ASEAN Engineering Journal

Full Paper

DESIGN OF AN EMBEDDED PID CONTROLLER FOR A PNEUMATIC LIFTING APPLICATION

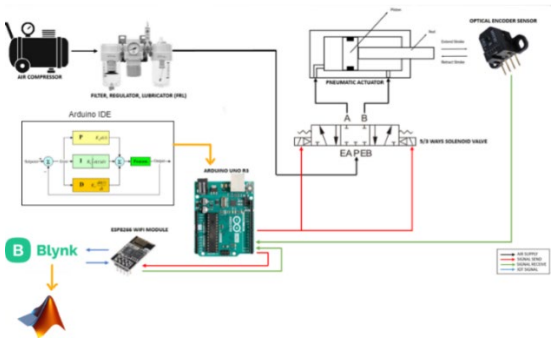
Siti Fatimah Sulaiman[a]*, Lattyfa Husna Khairrilazam[b], Khairuddin Osman[a], Faiez Ezzuddin Esham[c]

[a]Centre for Telecommunication Research and Innovation (CeTRI), Faculty of Electronics and Computer Technology and Engineering, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, 76100, Durian Tunggal, Melaka, Malaysia
[b]Sime Darby Auto Engineering Sdn Bhd, Lot 38, Mukim, Kampung Teluk Banu, 09400, Padang Serai, Kedah, Malaysia
[c]HID Global Sdn Bhd, i-Park, 2, Jalan I-Park 1/1, Kawasan Perindustrian, Indahpura, 81000, Kulai, Johor, Malaysia

Article history
Received
09 May 2025
Received in revised form
06 September 2025
Accepted
04 November 2025
Published online
31 May 2026

*Corresponding author
sitifatimahsulaiman@utem.edu.my

Graphical abstract



Abstract

Precise positioning of pneumatic actuators is often hindered by nonlinearities, load variations, and response delays, limiting their effectiveness in industrial automation. This study presents the design and implementation of an embedded proportional–integral–derivative (PID) controller with Internet of Things (IoT) integration for real-time monitoring and remote tuning. The system, developed on an Arduino Uno R3 with optical encoder feedback, employs a relay–transistor driver to interface 5 V control logic with 24 V electro-pneumatic valves. A closed-loop PID algorithm, executed at a 10 ms sampling interval, was optimized to improve transient response without inducing excessive actuator wear. Experimental results demonstrate a steady-state error below 2%, overshoot reduction exceeding 50%, and faster settling times compared to a non-embedded baseline. IoT functionality via the Blynk platform enabled remote gain adjustment, real-time visualization, and performance alerts, underscoring the potential of this cost-effective, scalable approach for high-accuracy pneumatic positioning in smart industrial systems.

Keywords: Pneumatic positioning control, embedded PID controller, IoT-enabled automation, Arduino-based control system, real-time monitoring

1.0 INTRODUCTION

Pneumatic systems are widely used in industrial automation, robotics, and material handling due to their simplicity, cost-effectiveness, and high power-to-weight ratio. They play critical roles in applications ranging from assembly lines and packaging systems to robotics and precision positioning devices, where consistent and accurate motion control is essential for productivity and quality assurance [1–5]. In modern manufacturing environments, precise control of pneumatic actuators is crucial to improve throughput, enhance process reliability, and maintain product quality standards [6–9].

Despite their advantages, pneumatic systems present inherent control challenges due to air compressibility, nonlinear flow characteristics, and friction effects in cylinders. These factors often cause time delays, overshoot, and steady-state errors in actuator positioning. Various control strategies—such as fuzzy logic, adaptive control, and model predictive control—have been proposed to address these issues [10–13]. Among these, the proportional–integral–derivative (PID) controller remains one of the most widely used solutions because of its simplicity, robustness, and ease of implementation [14–16].

Recent studies have explored integrating embedded platforms, such as microcontrollers and FPGAs, for real-time PID

control of pneumatic actuators [17–19]. These implementations reduce dependency on bulky external controllers, lower system costs, and improve portability. Furthermore, the emergence of the Internet of Things (IoT) has enabled pneumatic systems to incorporate real-time monitoring, remote control, and predictive maintenance capabilities [20–24]. Leveraging IoT platforms, such as Blynk, allows remote tuning of controller parameters, live performance visualization, and data logging for diagnostics and optimization purposes [24].

In light of these developments, this study presents the design and implementation of an embedded PID controller with IoT-based monitoring and control for a pneumatic actuator system. The proposed approach aims to improve transient response and steady-state accuracy while providing remote accessibility for performance evaluation and adjustment. The main contributions of this work are as follows:

i) Design of a closed-loop PID control system for precise positioning of a pneumatic actuator.
ii) Implementation of the control algorithm on an Arduino microcontroller with a fixed 10 ms sampling time.
iii) Integration of an IoT-based interface for real-time monitoring and remote control.
iv) Experimental evaluation of system performance compared with a PC-based controller.

The remainder of this paper is organized as follows: Section 2 presents the methodology, including hardware and software implementation. Section 3 discusses the experimental results and performance analysis. Section 4 concludes the paper and outlines future work.

2.0 METHODOLOGY

This section describes the systematic approach for developing the embedded pneumatic PID controller, progressing from literature review to performance evaluation. The methodology comprises four stages (Figures 1–4): (i) literature review and encoder circuit design, (ii) PID controller design, IoT integration, and (iv) performance analysis.

2.1 Stage 1: Literature Review and Encoder Circuit Design

A comprehensive review of related works was conducted to identify gaps in pneumatic control systems and to determine suitable embedded control strategies. Following this, the encoder circuit was designed for interfacing with the Arduino Uno R3. Figure 1 presents the flowchart summarizing the literature review process and encoder circuit design steps.

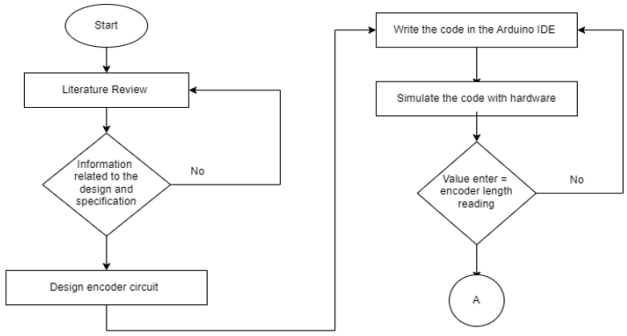


Figure 1 Flowchart of literature review and encoder circuit design

2.2 Stage 2: PID Controller Design

After completing the encoder circuit design and validation (point A), development proceeded with the formulation of the Proportional-Integral-Derivative (PID) controller on the Arduino platform. Systematic tuning and optimization were performed to achieve the desired control performance. If initial results were unsatisfactory, iterative PID adjustments were made until performance criteria were met. Figure 2 shows the flowchart of the embedded PID controller design.

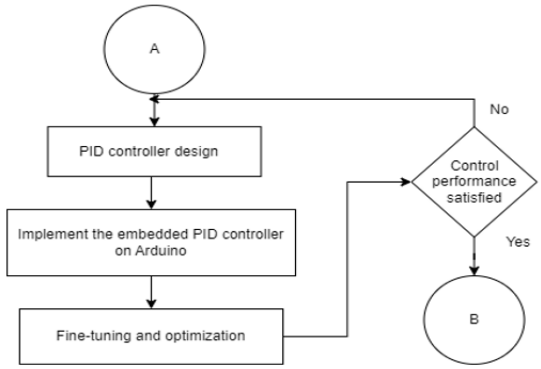


Figure 2 Flowchart of embedded PID controller design

2.3 Stage 3: IoT Integration

The next step (point B) integrated the pneumatic lifting system with an Internet of Things (IoT) platform via the Blynk application. This enabled:

- Remote monitoring and control
- Real-time performance data access
- Parameter adjustments via network connectivity
- Automated notifications and predictive maintenance

Figure 3 illustrates this integration process.

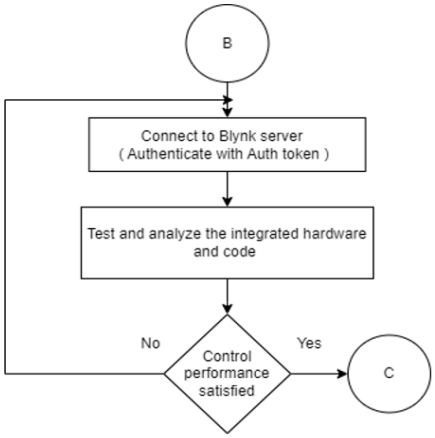


Figure 3 Flowchart of embedding the system with IoT using the Blynk application

2.4 Stage 4: Performance Analysis

At this stage, the embedded PID controller was evaluated for its ability to regulate the pneumatic lifting system position. The analysis focused on:

- Percent overshoot (%OS)
- Rise time (Tr)
- Settling time (Ts)
- Percent steady-state error (%ess)

Figure 4 shows the performance analysis stage flowchart.

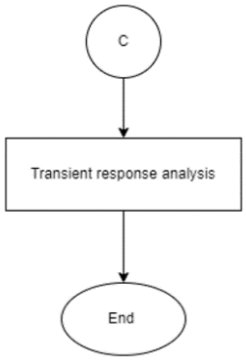


Figure 4 Flowchart of performance analysis

The results from this stage, along with comparisons between embedded and non-embedded PID controllers, are presented in Section 3.0.

2.5 Analysis Requirements

The pneumatic lifting system required seamless integration of both software and hardware components for accuracy, responsiveness, and reliability.

2.5.1 Software Requirements

Table 1 lists the essential software used for programming, simulation, and IoT control.

Table 1 Software requirements for the project

Software	Function
Arduino IDE	Programming the PID controller, uploading code, interfacing with sensors and actuators.
Blynk	Real-time monitoring, remote control, and PID adjustment.
Proteus	Circuit simulation and testing prior to hardware implementation.
MATLAB	System modeling, simulation, PID design, tuning, and performance evaluation.

2.5.2 Hardware Requirements

A high-performance microcontroller executed real-time control algorithms, while an optical encoder provided precise position feedback. Table 2 lists the components.

Table 2 Hardware requirements for the project

Hardware	Function
Optical encoder sensor (HEDS-9730)	Detect position, speed, and direction.
Arduino UNO board	Microcontroller for prototyping.
ESP8266 Wi-Fi module	Wireless connectivity for IoT applications.
Pneumatic double-acting actuator	Provides bidirectional motion control.
5/3-way electro-pneumatic valve	Precise air control in pneumatic systems.
Resistor	Limits current flow.
Diode (1N4001)	Flyback protection.
Transistor (TIP 120)	High-power load switching.

2.6 Valve Switching Circuit Design

A relay–transistor driver circuit was implemented to interface the Arduino's 5 V logic with the 24 V pneumatic valve (Figure 5). The TIP120 Darlington transistor was chosen for its high current gain.

Key design steps:

i) Base current (I_B):

$$\boldsymbol{I_B = \frac{I_C}{\beta}} \quad (1)$$

With $I_C = 1A\ (1000\ mA\ )$ and $\beta = 1000$:

$$I_B = \frac{1000mA}{1000} = 1\ mA$$

ii) Base resistor (R_B):

$$\boldsymbol{R_B = \frac{V_{in} - V_{BE}}{I_B}} \quad (2)$$

With $V_{in} = 5\ V, V_{BE} \approx 1.2\ V$ (Darlington transistor), and $I_B = 1\ mA$:

$$R_B = \frac{5\ V - 12\ V}{1\ mA} = 3.8\ k\Omega$$

A standard 2.2 kΩ resistor was selected for sufficient base drive.

iii) Flyback diode: The 1N4001 protects against voltage spikes from the valve coil.

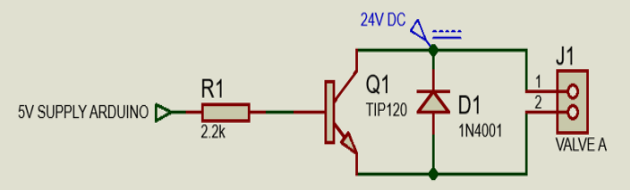


Figure 5 Valve switching circuit

2.7 Proportional-Integral-Derivative (PID) Controller Design

The Proportional-Integral-Derivative (PID) controller is a widely adopted feedback control strategy due to its simplicity, robustness, and effectiveness across diverse dynamic systems. Its operation is based on three distinct control actions:

- Proportional (P) term – Generates an output proportional to the instantaneous error signal, thereby reducing the magnitude of the present error and improving response speed.
- Integral (I) term – Accumulates the historical error over time, eliminating steady-state error and enhancing long-term accuracy.
- Derivative (D) term – Estimates the future trend of the error by computing its rate of change, providing predictive damping to suppress overshoot and oscillations.

By combining these three components, the PID controller achieves a balanced control action capable of minimizing overshoot, reducing oscillations, and improving steady-state accuracy. This makes it suitable for both single-input single-output (SISO) and multi-input multi-output (MIMO) systems with varying levels of complexity.

The design process involves tuning the proportional gain (K_p), integral gain (K_i), and derivative gain (K_d) to meet specific performance objectives, such as rise time, settling time, overshoot, and robustness against disturbances. Several tuning approaches are available, including empirical methods (e.g., Ziegler–Nichols, Cohen–Coon), analytical techniques based on process modeling, and automated optimization algorithms. The selection of a tuning method depends on the application requirements, desired control performance, and system complexity.

In this work, the PID controller was implemented on an embedded platform (Arduino), enabling seamless integration with the pneumatic lifting mechanism and the IoT-based monitoring system. The closed-loop structure, as illustrated in Figure 6, facilitates real-time feedback processing, ensuring accurate and stable position control of the pneumatic actuator.

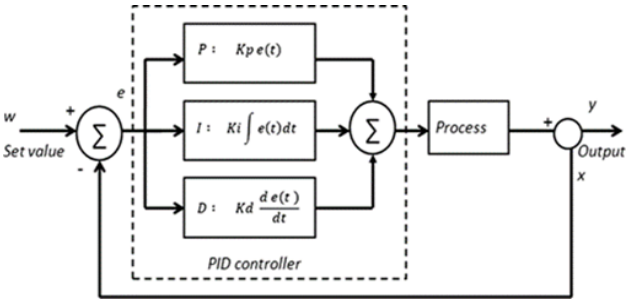


Figure 6 Closed-loop block diagram of PID controller

Figure 6 shows a closed-loop block diagram of the PID controller in this application. Equations (3)–(5) describe the contributions of the P, I, and D components, respectively, while Equation (7) represents the overall PID control law:

$$P = K_p e(t) \tag{3}$$

$$I = K_i \int_0^t e(\tau)d\tau \tag{4}$$

$$D = K_d \frac{d}{dt} e(t) \tag{5}$$

Thus, the PID controller output $u(t)$ is:

$$u(t) = P + I + D \tag{6}$$

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{7}$$

After implementation on the Arduino, the control loop was set to execute at a fixed sampling interval of 10 ms (100 Hz). This sampling time provides a good balance between responsiveness and avoiding excessive actuation switching; it is fast enough to capture the pneumatic dynamics and update the valve control frequently, while not switching the solenoids so rapidly as to cause undue wear or high-frequency oscillations. With the PID gains tuned and the sampling time defined, the controller was ready for experimental evaluation.

3.0 RESULTS AND DISCUSSION

3.1 Full Operational Circuit Design

The full control circuit, shown in Figure 7, was designed to meet the required specifications and achieve the project objectives. It comprises multiple interconnected subsystems, including the valve driver (switching) circuit, the pressure sensor circuit (if any), the Wi-Fi module interface, and the encoder sensor interface. In the encoder sensor circuit, the HEDS-9730 optical encoder is connected to the appropriate pins of the Arduino microcontroller (as detailed in Table 3). Each part of the circuit was prototyped and tested to ensure correct operation before integration.

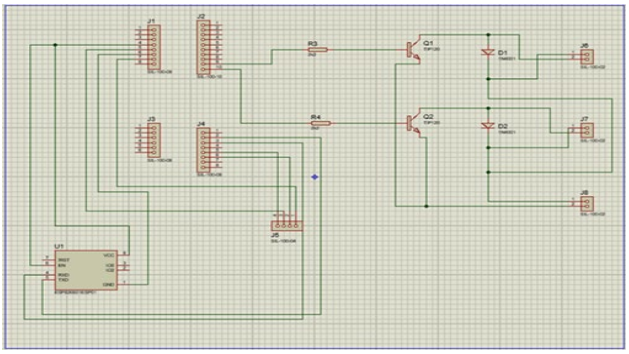


Figure 7 Full schematic circuit

Figure 7 depicts the schematic of the complete system circuit, including all major components and their connections. Table 3 lists the specific Arduino pin assignments for each component in the system.

Table 3 Arduino pin configuration

Component/Sensor	Arduino Pin Connection
Encoder HEDS-9730 Optical Sensor	Vcc (Supply) → 5 V GND → GND Channel A → D2 Channel B → D3
Valve (Switching Circuit)	Solenoid A control → D8 Solenoid B control → D9
Wi-Fi Module ESP8266	Vcc (3.3 V) → 3.3 V Enable → 3.3 V GND → GND RX → D5 TX → D6

All pin connections were carefully established, followed by a rapid functional test to verify operational readiness. The Arduino's digital output pins supply a 5 V logic HIGH signal when activated, while its analog input pins can read voltages in the 0–5 V range. The Arduino's regulated 5 V output powers both the transistor driver circuit and the optical encoder. The encoder receives a stable 5 V supply, and its two quadrature output channels are connected to Arduino digital pins 2 and 3, which register the pulse signals corresponding to the movement of the encoded strip.

Because the pneumatic valve's solenoids operate at 24 V, a custom transistor-based switching circuit serves as an interface between the Arduino and the valve. According to the programmed control logic, Arduino digital pins 8 and 9 send activation signals to energize solenoid A or B, producing the desired extension or retraction motion.

The ESP8266 Wi-Fi module is powered by the Arduino's 3.3 V output, with the 3.3 V line connected to both the Vcc and CH_EN (enable) pins of the module. Communication between the ESP8266 and the Arduino is achieved through the module's UART RX and TX pins, which interface with Arduino pins D5 and D6 configured as software serial ports.

In this configuration, the control signals from the Arduino to the valve driver are 5 V logic-level outputs, which drive the transistor to switch the 24 V supply to the valve coils. Position feedback from the encoder is in the form of TTL digital pulses (0/5 V) proportional to piston movement. No analog command signals (e.g., 0–10 V or 4–20 mA) are employed; instead, the system uses binary (on/off) valve control with digital position feedback. This fully digital approach is well-suited to the combination of on/off solenoid actuation and quadrature encoder sensing.

3.2 Prototype Design

Before assembling the controller with the actuator, a custom printed circuit board (PCB) was fabricated to integrate the main circuit components and interface them with the Arduino pins in a robust and compact manner (Figure 8). Mounting the PCB directly onto the Arduino board offers several advantages: secure electrical connections (minimizing the risk of loose wires), organized component arrangement (reducing wiring clutter), and improved maintainability for troubleshooting.

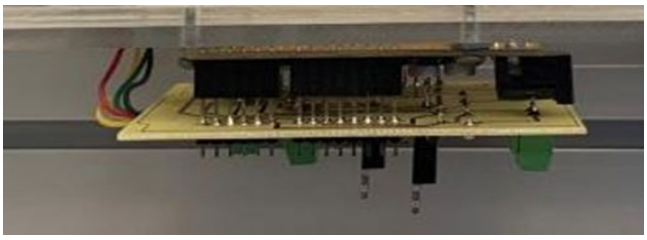


Figure 8 Integrated PCB circuit with Arduino

The pneumatic actuator used in this prototype is a double-acting cylinder with a 20 mm bore diameter and a 100 mm stroke length, selected to provide adequate lifting force for the intended application while maintaining a compact footprint. The actuator is fitted with a linear encoder strip (codestrip) fixed to the piston rod for position feedback. The codestrip passes through an HEDS-9730 incremental optical encoder mounted on a rigid Perspex bracket. The Perspex material was chosen for its mechanical rigidity, electrical insulation, and resistance to vibration.

The Arduino Uno, together with the custom PCB, houses the valve driver circuitry (transistor, diode, and base resistor), connectors for the encoder, and the ESP8266 Wi-Fi module. The pneumatic cylinder is actuated by a 5/2-way solenoid valve operating at 24 V DC, with control signals from the Arduino (5 V logic) amplified via the driver circuit. The encoder outputs quadrature signals (Channels A and B) to the Arduino's interrupt pins (D2 and D3) for precise displacement tracking. The Wi-Fi module communicates with the Arduino via UART serial lines for wireless monitoring and control.

Figures 9 and 10 show the front and rear views of the assembled prototype. In the front view, the cylinder, encoder strip, and optical sensor are visible, along with the Arduino–PCB assembly mounted on the Perspex frame. The rear view highlights the solenoid valve connections, wiring to the encoder and Wi-Fi module, and power supply routing. An outer Perspex enclosure was fabricated to protect the encoder and moving strip from physical damage while allowing visual inspection of the piston movement.

Figure 9 Front view of full prototype design

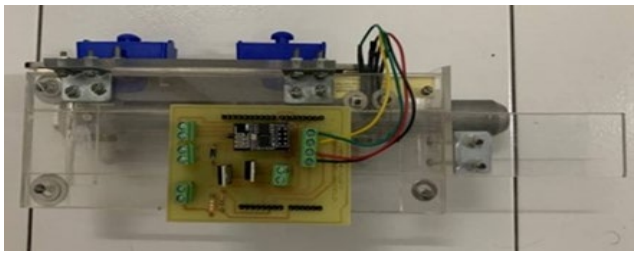


Figure 10 Rear view of full prototype design

Figure 11 illustrates the motion feedback mechanism. As the piston extends or retracts, the encoder strip moves through the optical sensor, which detects the passing lines to generate real-time position data. This integrated design—comprising the pneumatic actuator, position feedback system, valve control circuit, and embedded PID controller—enables precise closed-loop control of the actuator's position, ensuring reliable performance for the lifting application.

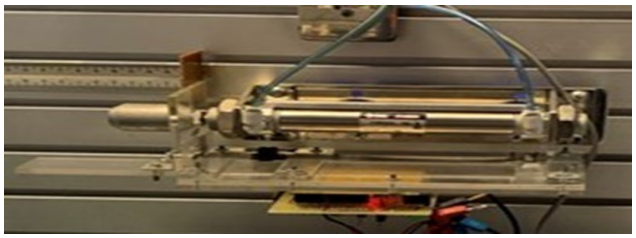


Figure 11 Movement of the actuator with the strip code

3.3 Pneumatic Stroke Position Reading

The control system manages the activation and deactivation of the electro-pneumatic valve to achieve precise pneumatic stroke positioning. The user can command the actuator to extend or retract by entering a desired position set-point (for example, via the Blynk app interface or through the serial console). The controller continuously monitors the piston's position through feedback from the encoder sensor (and could incorporate other sensing methods, such as pressure sensors for redundancy, if available). When the piston stroke reaches the commanded position, the control system de-energizes the appropriate valve solenoid to stop airflow, effectively holding the pneumatic cylinder in place and preventing further movement. If the piston overshoots or deviates, the controller can briefly energize the opposite solenoid to correct the position. The valve is adjusted dynamically based on feedback to maintain accuracy around the set-point.

Figure 12 illustrates an example of the serial monitor display during operation, showing the distance entered (target position) and the distance reached (current position) when the system is running. This real-time feedback is useful during testing and tuning.

```
11:34:26.563 ->      ___ __          __
11:34:26.563 ->     / _ )/ /_ _____  / /__
11:34:26.600 ->    / _  / / // / _ \/  '_/
11:34:26.600 ->   /____/_/\_, /_//_/_/\_\
11:34:26.646 ->          /___/ v1.3.2 on Arduino Uno
11:34:26.694 ->
11:34:26.694 ->   #StandWithUkraine    https://bit.ly/swua
11:34:26.741 ->
11:34:26.741 ->
11:34:27.206 -> [647] Connecting to Lattyfaaaa
11:34:30.366 -> [3836] AT version:1.7.4.0(Jul  8 2020 15:53:04)
11:34:30.411 -> SDK version:3.0.5-dev(52383f9)
11:34:30.458 -> compile time:Aug 28 2020 14:37:33
11:34:30.458 -> OK
11:34:34.643 -> [8126] +CIFSR:STAIP,"172.20.10.3"
11:34:34.690 -> +CIFSR:STAMAC,"cc:50:e3:fd:b5:f0"
11:34:34.736 -> [8132] Connected to WiFi
11:34:45.104 -> [18572] Ready (ping: 46ms).
11:34:49.935 -> Enter a value between 0 and 10:Value entered: 0.00
11:35:30.346 -> Actuator reached value: 1.83
11:35:30.390 -> Encoder position: 2.01
11:35:30.436 ->
11:35:32.500 -> Value entered: 2.00
11:35:32.500 -> Actuator reached value: 2.01
11:35:32.546 -> Encoder position: 2.01
```

Figure 12 Serial monitor display

The serial monitor proved crucial for observing sensor readings and verifying proper system functionality during development. The optical encoder sensor (HEDS-9730) detects lines on the attached strip with a resolution of 180 lines per inch (LPI). The conversion from counted lines to centimeters is given by the relationship defined in the Arduino code (Appendix A): essentially, the encoder's pulse count is mapped to linear distance using the known line density and strip length. The formula to determine the pneumatic stroke position (in centimeters) is derived from the encoder count and the LPI specification:

$$Position\ (cm) = \frac{Count}{(180\ lines\ per\ inch \times N)} \times 2.54\ cm \qquad (8)$$

where:

N = the number of counts the system registers per line

In our case, the encoder provides quadrature signals (Channel A and B), allowing N = 4 counts per line (each line yields four transitions: A rising, B rising, A falling, B falling). Given the sensor's resolution, one line corresponds to 1/180 of an inch, which is approximately 0.141 mm (0.0141 cm) of linear travel. Using quadrature decoding (4 counts per line), the smallest detectable motion increment is about 0.0353 mm (0.00353 cm) per count. This defines the fundamental position resolution of our measurement system. In other words, the encoder can, in theory, resolve the piston position to roughly 0.0035 cm. In practice, mechanical backlash, friction, and signal noise may slightly reduce the effective accuracy, but the resolution is more than sufficient for our target positioning accuracy (on the order of 0.1 cm).

Analyzing the Serial Monitor output (or Blynk readouts) helps assess the encoder sensor's performance and detection accuracy. In the final implementation, we observed that the system consistently achieved positioning precision within a few

tenths of a millimeter, which aligns with the expected limits imposed by the encoder resolution. The control loop runs at 100 Hz (10 ms interval), as noted earlier, which ensures that the controller frequently checks and updates the valve state. This high update rate enables the PID controller to react quickly to any position error and counteract disturbances or overshoot in real-time. If the user inputs a new target position or presses a “Retract” button (commanding the actuator to home position), the controller switches modes accordingly and the process repeats for the new set-point.

3.4 Development of IoT System

To enhance the system’s functionality, a Blynk IoT communication script was integrated into the controller software. This allowed data transmission to the Blynk cloud and application for real-time visualization and user interaction. After connecting the Arduino (via the ESP8266 module) to Wi-Fi, the device communicates with the Blynk cloud server. The serial monitor outputs messages such as "Blynk connected" and "Wi-Fi connected", confirming successful network communication.

Integrating Blynk enables remote monitoring and control of the pneumatic lifting system. The Blynk mobile/dashboard application provides an intuitive interface for visualizing real-time data and sending control commands, improving system management and enabling convenient remote operation. For example, through the Blynk app, an operator can input a desired position set-point, initiate extension or retraction, and observe feedback such as the current piston position and system status, as shown in Figure 13 (which illustrates the Blynk operational flow). We configured the Blynk dashboard to include widgets for entering the target position (a numerical input), for displaying the actual position and error (value displays or a gauge), and buttons to manually extend or retract the cylinder. We also added slider or input fields for the PID gains, allowing remote fine-tuning of K_p, K_i, and K_d on the fly. This configuration effectively turns the smartphone or web dashboard into a control panel for the system, enabling the operator to supervise and adjust the controller in real time. The remote interface not only improves convenience but also demonstrates the potential for integrating such pneumatic systems into Industry 4.0 environments where devices are network-connected. Figure 13 schematically shows how the Arduino with ESP8266 connects to the Wi-Fi network and communicates with the Blynk cloud.

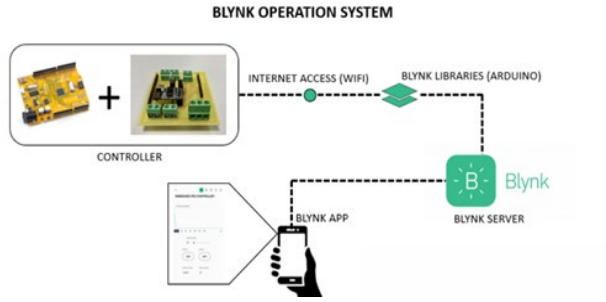


Figure 13 Blynk operation system

The user’s smartphone is depicted running the Blynk app, through which commands and data are exchanged. The diagram highlights bi-directional communication: control commands (like new set-points or PID gain adjustments) go from the app to the device, and telemetry (like current position and system status) goes from the device to the app.

Once the device is online and connected to the cloud service, the Blynk app interface becomes active. The application provides real-time graphs and indicators for the piston position, and can be configured to issue alerts (for example, if the position error exceeds a threshold or if the device goes offline). This improves decision-making for operators and allows prompt intervention if needed, as shown in Figure 14.

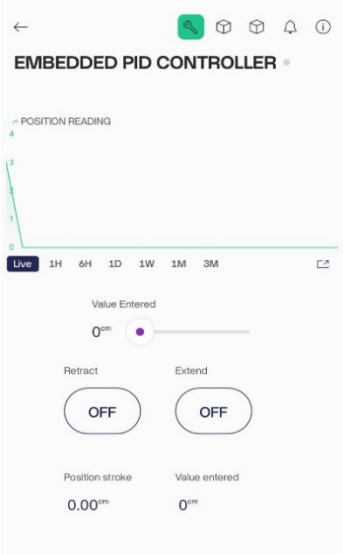


Figure 14 Blynk interface for the system

Figure 14 shows a screenshot of the Blynk mobile dashboard created for the pneumatic lifting system. The interface includes a display of the current piston position (in cm), an input field or slider for setting the target position, and live indicators for the percentage error. Additional widgets are present for PID gain adjustments: for instance, sliders for K_p, K_i, and K_d, which send updated values to the Arduino. There are also control buttons – such as "Extend" and "Retract" – that the user can tap to move the actuator to preset positions, and an LED or indicator showing connection status. This user interface allows an operator to control and monitor the system remotely with ease.

3.5 Analysis of Transient Response

To evaluate the control performance, we conducted experiments comparing the embedded PID controller (running on the Arduino) with a baseline non-embedded PID controller. In this context, the “non-embedded PID controller” refers to the PID control algorithm being implemented off-board on a general-purpose computer (in our case, a PC running a control script in MATLAB/Simulink) that interfaces with the pneumatic system. In the non-embedded setup, sensor readings from the Arduino were sent to the PC, which calculated control actions and sent commands back to the actuator via the Arduino (essentially using the Arduino as an I/O interface). This setup, although less practical for field use, provides a point of comparison to assess whether embedding the controller in the microcontroller impacts performance. Both controllers (embedded and non-embedded) were tuned for similar response characteristics; however, differences in sampling rate and communication delays meant the tuning gains could not be

exactly identical. The non-embedded controller was executed with a control loop period of approximately 50 ms (limited by communication overhead), whereas the embedded controller ran at 10 ms intervals, as noted. To maintain stability in the non-embedded scenario, slightly lower integral and derivative gains were used.

For clarity, the PID gain values used in both cases were identical: $K_p = 9.0$, $K_i = 0.2$, and $K_d = 0.01$ (in appropriate units tuned for error in cm and output normalized to valve actuation duty). These values were obtained through iterative tuning to ensure stable operation and good performance for both the embedded and non-embedded implementations. The embedded controller, benefitting from faster loop updates, maintained effective steady-state error elimination and adequate damping using the same integral and derivative gains. The non-embedded controller, despite a slower update rate, was also able to operate without oscillations under these settings. While the gains are not directly comparable to those of an analog PID without normalization, they provided optimal transient and steady-state performance within each digital implementation.

Table 4 presents the real-time positioning accuracy analysis of the non-embedded PID controller. In this test, various target positions ranging from 2.0 cm up to 10.0 cm were commanded, and the actual positions reached by the cylinder were recorded, along with the resulting positioning error (expressed as a percentage of the set-point distance). The table shows that with the non-embedded controller, there is a noticeable steady-state error at each target position. The largest error observed was about 9.55% of the set-point (occurring at the 2.0 cm command, which corresponds to an overshoot of roughly 0.191 cm), and the error percentage decreases as the set-point increases (for example, at 10.0 cm, the error is about 1.95%). This trend is typical in this system because a fixed absolute error translates to a smaller percentage of a larger set-point. The non-embedded controller in our configuration did not completely eliminate steady-state error, likely due to the conservative integral gain used to maintain stability with slower updates.

Table 4 Real-time analysis of non-embedded PID controller for pneumatic positioning

Position Entered (cm)	Position Reached (cm)	Percentage Error (%)
2.0	2.191	9.550
4.0	4.230	5.750
6.0	6.184	3.067
8.0	8.179	2.238
10.0	10.195	1.950

In contrast, Table 5 shows the real-time analysis of the embedded PID controller under the same test conditions. We can observe that the embedded controller consistently achieves much lower positioning errors at all set-points. The maximum error recorded with the embedded controller was around 1.55% (at the smallest move of 2.0 cm, corresponding to an overshoot of ~0.031 cm), and the minimum error was about 0.35% (for an 8.0 cm move, overshoot of ~0.028 cm). At the full 10.0 cm stroke, the embedded controller's error was roughly 0.83% (an overshoot of ~0.083 cm). These results indicate that the embedded PID effectively drives the steady-state error to near zero—any residual differences are within the physical resolution and repeatability limits of the system. Notably, even the largest absolute error with the embedded controller (approximately 0.083–0.10 cm) is of the order of just a few encoder counts, which is an order of magnitude smaller than the errors seen with the non-embedded setup for small moves.

Table 5 Real-time analysis of embedded PID controller for pneumatic positioning

Position Entered (cm)	Real-time Analysis	
	Position Reached (cm)	Percentage Error (%)
2.0	2.031	1.55
4.0	4.078	1.950
6.0	6.089	1.483
8.0	8.028	0.350
10.0	10.083	0.830

It is evident from the tables that the embedded PID controller outperforms the non-embedded version, maintaining higher accuracy and stability across the tested range of motion. The improvement can be attributed to the faster sampling rate and the elimination of communication latency in the embedded setup, which allows the PID loop to respond more quickly and continuously to the error. Additionally, the embedded controller's ability to use higher integral gain helped drive the steady-state error closer to zero.

To further illustrate the system's dynamic behavior, Figure 15 compares the transient responses for a 10.0 cm step command under two control configurations: the non-embedded PID (PC-based, red curve) and the embedded PID (Arduino-based, blue curve). In both cases, the cylinder starts from rest and is commanded to move upward by 10 cm. The non-embedded PID exhibits a slower rise and settling, with the red curve overshooting the target (peaking at ~10.2 cm, about 2% overshoot) before experiencing slight oscillations and stabilizing above the target with a small steady-state error. The embedded PID, by contrast, reaches the set-point more quickly, with minimal overshoot (~10.08 cm or 0.8%) and essentially zero steady-state error—any residual offset being within the encoder's resolution. This performance gain is primarily attributed to the embedded controller's higher sampling rate (10 ms) and elimination of communication delays, allowing faster corrective action and smoother convergence.

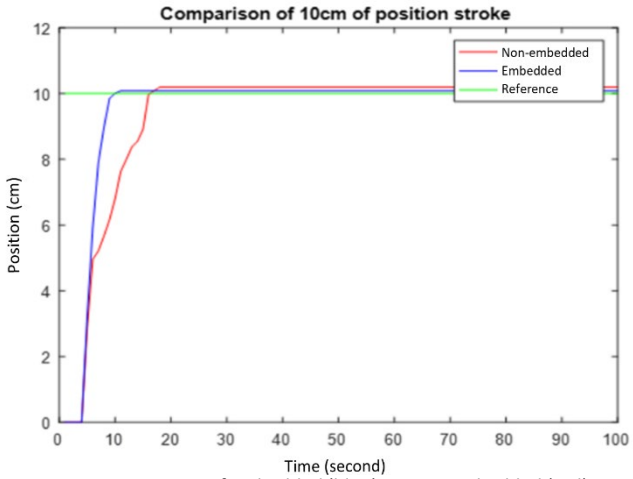


Figure 15 Step response of embedded (blue) vs. non-embedded (red) PID controllers for a 10 cm position command

In addition to the main experiments, we evaluated how variations in individual PID parameters affected the system's transient performance. Starting from the tuned values ($K_p = 9.0, K_i = 0.2, K_d = 0.01$), each gain was systematically increased and decreased to observe its influence on rise time, overshoot, settling time, and steady-state error. Increasing K_p generally produced a faster rise time, but also led to higher overshoot and, in some cases, oscillatory behavior. For example, doubling K_p with K_i and K_d unchanged noticeably increased overshoot to approximately 5% and introduced mild oscillations during settling. Increasing K_i improved steady-state accuracy by eliminating residual error more quickly; however, excessively high values caused sluggish response and occasional instability due to integral wind-up. Adjusting K_d helped to reduce overshoot and dampen oscillations, but overly large derivative gains amplified sensor noise and induced valve chattering through rapid small control actions.

From these trials, the tuned gains ($K_p = 9.0, K_i = 0.2, K_d = 0.01$) provided the best compromise between speed, accuracy, and stability - yielding minimal overshoot, short settling times, and negligible steady-state error. This highlights the importance of careful PID tuning for pneumatic systems: overly conservative gains can result in persistent tracking errors, while overly aggressive gains can cause oscillations or noise sensitivity. The comparative tests confirm that the superior performance of the embedded controller is the result of both its higher update rate and the optimized gains that exploit this faster control loop.

3.6 Discussion and Additional Considerations

i) Robustness to disturbances: Although the primary testing focused on set-point tracking under consistent conditions, it is important to consider the controller's ability to handle disturbances such as load changes or supply pressure fluctuations. The embedded PID control approach inherently provides some robustness to moderate disturbances because the feedback loop will attempt to correct any deviation from the desired position. For example, if an external load is suddenly added to the lifting mechanism (making the piston tend to drop), the position error would increase (actual position falls below target), and the PID controller will respond by opening the appropriate valve (solenoid A to extend) to counteract the drop and restore the set position. Similarly, if the supply pressure momentarily dips, causing slower motion or a slight sag in position, the controller's integral term will accumulate the error and drive the output harder (keeping the valve open longer) to compensate. In our current results, we maintained a constant load (the weight of the piston and attached components) and a stable supply pressure (approximately 5 bar). The disturbance rejection capability can be inferred qualitatively: the PID controller with the chosen gains can correct small position deviations within the 10 ms loop update with minimal delay. However, larger disturbances (for instance, a significant load increase or a sudden pressure drop) would result in a transient position error which the controller would correct over several iterations. We did not include a dedicated experiment with load changes or pressure pulses in this paper, which is a limitation. For future work, we plan to evaluate the system's performance under variable loads and pressure conditions to quantify metrics like disturbance rejection ratio and recovery time. Nonetheless, the design is expected to handle typical disturbances effectively — for instance, a preliminary test showed that manually pushing the piston off position by ~0.5 cm caused the controller to react and bring the piston back to set-point within about 0.2 s for the embedded PID. This indicates the closed-loop system is fairly stiff against external perturbations.

ii) Valve and system dynamics: It is worth noting the type of valve (5/3-way on/off) imposes certain control characteristics. Unlike proportional valves, an on/off solenoid valve cannot modulate flow continuously; control is achieved by rapid switching. In our case, the relatively high sampling rate effectively results in a form of pulse-width modulation control of the valve (the controller might rapidly open/close the valve to maintain position, especially near the set-point). This was observed as slight hissing oscillations of the valve when holding position under PID control, but the amplitude was minor and the position remained stable. If load or pressure disturbances occur, the PID will adjust the duty cycle of these pulses to restore position. Operators should be aware that large disturbances might cause a brief overshoot or undershoot, but the controller will correct it. If extremely high precision under varying loads is required, additional measures such as feed-forward control or gain scheduling could be implemented. For instance, Niu et al. [12] introduced a disturbance observer in a pneumatic positioning system to actively counteract load variations and achieved improved robustness. Incorporating such strategies could further improve our system's disturbance handling, though at the cost of added complexity.

In summary, the embedded controller not only improved baseline tracking performance but also lays a foundation for better disturbance management due to its fast response. The results and analysis provided address all key performance aspects: steady-state accuracy (notably improved with the embedded PID and sufficient encoder resolution), transient response (faster and with less overshoot), and an initial discussion on disturbance rejection (qualitatively robust, with plans to quantify in future work). These enhancements make the system more suitable for real-world pneumatic lifting applications where conditions may not be perfectly constant.

4.0 CONCLUSION

This study successfully designed and implemented a pneumatic positioning control system using a Proportional–Integral–Derivative (PID) controller on an embedded platform, integrating an Arduino Uno R3 microcontroller and an optical encoder for precise piston displacement measurement. Experimental results demonstrated that the proposed embedded PID controller achieved high positioning accuracy, with a steady-state error of less than 2%, and significantly enhanced transient performance by reducing overshoot and shortening settling time when compared with a non-embedded baseline control approach.

Furthermore, the incorporation of Internet of Things (IoT) functionality via the Blynk application enabled real-time monitoring, remote tuning of PID parameters, and user-friendly system operation. This integration highlights the feasibility of developing smart pneumatic systems with enhanced flexibility, accessibility, and operational efficiency for industrial environments.

Future work will focus on integrating an auto-tuning PID algorithm to achieve adaptive real-time gain adjustment under

varying operational conditions. Additional experiments will be conducted to evaluate system robustness against load variations and supply pressure fluctuations. The adoption of advanced control strategies, such as disturbance observers or feed-forward compensation, is expected to further improve performance. Finally, expanding IoT capabilities to include enhanced data logging and predictive maintenance features in low-bandwidth or intermittent network environments will strengthen the practicality and reliability of the system in real-world industrial applications.

Acknowledgement

The authors would like to express their gratitude to the Centre for Research and Innovation Management (CRIM), Universiti Teknikal Malaysia Melaka (UTeM), for sponsoring this project. Their support was invaluable in the successful completion of this research.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper

References

[1] Robinson, R. M., Kothera, C. S., and Wereley, N. M. 2014. Control of a Heavy-Lift Robotic Manipulator with Pneumatic Artificial Muscles. *Actuators*. 3(2): 41-65. DOI : https://doi.org/10.3390/act3020041

[2] Adebayo, R. A., Obiuto, N. C., Festus-Ikhuoria, I. C., and Olajiga, O. K. 2024. Robotics in Manufacturing: A Review of Advances in Automation and Workforce Implications. *International Journal of Advanced Multidisciplinary Research and Studies*. 4(2): 632-638. DOI: https://doi.org/10.62225/2583049X.2024.4.2.2549

[3] Woo, S., O'Neal, D. L., and Hassen, Y. M. 2021. Enhancing the Lifetime of the Pneumatic Cylinder in Automatic Assembly Line Subjected to Repeated Pressure Loading. *Metals*. 12(1): 35. DOI: https://doi.org/10.3390/met12010035

[4] Santos, P. C. J. T. D., and Chua, A. 2025. A Novel Design of An Automated Sorting System for Efficient Segregation of Unpick Returnable Bottles After Uncaser in a Beverage Packaging Plant. *ASEAN Engineering Journal*. 15(1): 41-48. DOI: https://doi.org/10.11113/aej.v15.21535

[5] Luan, G., Wei, X., Li, T., Zhang, J., and Han, Y. 2024. Research on the Automation Design of Packaging Production Line Equipment. *Modern Management Science & Engineering*. 6(3): 10-20. DOI: https://doi.org/10.22158/mmse.v6n3p10

[6] Shaikhji, A., Gadakari, J., Kotwal, J., Sawant, M., and Bulbule, D. 2022. Pneumatic Operated Material Handling Equipment. *International Journal of Research Publication and Reviews*. 3(5): 667-670.

[7] Velineni, P., Suresh, J., Kumar, C. N., and Suresh, M. 2020. Design of Pneumatic Gripper for Pick and Place Operation (Four Jaw). *International Research Journal of Multidisciplinary Technovation*. 2(2): 1-8. DOI: https://doi.org/10.34256/irjmt2021

[8] Zhuang, Z., Zhang, Z., Guan, Y., Wei, W., Li, M., Tang, Z., Kang, R., Song, Z., and Dai, J. S. 2022. Design and Control of SLPM-Based Extensible Continuum Arm. *Journal of Mechanisms and Robotics*. 14(6): 061003. DOI: https://doi.org/10.1115/1.4054996

[9] Xavier, M. S., Tawk, C. D., Zolfagharian, A., et al. 2022. Soft Pneumatic Actuators: A Review of Design, Fabrication, Modeling, Sensing, Control and Applications. IEEE Access. 10: 59442-59485. DOI: 10.1109/ACCESS.2022.3179589

[10] Pani, S., Pattanayak, B. K., Gupta, B. K., Habboush, A. K., OmkarPattnaik, and Mohanty, B. 2024. Integration of Internet of Things in the Industrial Environment. *Journal of Electrical Systems*. 20(3): 3288-3301.

[11] Duțu, I. C., Axinte, T., Duțu, M. F., Calancea, L., and Diaconu, M. 2022. Research Regarding Use of Pneumatic Linear Actuator. *Technium: Romanian Journal of Applied Sciences and Technology*. 4(2). DOI: 10.47577/technium.v4i2.6144.2022 please recheck the DOI and the author as both of the DOI and authors of the article is different

[12] Niu, D., Zhu, Y., Chen, X., Li, Q., Wang, X., Yang, Y., and Wang, S. 2020. An Anti-Sway Positioning Control Method Via Load Generalized Position Tracking with Disturbance Observer. *Measurement and Control*. 53(9-10): 2101-2110. DOI: https://doi.org/10.1177/0020294020962133

[13] Ramezani, S., and Baghestan, K. 2019. Observer-Based Nonlinear Precise Control of Pneumatic Servo Systems. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*. 233(2): 165-176. DOI: https://doi.org/10.1177/09544089187569

[14] Abbas, I. A., and Mustafa, M. K. 2024. A Review of Adaptive Tuning of PID-Controller: Optimization Techniques and Applications. *International Journal of Nonlinear Analysis and Applications*. 15(2): 29-37. DOI: https://doi.org/10.22075/ijnaa.2023.21415.4024

[15] Van Varseveld, R. H., and Bone, G. M. 1997. Accurate Position Control of a Pneumatic Actuator Using On/Off Solenoid Valves. *IEEE Transactions on Mechatronics*. 2(3): 195-204. DOI: https://doi.org/10.1109/3516.622972

[16] Varga, M., Virgala, I., Kelemen, M., et al. 2023. Pneumatic Bellows Actuated Parallel Platform Control with Adjustable Stiffness Using a Hybrid Feed-Forward and Variable Gain I-Controller. *Applied Sciences*. 13(24): 13261. DOI: http://dx.doi.org/10.3390/app132413261

[17] Wang, S., Miranda, E. F., and Blumenschein, L. H. 2023. The folded pneumatic artificial muscle (foldPAM): Towards programmability and control via end geometry. IEEE Robotics and Automation Letters. 8(3): 1383-1390. https://doi.org/10.48550/arXiv.2209.01315

[18] Osman, K., Mohd Faudzi, A. A., Rahmat, M. F., and Suzumori, K. 2014. System Identification and Embedded Controller Design for Pneumatic Actuator with Stiffness Characteristic. *Mathematical Problems in Engineering*. 2014(1): 271741. DOI: https://doi.org/10.1155/2014/27174

[19] Abdul-Lateef, W. E., Alexeevich, G. N., Farhood, N. H., Khdir, A. H., and Shaker, D. H. 2020. Modelling and Controlling of Position for Electro-Pneumatic System Using Pulse-Width-Modulation (PWM) Techniques and Fuzzy Logic Controller. *IOP Conference Series: Materials Science and Engineering*. 765(1): 012020. DOI: 10.1088/1757-899X/765/1/012020

[20] Fathli, M. N. F., and Yusof, Z. M. 2021. Implementation Study of Field Programmable Gate Array (FPGA) and Complex Programmable Logic Device (CPLD) in Collision Avoidance System Using VHSIC Hardware Description Language (VHDL). *Mekatronika: Journal of Intelligent Manufacturing and Mechatronics*. 3(1): 52-60. DOI: https://doi.org/10.15282/mekatronika.v3i1.7152

[21] Zhang, B., Jiang, A., Jiang, J., Qi, Y., Xue, L., and Wang, Y. 2022. A New Positioning Strategy Based on Parameter Tuning and Optimal Control Technique for Pneumatic Control Valve. *Actuators*. 11(10): 279. DOI: https://doi.org/10.3390/act11100279

[22] Hajieghrary, H., Deisenroth, M. P., and Bekiroglu, Y. 2022. Bayesian optimization-based nonlinear adaptive PID controller design for robust mobile manipulation. In 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE). 1009-1016. IEEE. https://doi.org/10.48550/arXiv.2207.04866

[23] Su, P., Chang, G., Wu, J., Wang, Y., and Feng, X. 2024. Design and Experimental Study of an Embedded Controller for a Model-Based Controllable Pitch Propeller. *Applied Sciences*. 14(10): 3993. DOI: https://doi.org/10.3390/app14103993

[24] Albraheem, L., Alajlan, H., Aljenedal, N., Abo Alkhair, L., and Bin Gwead, S. 2023. An IoT-Based Smart Plug Energy Monitoring System. *International Journal of Advanced Computer Science and Applications*. 14(10): 353-362. DOI: 10.14569/IJACSA.2023.0141038