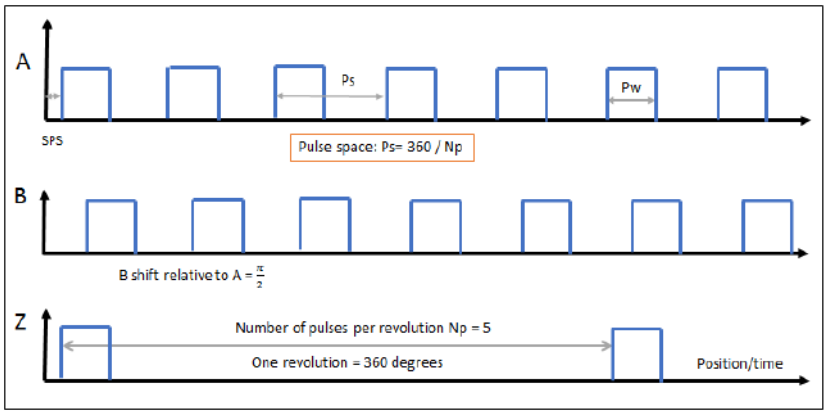

Figure1: Quadrature incremental encoder phases A, B and Z.

The sine function that used to generate the phase A is defined as following

$$f_A(\theta) = \sin\left(\frac{Np(\theta - SPS)\pi}{180}\right)$$

Therefore, the phase A function can now be defined as following:

$$A(\theta) = \begin{cases} High\ (1), & f_A(\theta) \geq 0; \\ Low\ (0), & f_A(\theta) < 0; \end{cases}$$

Similarly, the sine function to generate phase B

$$f_B(\theta) = \sin\left(\frac{Np(\theta - SPS)\pi}{180} - \frac{\pi}{2}\right)$$

And the phase B function is given by:

$$B(\theta) = \begin{cases} High\ (1), & f_B(\theta) \geq 0; \\ Low\ (0), & f_B(\theta) < 0; \end{cases}$$

Figure 2 shows the constructed Simulink block diagram of the rotational relative incremental quadrature encoder based on sine function.

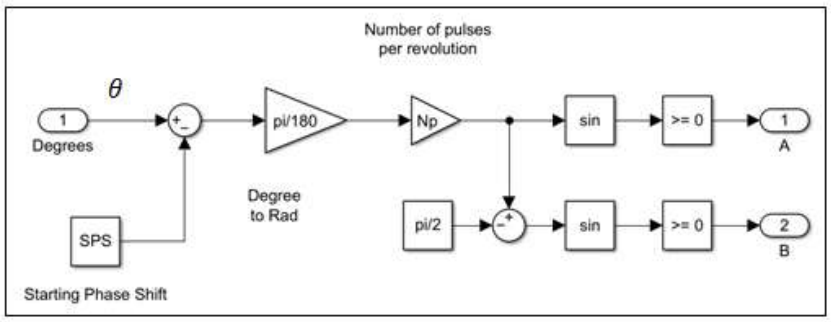

Figure 2: Simulink block diagram of quadrature incremental encoder module.

Now the train of repeating pulses of phase A and B can be imagined to be generated from one pulse cycle disc of a single pulse period as shown in figure 3a, and to get higher resolution the

single pulse period disc is geared up with the measured shaft, so that the gear ratio will determine the number of pulses per revolution or the encoder resolution, the degree per pulse ratio. Starting the simulation with zero initial position could lead to an initial loss or miss count. This could be avoided by starting simulation with starting phase shift (SPS), see figure 1 and 3b, also in practical cases where the initial starting angle of the encoder could be randomly any value.

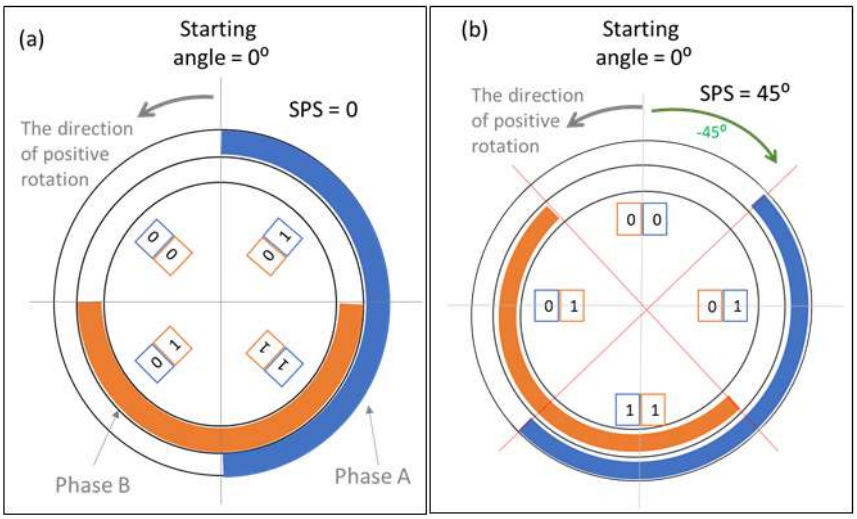

Figure 3: One pulse cycle disc of a single pulse period,

(a) without and (b) with starting phase shift.

Usually, phase Z, the zero pulse or home index, is used in applications where the absolute value of the rotational angle is needed for only one revolution, limited between zero to 360 degrees only, and the number of previous revolutions is not important. Nevertheless, the phase Z is not required for applications when the rotational angle is needed to retrieve how many degrees are achieved in both directions. In any way, the implication of phase Z to decoder algorithms is not included and hence not modelled in this work. But if needed, a more complex logic-based encoder model simulators are recommended, see for example [10, 11], also a quadrature encoder function is also available in Matlab-Simulink library but works only in Simscape as a physical environment emulator.

III. INCREMENTAL QUADRATURE DECODERS

A. Simple Incremental 1X Decoder

As a simple decoder algorithm, one can observe the phase A as it triggers from low to high, check the phase B, if phase B is low this is one direction of rotation, or high then the shaft rotates in the other opposite direction. This detection is used to initiate count up or count down action according to the last condition, as demonstrated by the following figure 4.

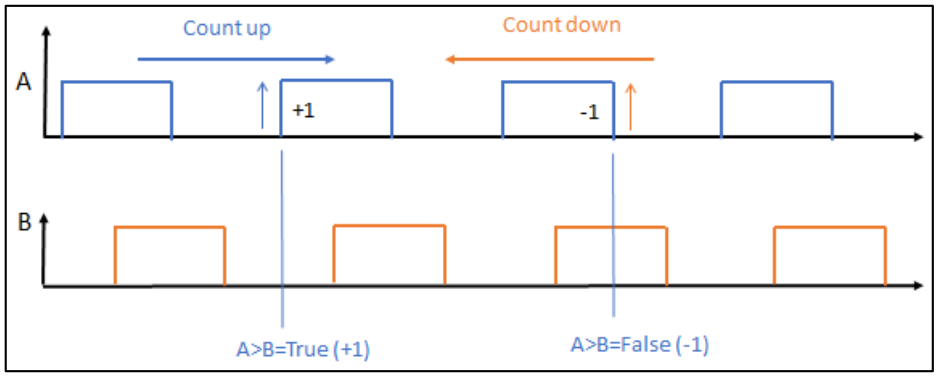

Figure 4: Phase A and B direction detection for count up or down action.

The algorithm can be written down using a pseudo code as following:

```
Define Interrupt: A (activation mode: low to high)
Initialize:  Pulse_count = 0;
Interrupt A:
        If A > B then
                Pulse_count = Pulse_count + 1;
        Else
                Pulse_count = Pulse_count - 1;
        End if
Terminate:
```

However, the algorithm is reprogramed or modeled using Matlab-Simulink graphical modeling environment, so that this model can be compiled by Matlab-Simulink to generate real time target software for real time hardware implementations, using for example, Arduino, dSPACE or any other target hardware. The simple 1x decoder block diagram is shown in figure 5, it is a triggered subsystem that checks which pulse edge comes first, the positive direction when phase A pulse edge comes first before phase B, otherwise it is the opposite direction.

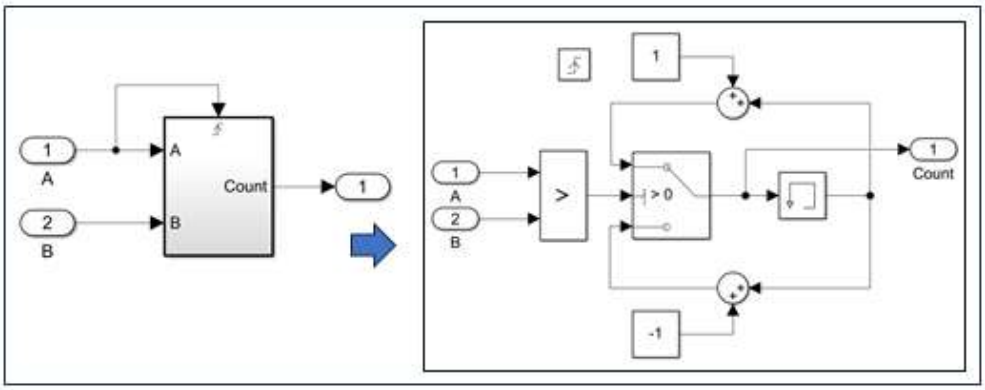

Figure 5: Simulink block diagram of a simple 1x decoder algorithm.

B. Logic 1X Decoder Algorithms

Now, there are four states for A and B phases within a cycle of one pulse period disc, see figure 6, the decoder is designed to detect the (A, B) state transition from old state (0, 0) to new state (1, 0) detecting a need to count up action, and the state transition from old state (0, 0) to new state (0, 1) to trigger the count down action in the opposite direction, and do not care about the other state transitions. This detection is done by using the logic circuit in figure 7a for count up and the logic circuit shown in figure 7b for count down action, note that old states are subscribed with o letter. Figure 8 shows the counter block diagram used for count up and down actions. The complete block diagram of the logic 1x decoder is shown in figure 9.

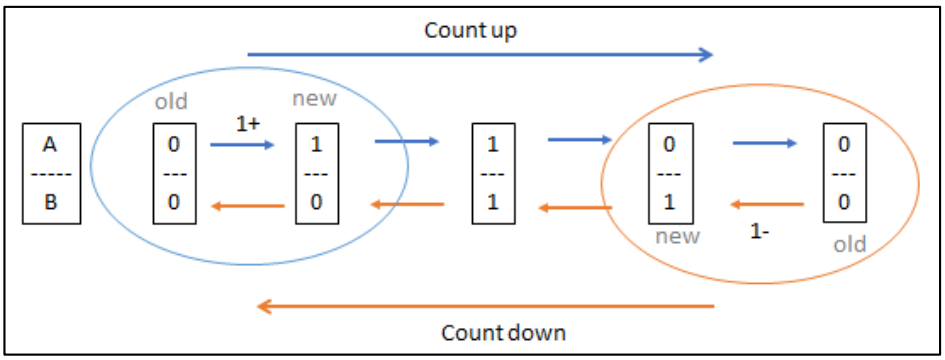

Figure 6: State transition of 1x logic decoder algorithm.

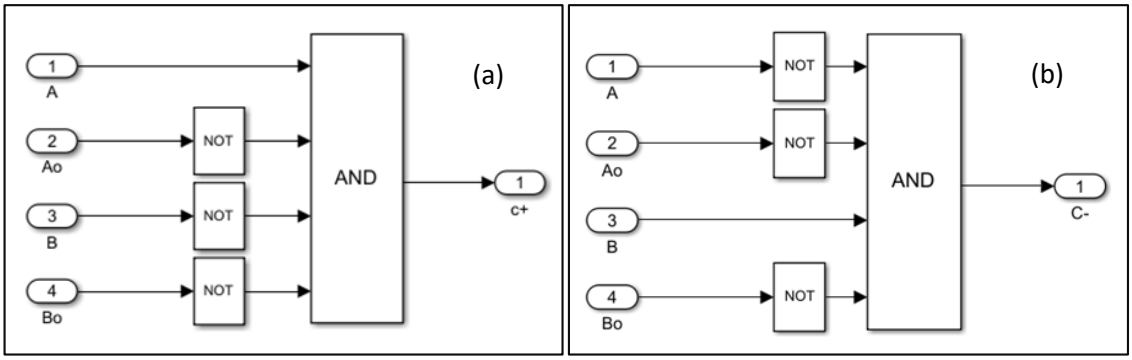

Figure 7: State transition detection logic (a) for count up action, (b) for count down action.

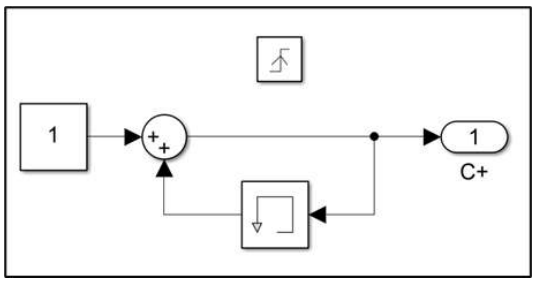

Figure 8: Counter block diagram for count up and count down actions.

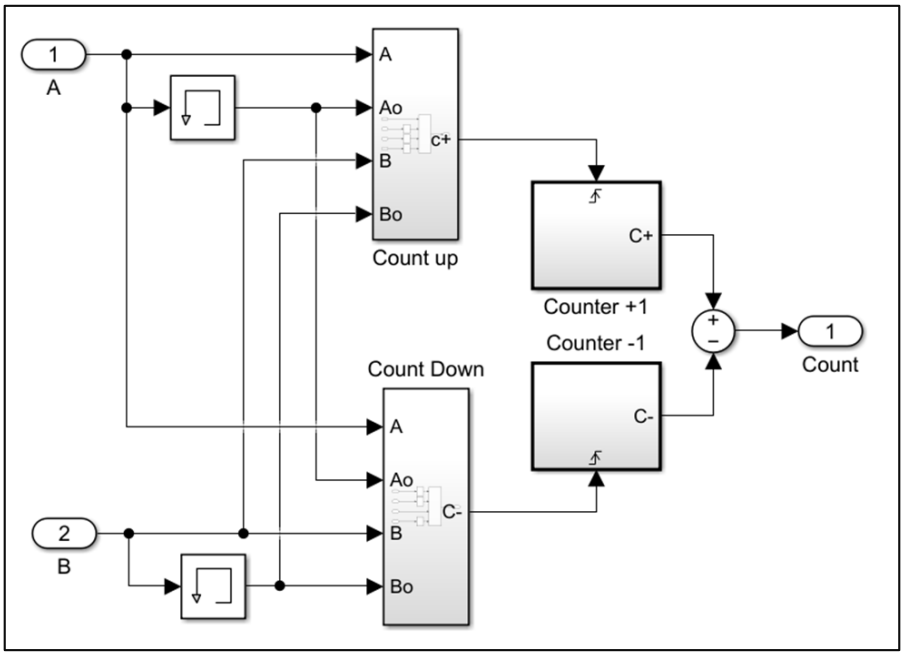

Figure 9: The block diagram of 1x logic decoder algorithm.

C. Logic 4X Decoder

Similar to logic 1x decoder, where it counts only one state transition pro direction, but for logic 4x decoder, all four state transitions are detected and counted for each direction. This makes the resolution to increase 4 times than the logic 1x decoder, hence the name logic 4x decoder is given. In other words, as shown in figure 10, in one pulse cycle for A and B phases there are four state transitions happen in one cycle, if we count all of them, we can increase the sensor resolution 4 times than if we count one pulse only. Therefore, all state transitions are detected and counted up or down accordingly, see figure 11, so similar to logic 1x decoder, four logic circuit are designed to detect count up action and other four logic circuits are for count down action detection, see figure 12.

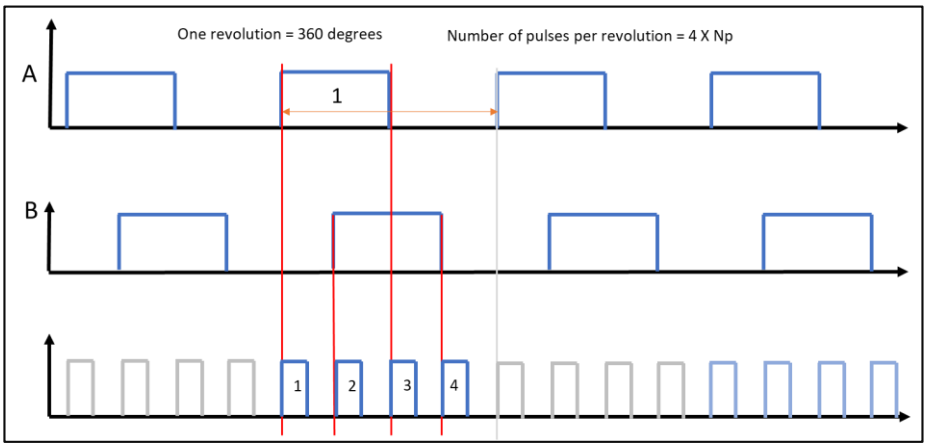

Figure 10: the principal of quadrature incremental 4x decoding.

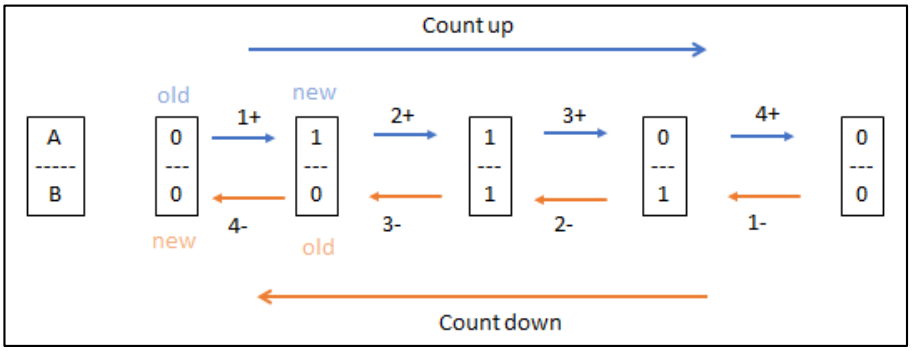

Figure 11: State transitions in one pulse cycle of A and B phases.

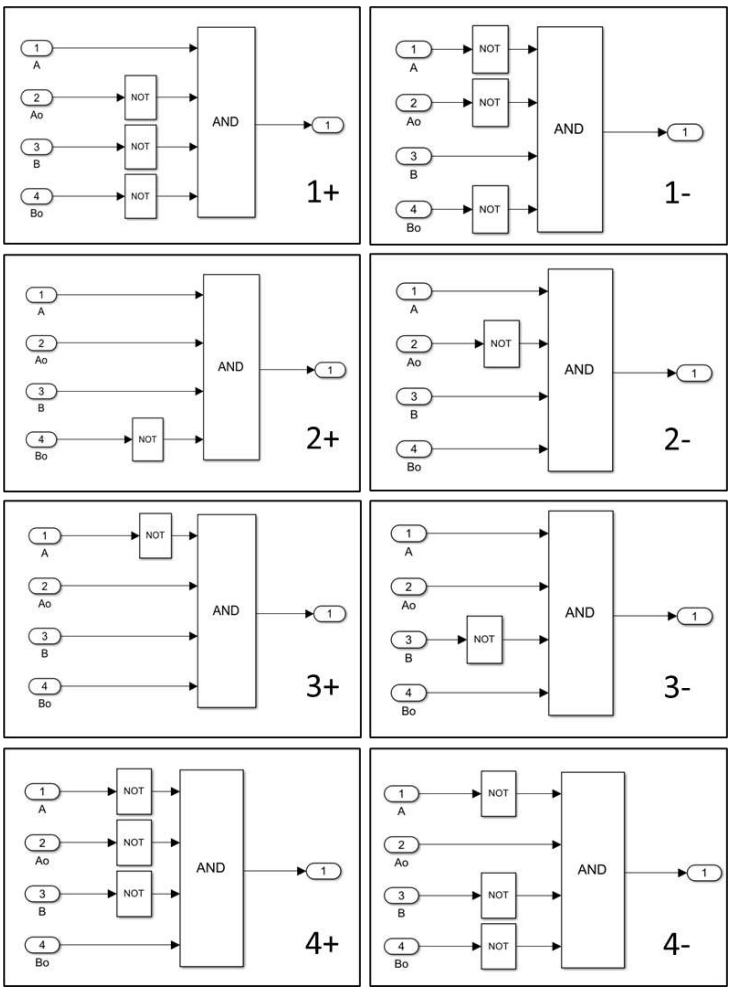

Figure 12: Logic diagrams for the four count up and count down permutations.

Moreover, figures 13a and 13b, shows the interconnections of count up and count down logic circuits, the block diagram of logic 4x decoder is exactly similar to the block diagram of logic 1x decoder shown in figure 9. Figure 14 shows a comparison between 1x, logic 1x and logic 4x decoders as well as the matlab22-Simulink quadrature decoder. The test is done for an encoder with very low resolution, 10 pulses per revolution, 36 degrees for every pulse for the

case of 1x decoders, but 9 degrees for 4x logic decoder, this will also interpret the errors of 1x decoders value of 36 degrees while the logic 4x decoder has an error of 9 degrees.

From the figure 14, it is obvious that the best decoder is the logic 4x decoder, while matlab22 quadrature decoder is the worst, but only because of the counter output is delayed and it can be removed by considering the output without delay as the presented algorithms.

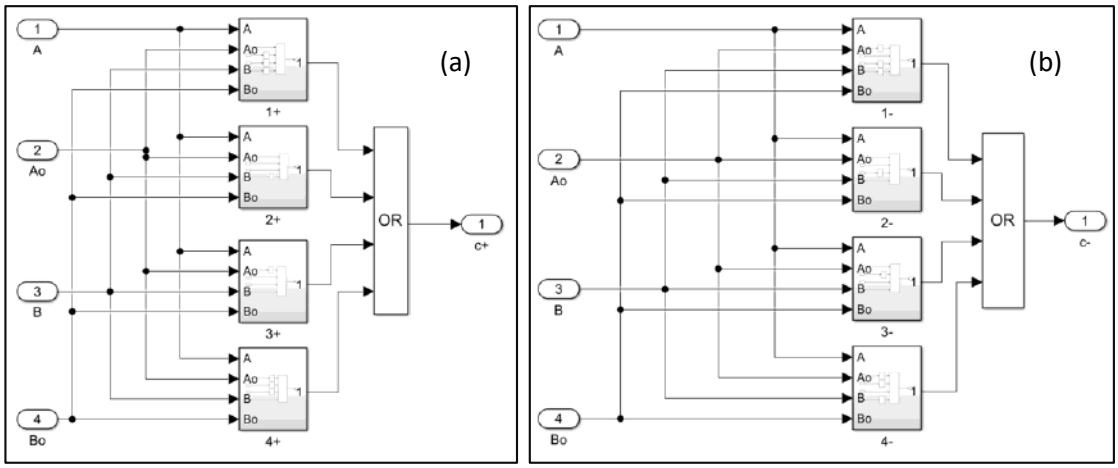

Figure 13: The interconnections of (a) count up block, (b) count down block.

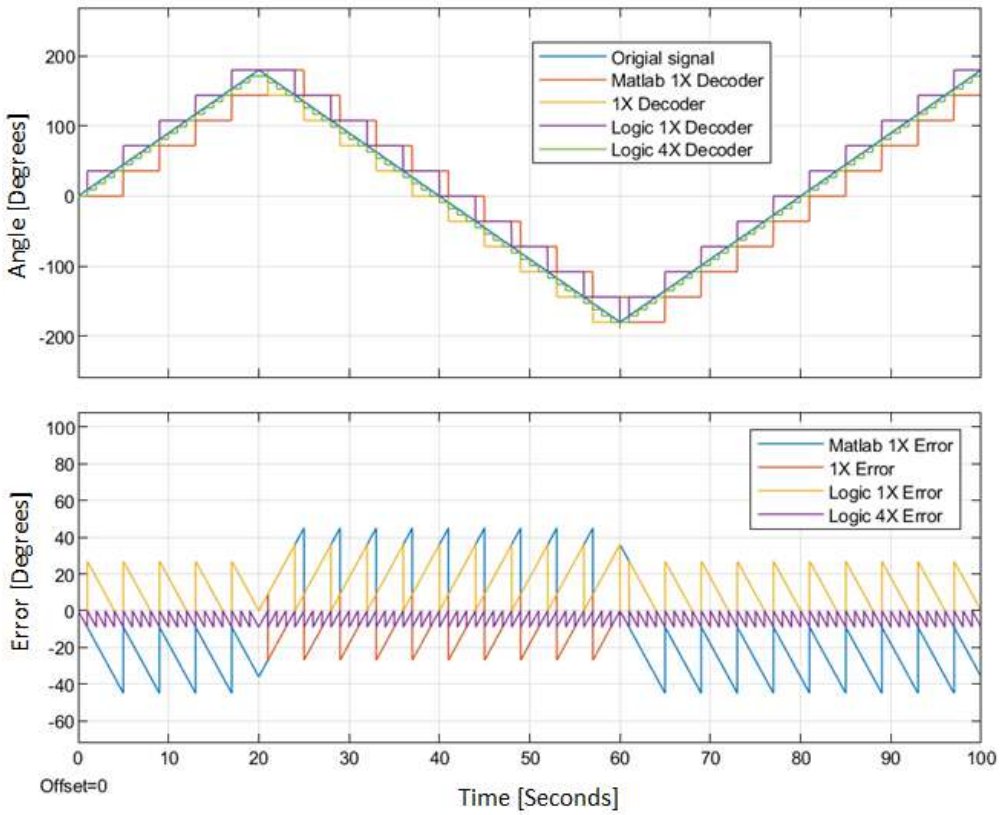

Figure 14: Original and measured signals of the different decoders and their errors.

IV. APPLICATION IN SERVO CONTROL SYSTEM

In this section, the developed decoders, 1x, logic 1x and logic 4x, together with the incremental quadrature encoder simulator are tested in closing the loops of position servo control systems so that their performances are observed by comparing their actual system outputs with an ideal sensor output of a linear system, see figure 15. All the systems for the sake of comparison are identical and represented by a pseudo dc motor model. The block diagram of the experimental setup is presented in figure 15, figure 16 shows the step responses for the actual system outputs of 1x decoder, logic 1x decoder and logic 4x decoder as well as the output of linear system with an ideal sensor, this is to demonstrate and to compare the effect of the decoders on the closed loop behaviour.

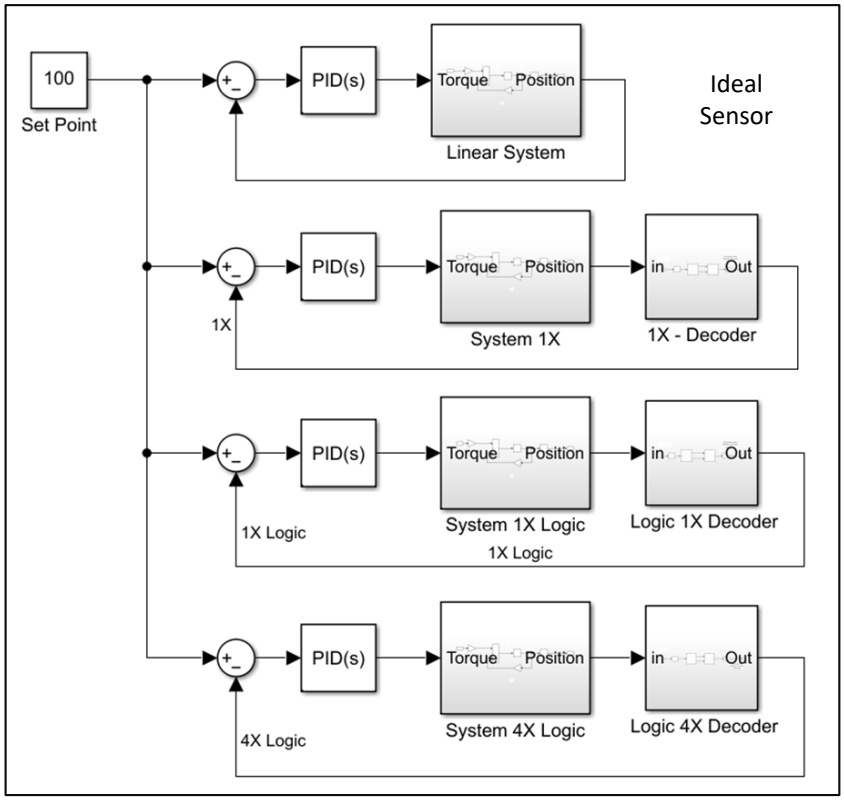

Figure 15: Simulink Block diagram for the comparison test of servo control systems.

The encoder simulators for all decoders are the same and all have 360 pulses per revolution, this means one degree per pulse resolution for 1x decoders, and the PID controller parameters are set by (P=2, I =0.2, D=0). Figure 16 shows clearly that logic 4x is superior to the others with the smallest steady state error and a performance near the linear system, second comes the logic 1x and the worst performance is the simple 1x decoder. Note again that the performance is measured by how much close the actual system output to the desired value compared to the ideal sensor linear system output, and not the measured one.

Moreover, the experiments are redone again with a relatively very low-resolution encoder of 20 pulses per revolution, 18 degrees per pulse resolution for 1x decoder and 4.5 degrees for

4x decoder. The actual system output step responses are plotted in figure 17. Again, the logic 4x decoder actual system output is the best closest to the linear system output with ideal sensor, second comes the actual system output of logic 1x decoder and the worst in this comparison is the actual system output of 1x decoder.

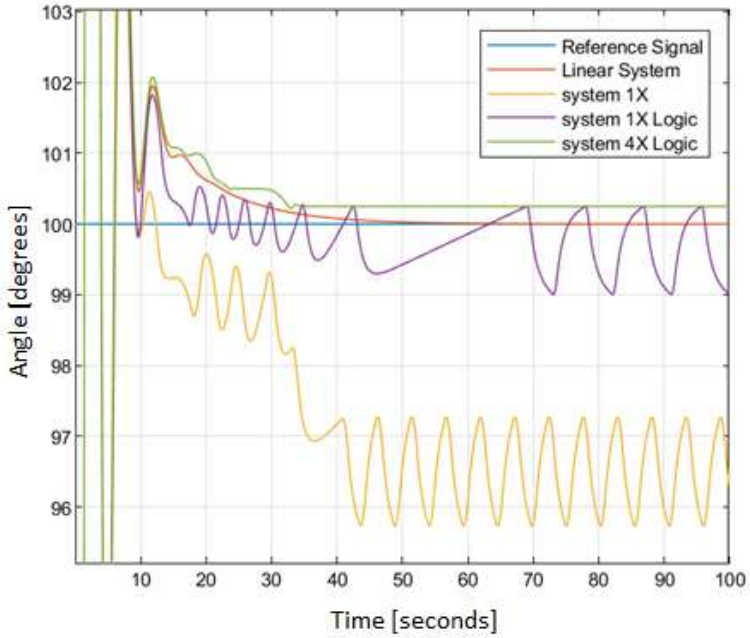

Figure 16: Step response of the actual system outputs compared to the output of an ideal sensor linear system with 360 ppr encoders.

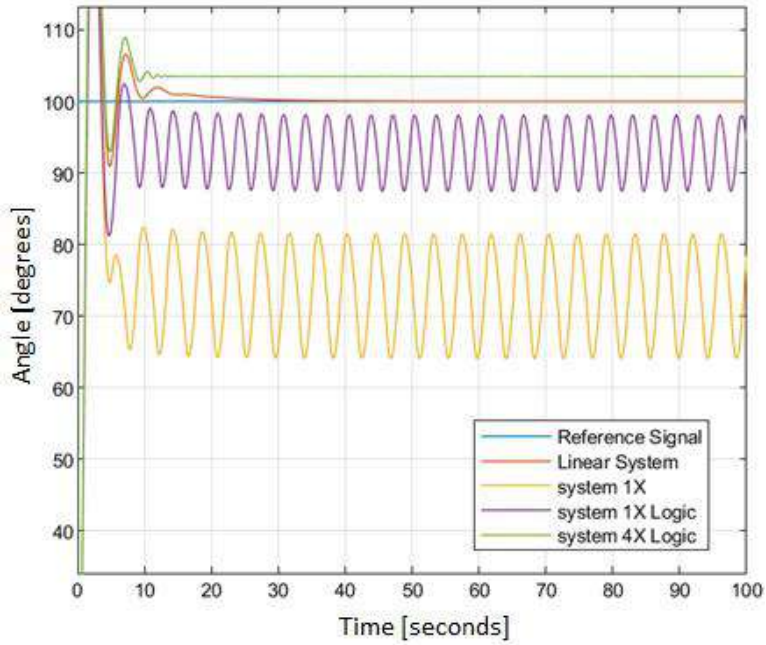

Figure 17: Actual system output step responses using 20 ppr encoders.

Now, in the figure 18, the measured output signals of the 1x, logic 1x and logic 4x decoders are plotted and compared to the output of the linear system, which represents an ideal sensor output. Where, it can be seen that the measured outputs of 1x decoders are oscillating within their resolution range 18 degrees, while 4x decoder within 4.5 degrees, therefore, the measured output of the logic 4x decoder is the most precise but also with most oscillations at higher frequencies. These oscillations make the closed loop behaves as two-point nonlinear control system. Of course, this nonlinear behaviour can be reduced by increasing the encoder resolution.

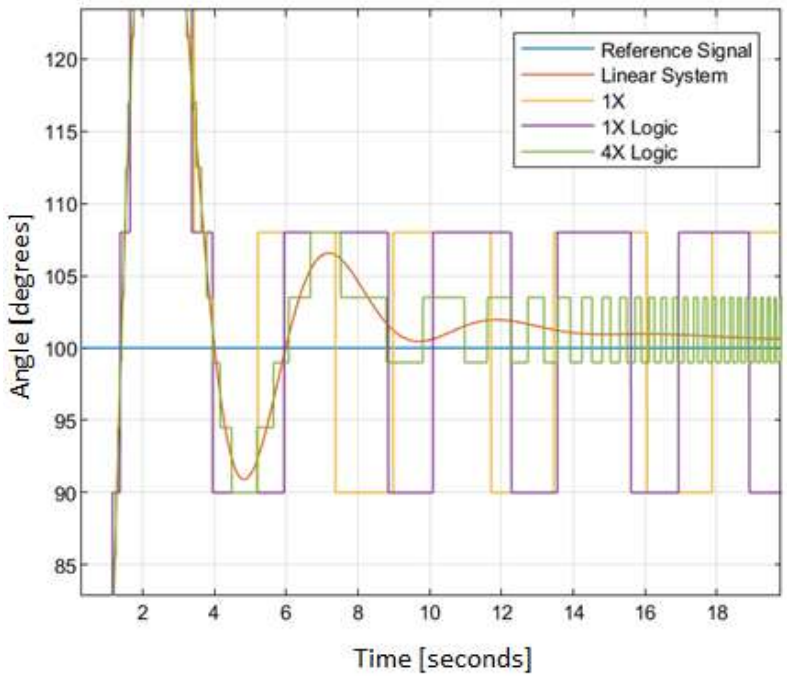

Figure 18: Measured output signals of 1x, logic 1x and logic 4x decoders compared to the ideal sensor output.

V. REAL TIME IMPLEMENTATION USING ARDUINO

The decoder algorithms are tested using a simple rotary encoder switch, module KY-040, as an angular position sensor with 20 pulses per revolution, which is mechanically coupled to the small yellow geared dc motor driven by H-bridge type L298N, the H-bridge control and the encoder signals are connected to Arduino Mega 2560 board. The details of the mechanical and the electrical interconnections are illustrated in figure 19. The Matlab-Simulink model of the position servo control system is presented in figure 20, this is actually a graphical program, which is compiled by Simulink real time library to Arduino C code and uploaded to the Arduino Mega board and run in real time. The block diagram (HB_DCM_DEC) of the H-bridge control with its subsystem details as well as the encoder decoder subsystem is also shown in figure 21.

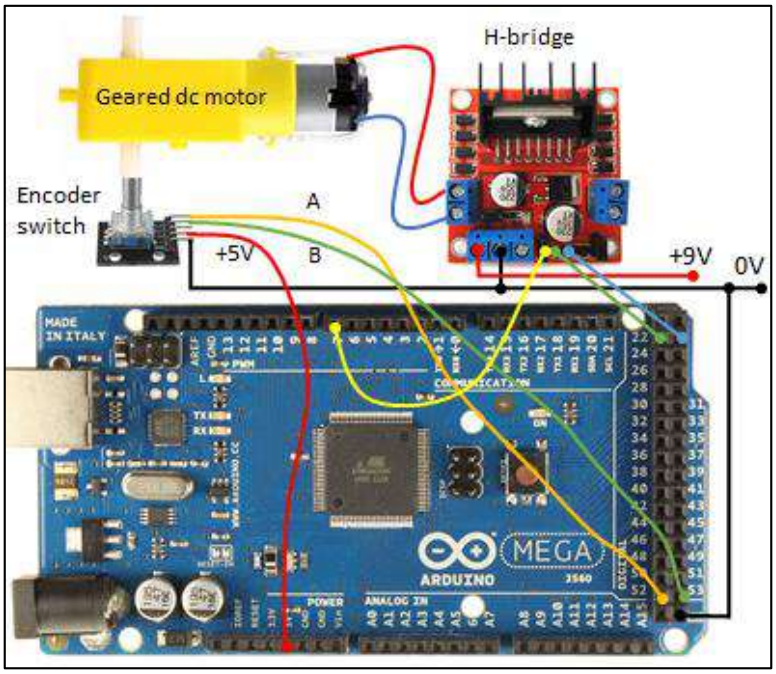

Figure 19: the mechanical and electrical interconnections of the Hardware.

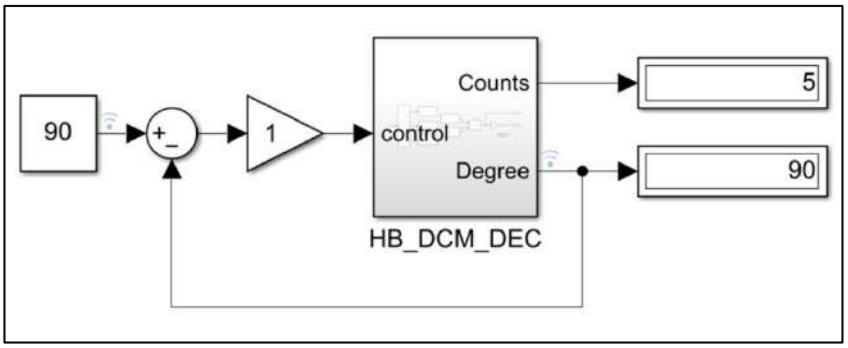

Figure 20: Simulink block diagram of the real time servo control system.

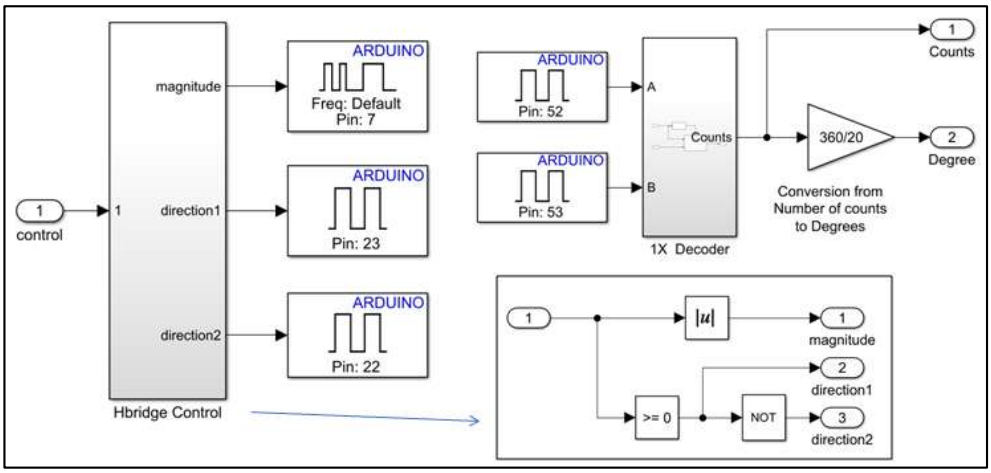

Figure 21: Block diagram of H-bridge dc motor and encoder-decoder subsystem.

Moreover, after successful hardware and software operation, figure 22a shows the measured output response to a step input from 90 to 180 degrees for servo system, where the 1x decoder has counted 5 pulses from the encoder signals, phase A and B, indicating 90 degrees increase. On the other hand, figure 22b shows the output response for the same servo system but with a logic 4x decoder, where it took 20 counts for 90 degrees increase.

Although the error between the measured output signal and the demanded input signal is zero in both cases of decoders 1x and 4x, but by direct observation and manual measurement of the actual output angular position, there is an error between the actual output or the true output and the demanded input. The errors are caused by low quality and low-resolution sensors and a wide dead zone that could possibly be caused by the H-bridge drive electronics as well as the mechanical and electrical characteristics of the geared motor. These have led to that the closed loop feedback servo system exhibits nonlinear behaviour largely contributed by the drive and the sensor sides.

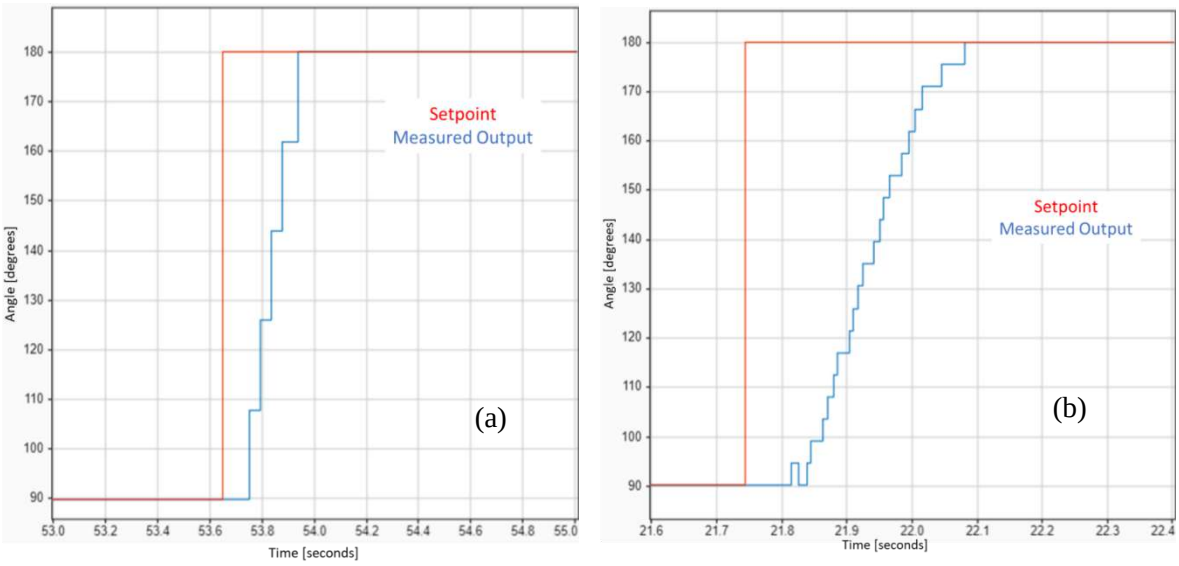

Figure 22: Measured output step response using 20 ppr encoder with

(a) 1x decoder, (b) 4x decoder.

VI. CONCLUSIONS

In this paper, the design and the development of the encoder simulator based on sine function has helped in turns to design and develop algorithms of 1x and 4x decoders, to decode the incremental encoder signals, phases A and B, successfully. In other words, the encoder simulator helped in testing, problem debugging and proving these decoder algorithms.

Furthermore, the Matlab Simulink model of the rotary incremental angular position sensor is constructed by combining the encoder and the decoder models, this gives the opportunity to emulate the encoder and the decoder together as a real angular position sensor, and to study its impact on the behaviour and the performance of closed loop feedback servo control

systems, in terms of number of pulses per revolution, stability, nonlinear behaviour, sampling time and beside to a number of other parameters.

The testing is done first in simulation and then in real time to control a real hardware by using a cost-effective test platform of rotary angular position servo control system with a very low-resolution sensor. The merge, between using graphical programming and the computational engineering analysis library of Matlab-Simulink together with Arduino boards, this has accelerated the design, development and the implementation processes and shortened the time that is usually taken. This type of workflow is generally called rapid control prototyping [12, 13], and usually used to develop and test new products in industry as well as for educational purposes very quickly and cheaply.

REFERENCES

[1] Heinz Heisler, *Advance Vehicle Technology*, Butterworth-Heinemann, second edition 2002.

[2] C. W. de Silva, *Sensors and Actuators: Engineering System Instrumentation*, Taylor & Francis Group: CRC Press, 2016.

[3] B. Drury, *Control Techniques Drives and Controls Handbook*, Institution of Engineering and Technology, second edition 2009.

[4] J. G. Webster, *The Measurement, Instrumentation and Sensors Handbook*, Taylor & Francis Group: CRC Press, 2014.

[5] R. M. Kennel, "Why do incremental encoders do a reasonably good job in electrical drives with digital control?" *in Conference Record of the 2006 IEEE Industry Applications Conference Forty-First IAS Annual Meeting*, vol. 2, Oct 2006, pp. 925–930.

[6] H. -L. Lin, K. -Y. Peng and J. -Y. Chang, "Design and Evaluation of an Open-Structure Rotary Magnetic Encoder for Subaqueous Environments," *in IEEE Transactions on Magnetics*, vol. 60, no. 9, pp. 1-5, Sept. 2024, Art no. 8000505, doi: 10.1109/TMAG.2024.3416968.

[7] R. Petrella, M. Tursini, L. Peretti, and M. Zigliotto, "Speed measurement algorithms for low-resolution incremental encoder equipped drives: a comparative analysis," *in Electrical Machines and Power Electronics, (ACEMP)*, Sept. 2007, pp. 780–787.

[8] R. Petrella and M. Tursini, "An embedded system for position and speed measurement adopting incremental encoders," *IEEE Transactions on Industry Applications*, vol. 44, no. 5, Sept. 2008, pp. 1436–1444.

[9] F. Briz, J. A. Cancelas, and A. Diez, "Speed measurement using rotary encoders for high performance ac drives," *in Industrial Electronics, Control and Instrumentation, (IECON) '94*, vol. 1, Sept. 1994, pp. 538–542.

[10] I. I. Incze, C. Szabó, and M. Imecs, "Modeling and simulation of an incremental encoder used in electrical drives," *in Proceedings of 10th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics*, Budapest, Hungary, November, 2009, pp. 12–14.

[11] I. I. Incze, C. Szabó, and M. Imecs, Incremental Encoder in Electrical Drives: Modeling and Simulation. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 287–300.

[12] D. Abel, and A. Bollig, *Rapid Control Prototyping*. Springer-Verlag, Berlin, 2006.

[13] G. Bodo, F. Tessari, S. Buccelli and M. Laffranchi, "A Rapid Control Prototyping and Hardware-in-the Loop Approach for Upper Limb Robotic Exoskeletons Control," *Applied Sciences*. 2024, 14(5):2017. https://doi.org/10.3390/app14052017