

# EMBEDDED PID CONTROLLER FOR PNEUMATIC LIFTING APPLICATION

LATTYFA HUSNA BINTI KHAIRRILAZAM

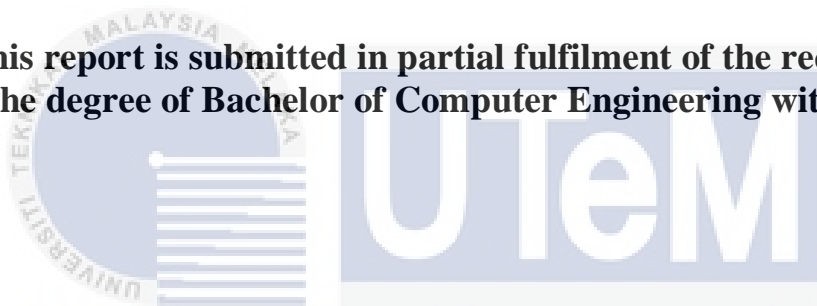


UNIVERSITI TEKNIKAL MALAYSIA MELAKA

# **EMBEDDED PID CONTROLLER FOR PNEUMATIC LIFTING APPLICATION**

**LATTYFA HUSNA BINTI KHAIRRILAZAM**

**This report is submitted in partial fulfilment of the requirements  
for the degree of Bachelor of Computer Engineering with Honours**



**Faculty of Electronic and Computer Technology and  
Engineering**

**Universiti Teknikal Malaysia Melaka**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**2024**

## DECLARATION

I declare that this report entitled “Embedded PID Controller for Pneumatic Lifting Application” is the result of my own work except for quotes as cited in the references.



اونيورسيٲى ٲكنيكل ماليسيا ملاك

Signature :  UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Author : LATTYFA HUSNA BINTI KHAIRRILAZAM

Date : 14 JUNE 2024

## APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Computer Engineering with Honours.



اونيور سيور تيكنيكل مليسيا ملاك

Signature : 

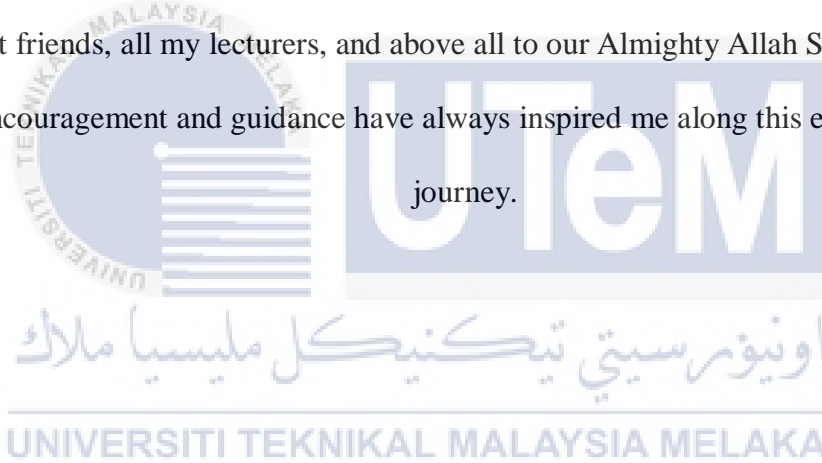
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Supervisor Name : TS. DR. SITI FATIMAH BINTI SULAIMAN

Date : 18 JUNE 2024

## DEDICATION

I wholeheartedly dedicate this thesis to my parents, my supervisor, my partner, my best friends, all my lecturers, and above all to our Almighty Allah S.W.T. Their encouragement and guidance have always inspired me along this educational journey.



## ABSTRACT

Pneumatic lifting applications are widely used in manufacturing, automotive, and aerospace industries. It uses compressed air to power the pneumatic actuators to move objects, perform tasks, or create force. However, one of the main challenges is accuracy that the lift can overshoot or undershoot the desired setpoint, which can be affected by the friction in the system, the response time of the pneumatic valve and variations in air pressure, temperature, and load. This study aims to address this problem by designing and implementing an embedded Proportional-Integral-Derivative (PID) controller for a pneumatic lifting application. The embedded PID controller is integrated into a microcontroller, such as an Arduino. The PID controller was compensated for these variations, resulting in a more accurate and robust system. The advantage of using a PID controller is that it is a simple and effective way to control a system. It is also relatively easy to implement on an embedded system. At the end of this study, an accurate and robust pneumatic lifting system that can be used in a variety of industrial applications was developed.

## ABSTRAK

*Aplikasi pengangkat pneumatik digunakan secara meluas dalam industri pembuatan, automotif dan aeroangkasa. Udara termampat digunakan untuk menggerakkan penggerak pneumatik, untuk menggerakkan objek, melaksanakan tugas, atau mencipta daya. Walau bagaimanapun, salah satu cabaran utama ialah ketepatan bahawa lif boleh melepasi atau melepasi titik set yang dikehendaki, yang boleh dipengaruhi oleh geseran dalam sistem, masa tindak balas injap pneumatik dan variasi dalam tekanan udara, suhu dan beban. Kajian ini bertujuan untuk menangani masalah ini dengan merekabentuk dan melaksanakan pengawal Terbitan-Kamiran-Berkadaran (PID) terbenam untuk aplikasi mengangkat pneumatik. Pengawal PID tertanam akan disepadukan pada mikropengawal, seperti Arduino. Pengawal PID akan mengimbangi variasi ini, menghasilkan sistem yang lebih tepat dan teguh. Kelebihan menggunakan pengawal PID ialah ia merupakan cara yang mudah dan berkesan untuk mengawal sesuatu sistem. Ia juga agak mudah untuk dilaksanakan pada sistem terbenam. Di penghujung kajian ini, diharapkan sistem pengangkat pneumatik yang tepat dan mantap yang boleh digunakan dalam pelbagai aplikasi industri akan dibangunkan.*

## ACKNOWLEDGEMENTS

First and foremost, I am grateful to Almighty Allah for the continuous blessing and giving me strength, good health, well-being, and an opportunity to complete this study. I would like to express appreciation to my parents for their willingness to send me to UTeM, to pursue my studies and always supports and gave me encouragement along my journey as a student in UTeM.

In addition, I want to thank my supervisor, Ts. Dr. Siti Fatimah Binti Sulaiman, for her guidance during this final year project, from the beginning until the end. Dr. Siti Fatimah was always willing to share her thoughts and teach me a lot of extra knowledge, which helped this study. Although she is full of schedules as a lecturer and many other final year students' supervisors, Dr. Siti Fatimah always makes her time to follow up with my progress. I am very thankful to have Dr. Siti Fatimah as my supervisor.



Finally, thanks to all my family and friends, especially my parents and partner, who supported me through ups and downs to complete my final year project. Without them, it is difficult for me to complete this report.



## TABLE OF CONTENTS

<b>Declaration</b>	
<b>Approval</b>	
<b>Dedication</b>	
<b>Abstract</b>	<b>i</b>
<b>Abstrak</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Table of Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xii</b>
<b>List of Symbols and Abbreviations</b>	<b>xiii</b>
<b>List of Appendices</b>	<b>xiv</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Introduction	1
1.2 Problem Statement	3
1.3 Objectives	4
1.4 Scope and Limitation	4

1.5	Important and Significant	5
1.6	Thesis Outline	6
<b>CHAPTER 2 LITERATURE REVIEW</b>		<b>7</b>
2.1	Overview of Pneumatic System Applications	7
2.2	Pneumatic Systems	9
2.3	Pneumatic Valves	10
2.4	Sensors	12
2.4.1	Encoder Sensors	12
2.5	Microcontroller	14
2.5.1	Embedded Systems	16
2.6	Embedded Control for Pneumatic Position Systems	17
2.6.1	Summary of Embedded Control for Pneumatic position Systems	19
2.7	Internet of Things (IoT)	20
<b>CHAPTER 3 METHODOLOGY</b>		<b>23</b>
3.1	Project Planning	23
3.2	Requirement Analysis	27
3.2.1	Software Requirement	28
3.2.2	Hardware Requirement	30
3.3	Requirement Specifications for the Project	31
3.3.1	Pneumatic Double Acting Actuator	31

3.3.2	5/3-Way Electropneumatic Valves	32
3.3.3	Arduino	34
3.3.4	Encoder Sensors	36
3.3.5	Internet of Things Communication	39
3.3.6	Blynk	39
3.4	Controller Design	41
3.4.1	Proportional-Integral-Derivative (PID)	42
3.5	Transient Response Analysis	43
3.5.1	Rise Time	44
3.5.2	Settling Time	44
3.5.3	Overshoot	44
3.5.4	Steady-state Error	45
<b>CHAPTER 4 RESULTS AND DISCUSSION</b>		<b>46</b>
4.1	Circuit Design	47
4.1.1	Valve Switching	48
4.1.2	Complete Circuit Design and Fabrication	52
4.2	Prototype Design	56
4.3	Pneumatic Stroke Position Reading	59
4.4	Fine-tuning PID Controller	67
4.5	Development of IoT System	67

4.6	Analysis of Transient Response	70
<b>CHAPTER 5 CONCLUSION AND FUTURE WORKS</b>		<b>79</b>
5.1	Conclusion	79
5.2	Future Work	80
<b>REFERENCES</b>		<b>82</b>
<b>APPENDICES</b>		<b>88</b>



## LIST OF FIGURES

<b>Figure 2.1 : The pneumatic lifting application</b>	8
Figure 2.2 : Schematic diagram of pneumatic valves	11
Figure 2.3 : Basic microcontroller structure	14
Figure 2.4 : Embedded system architecture	16
Figure 2.5 : Internet of things (IoT) architecture	20
Figure 3.1 : Flowchart of literature review and encoder circuit design	24
Figure 3.2 : Flowchart of embedded PID controller design	25
Figure 3.3 : Flowchart of embedding the system with IoT using the Blynk application	26
Figure 3.4 : The flowchart of performance analysis	27
Figure 3.5 : Pneumatic double-acting actuator	32
Figure 3.6 : 5/3-way electropneumatic valve	33
Figure 3.7 : 5/3-way electropneumatic valve schematic	33
Figure 3.8 : Arduino UNO R3	34
Figure 3.9 : Input/Output pins for Arduino board	35
Figure 3.10 : Arduino IDE interface	36
Figure 3.11 : HEDS-9730 optical encoder sensor	37
Figure 3.12 : Block diagram of HEDS-9730 optical encoder sensor	37
Figure 3.13 : Output waveform of encoder sensor	38

Figure 3.14 : ESP8266 Wi-Fi module	39
Figure 3.15 : Blynk interface	40
Figure 3.16 : C++ code to generate communication between the board and Blynk C++	41
Figure 3.17 : Closed-loop block diagram of PID controller	43
Figure 4.1 : Block diagram of the project	47
Figure 4.2 : Valve switching circuit	48
Figure 4.3 : Full schematic circuit	52
Figure 4.4 : PCB schematic design in Proteus software	54
Figure 4.5 : Completed PCB circuit	55
Figure 4.6 : Integrated PCB circuit with Arduino	55
Figure 4.7 : Front view of full prototype design	57
Figure 4.8 : Back view of full prototype design	57
Figure 4.9 : The movement of the actuator with the strip code	58
Figure 4.10 : Arrangement of the pneumatic system	58
Figure 4.11 : Serial monitor display	60
Figure 4.12 : Flow code for starting to initiate the program with the IoT system	62
Figure 4.13 : Flow code for initial state	63
Figure 4.14 : Flow code for step B	64
Figure 4.15 : Flow code for step D	65
Figure 4.16 : Flow code for step C	66
Figure 4.17 : Three parameters of PID	67
Figure 4.18 : Blynk operation system	68
Figure 4.19 : Serial print showed Blynk connected	68

Figure 4.20 : Blynk interface set for the system	69
Figure 4.21 : Graph of comparison 2cm of position stroke	72
Figure 4.22 : Graph of comparison for 4cm of position stroke	74
Figure 4.23 : Graph of comparison for 6cm of position stroke	75
Figure 4.24 : Graph of comparison for 8cm of position stroke	76
Figure 4.25 : Graph of comparison for 10cm of position stroke	77





## LIST OF TABLES

Table 2.1 Summary of the embedded control techniques for pneumatic position systems	19
Table 3.1 Table of software requirements for the project	28
Table 3.2 Software requirements for the project	30
Table 4.1 TIP120 specifications	50
Table 4.2 Arduino pins configuration	53
Table 4.3 Real-time analysis of non-embedded PID controller	70
Table 4.4 Real-time analysis of embedded PID controller	71
Table 4.5 Transient response for distance 2cm	73
Table 4.6 Transient response for distance 4cm	74
Table 4.7 Transient response for distance 6cm	75
Table 4.8 Transient response for distance 8cm	76
Table 4.9 Transient response for distance 10cm	77

## LIST OF SYMBOLS AND ABBREVIATIONS

PID	:	Proportional Integral Derivative
IoT	:	Internet of Things
PWM	:	Pulse-Width-Modulation
IP	:	Internet Protocol
USB	:	Universal Serial Bus
FRL	:	Filter, Regulator and Lubricator
I/O	:	Input/Output
$V_{CE}$	:	Collector-Emitter Voltage
$I_C$	:	Collector Current
$I_B$	:	Base Current
V	:	Voltage
I	:	Current
R	:	Resistor
$\Omega$	:	Ohm
Cm	:	centimeter
PFC-O	:	Predictive Functional Control with Observer
NI	:	National Instruments

## LIST OF APPENDICES

Appendix A: Source Code .....	102
-------------------------------	-----



# CHAPTER 1

## INTRODUCTION



### 1.1 Introduction

In the dynamic landscape of industrial automation, precise control is crucial for the optimal performance of mechanical systems. Pneumatic lifting applications, utilized across sectors like manufacturing, material handling, and robotics, require sophisticated control mechanisms to ensure seamless operation.

In the rapidly evolving field of automation and control systems, the integration of Internet of Things (IoT) technologies has revolutionized efficiency and intelligence. This thesis introduces the development and implementation of an embedded Proportional-Integral-Derivative (PID) controller for a pneumatic lifting system, harnessing IoT capabilities via the Blynk application.

Pneumatic systems are widely employed in various industrial applications due to their simplicity, reliability, and cost-effectiveness, playing integral roles in modern industrial processes [1]. The ability to lift and position loads accurately is crucial, necessitating advanced control strategies. A typical pneumatic lift employs a hollow cylinder and piston mechanism driven by an external motor or pump. This setup increases air pressure within the cylinder, enabling linear force generation for extending lift components. One of the primary advantages of pneumatic lifts lies in their versatility. However, achieving precise control over such systems can be challenging due to inherent nonlinearities and dynamic characteristics.

Traditional control methods often struggle to provide the required accuracy and responsiveness. To address these challenges, the Proportional-Integral-Derivative (PID) control algorithm, renowned for its robustness and simplicity, is employed. By adjusting the proportional, integral, and derivative gains, the PID controller effectively minimizes the error between desired and actual positions of the pneumatic actuator, ensuring smooth and accurate lifting operations.

The integration of IoT enhances the functionality and flexibility of the control system. Leveraging the Blynk application, a versatile IoT platform, enables remote monitoring and control of the pneumatic lifting system. Operators can set target positions, adjust PID parameters, and monitor system performance in real-time from any location with internet access. This not only improves operational efficiency but also allows for timely interventions and adjustments, enhancing overall reliability and safety.

The primary objectives of this project are to design, implement, and validate an embedded PID controller for a pneumatic lifting system and integrate this system with

the Blynk application for enhanced remote monitoring and control. This involves developing both hardware and software components, including selecting appropriate sensors and actuators, programming the microcontroller for PID control, and configuring the Blynk application for IoT connectivity.

The successful implementation of this project demonstrates the potential of combining traditional control algorithms with modern IoT technologies to create smart, efficient, and user-friendly control systems. It provides valuable insights into the design and deployment of IoT-based control solutions, paving the way for further advancements in industrial automation.

## 1.2 Problem Statement

This study addresses several key issues:

### Control Accuracy:

- The primary challenge is achieving precise control over the positioning of the pneumatic cylinder stroke.
- Pneumatic lifting applications often experience issues of overshooting or undershooting the desired set-point.

### Factors Contributing to Control Challenges:

- These issues arise due to various factors such as load variations, system friction, and the response time of pneumatic valves.

- Additionally, pneumatic systems can be sensitive to disturbances, where changes in air pressure may cause unintended movements despite constant control signals [2].
- To mitigate these challenges, this study proposes implementing a PID controller as an effective strategy for controlling the cylinder stroke position in pneumatic systems.

### 1.3 Objectives

The objectives of this study are as follows:

- To design an embedded PID controller for achieving precise control of a pneumatic lifting application.
- To integrate an IoT device and control the position of the pneumatic stroke using the Blynk application.
- To analyze the transient response performance of the pneumatic lifting application using the embedded PID controller proposed in this study.

### 1.4 Scope and Limitation

- Designing a pneumatic controller capable of monitoring and controlling the system using an Arduino microcontroller integrated with Internet of Things (IoT) devices.
- Using a 24V DC power supply was used to activate the solenoid valve. Developing an embedded Proportional-Integral-Derivative (PID) controller as the primary control strategy for positioning the pneumatic system in lifting applications.

- iii. Employing Arduino as the microcontroller platform for embedding the PID controller.
- iv. Limiting the maximum stroke length of the pneumatic system to 10cm due to system constraints.
- v. Integrating the system with IoT through the Blynk application for remote monitoring and control.
- vi. Analyzing the performance of the PID controller in terms of transient response to evaluate its effectiveness in controlling position.

### **1.5 Important and Significant**

The project holds substantial potential for commercialization across various industries such as material handling, manufacturing, automated packaging, assembly lines, and automotive sectors. The embedded PID controller proposed in this study contributes to sustainability and environmental friendliness by addressing SDG 12 on responsible consumption and production. By optimizing control signals, the PID controller reduces energy consumption in pneumatic lifting systems, leading to significant energy savings, particularly in applications where these systems are extensively used.

Additionally, the project aligns with SDG 3 on good health and well-being by enhancing safety in industrial processes. The PID controller prevents overextension or under-extension of pneumatic actuators, thereby reducing risks to human health and promoting a safer work environment. This improvement can also minimize downtime caused by accidents, enhancing operational efficiency.



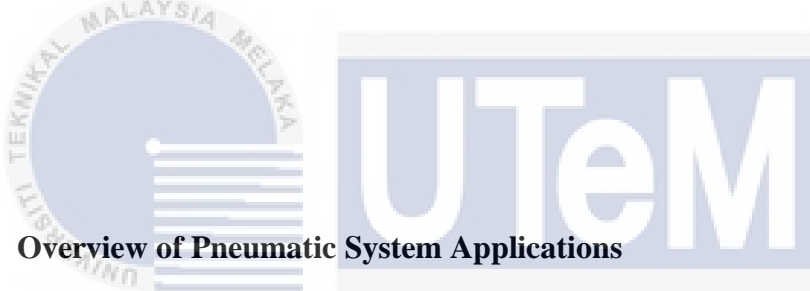
Moreover, the project supports SDG 9 on industry, innovation, and infrastructure by fostering technological advancements in industrial automation. The development of an innovative embedded system for pneumatic actuators, coupled with IoT technology for remote monitoring and control, enhances infrastructure by facilitating data transfer and optimizing system functionality.

## 1.6 Thesis Outline

This thesis is organized into five chapters. Chapter 1 introduces pneumatic systems, covering their principles, applications, and the use of PID controllers for position control. It defines the problem statement, objectives, scope, and significance of the study. Chapter 2 reviews relevant literature on pneumatic systems and control strategies. Chapter 3 outlines the methodology, detailing the process flow, component specifications, conceptual framework, and circuit design principles for implementation. Chapter 4 presents the study's outcomes, including fabrication results, circuit explanations with IoT integration, and analysis of the PID controller's performance in position control. Finally, Chapter 5 concludes with a summary of achievements, implications of findings, and recommendations for future research to enhance system quality and applicability.

## CHAPTER 2

### LITERATURE REVIEW



#### 2.1 Overview of Pneumatic System Applications

Pneumatic systems have become integral to numerous industrial applications due to their efficiency and versatility. They are extensively used in industrial manipulators and robotics for their precise and rapid movement capabilities. Pneumatic actuators excel in applications requiring high speeds and repetitive motions, offering quick acceleration and deceleration. Their resilience in harsh environments, such as high temperatures and explosive atmospheres, further enhances their utility in industrial settings.

Figure 2.1 depicts a typical pneumatic lifting application, showcasing the fundamental components used in such systems. It includes a hollow cylinder and piston arrangement driven by pressurized air supplied through an external motor or pump. This setup allows for the generation of linear force necessary to extend the lift

components, highlighting the simplicity and effectiveness of pneumatic actuators in industrial environments.



**Figure 2.1 : The pneumatic lifting application**

The core of a pneumatic lift system typically comprises a hollow cylinder and piston. When pressurized air is supplied via an external motor or pump, the piston moves within the cylinder, generating linear force to extend the lift components. One significant advantage of pneumatic lifts is their adaptability to various applications, including those demanding reliable operation in extreme environmental conditions. Unlike hydraulic actuators, pneumatic systems are valued for their precision, capability to generate substantial force, and cost-effectiveness [3].

## 2.2 Pneumatic Systems

In recent years, pneumatic systems have gained significant traction and found extensive implementation in various applications. Industrial manipulators and robotics heavily rely on pneumatic actuators for their precise and efficient movement capabilities. Pneumatic actuators are particularly well-suited for applications that require rapid and repetitive motion, as they can achieve high speeds and acceleration [4]. Moreover, their ability to operate in harsh and demanding environments, such as high temperatures or explosive atmospheres, makes them highly versatile in industrial settings [5].

The pneumatic cylinder, a fundamental component of a pneumatic actuator, is crucial in transforming compressed air energy into mechanical motion. It typically consists of a piston, a cylinder, and valves. When pressurized air is introduced into the cylinder, it exerts a force on the piston, causing it to move along its length. This linear motion can be further utilized to perform various tasks, such as actuating valves, opening and closing doors, or driving mechanical systems [6].

To meet the evolving demands of different applications, the development of pneumatic actuators focuses on several vital aspects. This includes improving efficiency, enhancing the power-to-weight ratio, rationalizing construction, and introducing innovative designs and technologies [7]. By addressing these areas, researchers and engineers aim to optimize the performance and capabilities of pneumatic actuators.

The history of pneumatic actuators can be traced back to pneumatic system tools, such as hammer drills, air pumps, compressors, and tubes [8]. Over time, pneumatic systems have become more advanced and intelligent, incorporating modern

technologies and control mechanisms [9]. Modern pneumatic technology relies on diverse sensing arrays comprising sensing elements responsive to different stimuli, including pressure, force, heat, and displacement [10]. Integrating sensing technologies in pneumatic systems enhances their performance, precision, and control capabilities.

In the automotive industry, pneumatic systems play a significant role, and various sensors have contributed to their advancements. Fleming provides a comprehensive review of automotive sensor technologies, emphasizing the impact these sensors have on developing automotive systems, including pneumatic systems [11]. With nearly 50 different types of automotive sensors in use, the application of sensors in pneumatic systems has contributed significantly to their functionality and efficiency.

### **2.3 Pneumatic Valves**

Pneumatic valves play a critical role in controlling the flow of compressed air or gases across various industrial applications. Over the past decade, significant advancements in pneumatic valve technology have focused on improving efficiency, reliability, and overall performance. This literature review aims to provide a comprehensive overview of key developments, applications, and research trends in pneumatic valves during this period.

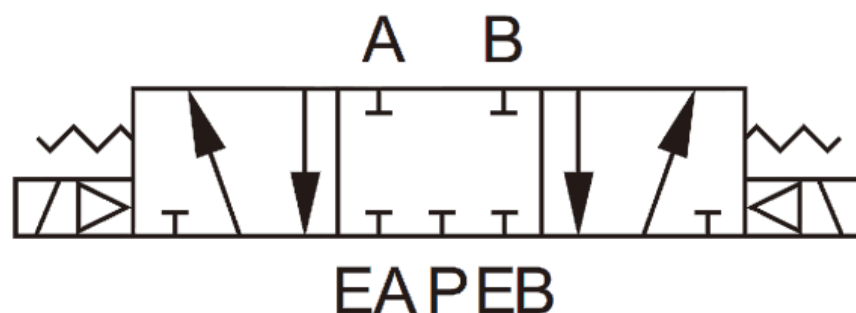
Several notable studies contribute to our understanding of pneumatic valve advancements. Detailed analyses of different pneumatic valve designs and actuation mechanisms emphasize optimizing these designs for efficient flow control [12]. Advances in smart pneumatic valves highlight the integration of sensing technologies,

intelligent control algorithms, and communication protocols to enhance functionality within industrial automation systems [13].

In the realm of energy-saving applications, Tanaka and Yamamoto discuss the development of pneumatic proportional control valves, showcasing the application of advanced control algorithms such as fuzzy logic and neural networks to optimize valve performance and reduce energy consumption [14]. In manufacturing techniques, Zhang explores additive manufacturing for pneumatic valves, highlighting its potential to revolutionize complex component production through design flexibility and material selection [15].

Finally, advanced control strategies for pneumatic valve systems in robotics have focused on model-based control techniques [16], including adaptive and predictive control, to improve accuracy, response time, and overall performance in robotic applications.

Figure 2.2 illustrates the essential components and flow paths of pneumatic valves used in industrial applications. It highlights the actuation mechanisms and control points crucial for regulating the flow of compressed air or gases.



**Figure 2.2 : Schematic diagram of pneumatic valves**

## 2.4 Sensors

Sensors are essential devices that detect and measure physical quantities or environmental conditions, converting them into electrical signals. They are integral across engineering, science, medicine, and consumer electronics, facilitating functions such as temperature, pressure, light, and motion sensing. Advancements in semiconductors, Micro-Electro-Mechanical Systems (MEMS), and nanotechnology have led to smaller, more sensitive sensors. Integration with artificial intelligence (AI) and wireless connectivity has addressed challenges like calibration and power consumption. Sensors play a pivotal role in data collection and utilization in today's interconnected world. When integrated with microcontrollers, which act as the central processing units, sensors enable efficient data processing and decision-making. This integration is particularly advantageous in applications where low latency and high reliability are critical, such as industrial automation, robotics, and embedded systems. Standardization ensures seamless communication between sensors and devices, further enhancing their utility. Ongoing advancements, including flexible and wearable sensors, continue to broaden their applications, driving innovation across diverse industries.

### 2.4.1 Encoder Sensors

Encoder sensors are essential for accurately measuring position and velocity in control systems, particularly in pneumatic actuators. Optical encoders have been successfully applied to achieve precision control in pneumatic systems. Research has demonstrated their effectiveness in providing accurate position feedback and enabling precise motion control [17].

Magnetic encoder sensors have also been investigated for reliable position sensing in pneumatic actuators. Studies have focused on designing and implementing magnetic encoder systems that ensure robust position sensing in pneumatic applications [18].

Incremental encoder sensors have been utilized for closed-loop control of pneumatic actuators, enhancing position control and tracking performance. These systems leverage incremental encoder feedback to achieve accurate control in pneumatic actuator setups [19].

Optoelectronic encoder sensors offer high-speed control capabilities for pneumatic actuators. Research has shown their effectiveness in high-resolution position sensing and rapid response times in dynamic control scenarios involving pneumatic actuators [20].

Integrated wireless encoder sensors have been developed for real-time monitoring and control of pneumatic actuator performance in IoT-based systems. These systems enable IoT-driven monitoring and control of pneumatic actuators, enhancing connectivity and intelligence in industrial applications [21].

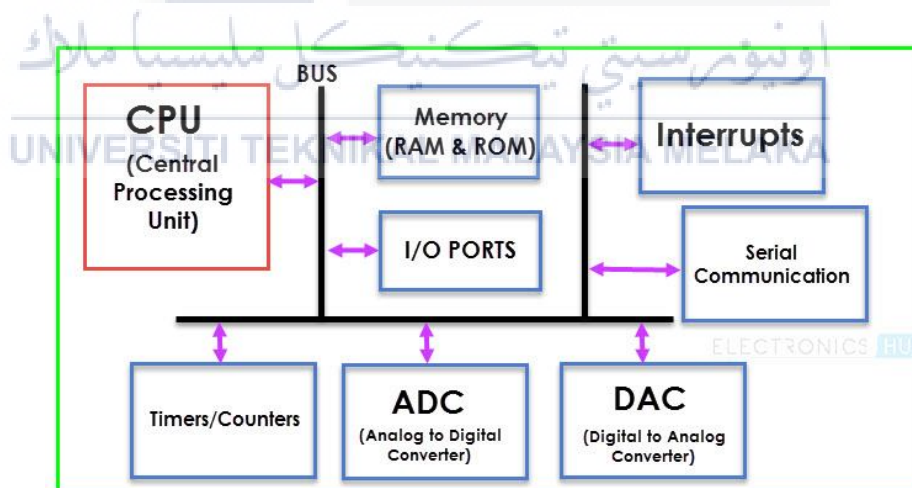
Overall, advancements in encoder sensor technology play a crucial role in improving the precision and efficiency of pneumatic actuator control systems, expanding their applications across various industries and systems.



## 2.5 Microcontroller

In recent years, microcontrollers have undergone significant advancements, revolutionizing embedded systems across industries such as automotive, consumer electronics, medical devices, and industrial automation [22]. Their compact size, high integration levels, low power consumption, and real-time control capabilities have driven their widespread adoption. Understanding the latest developments in microcontroller technology is crucial as demand grows for intelligent and interconnected devices.

When designing systems with microcontrollers, considerations like architecture, processing power, memory capacity, input/output options, and peripheral support are paramount [16], as illustrated in Figure 2.3. These factors ensure that the microcontroller meets specific application requirements, optimizing performance and functionality.



**Figure 2.3 : Basic microcontroller structure**

For example, the Arduino platform exemplifies a popular microcontroller ecosystem, leveraging various chips for open-source hardware and software development. Arduino boards, featuring microcontrollers like the ATmega328P, facilitate rapid prototyping and project development for hobbyists, students, and professionals. The ATmega328P, found in Arduino Uno boards, boasts an 8-bit AVR architecture, 32KB flash memory, 2KB RAM, and versatile I/O options. It supports communication protocols like I2C, SPI, and UART, enabling connectivity with sensors, actuators, and external devices. Arduino's simplified programming language, based on Wiring, includes a vast library for streamlined development.

Ongoing research focuses on enhancing microcontroller energy efficiency, emphasizing power management techniques and optimization algorithms to extend battery life and reduce environmental impact [23]. These energy-efficient microcontrollers are pivotal for sustainable technology solutions.

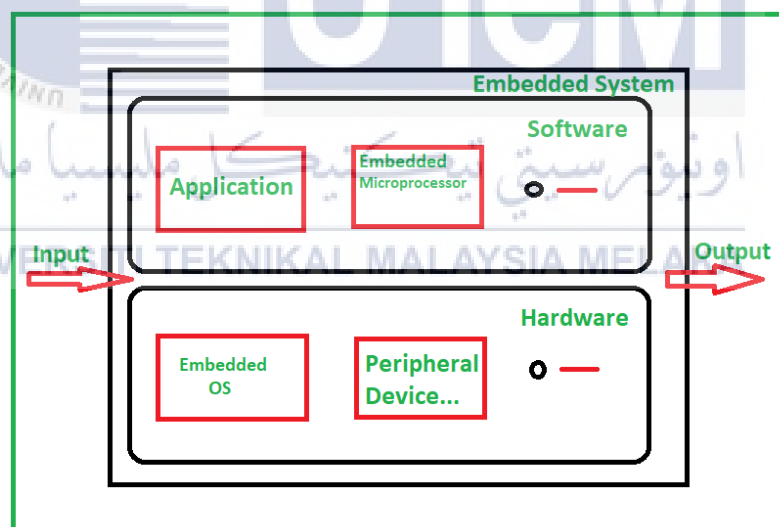
Microcontrollers also play a crucial role in IoT deployments [24], requiring low-power communication protocols and efficient data-handling algorithms. Their ability to connect seamlessly with IoT devices and systems supports applications in smart homes, cities, and industrial automation.

Furthermore, integrating AI capabilities into microcontrollers is a burgeoning area. AI-enabled microcontrollers utilize machine learning algorithms and neural networks for edge computing, enabling intelligent decision-making and enhancing efficiency in real-time applications.

This review underscores the diverse applications and continuous evolution of microcontroller technology, driving innovations across multiple sectors and paving the way for smarter, more efficient embedded systems.

### 2.5.1 Embedded Systems

Embedded systems are specialized computer systems designed to perform specific tasks within larger devices or systems. They are characterized by their dedicated functionality, real-time operation, and integration into broader systems [25]. Typically, embedded systems feature a core processing unit, such as a microcontroller or microprocessor, along with additional hardware components like sensors, actuators, and memory modules [26]. The architecture of an embedded system is illustrated in Figure 2.4.



**Figure 2.4 : Embedded system architecture**

Embedded systems find application across diverse domains, including consumer electronics, automotive systems, medical devices, and industrial automation [27]. In

automotive applications, embedded systems control critical functions such as engine management, anti-lock braking systems, and airbag deployment, ensuring safety and operational efficiency [22]. These systems are engineered to execute predefined tasks within precise time constraints, enhancing the overall reliability and performance of automotive systems.

A prominent example of an embedded system is the intelligent thermostat used in home automation. This device integrates a microcontroller to monitor temperature, process user inputs, and regulate the heating or cooling system accordingly [28]. By leveraging embedded systems, the smart thermostat achieves precise temperature control, energy efficiency, and interoperability with other smart devices in a home network.

Due to their resource constraints, including limited processing power, memory, and energy consumption allowances [22], embedded systems require efficient software development practices. These include the use of real-time operating systems and strategies for optimizing power consumption to ensure optimal performance and reliability in various applications.

## 2.6 Embedded Control for Pneumatic Position Systems

Osman *et al.* (2014) introduced a predictive functional control with observer (PFC-O) strategy for pneumatic systems, demonstrating its effectiveness in simulations and real-time experiments using MATLAB and NI devices. They achieved a settling time of 0.79 seconds for position control, negligible steady-state error in force control, and no overshoot in force analysis [29].

Abdul-Lateef *et al.* (2020) proposed a model employing PWM and a fuzzy PD controller to enhance control accuracy in electro-pneumatic systems. Their research focused on using PWM to simulate proportional valves at 50Hz with 5 bar pressure, coupled with a fuzzy PD controller in MATLAB/Simulink for precise control. This approach aimed to offer cost-effective alternatives for industrial automation and mechatronic applications, showing improved performance compared to traditional models [30].

Ahmad (2021) designed an FPGA-based controller for pneumatic position systems, achieving less than 1% steady-state error and settling times under 2 seconds across various inputs. The FPGA implementation provided higher sampling rates and better noise immunity compared to conventional microcontroller-based systems [31].

Zhang (2022) developed a model-based controller using the linear quadratic regulator (LQR) method for pneumatic systems, demonstrating less than 0.5% steady-state error and settling times under 1 second. The controller showed robust performance against model uncertainties, maintaining high accuracy even with parameter variations up to 20% [32].

Ghazaly (2022) utilized a mathematical model based on the ideal gas law and continuity equation, implementing a PID controller tuned via the Ziegler-Nichols method. Their study reported 98% accuracy and 2-second settling times for step and ramp inputs, demonstrating the PID controller's effectiveness in pneumatic position control [33].

In another study, Zhang (2023) designed and implemented a PID controller on an embedded platform for pneumatic cylinders, achieving satisfactory performance

across step, ramp, and sinusoidal inputs. Their findings emphasized the significant impact of proportional and integral gains on settling time and steady-state error, respectively, showcasing the PID controller's versatility and effectiveness in various operating conditions [34].

### 2.6.1 Summary of Embedded Control for Pneumatic position Systems

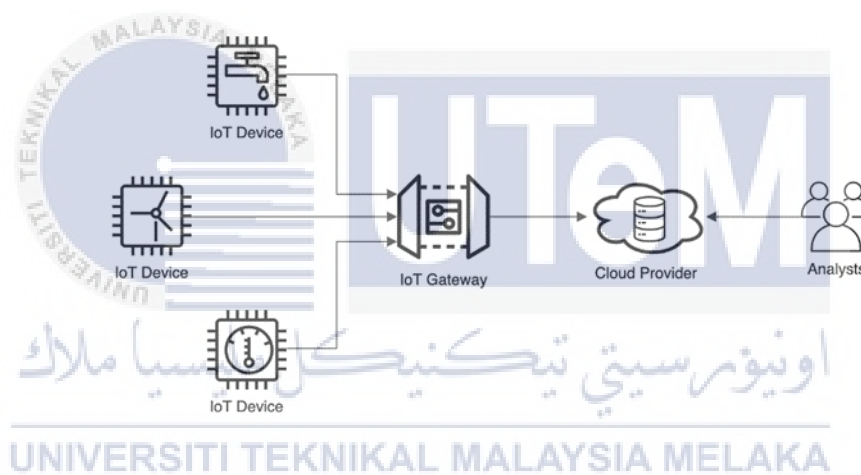
Several control strategies have been investigated for the precise control of pneumatic position systems, encompassing PID controllers, FPGA-based solutions, and model-based approaches. Table 2.1 summarizes the embedded control techniques employed by previous researchers in pneumatic position systems.

**Table 2.1 Summary of the embedded control techniques for pneumatic position systems**

Author	Control Strategies	Steady-state Error ( $e_{ss}$ )	Settling Time ( $t_r$ )	Embedded System
Osman <i>et al.</i> (2014) [29]	Predictive functional control with observer (PFC-O)	Smaller	0.79 seconds	Yes
Wisam Essmat Abdul-Lateef <i>et al.</i> (2020) [30]	Fuzzy Logic Controller	Smaller		No
M. Ahmad (2021) [31]	FPGA-based controller	1%	2 seconds	No
J. Zhang (2022) [32]	Model-based controller	2%	4 seconds	No
M. Ghazaly (2022) [33]	PID controller		2 seconds	No
H. Zhang (2023) [34]	PID controller	1%	2 seconds	No

## 2.7 Internet of Things (IoT)

The Internet of Things (IoT) has emerged as a transformative concept in recent years, revolutionizing how physical objects and devices interact and communicate with each other and with humans. It entails a vast network of interconnected objects embedded with sensors, software, and connectivity, enabling them to collect and exchange data over the internet. This network encompasses various items, ranging from standard devices like smartphones, wearable gadgets, and home appliances to complex industrial machinery, smart city infrastructure, and autonomous vehicles [34].



**Figure 2.5 : Internet of things (IoT) architecture**

Since its inception, IoT has experienced remarkable growth and development, becoming one of the most discussed and rapidly evolving fields in information technology. The IoT landscape has witnessed significant advancements, including 40 technological innovations, standardization efforts, and a proliferation of connected devices and applications. These advancements have deepened our understanding of IoT's potential and expanded its reach into various industries and domains.

The fundamental principle behind IoT is to enable seamless communication and interaction among objects, as well as between objects and humans, creating a cohesive and interconnected ecosystem [27]. By leveraging data collected from diverse sources, IoT devices can offer valuable insights, automate processes, and enhance efficiency and convenience in numerous fields. This ability to harness and analyze vast amounts of data has created new opportunities for businesses, governments, and individuals to make data-driven decisions and optimize operations.

In IoT, each object is assigned a unique identity, allowing it to be easily recognized and connected within the network. This identity can be established through various means, such as radio-frequency identification (RFID) tags, embedded sensors, or machine-readable codes. These identifiers facilitate seamless connectivity and communication across geographical boundaries, enabling objects to interact regardless of their physical location or time zone. With the exponential growth of IoT devices, vast amounts of data are being generated at an unprecedented rate. This data, often referred to as Big Data, holds immense potential for deriving valuable insights and enabling data-driven decision-making.

However, managing and analyzing such enormous volumes of data presents technical and logistical challenges. Therefore, advanced data analytics techniques, including artificial intelligence (AI) and machine learning, have become crucial in extracting meaningful information from the vast data streams flowing through IoT [35].

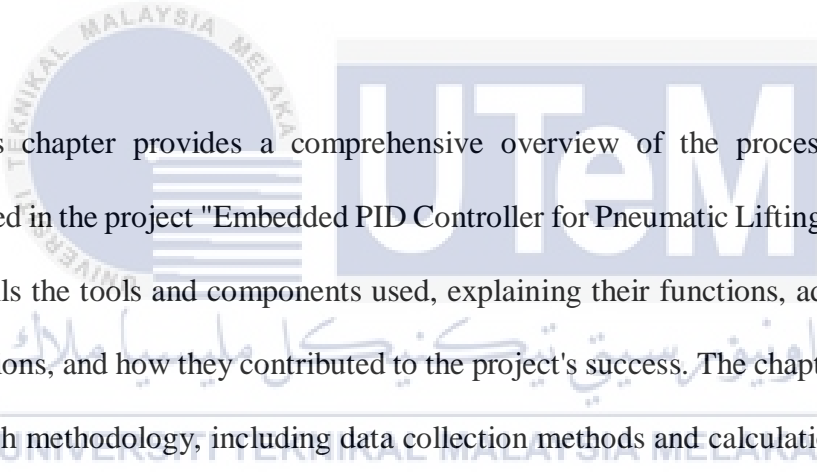


As IoT continues to evolve, its impact spans numerous sectors. IoT devices enable remote patient monitoring, real-time health tracking, and personalized medicine in healthcare. In agriculture, IoT sensors assist in optimizing irrigation, monitoring crop health, and improving yield. Smart cities leverage IoT technology for efficient resource management, traffic control, and environmental monitoring.

Moreover, manufacturing, logistics, energy, and transportation industries are undergoing transformative changes by integrating IoT solutions. The adoption of IoT technology has not been without challenges. Security and privacy concerns are among the primary considerations. Securing the IoT ecosystem with billions of connected devices becomes a complex task. Ensuring the confidentiality, integrity, and availability of data transmitted and stored by IoT devices is crucial to protect against cyber threats and unauthorized access. Additionally, privacy concerns arise from collecting and analyzing personal data, highlighting the need for robust data protection regulations and frameworks [36].

## CHAPTER 3

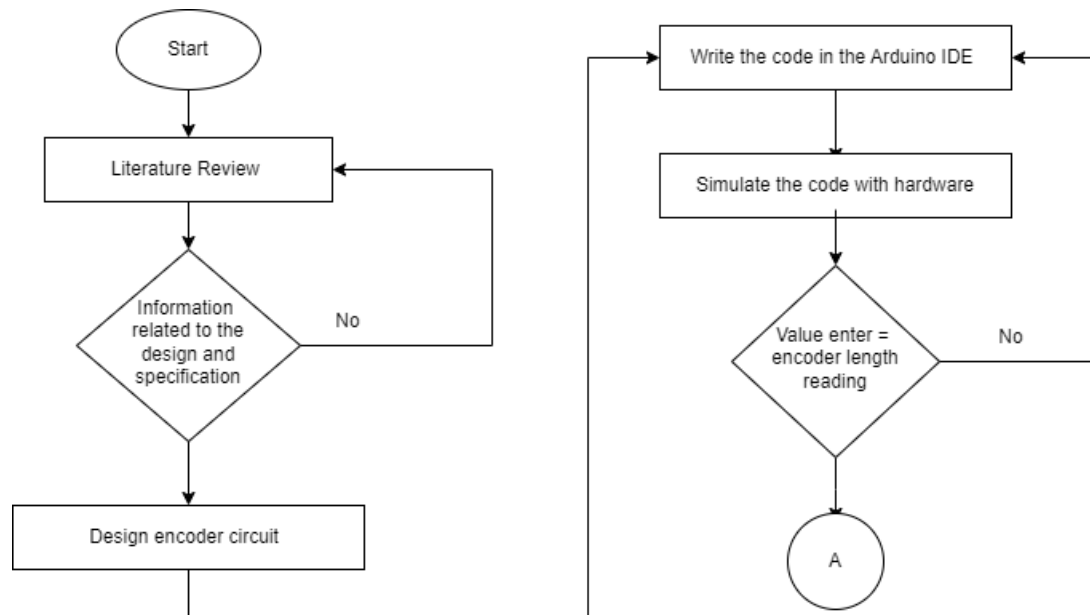
### METHODOLOGY



This chapter provides a comprehensive overview of the processes and steps involved in the project "Embedded PID Controller for Pneumatic Lifting Application." It details the tools and components used, explaining their functions, advantages, and limitations, and how they contributed to the project's success. The chapter outlines the research methodology, including data collection methods and calculation techniques, used to develop an innovative pneumatic system controller and monitor integrated with Internet of Things (IoT) technology.

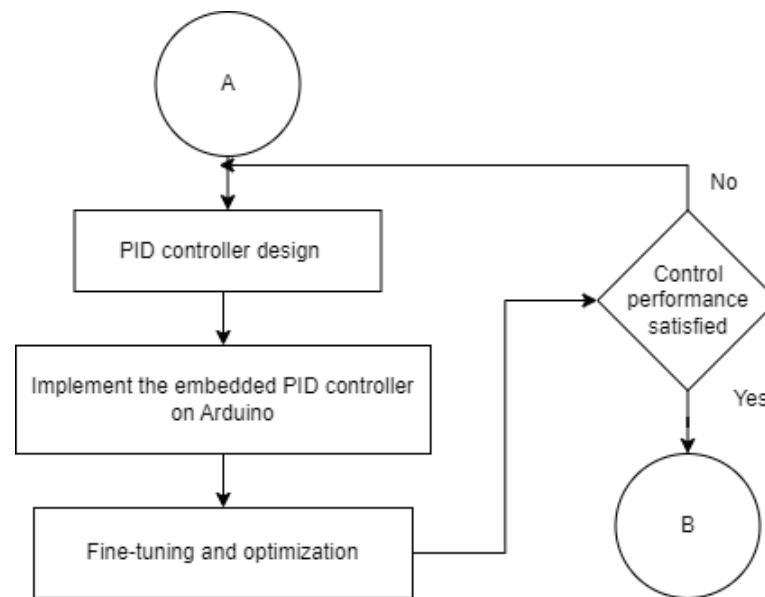
#### 3.1 Project Planning

This study consisted of four main stages: literature review and encoder circuit design, controller design and simulation testing, embedding the system with Internet of Things (IoT) using the Blynk application, and performance analysis.



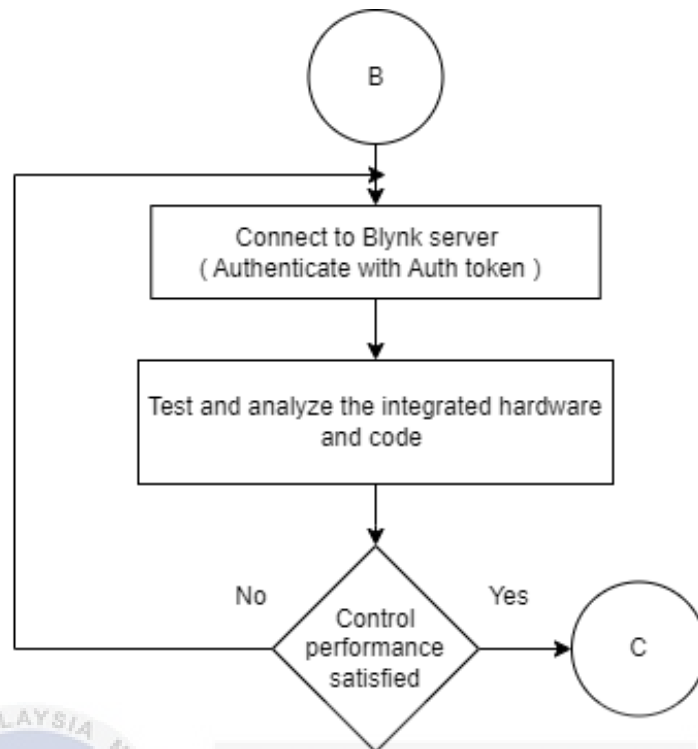
**Figure 3.1 : Flowchart of literature review and encoder circuit design**

The literature review is the initial stage of this project. The main objective of this stage is to highlight the issues related to the position control of a pneumatic system. If the information is sufficient, the next step is to design the encoder circuit. This involves designing and implementing the encoder circuit using Arduino. Once the circuit design is complete, the code is written in the Arduino IDE and then simulated with hardware. The simulation step includes verifying if the entered value matches the encoder length reading. If it does not, adjustments are made, and the process loops back to writing and simulating the code again. If the values match, the process is considered complete.



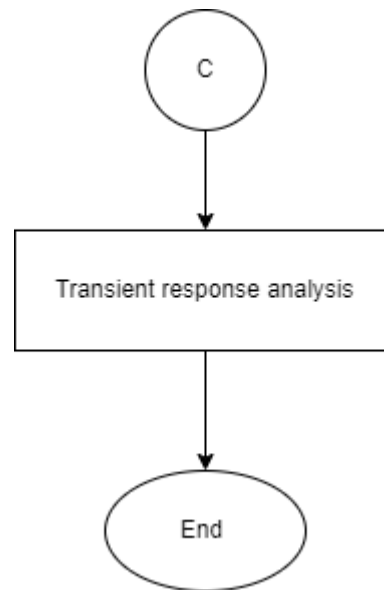
**Figure 3.2 : Flowchart of embedded PID controller design**

In the third stage of this project, the steps following the initial encoder circuit design and code validation (denoted by point A) are undertaken. This stage begins with the design of a PID (Proportional-Integral-Derivative) controller. Once the PID controller design is complete, it is implemented on the Arduino platform. Following implementation, fine-tuning and optimization of the PID controller are conducted to enhance performance. The system's control performance is then evaluated. If the performance is not satisfactory, the process loops back to the fine-tuning and optimization step. If the control performance is satisfactory, the process proceeds to the next stage (denoted by point B).



**Figure 3.3 : Flowchart of embedding the system with IoT using the Blynk application**

A framework is embedded into the system to further enhance its capabilities. Figure 3.3 depicts the integration of the pneumatic system with an IoT system. This integration allows for remote monitoring and control of the pneumatic system using network connectivity and data communication protocols. The IoT functionality enables industrial users to access real-time data, receive alerts, and remotely adjust system parameters, thereby enhancing operational efficiency and enabling proactive maintenance.



**Figure 3.4 : The flowchart of performance analysis**

In the final stage of this project (Stage 4), the performance of the embedded PID controller in controlling the position of the pneumatic system will be analyzed to determine necessary improvements. This analysis will focus on evaluating the transient response, including percent overshoot ( $\%OS$ ), rise time ( $Tr$ ), settling time ( $Ts$ ), peak time ( $Tp$ ), and percent steady-state error ( $\%ess$ ). Finally, the design of the embedded PID controller will be fine-tuned and optimized.

### 3.2 Requirement Analysis

To ensure the proper development of the system, a harmonious integration of both software and hardware components was deemed indispensable. The following sections elaborate meticulously on the software and hardware requirements employed throughout the system's developmental process. By carefully considering and implementing these requirements, the development team aimed to create an environment conducive to accuracy and efficiency in system development.

### 3.2.1 Software Requirement

The software requirements encompassed several crucial elements to ensure the accurate creation of the system. Firstly, selecting an appropriate operating system was essential, considering compatibility and performance. Secondly, an integrated development environment (IDE) provided a comprehensive suite of tools for coding, debugging, and testing purposes. The programming languages and frameworks were carefully chosen to align with the system's requirements and the team's proficiency, facilitating efficient development. Lastly, a reliable database management system (DBMS) was utilized to ensure efficient data storage and retrieval, thereby facilitating seamless system information management.

**Table 3.1 Table of software requirements for the project**

Software	Function
Arduino IDE	Includes a code editor, a message area, a text console, a toolbar with buttons for standard functions, and a series of menus. It communicates with and uploads programs to the Arduino hardware.
Blynk	An IoT platform for iOS or Android smartphones that uses the Internet to control Arduino, Raspberry Pi, and NodeMCU. This application compiles and provides the appropriate address on the available widgets to create a graphical or human machine interface (HMI) interface.
Proteus	An excellent open-source tool for teaching, sharing, and prototyping electronic projects is now available. It enables the creation of professional-looking wiring diagrams and schematics by incorporating custom-designed parts. Furthermore, it allows you to design and fabricate PCBs from the files you create, providing a complete solution for electronic project development.

<p>Microsoft Office</p>	<p>a. Microsoft Word, a vital tool for thesis writing, offers a comprehensive range of features and functionalities to assist scholars in organizing ideas, formatting text, managing references, and presenting their work professionally. Its user-friendly interface and robust capabilities make it an indispensable companion for crafting impactful and polished academic manuscripts.</p> <p>b. Microsoft Excel, an eminent and formidable tool, is pivotal in data collection and analysis. Its robust capabilities empower users to seamlessly gather and scrutinize vast amounts of information, unravelling profound insights and illuminating the path toward informed decision-making.</p> <p>c. Microsoft PowerPoint, a sublime and commanding tool, reigns as the undisputed champion of slide preparation. Its majestic features and beautiful design templates empower users to create visually stunning presentations that leave a lasting impact on audiences.</p>
-------------------------	---

Table 3.1 details the software utilized throughout the project, providing insight into its successful execution. This regal tableau demonstrates how software tools were carefully chosen to oversee the development process, balancing complicated jobs and organizing technological capability. The project was meticulously planned and supported by superb software collaborators, resulting in a magnum opus that will be remembered for its achievements.



### 3.2.2 Hardware Requirement

Hardware requirements are the basis of successful system implementation since they include all the physical components required for it to work properly. These requirements provide vital infrastructure for software development and support. One important part of hardware requirements is carefully selecting a powerful and appropriate processor capable of handling the system's processing demands. Establishing a reliable network connection is crucial for effective communication and data movement within the system. In summary, hardware requirements are critical to guaranteeing a system's smooth operation and optimal performance, as well as aiding the right integration of software components.

**Table 3.2 Software requirements for the project**

Hardware	Function
Optical encoder sensor (HEDS-9730)	A sensor that converts mechanical motion into electrical signals. It uses patterns or slots on a rotating or linearly moving disk to detect position, speed, and direction. It is widely used in robotics, automation, and motion control systems for accurate feedback and control.
Arduino UNO board	A popular microcontroller board that provides an easy-to-use platform for developing and prototyping electronic projects. It has a variety of input/output pins, analogue inputs, and communication interfaces, making it compatible with various components. Its ease of use and low cost make it a popular choice for DIY electronics and rapid prototyping.
ESP8266 Wi-Fi module	A popular and versatile component that adds wireless connectivity to electronic projects. It integrates a

	microcontroller with built-in Wi-Fi capabilities, ideal for IoT applications and wireless communication.
Pneumatic double-acting actuator	A mechanical device for bidirectional motion control.
5/3 way electro-pneumatic valve	A valve for precise compressed air control in pneumatic systems.
Resistor	An electronic component that restricts current flow in circuits.
Diode (1N4001)	A rectifier diode for current direction control.
Transistor (TIP 120)	A high-power switching transistor for driving loads.

### 3.3 Requirement Specifications for the Project

The requirement specifications for the project serve as a concise blueprint outlining the essential criteria and functionality that the system must adhere to. These specifications define the specific features, capabilities, and constraints that guide the development process. They establish clear goals and expectations to ensure alignment and clarity throughout the project lifecycle. By outlining these specifications, the project's design, execution, and testing phases are guided with precision, ensuring accuracy and coherence in the final deliverable.

#### 3.3.1 Pneumatic Double Acting Actuator

A pneumatic double-acting actuator is a mechanical device powered by compressed air to control the movement of a mechanism in both extending and retracting directions. It consists of a piston housed within a cylinder, with two ports for air supply: the "extend" and "retract" ports, as depicted in Figure 3.5.

When compressed air is supplied to the extend port, it pushes the piston forward, causing the actuator to extend and perform a specific action. Conversely,

directing compressed air to the retract port pushes the piston in the opposite direction, retracting the actuator.



**Figure 3.5 : Pneumatic double-acting actuator**

In this project, the SMC CD85N20-100-B actuator was selected due to its specifications. The CD85N20-100-B actuator features a bore size of 20 mm and a stroke length of 100 mm. These specifications denote the diameter of the actuator's cylinder and the distance the piston can travel within it. It is important to note that the CD85N20-100-B model may have additional variations and specifications tailored to specific application requirements.

### 3.3.2 5/3-Way Electropneumatic Valves

The 5/3-way electropneumatic valve depicted in Figure 3.6 is a type of valve used for precise control of compressed air in pneumatic systems. It features five ports and three distinct flow paths. These ports are labeled as "supply," "exhaust," and three "working" ports typically designated as "A," "B," and "P," as illustrated in Figure 3.7.



**Figure 3.6 : 5/3-way electropneumatic valve**

The 5/3-way valve enables diverse operational modes and functions by directing airflow through its multiple ports and flow paths. Depending on the specific configuration and control signals applied, the valve can achieve various combinations of pressurized air supply, exhaust, and flow direction to effectively control the motion of pneumatic actuators.



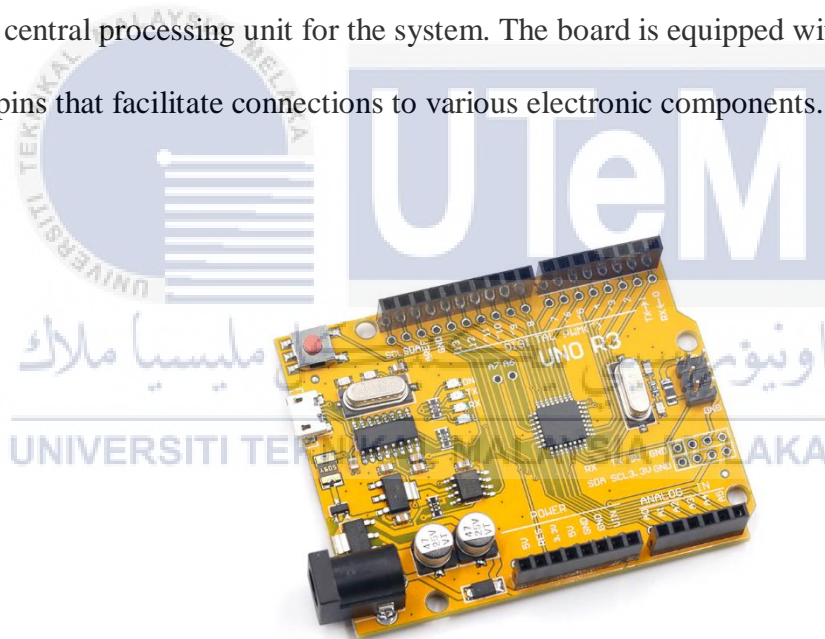
**Figure 3.7 : 5/3-way electropneumatic valve schematic**

In its neutral or rest position, the valve blocks compressed air flow to the working ports, maintaining the actuators in a stable state. Upon application of an electrical signal, the valve activates and redirects airflow to the appropriate working port, enabling the actuator to extend or retract as needed.

The 5/3-way electropneumatic valve finds widespread use in automation systems, pneumatic control circuits, and industrial machinery, where precise control over the direction and flow of compressed air is critical. It facilitates efficient and reliable control of pneumatic actuators, enabling accurate positioning, motion control, and operational flexibility across various pneumatic applications.

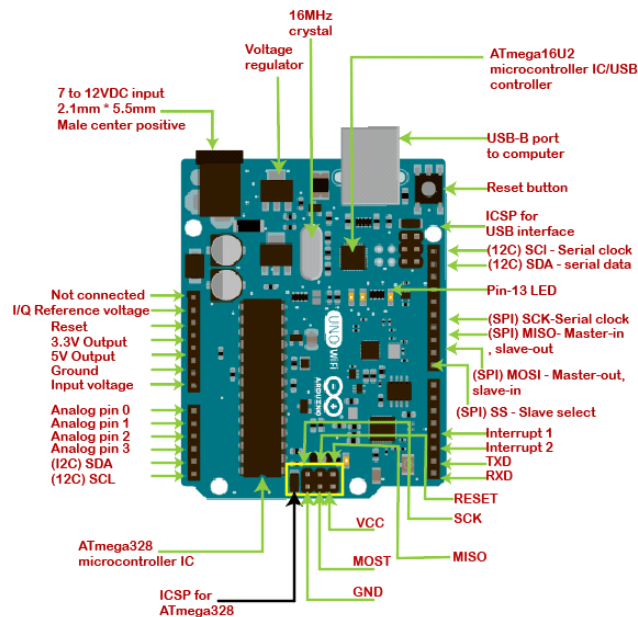
### 3.3.3 Arduino

Arduino, depicted in Figure 3.8, is an open-source electronics platform that offers hardware and software components for developing interactive projects. At its core, Arduino utilizes a microcontroller board, such as the Arduino Uno, which acts as the central processing unit for the system. The board is equipped with input/output (I/O) pins that facilitate connections to various electronic components.



**Figure 3.8 : Arduino UNO R3**

The schematic of an Arduino board typically includes essential components such as power supply connections, including a voltage regulator for stable power and a Micro USB connector for programming and communication purposes. It also incorporates crystal oscillators or resonators to ensure precise timing signals for the microcontroller.



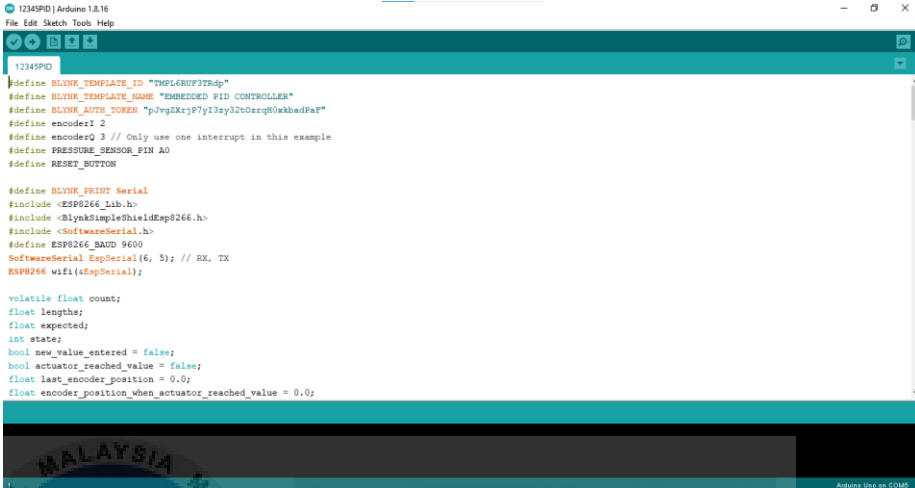
**Figure 3.9 : Input/Output pins for Arduino board**

The I/O pins on the Arduino board are categorized into digital and analog pins, illustrated in Figure 3.9. Digital pins can function as either inputs or outputs, enabling digital communication and control. These pins operate on binary logic, capable of toggling between a high state (typically 5V) and a low state (0V), indicating the presence or absence of a signal. Analog pins, on the other hand, facilitate the measurement of analog signals, such as sensor readings.

To interface with external devices, the Arduino board utilizes various communication protocols, including UART (Universal Asynchronous Receiver-Transmitter), I2C (Inter-Integrated Circuit), and SPI (Serial Peripheral Interface). These protocols facilitate seamless communication with sensors, displays, actuators, and other electronic components.

In addition to its hardware features, Arduino provides an integrated development environment (IDE) for writing and uploading code to the microcontroller

board. The Arduino IDE offers a user-friendly interface for programming in the Arduino language, based on C/C++, as depicted in Figure 3.10. It also includes a library of pre-written functions that simplify the programming process.



```

12345PID
#define BLYNK_TEMPLATE_ID "DMPL6RUF3TRdp"
#define BLYNK_TEMPLATE_NAME "EMBEDDED PID CONTROLLER"
#define BLYNK_AUTH_TOKEN "pDvg2KxyP7y13zy12t0rrq04khdPaP"
#define encoderI 2
#define encoderQ 3 // Only use one interrupt in this example
#define PRESSURE_SENSOR_PIN A0
#define RESET_BUTTON

#define BLYNK_PRINT Serial
#include <ESP8266 Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <SoftwareSerial.h>
#define ESP8266_BAUD 9600
SoftwareSerial EspSerial(6, 5); // RX, TX
ESP8266 wifi(&EspSerial);

volatile float count;
float lengths;
float expected;
int state;
bool new_value_entered = false;
bool actuator_reached_value = false;
float last_encoder_position = 0.0;
float encoder_position_when_actuator_reached_value = 0.0;

```

**Figure 3.10 : Arduino IDE interface**

The Arduino platform integrates both hardware and software components essential for developing interactive projects. By connecting electronic components to the Arduino board and utilizing the Arduino IDE for coding, users can transform their ideas into reality, enabling the creation of diverse applications and devices.

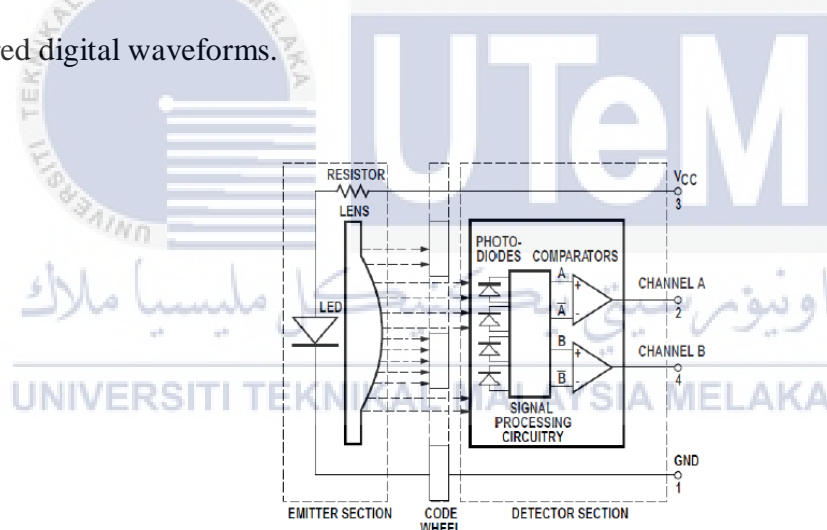
### 3.3.4 Encoder Sensors

The optical encoder plays a crucial role in detecting displacement within precision mechanical systems. Among the available options, the HEDS-9730 shown in Figure 3.11 is popular due to its cost-effectiveness and high performance as an incremental encoder module. This module detects rotary and linear positions using a code wheel or code strip.



**Figure 3.11 : HEDS-9730 optical encoder sensor**

The HEDS-9730 optical encoder module consists of a lensed LED source and a detector IC enclosed in a compact C-shaped plastic package. A single lens positioned directly above the LED emits collimated light. On the opposite side, the detector IC includes multiple photodetectors and signal-processing circuitry to generate the required digital waveforms.

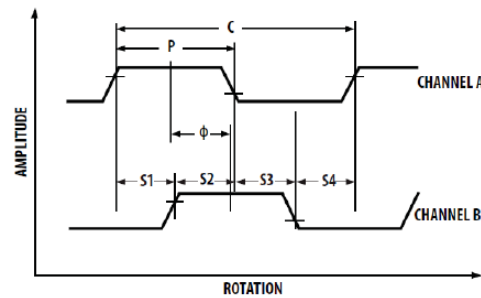


**Figure 3.12 : Block diagram of HEDS-9730 optical encoder sensor**

Figure 3.12 depicts the block diagram illustrating the movement of the code wheel or code strip between the emitter and detector. The pattern of spaces and bars on the code surface interrupts the light beam. Photodiodes within the module detect these interruptions, arranged in a pattern corresponding to the radius and count density of the code wheel or strip. These photodiodes are strategically positioned so that when one pair detects a light period, the adjacent pair experiences a dark period. This spatial



arrangement ensures that the photodiode outputs pass through signal processing circuitry, with two comparators generating the final outputs for channels A and B. The digital output of channel A is in quadrature with channel B (90 degrees out of phase) due to this integrated phasing technique, as shown in Figure 3.13.



**Figure 3.13 : Output waveform of encoder sensor**

An intriguing feature of the HEDS-9730 module is its exceptional tolerance to mounting misalignment. This capability is achieved through a combination of a highly collimated light source and a unique photodetector array. Even when the module is not perfectly aligned with the code wheel or code strip, it can still accurately detect interruptions and generate the corresponding digital output.

Overall, the HEDS-9730 optical encoder module provides reliable and precise displacement sensing in precision mechanical systems. Its affordability, high performance, and tolerance to misalignment make it an indispensable component across various industries, including robotics, automation, CNC machines, and motion control systems.

### 3.3.5 Internet of Things Communication

The Internet of Things (IoT) integrates physical devices and objects with the Internet, enabling them to collect, communicate, and manage data. This connectivity revolutionizes industries, enhances efficiency, and facilitates innovative systems for improved decision-making and quality of life. While security and privacy concerns remain challenges, IoT holds immense potential to transform our lives through connectivity and automation.



Figure 3.14 : ESP8266 Wi-Fi module

This project utilized the ESP8266 Wi-Fi module (Figure 3.14) as a wireless interconnector to connect the board online via Wi-Fi. The module needs to be within range of a nearby Wi-Fi network. Data from the input pin, connected to the sensor, is first sent to the cloud. Subsequently, the data is transmitted to the Wi-Fi module, which then forwards it to the Arduino board for display on the serial monitor.

### 3.3.6 Blynk

Blynk is a comprehensive platform designed specifically for IoT projects, offering a range of features. It provides users with a mobile app and a cloud-based

infrastructure for remote control and monitoring of connected devices. Users can easily create interfaces and widgets to interact with their IoT devices using Blynk apps, as illustrated in Figure 3.15. This enables seamless control and real-time feedback.

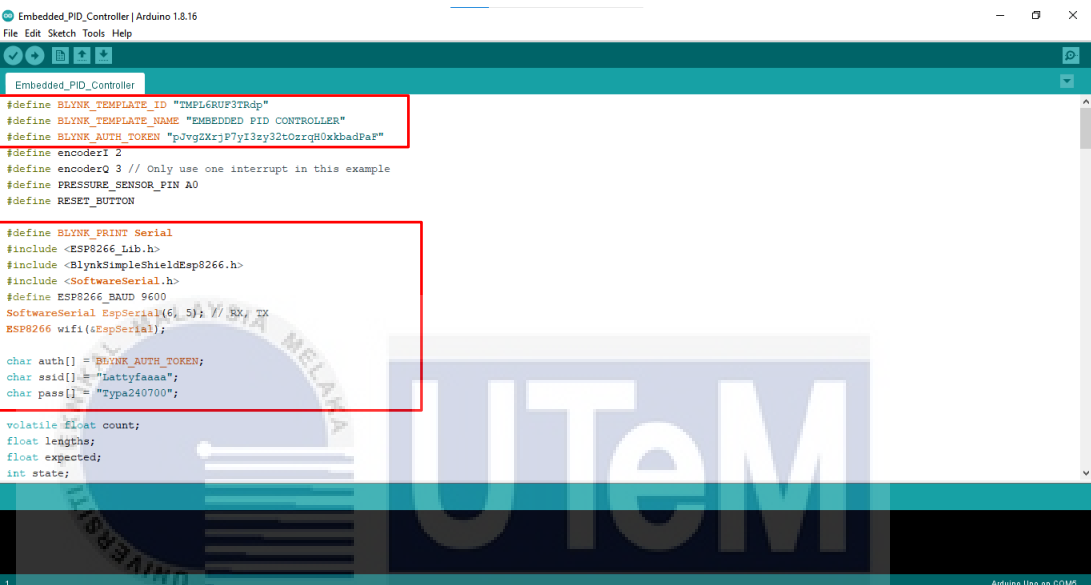


**Figure 3.15 : Blynk interface**

One of the primary advantages of Blynk is its cloud storage and analytics capabilities, which allow users to collect, analyze, and visualize sensor data from their IoT projects. This feature enables comprehensive evaluation of trends, patterns, and correlations, providing valuable insights for improving system performance and making data-driven decisions.

Blynk's cloud-based infrastructure ensures secure data transmission and storage, safeguarding user privacy and data integrity. The platform supports a wide range of hardware platforms and communication protocols, enhancing its adaptability and interoperability with diverse IoT devices.

With Blynk, users can create powerful and user-friendly IoT applications without extensive coding or infrastructure setup. The platform accelerates the prototyping and deployment of IoT solutions by streamlining the development process. Blynk simplifies applications in IoT solutions, home automation, environmental monitoring, and industrial processes.



```

Embedded_PID_Controller | Arduino 1.8.16
File Edit Sketch Tools Help

Embedded_PID_Controller

#define BLYNK_TEMPLATE_ID "EMPL6RUF3TRdp"
#define BLYNK_TEMPLATE_NAME "EMBEDDED PID CONTROLLER"
#define BLYNK_AUTH_TOKEN "pJvqzXrjP7yI3zy32t0zrqH0xkbadFaF"

#define encoderI 2
#define encoderQ 3 // Only use one interrupt in this example
#define PRESSURE_SENSOR_PIN A0
#define RESET_BUTTON

#define BLYNK_PRINT Serial
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <SoftwareSerial.h>
#define ESP8266_BAUD 9600
SoftwareSerial EspSerial(6, 5); // RX, TX
ESP8266 wifi(&EspSerial);

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Lattyfaaaa";
char pass[] = "Ttpa2407000";

volatile float count;
float lengths;
float expected;
int state;

```

**Figure 3.16 : C++ code to generate the communication between the board and Blynk C++**

### 3.4 Controller Design

This project proposes the use of a Proportional-Integral-Derivative (PID) controller as a new control strategy for precisely controlling the positioning of a pneumatic cylinder stroke system. Subsequently, the performance of the proposed PID controller is compared with that of a non-PID controller. This section outlines the procedures employed to design the PID controllers used in this study.

### 3.4.1 Proportional-Integral-Derivative (PID)

The PID controller adjusts the control input based on three terms: proportional, integral, and derivative, corresponding to present, past, and future errors, respectively. The proportional term produces an output proportional to the current error, reducing its magnitude. The integral term accumulates past errors to eliminate steady-state error over time. The derivative term predicts future error changes, providing damping to improve system stability against rapid changes. Together, these terms offer a balanced response, minimizing overshoot, oscillations, and steady-state error, making PID controllers versatile and robust for dynamic systems.

Designing a PID controller involves tuning the proportional gain, integral gain, and derivative gain to achieve desired system performance. Tuning methods can be empirical, analytical (such as Ziegler-Nichols or Cohen-Coon), or automated using optimization algorithms. Each method has specific advantages and limitations, chosen based on application requirements like response speed, stability, and robustness. Once tuned, PID controllers effectively manage a wide range of system dynamics, from simple single-input single-output (SISO) to complex multi-input multi-output (MIMO) systems.

Figure 3.17 illustrates the closed-loop block diagram of a PID controller. Equations (3.1) to (3.3) detail the basic equations for the P, I, and D controllers, with Equation (3.3) describing the PID controller's overall equation.

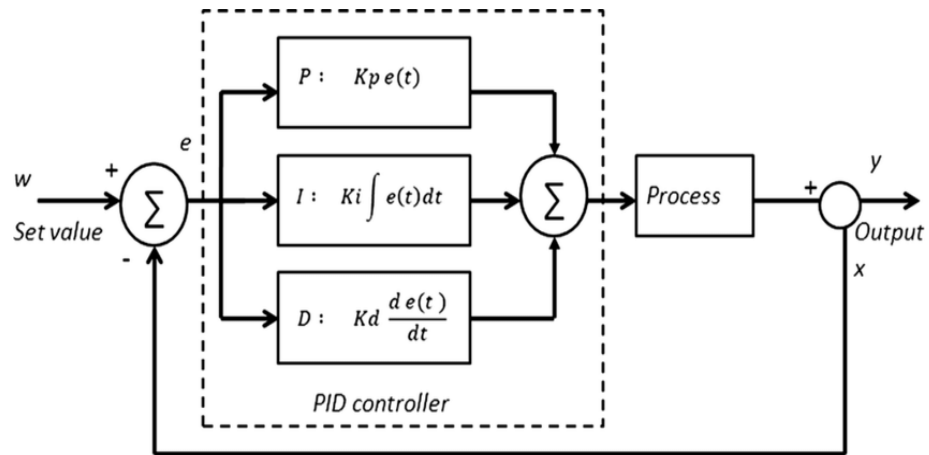


Figure 3.17 : Closed-loop block diagram of PID controller



$$P = K_p e(t) \quad (3.1)$$

$$I = K_i \int_0^t e(\tau) d\tau \quad (3.2)$$

$$D = K_d \frac{de(t)}{dt} \quad (3.3)$$

Therefore,

$$\text{PID} = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (3.4)$$

### 3.5 Transient Response Analysis

Analyzing the transient response of a PID (Proportional-Integral-Derivative) controller for a pneumatic lifting application involves evaluating how effectively the controller adjusts the lift's position in response to input changes, such as desired height adjustments. Key performance metrics like rise time, peak time, overshoot, and settling time are crucial for ensuring the lift reaches the target position smoothly and

quickly, minimizing oscillations. The PID controller's parameters (proportional gain, integral time, and derivative time) are tuned to optimize these metrics, balancing fast response with minimal overshoot. Evaluating the system's behavior typically involves simulations and real-time testing to assess the controller's ability to handle disturbances and varying loads. Fine-tuning PID parameters ensures the pneumatic lift operates efficiently, achieving precise positioning critical for safety and performance in industrial applications.

### 3.5.1 Rise Time

Rise time is a critical parameter in transient response analysis, indicating how quickly the system's output reaches from a lower percentage to a higher percentage of its final steady-state value after a step input. Commonly measured from 10% to 90% of the final value, it signifies how fast the actuator moves from its initial position to near the new set-point when adjusted.

### 3.5.2 Settling Time

Settling time is essential in transient response analysis, representing how long it takes for the system's output to remain within a certain percentage (typically 2% or 5%) of its final steady-state value after a disturbance or input change. After adjusting the set-point, the actuator may oscillate before settling down close to the set-point. Settling time measures the duration until these oscillations dampen and the actuator stabilizes.

### 3.5.3 Overshoot

Overshoot is crucial in transient response analysis, indicating how much the system's output exceeds its final steady-state value initially after a disturbance or input

change. Expressed as a percentage of the final value, overshoot measures how far beyond the set-point the actuator moves upon reaching it initially.

#### 3.5.4 Steady-state Error

Steady-state error assesses a control system's performance, indicating the difference between the desired and actual output once the system has stabilized. It quantifies how closely the system maintains its desired output value in the presence of a constant input or set-point. Equation (3.5) describes the basic calculation for steady-state error:

$$\text{Steady - state error ( cm) = Peak - Reference} \quad (3.5)$$





## CHAPTER 4

### RESULTS AND DISCUSSION

This chapter presents the outcomes of implementing the embedded Proportional-Integral-Derivative (PID) controller for a pneumatic lifting application. It thoroughly examines the design and integration process of the PID controller within the embedded system, followed by its implementation for precise position control of the pneumatic stroke. The integration of Internet of Things (IoT) devices via the Blynk application is also discussed, highlighting the user interface and remote monitoring capabilities. A detailed analysis of the system's transient response performance is provided, showcasing the controller's effectiveness in minimizing rise time, overshoot, settling time, and steady-state error. Comparative results before and after implementing the PID controller are presented, demonstrating significant improvements in the system's responsiveness and accuracy.

## 4.1 Circuit Design

Figure 4.1 illustrates the block diagram of a project involving an embedded PID controller for a pneumatic lifting application.

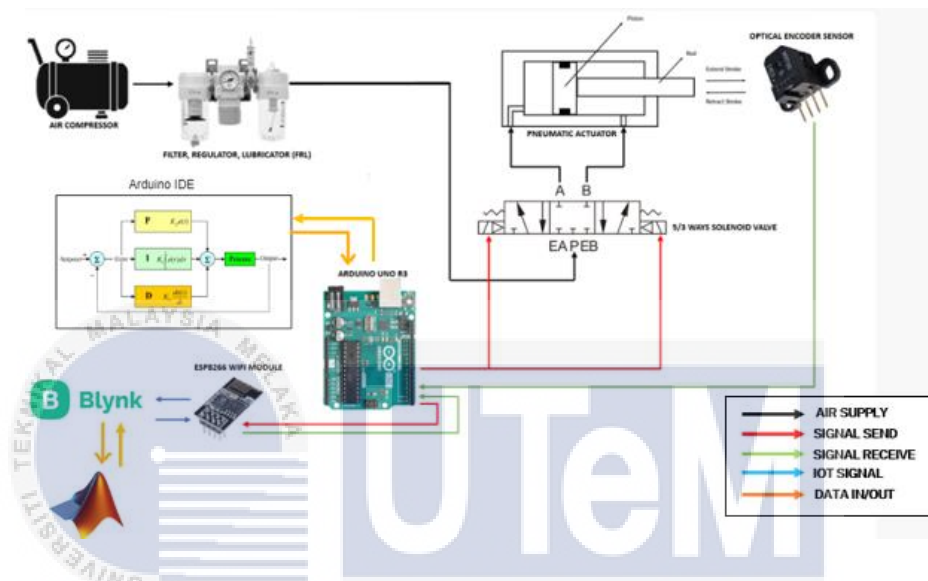


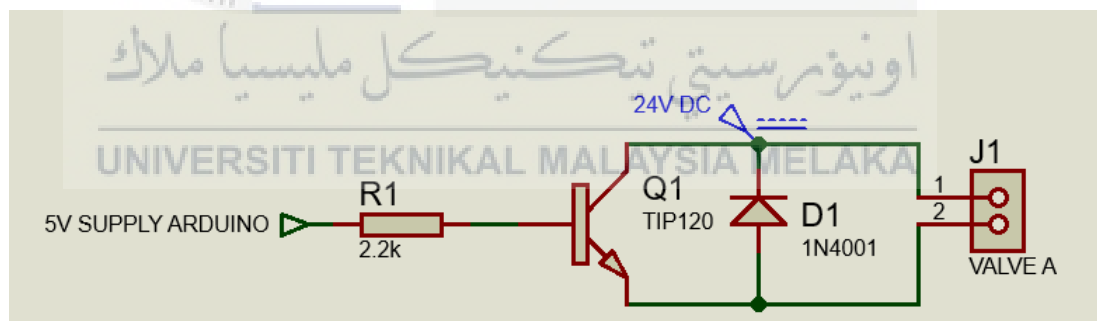
Figure 4.1 : Block diagram of the project

The system comprises an air compressor connected to a Filter, Regulator, and Lubricator (FRL) unit, ensuring a clean and regulated air supply. This air is directed to a pneumatic actuator controlled by a 5/3-way solenoid valve, which regulates the actuator's movement. The position of the actuator is monitored using an optical encoder sensor, providing feedback for precise control. The central control unit is an Arduino Uno running the PID control algorithm, programmed via the Arduino IDE. It communicates with an ESP8266 Wi-Fi module to enable IoT functionality, allowing remote monitoring and control through the Blynk application. The block diagram highlights different signal pathways: air supply, control signals sent and received by the Arduino, and data input/output between the Arduino and the encoder sensor. This

setup ensures efficient and precise control of the pneumatic actuator, demonstrating a practical application of embedded PID control in industrial automation with IoT integration for enhanced functionality.

#### 4.1.1 Valve Switching

In this project, a relay circuit was developed to convert a 5V supply to a 24V supply input, as shown in Figure 4.2, with pin 1 representing positive and pin 2 representing ground. The circuit contains various components, including resistors, diodes, and transistors, each of which performs a distinct function in the circuit design. The resistor limits the current flow across the circuit, which is established by using Ohm's law to find the suitable resistance value. The Ohm's law equation (4.1) can be used to calculate the appropriate resistor value for the circuit to ensure that the current does not overflow when the voltage input is applied.



**Figure 4.2 : Valve switching circuit**

$$V = IR \quad (4.1)$$

The circuit's transistor type was chosen based on the component's datasheet. The datasheet provides specifications for voltage rating, current handling, power

dissipation, base current, and gain based on the circuit's input and output voltages. This enabled the selection of the appropriate transistor type for the circuit's operation, as indicated by the calculated value. The computation below assumes  $V_{BE} = 1.2$ ,  $I_C = 1A(1000 mA)$ ,  $\beta = 1000$ , and  $V_{CE sat} = 2V$ .

Base current, ( $I_B$ ):

$$I_B = \frac{I_C}{\beta} \quad (4.1)$$

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

$$I_B = 1 mA$$

Base resistor, ( $R_B$ ):

$$R_B = \frac{(V_{in} - V_{BE})}{I_B} \quad (4.3)$$

$$R_B = \frac{(5V - 1.2V)}{1 mA}$$

$$R_B = 3.8 k\Omega$$

To be equivalent to the  $3.8k\Omega$  value of  $R_B$  ( $R1$ ), a standard close resistor of  $2.2k\Omega$  was selected. The TIP120 is a suitable transistor for controlling electro-pneumatic valves by operating the 5V signal and allowing current flow in the circuit. It connects to the load of the 24V supply. The TIP120 transistor, a Darlington transistor, is frequently used in high-power switching applications. Table 4.1 shows the TIP120 specifications.

**Table 4.1 TIP120 specifications**

Parameter	Required Value	Calculation Value
Collector-Base Voltage ( $V_{CE}$ )	60V	24V
Base-Emitter Voltage ( $V_{BE}$ )	1.5V	1.2V
Base Current ( $I_B$ )	120mA	1mA
Beta ( $\beta$ )	1000	1000
Collector Current ( $I_C$ )	5A	1A
Collector-Emitter Saturation Voltage ( $V_{CE sat}$ )	2V	2V
Base resistor ( $R_B$ )	-	2.2k $\Omega$

The TIP120 is a Darlington transistor commonly used for high-power switching applications. The parameters concerning each other and their significance for TIP120. The Collector-Emitter Voltage ( $V_{CE}$ ) specifies the maximum voltage that can be applied between the collector and emitter terminals. In this case, the requirement is 60V, but the calculated value is 24V. The circuit should ensure the ( $V_{CE}$ ) remains below 60V to avoid exceeding the transistor's maximum voltage rating.

The Base-Emitter Voltage ( $V_{BE}$ ) is the voltage drop between the base and emitter terminals. The requirement is 1.5V, while the calculated value is 1.2V. This voltage drop is crucial to ensure proper forward biasing of the base-emitter junction for the transistor to operate effectively.

The Base Current ( $I_B$ ) flows into the base terminal. The requirement is 120mA, but the calculated value is 1mA. The base current controls the transistor's amplification capabilities and is necessary to drive the transistor into saturation. Designing the circuit to provide sufficient base current ensures proper operation.

Next, a flyback diode method was employed by connecting a diode across the load to mitigate the effects of flyback, which refers to the sudden voltage spike

observed across an inductive load when its supply current is rapidly reduced or interrupted. This method suits circuits involving switching power supplies and inductive loads.

The selected diode for this purpose is the 1N4001 diode, chosen because it meets the circuit's specifications and requirements based on the data sheet, particularly the 24V voltage level. The 1N4001 diode has a maximum peak reverse voltage (VRRM) rating of 50V, sufficient for the 24V application.

Connecting the diode in a reverse-biased configuration across the load provides a low-resistance path for the inductive energy when the current flow is abruptly halted. This allows the energy to dissipate safely and prevents the voltage spike from damaging the circuit components.

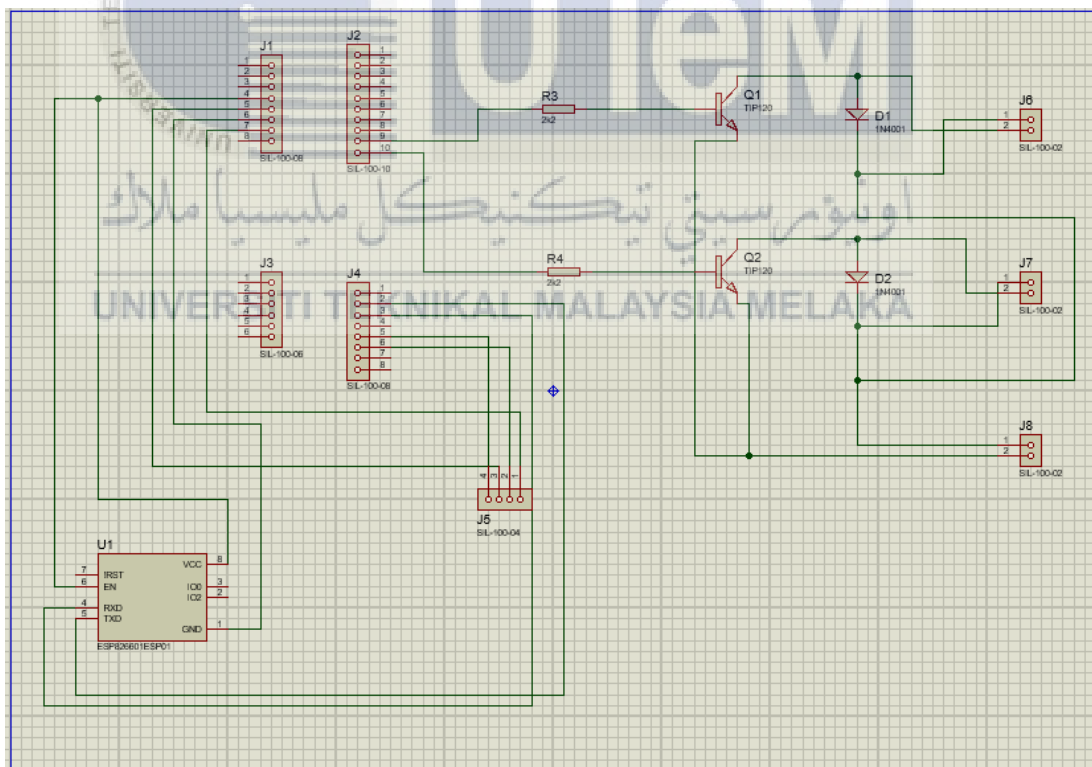
The flyback diode, such as the 1N4001, effectively suppresses the flyback voltage, safeguarding other components and ensuring the reliable operation of the circuit. It is essential in power electronic applications where inductive loads, such as solenoids, motors, or transformers, are present. By implementing the flyback diode method and selecting an appropriate diode like the 1N4001, the circuit can effectively manage the flyback phenomenon, enhancing the overall performance and longevity of the system.

The circuit was thoroughly analyzed to ensure that the voltage input and output were suitable to activate the electro-pneumatic valve powered by a 24V DC supply in channel 2. To do this, the circuit meticulously harnesses a 5V supply in channel one from an Arduino signal and efficiently converts it to the necessary 24V power source.

This rigorous monitoring guarantees that the voltage levels fit the electro-pneumatic valve's stringent specifications, resulting in maximum functionality and performance.

#### 4.1.2 Complete Circuit Design and Fabrication

The circuit has been designed to fulfill the required specifications and achieve the project objectives as shown in Figure 4.3. The circuit comprises multiple components, including the switching circuit, Wi-Fi module circuit, and encoder sensor circuit. In the encoder sensor circuit, the HEDS-9730 encoder sensor is connected to the appropriate pins of the Arduino microcontroller. The specific pin connections for each circuit to be controlled by the code in Appendix A, which contains the complete program code, are listed in Table 4.2.



**Figure 4.3 : Full schematic circuit**

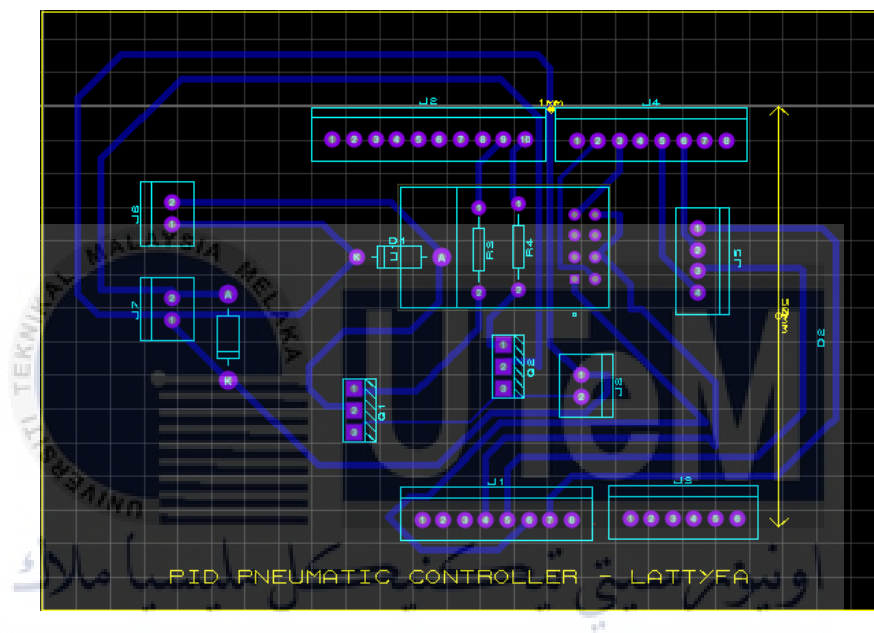
**Table 4.2 Arduino pins configuration**

Component/Sensors		Arduino Pin
<b>Encoder HEDS-9730</b> <b>Optical Sensor</b>	$(V_{SUPPLY})$	5V
	Ground	Ground
	Channel 1	D2
	Channel 2	D3
<b>Valve (Switching Circuit)</b>	A	D8
	B	D9
<b>Wi-Fi Module ESP8266</b>	$(V_{SUPPLY})$	3.3V
	Enable	3.3V
	Ground	Ground
	RX	D5
	TX	D6

The pin connections in the circuit have been accurately established, and a rapid system test was conducted to ensure its operational capability. The digital pin from the Arduino emits a 5V signal, while the analog pin can receive a 5V signal. The Arduino supplies the necessary 5V for the transistor circuit. In the case of the encoder sensor, a 5V signal is supplied by the Arduino, and the digital pins 2 and 3 are utilized to receive the binary readings from the sensor's stripe, enabling the detection of actuator distance. The switching circuit is an intermediary between the valve and the Arduino to connect the valve since the valve requires a 24V power supply. Per the code specified in Appendix A, the digital pins 8 and 9 send signals to turn the valve on or off based on the system's operation. Finally, a 3.3V supply is necessary for the Wi-Fi module circuit, which is obtained from the Arduino. The 3.3V supply is connected to both the Wi-Fi module's supply and enable pins, while the RX and TX pins are connected to pins 5 and 6, respectively, for communication with the module.



After completing all the necessary tests on the circuit, the next step was to fabricate the PCB (Printed Circuit Board). The PCB was designed using Proteus software, allowing it to be seamlessly integrated with the Arduino and optimized for its combined operation with the actuator. Figure 4.4 shows the designed PCB circuit, with the blue routes indicating the bottom copper layer and the red routes representing the top copper layer.

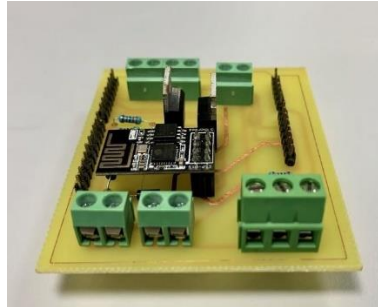


**Figure 4.4 : PCB schematic design in Proteus software**

The fabrication of the PCB enables a more streamlined and professional implementation of the circuit design. Embedding the circuit on a PCB ensures better reliability, improved connectivity, and reduced chances of interference or accidental damage. This step is crucial to ensure the successful integration of the circuit with the actuator, resulting in a robust and efficient system.

It is worth noting that using Proteus software for PCB design allows for precise layout planning, accurate component placement, and efficient routing. This helps

minimize signal integrity issues, optimize electrical performance, and ensure the overall functionality and durability of the circuit.



**Figure 4.5 : Completed PCB circuit**

Once the fabrication process was completed, as depicted in Figure 4.5, the PCB was seamlessly integrated with the Arduino. This integration involved physically embedding the PCB onto the Arduino board, allowing for a compact and unified system. The resulting combination of the PCB and Arduino is illustrated in Figure 4.6.



**Figure 4.6 : Integrated PCB circuit with Arduino**

Embedding the PCB with Arduino brings several advantages to the system. Firstly, it ensures a more secure and stable connection between the components, minimizing the risk of loose connections or accidental disconnections during

operation. Additionally, it enhances the overall aesthetics and organization of the system, reducing clutter and facilitating easier maintenance and troubleshooting.

The integration process involved aligning the appropriate pins and connectors on the PCB with the corresponding headers on the Arduino board. Careful attention was given to ensure proper alignment and secure attachment, guaranteeing reliable electrical connections between the PCB and the Arduino.

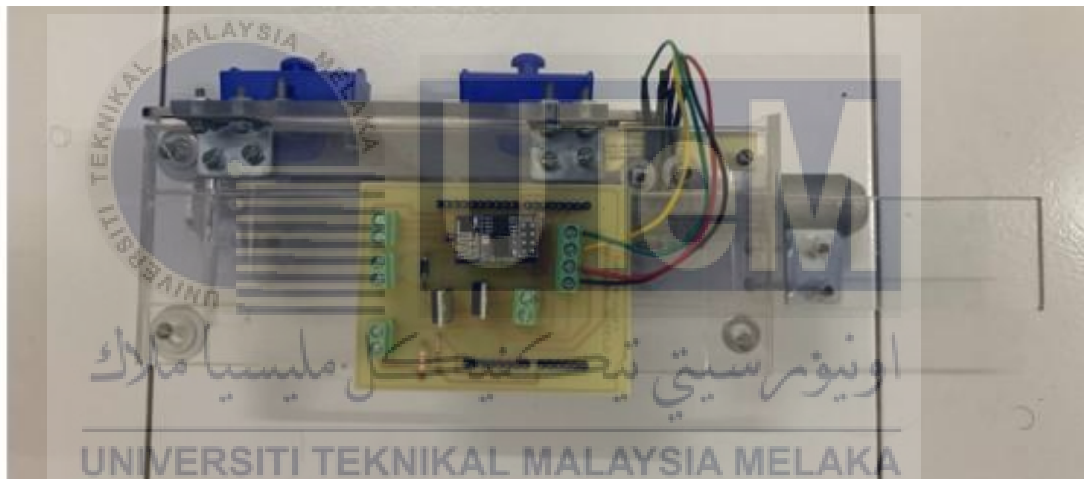
Finally, after completing the entire circuit design, fabrication, and integrating the PCB with the Arduino, the controller was attached to the actuator for comprehensive testing. This crucial step involves connecting the output signals from the controller to the valve and the air supply connected to the actuator, allowing the system to be evaluated in real-world conditions.

#### **4.2 Prototype Design**

The prototype incorporated the previously developed actuator and control system. Figures 4.7 and 4.8 show the front and back views of the prototype design, including the integration of the actuator's piston and binary strip. The encoder sensor precisely measures the piston's distance as the strip moves through it. The controller was attached to the actuator using 5mm Perspex material, which is strong enough to withstand the forces of pneumatic attraction and retraction.

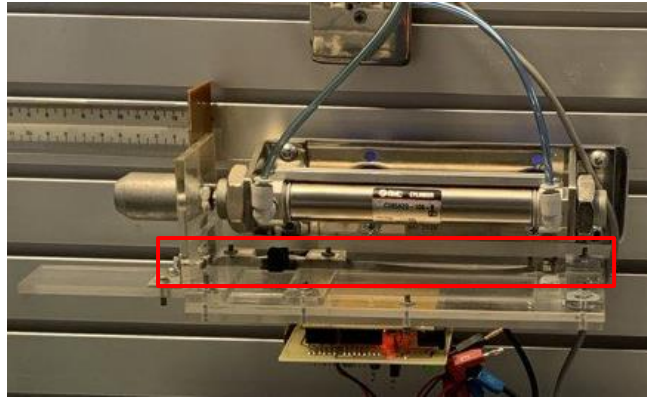


**Figure 4.7 : Front view of full prototype design**



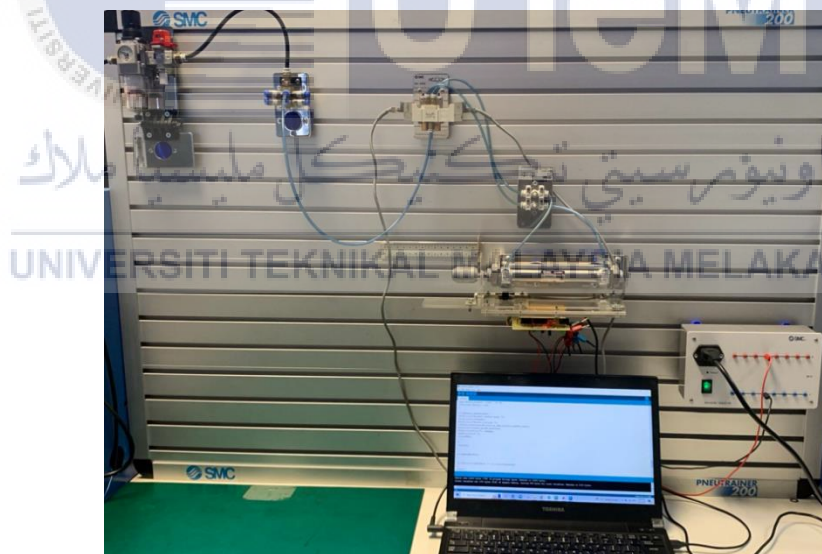
**Figure 4.8 : Back view of full prototype design**

An outer shell made of Perspex was also created for practicality and protection. This shell houses the encoder sensor, enabling it to read the strip code as it moves along with the actuator. In Figure 4.9, the encoder strip code can be seen moving with the slider that grips the actuator's piston. As a result, the precise displacement of the piston can be accurately determined.



**Figure 4.9 : The movement of the actuator with the strip code**

The binary strip, encoder sensor, and Perspex attachments provide precise measurement of the actuator's movements. This enables effective control and monitoring of the actuator position, enhancing system functionality.



**Figure 4.10 : Arrangement of the pneumatic system**

Figure 4.10 illustrates the pneumatic piping arrangement for the air pressure supply in the system. Compressed air flows from the supply to the proportional valve, then through the air control valve to reach the pneumatic actuator. This arrangement

controls airflow into the actuator, preventing excessive piston speed from interfering with the encoder sensor's accurate position measurement.

The proportional valve regulates air pressure and flow rate to meet the pneumatic actuator's requirements. The air control valve improves control by regulating airflow and keeping it within specific limitations. Proper airflow regulation ensures the actuator's piston moves smoothly and consistently, allowing the encoder sensor to accurately record position.

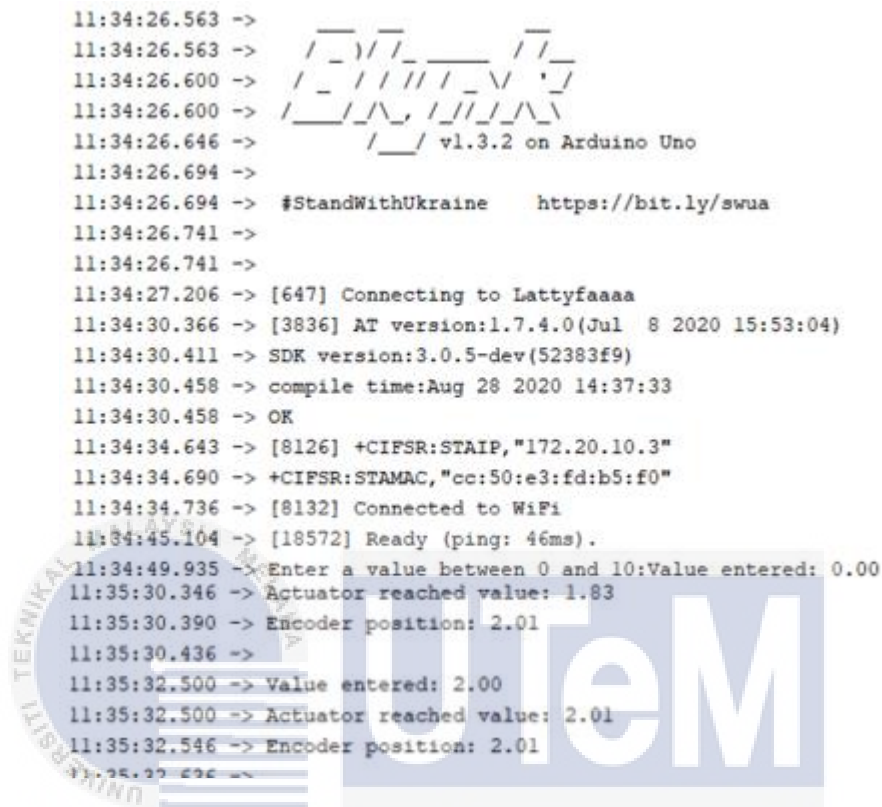
This pneumatic piping layout allows engineers to precisely control the pneumatic system's operation. Controlled airflow ensures reliable and accurate position readings from the encoder sensor by preventing unexpected or irregular piston movements, ultimately improving overall performance and functionality.

#### 4.3 Pneumatic Stroke Position Reading

The control system controls the activation and deactivation of the electro-pneumatic valve, ensuring precise pneumatic stroke position. The user can extend or retract the pneumatic by entering the desired setpoint. The control system continuously checks the pneumatic stroke's position through feedback from encoder sensors or other position-sensing methods.

When the pneumatic stroke reaches the desired position, the control system commands the valve to retract, stopping airflow to the pneumatic stroke. This keeps the pneumatic stroke in its intended position and prevents unwanted movements. The control system adjusts the valve depending on feedback to maintain the desired

position. Figure 4.11 illustrates how the serial monitor displays the distance entered and reached when the system is running.



```

11:34:26.563 ->
11:34:26.563 ->
11:34:26.600 ->
11:34:26.600 ->
11:34:26.646 ->
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 4.11 : Serial monitor display

The Serial Monitor plays a critical role in this system by enabling engineers to monitor sensor readings and ensure the system's proper functioning. The encoder sensor's line detection scans the encoder strip with a code of 180Lpi (180 lines per inch). To convert these line measurements into centimeters, utilize the "count" variable, which accurately represents the number of lines reads shown in equation 4.4.

$$1 \text{ inch} = 180 \text{ lines}$$

$$1 \text{ cm} = \frac{180}{2.54} = 70.866 \text{ lines}$$

Therefore, the formula used to determine the position of the pneumatic stroke.

$$lengths = \frac{count}{2(70.866)} \quad 4.2)$$

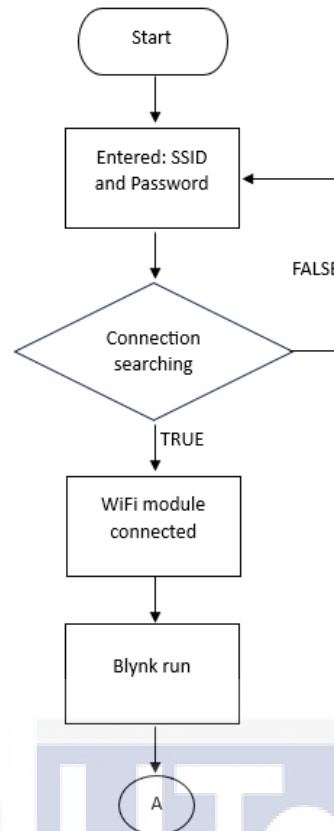
$$lengths = \frac{count}{141.73}$$

Analyzing the Serial Monitor output helps evaluate the encoder sensor's performance and line detection accuracy. This information is crucial for calibrating the equipment and achieving accurate measurements. Using a high-resolution encoder strip, like the 180Lpi version provided, allows for fine-grained position determination, improving total system precision.

The "count" variable is critical in converting from lines to centimeters. By combining the "count" value with the known characteristics of the encoder strip, Engineers can accurately calculate the displacement of the system's components. This knowledge is essential for maintaining precise positional control and ensuring the optimal operation of pneumatic actuators driven by the system.

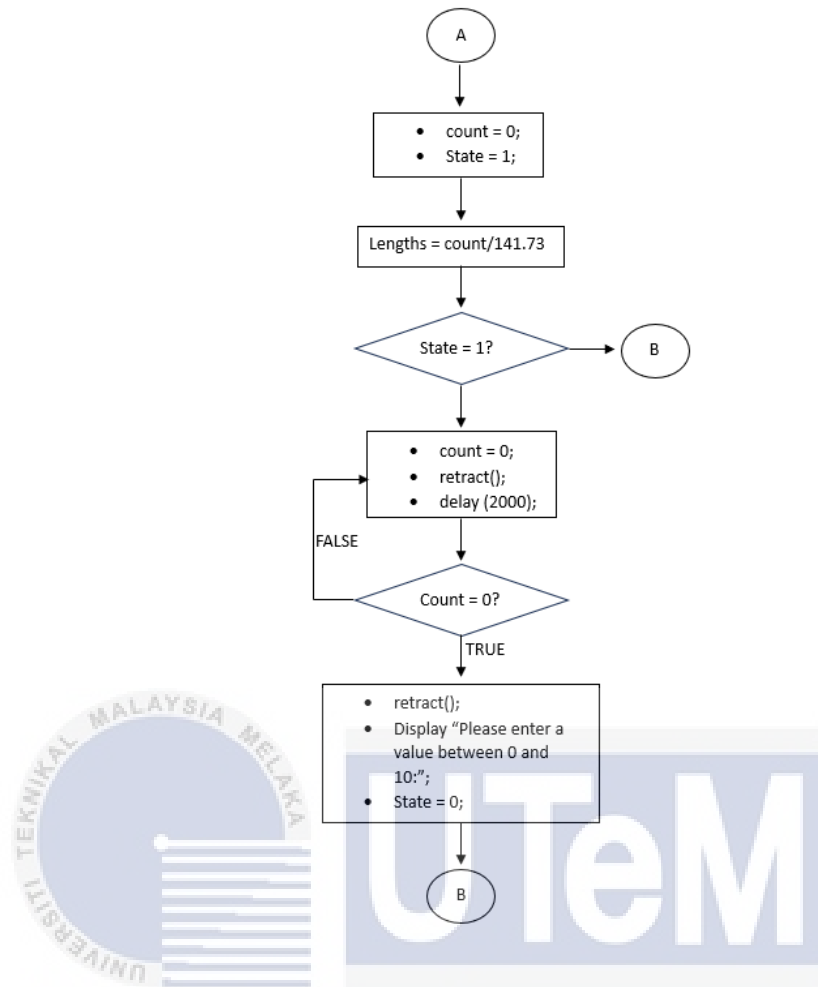
The process will repeat when the user sets a new desired setpoint or it could return to the 0cm position if user entered 0cm or press the retract button. This allows for smooth and controlled movements of the actuator to achieve the desired positions precisely. The flow code for the encoder sensor is shown below.





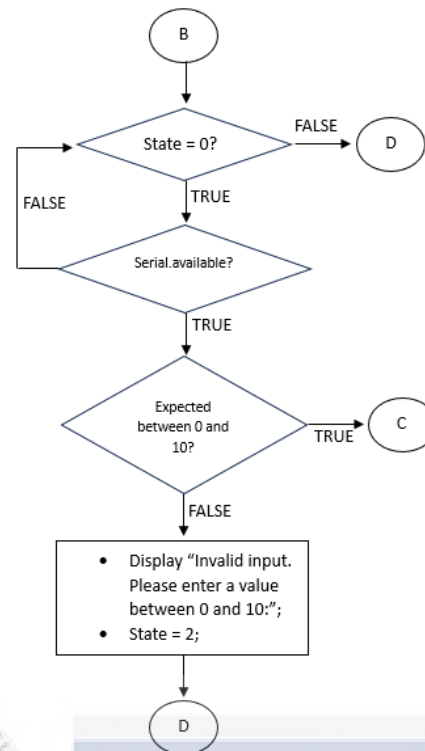
**Figure 4.12 : Flow code for starting to initiate the program with the IoT system**

The initial configuration of the coding system involves setting all parameters to their default values. The encoder sensor records the number of lines reads as "count," while the "state" variable indicates the current state of the pneumatic system.



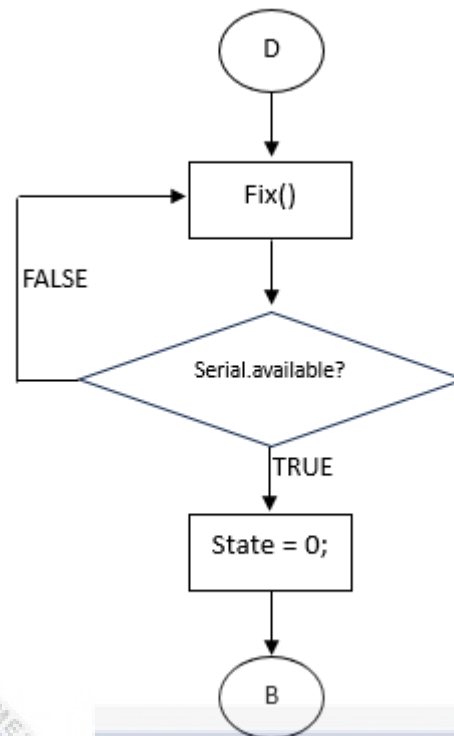
**Figure 4.13 : Flow code for initial state**

Upon system startup, all previous values and conditions are reset. The initial values for "count" and "state" are set to 0 and 1, respectively. Utilizing a pre-established formula, the displacement of the piston can be accurately determined. Subsequently, the system verifies whether the state equals 1 to confirm proper initialization. If the state is not equal to 1, the system proceeds to step B (depicted in Figure 4.13). Alternatively, suppose the state is equal to 1. In that case, the system initiates the retraction of the pneumatic actuator until the piston reaches its fully retracted position, commonly referred to as the initial position. Following this, the "state" variable is updated to 0, and the Serial Monitor prompts the user to input a value within the range of 0 to 10.



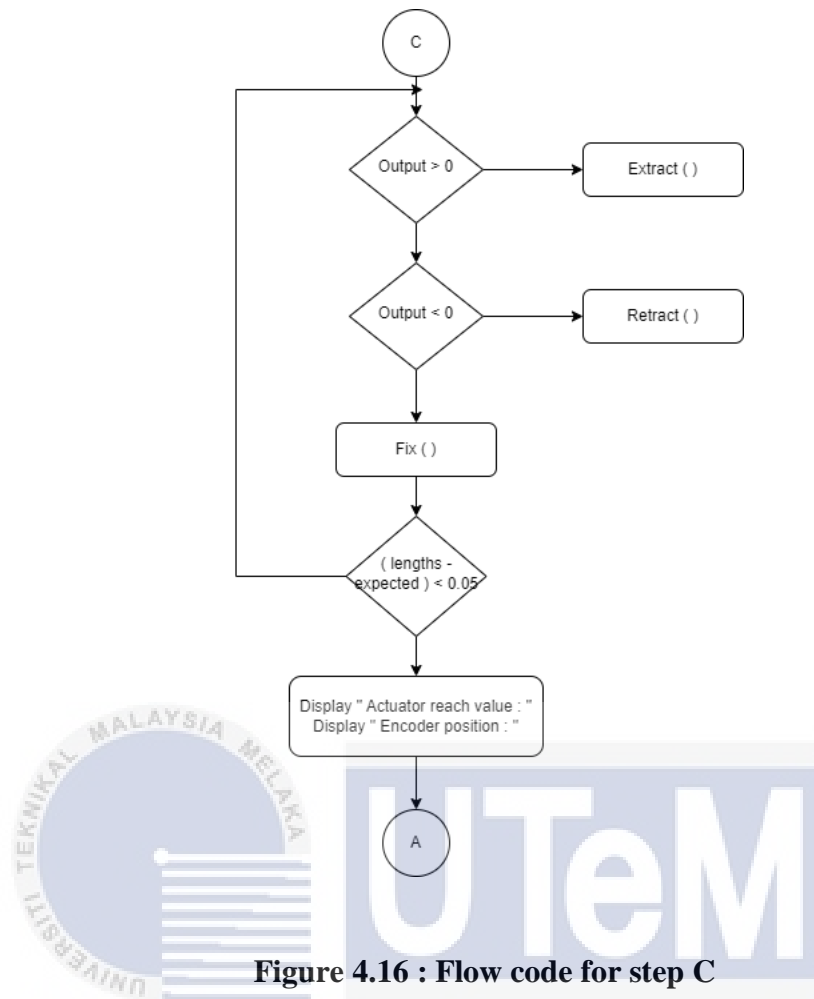
**Figure 4.14 : Flow code for step B**

In step B, the Arduino Uno microcontroller verifies if the “state” is set to 0. The system advances to step D if the “state” is not 0. Conversely, if the “state” is 0, the system continuously monitors the input from the Serial Monitor. As shown in Figure 4.14, during this monitoring process, if no input is detected, the system persists in a loop until an input is received. Once an input is detected, it is scrutinized to ensure its conformity within the range of 0 to 10. The system proceeds to step C if the input falls within this range. However, if the input is outside the valid range, the Serial Monitor displays the message “Invalid input. Please enter a value between 0 and 10”, and the system proceeds to step D.



**Figure 4.15 : Flow code for step D**

Figure 4.15 depicts the execution of the program in step D. Upon entering this phase, the position of piston is fixed until a new input from the Serial Monitor is received. The “state” variable is set to 0, indicating the end of this stage, and the system returns to step B to display the input value.



**Figure 4.16 : Flow code for step C**

In Figure 4.16, when the input value is correct in step B, the system advances to step C. If output greater than 0, the piston initiates extraction. But if the output is less than 0, the piston initiates retraction and if output is 0, the piston will stop the movement. Next, after determining the direction of the movement, the code checks if the current length is close enough to the expected length. If the absolute difference between lengths and expected is less than 0.05, the condition is true indicating that the actuator has reached the desired position. The system then prints the related information and returns to step A, awaiting a new input.

#### 4.4 Fine-tuning PID Controller

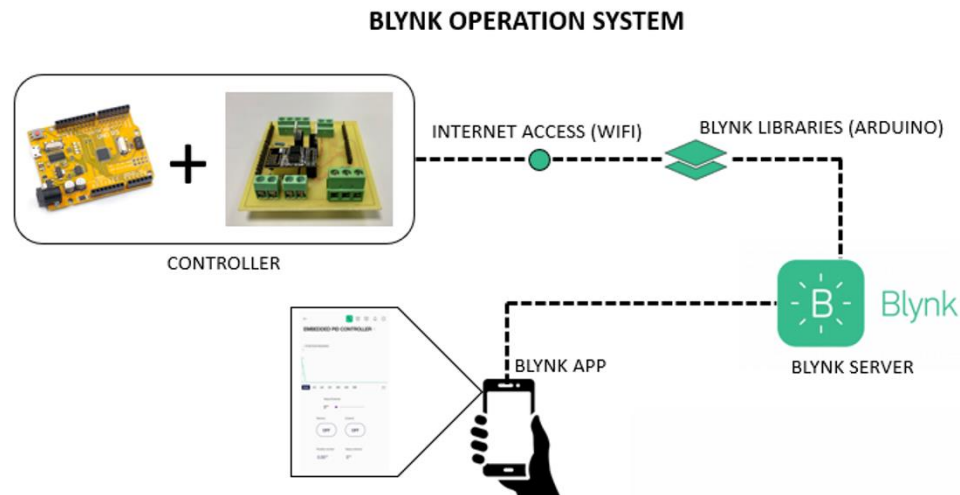
```
// PID variables
float setpoint, input, output;
float Kp = 9.0, Ki = 0.2, Kd = 0.01;
float integral = 0.0;
float previous_error = 0.0;
unsigned long last_time;
unsigned long current_time;
```

**Figure 4.17 : Three parameters of PID**

Figure 4.17 shows a code for a Proportional-Integral-Derivative (PID) controller setup in an Arduino IDE. It defines several variables used in the PID control process: setpoint, input, and output (all of type float), and the PID constants  $K_P$ ,  $K_I$ ,  $K_D$  with values 9.0, 0.2, and 0.01, respectively. Additionally, it declares integral and previous error (both initialized to 0.0) for storing the cumulative integral of the error and the previous error value. The last time and current time variables (both of type unsigned long) are used to track the time for calculating the derivative component and integrating the error over time.

#### 4.5 Development of IoT System

In order to enhance the functionality of the IoT system, as shown in Figure 4.18, a modification was made to the existing code by integrating a Blynk communication script. This modification enables the system to send data to the Blynk application for visualization and monitoring. The specific code implementation can be found in Appendix A.



**Figure 4.18 : Blynk operation system**

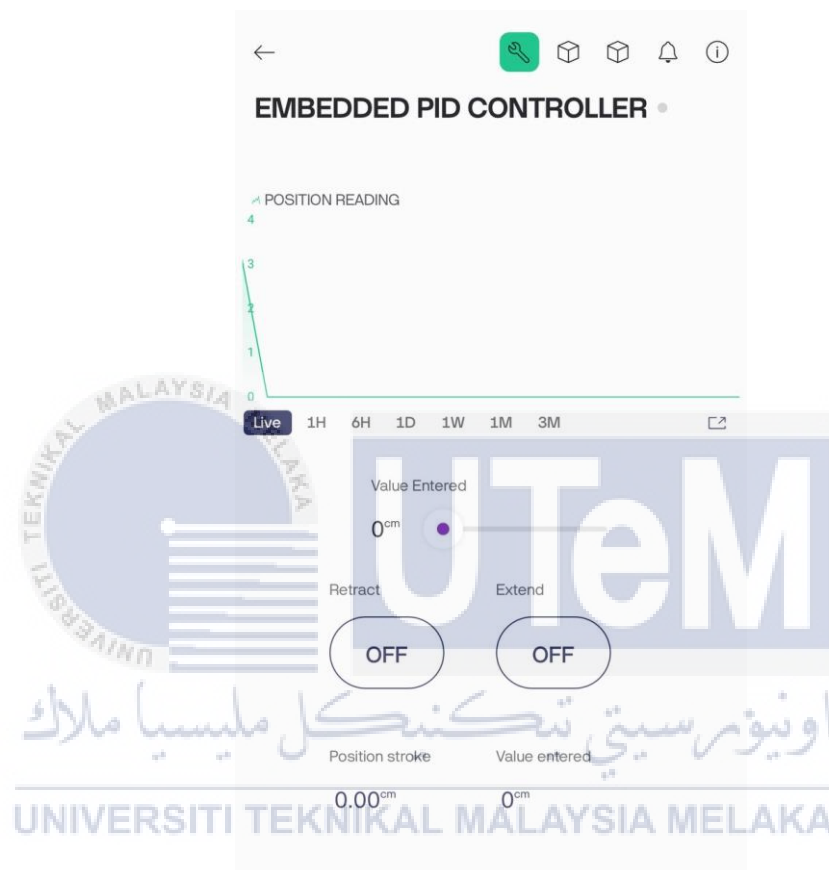
After connecting to the Wi-Fi module, the device starts communicating with the Blynk cloud. The serial monitor displays the messages "Blynk connected" and "Wi-Fi connected," shown in Figure 4.19. These signals indicate that the microcontroller successfully interfaces with the Wi-Fi module and sends data to the Blynk cloud for processing and display on the application.

```

11:34:26.563 -> [9]
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
  
```

**Figure 4.19 : Serial print showed Blynk connected**

By incorporating Blynk into the IoT system, users can remotely monitor and control various parameters and data points. The Blynk application provides an intuitive interface for visualizing real-time data, enabling efficient decision-making and system management, as shown in Figure 4.20.



**Figure 4.20 : Blynk interface set for the system**

In summary, adding the Blynk communication script to the code enhances the IoT system's capabilities by enabling data transmission to the Blynk cloud and display on the Blynk application. This integration empowers users to monitor and control the system remotely, facilitating efficient data analysis and decision-making processes.



#### 4.6 Analysis of Transient Response

Table 4.3 illustrates the real-time analysis of a non-embedded PID controller for a pneumatic lifting application. It shows the positions entered the system and the actual positions reached, along with the corresponding percentage errors. The data indicates that when a position of 2.0 cm is entered, the system reaches 2.191 cm, resulting in a significant percentage error of 9.55%. The errors decrease as the target positions increase, with the smallest error being 1.95% for a 10.0 cm input. Other notable errors include 5.75% for a 4.0 cm input and 2.238% for an 8.0 cm input, reflecting varying levels of accuracy across different setpoints.

**Table 4.3 Real-time analysis of non-embedded PID controller**

Position Entered (cm)	Real Time Analysis	
	Position Reach (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

Next, Table 4.4 displayed the real-time analysis of an embedded PID controller for a pneumatic lifting application. It presents data on the system's performance by comparing the positions entered by the system (in centimeters) with the actual positions reached. The table also includes the percentage error, which is calculated to assess the accuracy of the PID controller. For instance, when a position of 2.0 cm is entered, the system reaches 2.031 cm, resulting in a percentage error of 1.55%.

Similarly, other entries show varying degrees of error, with the smallest error being 0.35% for an 8.0 cm input and the largest error being 1.95% for a 4.0 cm input.

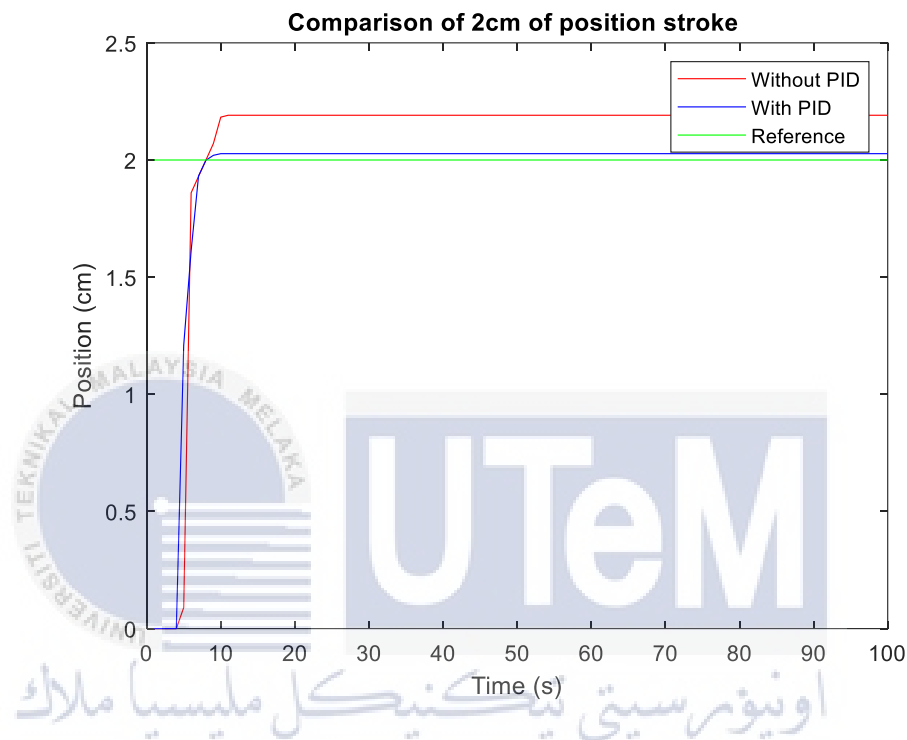
**Table 4.4 Real-time analysis of embedded PID controller**

Position Entered (cm)	Real Time Analysis	
	Position Reach (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

Comparing the results of the embedded and non-embedded PID controllers reveals distinct differences in performance. The embedded PID controller consistently demonstrates lower percentage errors across all setpoints, with the highest error being 1.95% for a 4cm input. In contrast, the non-embedded PID controller shows significantly higher errors, particularly at lower setpoints, with the highest error reaching 9.55% for a 2cm input. This indicates that the embedded PID controller provides more precise and reliable control, maintaining better accuracy in reaching the desired positions. The data underscores the superiority of the embedded PID controller in achieving stable and accurate positioning in pneumatic lifting applications, making it a more suitable choice for scenarios requiring high precision and consistency.

Figure 4.21 presents a comparative analysis of the transient response for a 2cm position stroke in a pneumatic lifting system, highlighting the performance of systems embedded PID controller and non-embedded PID controller. The graph illustrates the

position over time, with the blue line representing the system of non-embedded PID controller, the red line indicating the system of embedded PID controller, and the green line as the reference position. The peak positions reached are 2.191cm for the non-PID controlled system and 2.027cm for the PID-controlled system.



**Figure 4.21 : Graph of comparison 2cm of position stroke**

The embedded PID controller demonstrates superior performance compared to the non-embedded PID controller in all key metrics as shown in Table 4.5. The rise time for the embedded PID is slightly faster at 2.4949 seconds compared to 2.5256 seconds for the non-embedded PID. The settling time is significantly shorter for the embedded PID at 7.8128 seconds, compared to 9.6830 seconds for the non-embedded PID, indicating quicker stabilization. Additionally, the embedded PID system exhibits no overshoot (0%), whereas the non-embedded PID also shows no overshoot. The steady-

state error is notably lower for the embedded PID at 0.0270cm, compared to 0.1910cm for the non-embedded PID.

**Table 4.5 Transient response for distance 2cm**

Distance (cm) (Reference)	Transient Response	Control Strategy	
		Embedded PID	Non-Embedded PID
2.0	Rise time, $t_r$ (s)	2.4949	2.5256
	Settling time, ( $t_s$ (s)	7.8128	9.6830
	Overshoot, $OS$ (%)	0	0
	Steady-state error, $e_{ss}$ (cm)	0.0270	0.1910

The graph in Figure 4.22 illustrates the position over time, the peak positions reached are 4.230cm for the non-PID controlled system and 4.078cm for the PID-controlled system. In Table 4.6, the rise time for the embedded PID is slightly faster at 2.9864 seconds compared to 20.0613 seconds for the non-embedded PID. The settling time is significantly shorter for the embedded PID at 9.2306 seconds, compared to 25.6979 seconds for the non-embedded PID, indicating quicker stabilization. Additionally, the embedded PID system exhibits no overshoot (0%), whereas the non-embedded PID also shows no overshoot. The steady-state error is notably lower for the embedded PID at 0.0780cm, compared to 0.2300cm for the non-embedded PID.

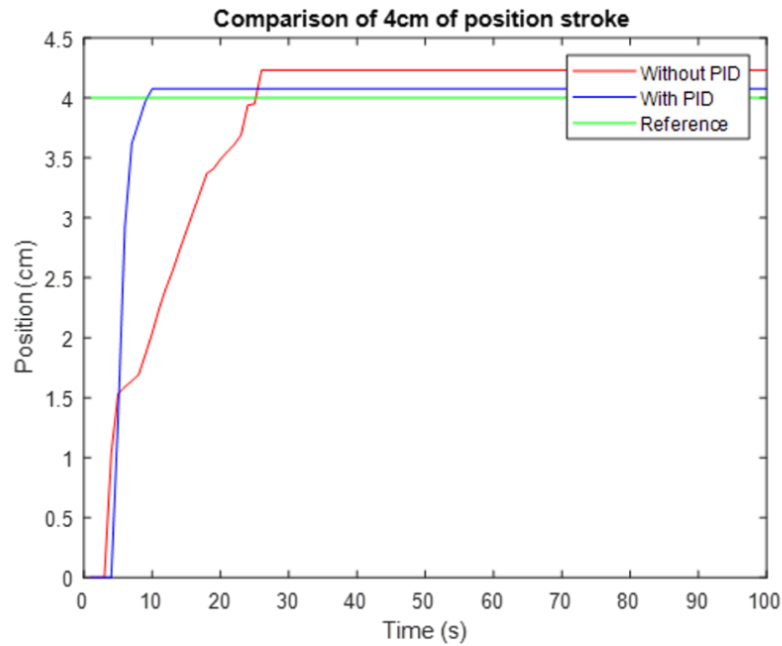


Figure 4.22 : Graph of comparison for 4cm of position stroke

Table 4.6 Transient response for distance 4cm

Distance (cm) (Reference)	Transient Response	Control Strategy	
		Embedded PID	Non-Embedded PID
4.0	Rise time, $t_r$ (s)	2.9864	20.0613
	Settling time, ( $t_s$ (s)	9.2306	25.6979
	Overshoot, $OS$ (%)	0	0
	Steady-state error, $e_{SS}$ (cm)	0.0780	0.2300

The graph shows the peak positions reached are 6.184cm for the non-PID controlled system and 6.089cm for the PID-controlled system as in Figure 4.23. For Table 4.7, it states that the embedded PID controller achieves a rise time ( $t_r$ ) of 1.7366 seconds and a settling time ( $t_s$ ) of 7.9741 seconds, with no overshoot and a steady-state error ( $e_{SS}$ ) of 0.0890cm. In contrast, the non-embedded PID controller

has a rise time of 6.4059 seconds, a settling time of 14.4715 seconds, with no overshoot but a higher steady-state error of 0.1840cm.

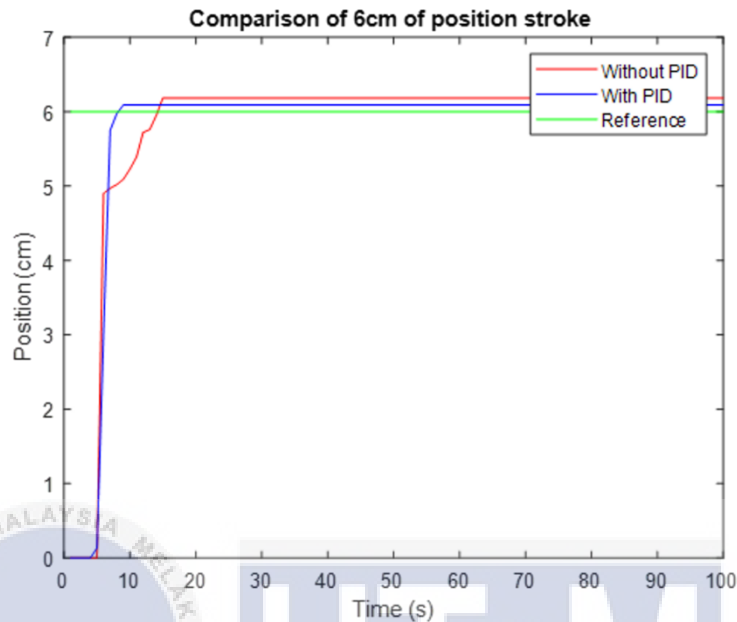


Figure 4.23 : Graph of comparison for 6cm of position stroke

Table 4.7 Transient response for distance 6cm

Distance (cm) (Reference)	Transient Response	Control Strategy	
		Embedded PID	Non-Embedded PID
6.0	Rise time, $t_r$ (s)	1.7366	6.4059
	Settling time, $t_s$ (s)	7.9741	14.4715
	Overshoot, $OS$ (%)	0	0
	Steady-state error, $e_{SS}$ (cm)	0.0890	0.1840

The peak positions reached by the graph in Figure 4.24 are 8.179cm for the non-PID controlled system and 8.028cm for the PID-controlled system. Table 4.8 shows that the embedded PID controller has a rise time ( $t_r$ ) of 2.4823 seconds, a settling time ( $t_s$ ) of 7.1132 seconds, no overshoot, and a steady-state error ( $e_{SS}$ ) of

0.0280cm. In contrast, the non-embedded PID controller has a rising time of 11.3182 seconds, a settling time of 19.5567 seconds, no overshoot, and a larger steady-state error of 0.1790cm.

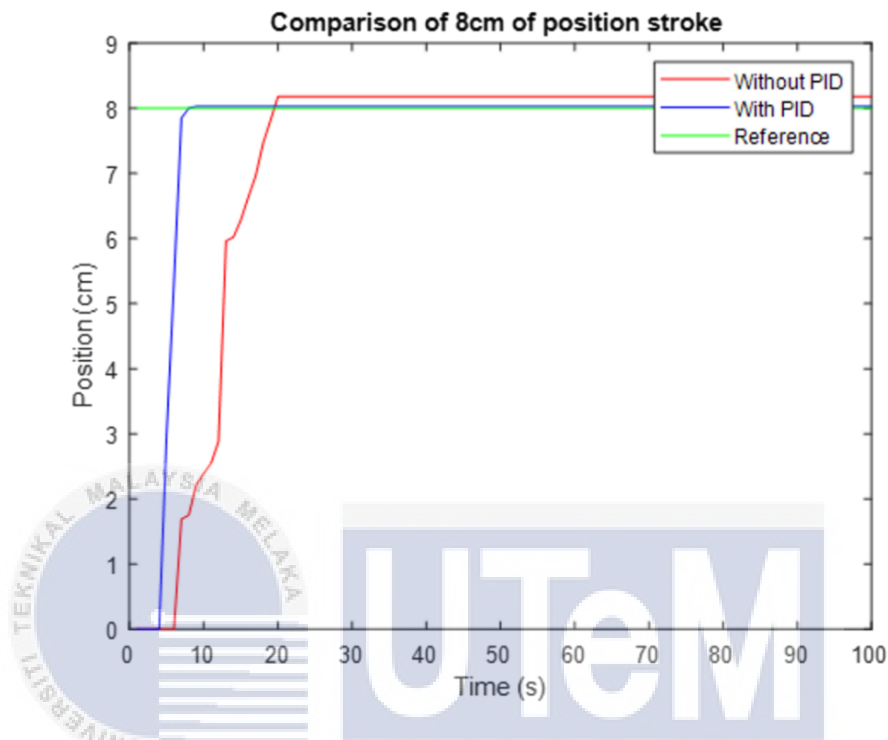


Figure 4.24 : Graph of comparison for 8cm of position stroke

Table 4.8 Transient response for distance 8cm

Distance (cm) (Reference)	Transient Response	Control Strategy	
		Embedded PID	Non-Embedded PID
8.0	Rise time, $t_r$ (s)	2.4823	11.3182
	Settling time, ( $t_s$ (s)	7.1132	19.5567
	Overshoot, $OS$ (%)	0	0
	Steady-state error, $e_{SS}$ (cm)	0.0280	0.1790

The graph in Figure 4.25 depicts the position over time; the peak positions achieved are 10.195cm for the non-PID controlled system and 10.083cm for the PID control

system. Table 4.9 shows that the embedded PID has a slightly faster rise time of 3.8026 seconds than the non-embedded PID, which is 10.8926 seconds. The embedded PID has a significantly shorter settling time which is 9.1959 seconds than the non-embedded PID (16.1259 seconds), indicating faster stabilization. Furthermore, the embedded PID system has no overshoot (0%), as does the non-embedded PID. The embedded PID has a significantly lower steady-state error which 0.0830cm than the non-embedded PID (0.1950cm).

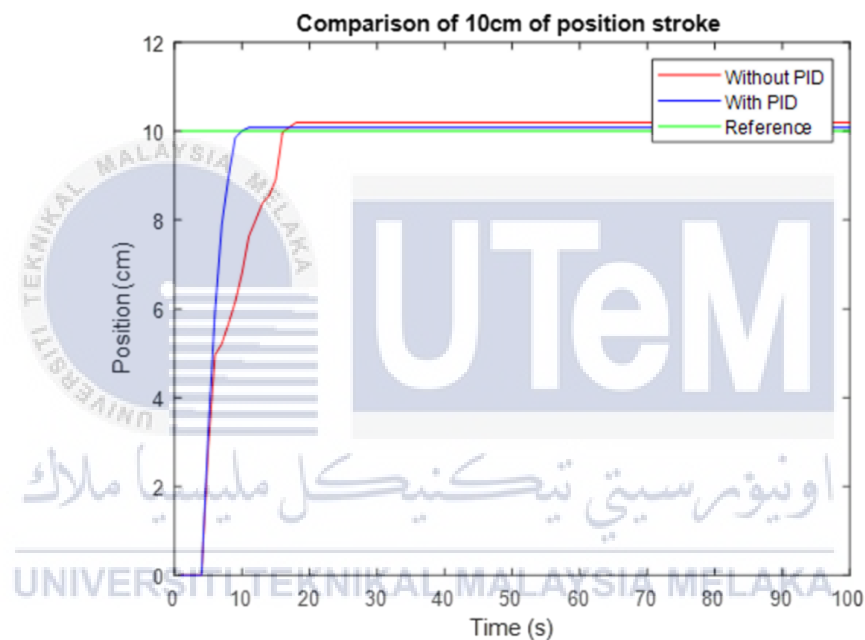


Figure 4.25 : Graph of comparison for 10cm of position stroke

Table 4.9 Transient response for distance 10cm

Distance (cm) (Reference)	Transient Response	Control Strategy	
		Embedded PID	Non-Embedded PID
10.0	Rise time, $t_r$ (s)	3.8026	10.8926
	Settling time, $t_s$ (s)	9.1959	16.1259
	Overshoot, $OS$ (%)	0	0
	Steady-state error, $e_{ss}$ (cm)	0.0830	0.1950



In summary, the transient response results of the pneumatic lifting system, controlled by the embedded PID controller, reveal significant insights into the system's dynamic performance under various conditions. The experiments involved subjecting the lift to step input changes and recording key performance metrics such as rise time, settling time, overshoot, and steady-state error.



## CHAPTER 5

### CONCLUSION AND FUTURE WORKS



#### 5.1 Conclusion

In this thesis, a pneumatic PID controller with an embedded system was successfully designed and built. The Arduino UNO R3 microcontroller was chosen as the driving and controlling unit for the piston's position. An optical encoder sensor detected the number of lines in the strip code, allowing the Arduino to accurately determine the piston lengths during extraction or retraction. The proposed PID controller achieved a high level of accuracy in controlling the pneumatic stroke, demonstrating that the percentage error of the embedded PID controller is less than 2%.

This study advances the design of embedded controllers and pneumatic monitoring systems, emphasizing the incorporation of Internet of Things (IoT) technologies for

industrial applications. By integrating IoT, the system gains connectivity and remote monitoring capabilities, enhancing overall functionality and control. This integration enables real-time data analysis, predictive maintenance, and remote operation, significantly advancing industrial applications.

The study also proves that transient response analysis, comparing non-embedded PID to embedded PID results, shows a decrease in steady-state error, indicating a faster system response. The system can continuously react to even slight deviations from the desired set-point. This is explained by the high proportional gain ( $K_P$ ) value, as a higher ( $K_P$ ) reduces steady-state error. The rise time and settling time also decrease, which is attributed to a small integral gain ( $K_I$ ) value, as a higher ( $K_I$ ) can slightly increase settling time. The derivative gain ( $K_D$ ) value is kept smallest to prevent the system from becoming overly sensitive to noise and fluctuations.

## 5.2 Future Work

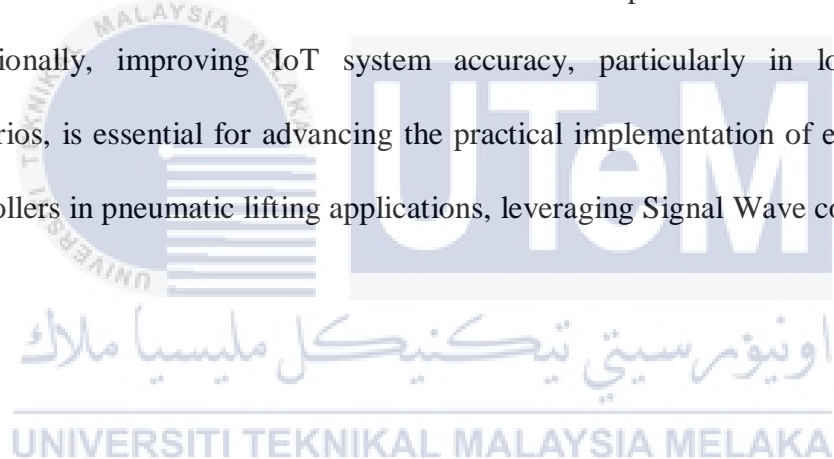
Based on the findings of this project, future research can focus on two key areas: enhancing PID controller implementation and improving IoT system data accuracy in low-connection scenarios.

In this project, PID parameters ( $K_P, K_I, K_D$ ) were determined through an experimental approach involving systematic adjustment to achieve desired performance characteristics such as minimal overshoot, rapid settling time, and steady-state accuracy. Future work aims to enhance PID control by implementing an auto-tuning mechanism. This mechanism would dynamically adjust  $K_P, K_I, K_D$  values in real-time based on feedback from the system. By automating this process,

the PID controller can adapt more effectively to varying operational conditions and disturbances, ensuring optimal performance without manual intervention.

Furthermore, the IoT system used in this project may require improvements to operate efficiently in low-connection environments. Future research can focus on replacing the current connectivity infrastructure with a more robust gateway solution. This enhancement would ensure reliable communication between the IoT system and the controller, even under poor network conditions or in remote locations.

In conclusion, future research should prioritize implementing an auto-tuning mechanism for the PID controller to enhance piston movement precision. Additionally, improving IoT system accuracy, particularly in low-connection scenarios, is essential for advancing the practical implementation of embedded PID controllers in pneumatic lifting applications, leveraging Signal Wave connectivity.



## REFERENCES

- [1] I. C. Duțu, T. Axinte, M. F. Duțu, L. Calancea, and M. Diaconu, "Research regarding use of pneumatic linear actuator," *Technium: Romanian Journal of Applied Sciences and Technology*, vol. 4, no. 2, 2022, doi: 10.47577/technium.v4i2.6144.
- [2] D. Saravanakumar, B. Mohan, and T. Muthuramalingam, "A review on recent research trends in servo pneumatic positioning systems," *Precis Eng*, vol. 49, pp. 481–492, Jul. 2017, doi: 10.1016/J.PRECISIONENG.2017.01.014.
- [3] Kagawa, T. (2008). "NEW PNEUMATIC TECHNIQUES AND APPLICATIONS". Proceedings of the JFPS International Symposium on Fluid Power, 2008(7-1), 3–8. <https://doi.org/10.5739/isfp.2008.3>
- [4] Asadi, M. T. Ahmadian, and F. Golnaraghi, "Pneumatic actuation systems in robotics," in *Intelligent and Robotic Systems*, Springer, 2019, pp. 149-175

- [5] M. Rogalski, J. Cegielski, and T. Muszyński, "Energy-efficient intelligent control of pneumatic actuators," *J. Intell. Manuf.*, vol. 32, no. 2, pp. 383-405, 2021
- [6] Lamberti, G., Spenillo, G., & Toto, F. (2019). Linear pneumatic actuators: An overview of architectures, controls, and applications. *Mechanism and Machine Theory*, 132, 19-39
- [7] L. Hakansson, J. O. Palmberg, and P. Krus, "Modern pneumatic actuator design," *Procedia Eng.*, vol. 201, pp. 466-471, 2016
- [8] Frey, C., Umlauf, G., Altemöller, M., Völlner, A., & Hesselbach, J. (2003). Miniaturized piezoelectric valveless pump based on a planar spiral coil structure. *Sensors and Actuators A: Physical*, 107(3), 235-242
- [9] Suzumori, K. (2008). Intelligent pneumatic actuator system. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (pp. 1582-1587).
- [10] Benhadj, M. L., Bois, G., & Sergent, M. (2000). Optimal sensor array design using a genetic algorithm: application to a pneumatic actuator. *Sensors and Actuators A: Physical*, 85(1-3), 244-253.
- [11] Fleming, P. J. (2001). State of the art in automotive sensors. *Measurement Science and Technology*, 12(9), 1342
- [12] Chen, J., & Li, X. (2017). A Review of Pneumatic Valve Design and Actuation Mechanisms. *Journal of Fluids Engineering*, 139(6), 061101. doi:10.1115/1.4035619

- [13] Santos, A., & Wang, L. (2018). Recent Advances in Smart Pneumatic Valves for Industrial Automation. *IEEE/ASME Transactions on Mechatronics*, 23(3), 1235-1246. doi:10.1109/TMECH.2018.2817609
- [14] Tanaka, S., & Yamamoto, Y. (2019). Development of Pneumatic Proportional Control Valves for Energy-Saving Applications. *Energy*, 166, 183-193. doi:10.1016/j.energy.2018.10.209
- [15] Zhang, Q., et al. (2020). Additive Manufacturing of Pneumatic Valves: A Review. *Journal of Manufacturing Science and Engineering*, 142(2), 020801. doi:10.1115/1.4045322
- [16] Liu, S., et al. (2021). Advanced Control Strategies for Pneumatic Valve Systems in Robotics. *Robotics and Computer-Integrated Manufacturing*, 69, 101991. doi:10.1016/j.rcim.2020.101991
- [17] Johnson, A., et al. (2005). "Optical encoders for precision pneumatic actuator control." *IEEE/ASME Transactions on Mechatronics*, 10(3), 319-326
- [18] Li, Q., et al. (2010). "Magnetic encoder sensors for pneumatic actuator position control." *Sensors and Actuators A: Physical*, 157(2), 291-298
- [19] Chen, H., et al. (2015). "Incremental encoder sensors in closed-loop control of pneumatic actuators." *Control Engineering Practice*, 42, 104-115
- [20] Zhang, W., et al. (2018). "Optoelectronic encoder sensors for high-speed pneumatic actuator control." *Measurement*, 118, 77-85

- [21] Park, J., et al. (2021). "Wireless encoder sensors for pneumatic actuator monitoring in IoT-based systems." *Sensors*, 21(2), 581.
- [22] Lee, O. Sokolsky, T. Chen, M. P. E. Heimdahl, and L. Sha, "Design Challenges for Reliable Autonomous Systems," *IEEE Des. Test*, vol. 33, no. 6, pp. 72-79, 2016
- [23] Li, J., Yang, L., Hu, S., Song, Y., & Chen, Y. (2021). A review on soft pneumatic actuators for wearable robotics. *Applied Sciences*, 11(6), 2597
- [24] S. Wang et al., "Towards smart factory for industry 4.0: a self-organized multiagent system with big data based feedback and coordination," *Computer Networks*, vol. 101, pp. 158-168, 2016
- [25] Dong, "Embedded Systems: Introduction to the MSP432 Microcontroller," Wiley, 2017.
- [26] R. H. Katz, "Contemporary Logic Design," 2nd ed. Pearson, 2016.
- [27] Aggarwal, R. and Lal Das, M. "RFID Security in the Context of "Internet of Things". First International Conference on Security of Internet of Things, Kerala", 17- 19 August 2012, 51-56, 2012, <http://dx.doi.org/10.1145/2490428.2490435>
- [28] Smith, A. et al. (2022). Emotion Theories and adolescent well-being: Results of an online intervention JOURNAL
- [29] Osman, K., Mohd Faudzi, A. 'Athif, Rahmat, M. F., & Suzumori, K. (2014). System Identification and Embedded Controller Design for Pneumatic Actuator



with Stiffness Characteristic. *Mathematical Problems in Engineering*, 2014, 1–13. <https://doi.org/10.1155/2014/27174>.

- [30] Abdul-Lateef, W. E., Alexeevich, G. N., Farhood, N. H., Khdir, A. H., & Shaker, D. H. (2020, March). Modelling and controlling of position for electro-pneumatic system using Pulse-Width-Modulation (PWM) techniques and Fuzzy Logic controller. In *IOP Conference Series: Materials Science and Engineering* (Vol. 765, No. 1, p. 012020). IOP Publishing.
- [31] M. Ahmad, A.R. Yusoff, and Z.A. Ibrahim. (2021). “Embedded Control of a Pneumatic Position System Using a Field Programmable Gate Array”. *IEEE/ASME Transactions on Mechatronics*.
- [32] J. Zhang, Y. Li, and X. Zhang. “Model-Based Control of a Pneumatic Position System Using an Embedded System”. *Journal of Dynamic Systems, Measurement, and Control*, 2022.
- [33] M. Ghazaly, M.F. Mohd Radzi, and N.A. Rahim. (2022). Research on “Design and Implementation of a PID Controller for a Pneumatic Position System Using an Embedded Microcontroller”. *2022 International Journal of Advanced Manufacturing Technology*.
- [34] H. Zhang, X. Chen, and Y. Li. Research on “Experimental Modeling and Control of a Pneumatic Cylinder Using an Embedded Controller”. *IEEE Transactions on Industrial Electronics* 2023.
- [35] Kosmatos, E.A., Tselikas, N.D. and Boucouvalas, A.C. “Integrating RFIDs and Smart Objects into a Unified Internet of Things Architecture. *Advances in* 109

Internet 54 of Things: Scientific Research”, 1, 5-12, 2011,  
<http://dx.doi.org/10.4236/ait.2011.11002>

[36] Reinhardt, A. “A Machine-To-Machine 'Internet Of Things'. Business Week”,  
(3880), 102-102, 2004

[37] Shao, W., & Li, L. “Analysis of the development route of IoT in China. Perking:  
China Science and Technology Information”, 24, 330-331, 2009.



## APPENDICES

### Appendix A

```

#define BLYNK_TEMPLATE_ID "TMPL6RUF3TRdp"
#define BLYNK_TEMPLATE_NAME "EMBEDDED PID CONTROLLER"
#define BLYNK_AUTH_TOKEN "pJvgZXrjP7yI3zy32tOzrqH0xkbadPaF"
#define encoderI 2
#define encoderQ 3 // Only use one interrupt in this example
#define RESET_BUTTON

#define BLYNK_PRINT Serial
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <SoftwareSerial.h>
#define ESP8266_BAUD 9600
SoftwareSerial EspSerial(6, 5); // RX, TX
ESP8266 wifi(&EspSerial);

volatile float count;
float lengths;
float expected;
int state;
bool new_value_entered = false;
bool actuator_reached_value = false;
float last_encoder_position = 0.0;
float encoder_position_when_actuator_reached_value = 0.0;
float vout = 0;

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Lattyfaaaa";
char pass[] = "Typa240700";

//BlynkTimer timer;

// PID variables
float setpoint, input, output;

```

```
float Kp = 9.0, Ki = 0.2, Kd = 0.01;
float integral = 0.0;
float previous_error = 0.0;
unsigned long last_time;
unsigned long current_time;
```

```
BLYNK_WRITE(V0)
{
  int resetButtonState = param.asInt();
  if (resetButtonState == HIGH) {
    count = 0;
    state = 1;
    new_value_entered = false;
    actuator_reached_value = false;
    last_encoder_position = 0.0;
    encoder_position_when_actuator_reached_value = 0.0;
    vout = 0;
  }
}
```

```
BLYNK_WRITE(V3)
{
  expected = param.asFloat();
  setpoint = expected; // Set the PID setpoint
  state = 2;
  new_value_entered = true;
  delay(2000);
}
```

```
void setup()
{
  Serial.begin(9600);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(encoderI, INPUT);
  pinMode(encoderQ, INPUT);
  //attachInterrupt(digitalPinToInterrupt(encoderI), handleEncoder, CHANGE);
  attachInterrupt(0, handleEncoder, CHANGE);
  count = 0;
  state = 1;
  EspSerial.begin(ESP8266_BAUD);
  delay(10);
  Blynk.begin(auth, wifi, ssid, pass, "blynk.cloud", 80);
  last_time = millis();
}
```

```
void loop()
{
  Blynk.run();
}
```

```

lengths = count / 141.73;
input = lengths; // Update the PID input with the current length

// PID computation
current_time = millis();
float elapsed_time = (current_time - last_time) / 1000.0; // Convert to seconds
float error = setpoint - input;
integral += error * elapsed_time;
float derivative = (error - previous_error) / elapsed_time;

output = Kp * error + Ki * integral + Kd * derivative;

previous_error = error;
last_time = current_time;

if (state == 1)
{
count = 0;
retract();
delay(2000);
if (count == 0)
{
Serial.print("Enter a value between 0 and 10:");
state = 0;
new_value_entered = true;
}
}
if (state == 0)
{
if (Serial.available() > 0)
{
expected = Serial.parseFloat();
if (expected >= 0 && expected <= 10)
{
setpoint = expected; // Set the PID setpoint
state = 2;
new_value_entered = true;
}
}
else
{
Serial.println("Invalid input. Please enter a value between 0 and 10:");
}
}
}
if (state == 2)
{
if (output > 0) {
extract(); // Move forward
} else if (output < 0) {

```

```

    retract(); // Move backward
  } else {
    fix(); // Stop movement
  }

  if (abs(lengths - expected) < 0.05) {
    fix();
    delay(1000);
    state = 0;
    actuator_reached_value = true;
    encoder_position_when_actuator_reached_value = (count) / 141.73;
  }
}

if (new_value_entered) {
  Serial.print("Value entered: ");
  Serial.println(expected);
  last_encoder_position = count / 141.73;
  new_value_entered = true;
}

if (actuator_reached_value) {
  Serial.print("Actuator reached value: ");
  Serial.println(lengths);
  Serial.print("Encoder position: ");
  Serial.println(encoder_position_when_actuator_reached_value);
  //Serial.println(last_encoder_position);
  Blynk.virtualWrite(V1, lengths);
  Serial.println(" ");
  actuator_reached_value = false;

  delay(2000);

}
delay(10);
}

void handleEncoder()
{

  if (digitalRead(encoderI) == digitalRead(encoderQ))
  {
    count++;
  }
  else
  {
    count--;
  }
}
void extract()

```

```
{  
  digitalWrite(9, LOW);  
  digitalWrite(8, HIGH);  
}  
void retract()  
{  
  digitalWrite(9, HIGH);  
  digitalWrite(8, LOW);  
}  
void fix()  
{  
  digitalWrite(9, HIGH);  
  digitalWrite(8, HIGH);  
}
```

