Jurnal Teknik Elektro: Electronic Control, Telecomunication, Computer Information and Power System

P-ISSN: 2502-0986 E-ISSN: 2686-0635

Temperature Control of Focused Beam Lamp: PID-Controlled Pulse Width Modulation using MATLAB Simulink and Arduino

Farid Baskoro [a,1,*], Miftahur Rohman [a,2], Dimas Arya S.F [b,1], Aristyawan Putra [a,3]

[a] Electrical Engineering, State University of Surabaya, St. Ketintang, Surabaya, Indonesia
[b] Aviation Polytechnic of Surabaya, St. Jemur Andayani I, Surabaya, Indonesia
[1] faridbaskoro@unesa.ac.id

ARTICLE INFO

Article History
Submission 19-04-2024
Revision 13-05-2024
Accepted 06-07-2024

Keywords
PID, Matlab, Temperature

ABSTRAK

This research explores the application of PID control on Arduino Uno using MATLAB Simulink to maintain temperature stability in a system. The main advantage of the PID approach is its adaptability to a variety of temperature control scenarios. Through MATLAB Simulink, users can easily optimize PID parameters such as P, I, and D according to system characteristics and specific needs. The three graphs analyzed depict the system response to changes in PID parameters. The first graph shows a moderate response with controlled fluctuations, reaching setpoint at 60 seconds. The second graph shows a dynamic and complex response, while the third graph produces a very active response with significant fluctuations. The results show that a higher P value results in a more active response and larger fluctuations, but the parameter adjustment must be carried out carefully to achieve a balance between fast response and system stability. In conclusion, the first graph strikes a good balance between responsiveness and stability, while the second and third graphs show the challenges in maintaining stability in certain situations.

1. Introduction

MATLAB's Simulink offers an intuitive and visual modeling approach, facilitating the design of complex systems with easy-to-understand graphical representations. Users can quickly assemble visual blocks representing system components and connect them to form a comprehensive model. This advantage provides great benefits, especially in understanding and developing system models with high efficiency [1] [2].

One of the significant advantages of Simulink MATLAB is its ability to perform real-time simulations. This allows users to test their models in conditions that are close to real operational situations. This real-time simulation capability not only helps in identifying potential problems, but also allows for fixes and improvements before physical implementation. Additionally, Simulink MATLAB provides fast analysis and optimization tools. Users can easily test the model with various parameters and conditions, allowing them to understand its impact on the system as a whole. This analysis supports the development of more efficient and reliable solutions [3] [4] [5] [6] [7].

With seamless integration with the MATLAB programming environment, Simulink provides an overall coherent and effective solution. This opens the door for users to combine the power of visual modeling with powerful analysis and programming capabilities, creating a holistic and efficient development environment. Thus, Simulink MATLAB is not just a modeling tool, but also an integrated solution for advanced system development [8] [9].

Matlab provides very powerful and flexible PID control features, enriching the user experience in designing control systems. Proportional, Integral and Differential (PID) control is one of the most commonly used control methods in various industrial applications. Matlab provides easy access to PID algorithms through a comprehensive library of functions, allowing users to implement PID control with parameters that can be adjusted according to system requirements [10] [11]. The advantage of PID control in Matlab lies in its ability to handle various types of dynamic systems with different responses. Users can adjust PID parameters, such as proportional, integral and differential gain, and observe their impact on system response in real-time through simulation. These features allow users to fine-tune their PID controls to achieve optimal performance [12] [13].

Matlab shows significant advantages with its ability to program Arduino, creating an integrated and efficient development ecosystem. Seamless integration between Matlab and Arduino allows users to design complex system models using Simulink, as well as simulate them before physical implementation. This advantage makes it easy to identify potential problems, analyze system characteristics, and perform virtual debugging before executing code on Arduino hardware. One of the main advantages is the existence of a function library in Matlab that supports control development, including PID control algorithms. With this, users can design, test, and optimize control algorithms before implementing them on Arduino. This not only saves time and resources, but also allows developers to achieve optimal performance on the control systems used.

In this research, Matlab is a tool that combines the power of modeling and simulation with direct implementation on Arduino hardware. Overall, the integration of Matlab with Arduino opens up vast opportunities for innovation, research, and development of projects involving microcontrollers.

2. Research Methods

This research method utilizes PID (Proportional, Integral, Differential) control to achieve temperature stabilization in the beam light focus control system. In this research, Pulse Width Modulation (PWM) technology was used combined with a PID controller to regulate the heating intensity of the focused beam lamp. The use of MATLAB Simulink is the first step in designing a system model, enabling visual modeling and simulation of the system's response to

temperature changes [14] [15]. After getting the desired simulation results, the PID program produced by Simulink can be uploaded directly to the Arduino UNO. Therefore,

The Arduino UNO acts as a physical controller that can apply the PID algorithm to regulate the energy input to the focused beam lamp, so that the temperature can be maintained at the desired level. This approach combines the advantages of MATLAB Simulink in system modeling with the practical capabilities of direct implementation on Arduino UNO hardware, creating a temperature control system.

This research involves the use of an LM35 temperature sensor to measure environmental temperature, while PID control is implemented using a RobotDyn dimmer to regulate the heating intensity of the focused beam lamp. LM35, as a precision temperature sensor, provides accurate temperature readings and is responsive to changes in ambient temperature. In order to achieve the research objective of maintaining the temperature at a stable value of around 32 degrees Celsius, system modeling was carried out using MATLAB Simulink.

The system model was built in Simulink with an LM35 temperature sensor as input, a RobotDyn dimmer as output, and a PID controller to regulate the heating of the focused beam lamp. Through simulation, the PID parameters are set in such a way that the system response reaches the desired temperature value, namely 32 degrees Celsius. After getting the optimal model, the PID program is implemented directly on the Arduino UNO to control the RobotDyn dimmer in real-time.

With this approach, the system is able to measure temperature using the LM35 sensor, apply PID control to adjust the RobotDyn dimmer, and maintain the focal temperature of the beam lamp at the targeted value. Thus, this research combines precision sensory technology and a PID controller to create an effective and responsive temperature control system using MATLAB Simulink and Arduino UNO.

3. Results and Discussion

Equation 1 represents the proportional, integral, and derivative controllers commonly used in automatic control systems. The proportional component (P) provides a response proportional to the magnitude of the current error between the setpoint and the system's actual value. The integral component (I) responds to errors over time, helps overcome steady-state errors, and improves response to system changes. The derivative component (D) responds to the error rate of change, helping to prevent overshoot and improve system stability.

Variable s is a Laplace variable used to analyze systems in the frequency domain. The N factor in the derivative component functions as a filter to regulate the characteristics of the system's response to change. The P,I, and D parameter settings are adjusted according to the characteristics of the particular control system to achieve the desired performance. The PID formula is an important tool in automatic control that is used to regulate the system effectively and achieve the desired setpoint value.

$$u(t) \quad = P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

Equation (1)

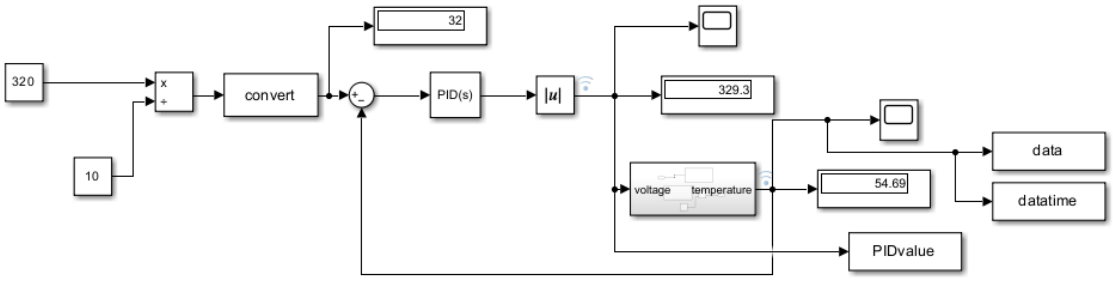

Figure 1. Block Simulink Matlab

In the Simulink circuit Figure 1, the input given is a value of 320. This value is then divided by 10 as a calibration step to set a temperature setpoint of 32. This setpoint is then connected directly to the display block to allow direct observation of the setpoint value. Furthermore, the setpoint value is also connected to the sum block which acts as a sum point in the system.

The calibrated setpoint values are also routed to the PID block, which encapsulates proportional, integral, and derivative controls to produce optimal control signals. The output of the PID block is then connected to the absolute block, which is used to produce the absolute value of the control signal. The absolute block is then connected to an absoluter block, which may indicate the need to provide a further absolute value to the control signal.

Next, the output of the absoluter block is connected to a subsystem. This subsystem may include additional logic and operations depending on temperature control needs. The output from the subsystem is directed back to the summation point via a feedback block, which allows information back into the system to correct and adjust temperature control based on the system's response to control signals.

Overall, the Simulink circuit represents a temperature control system that involves setpoint calibration, use of PIDs to generate control signals, absolute value processing, and integration with a subsystem to optimize temperature control. A feedback loop involving a sum block and a feedback block helps fine-tune the system's response to temperature changes and maintain its stability.

The "Subsystem" block function in MATLAB Simulink is a critical component that allows users to organize and group a number of Simulink blocks or elements into one separate unit. The subsystem acts as a container or "container" which helps in creating a more structured and easy to understand Simulink model. When a system has high complexity, using the “Subsystem” block allows for modular modeling, separating the various aspects of the system into smaller, manageable parts.

In this context, users can create a Subsystem to represent a function or module in a larger system. Subsystem blocks can be named and treated as independent entities, thereby facilitating efficient system engineering. These blocks can be placed freely within a Simulink model, and can help avoid cluttering overly complex model displays.

The "Subsystem" block also supports a higher level of code readability and maintainability, as users can provide additional documentation and comments to the block. Additionally, Subsystems can be called from other locations in the model, providing the ability to design models that are modular and reusable.

Overall, the use of the "Subsystem" block in MATLAB Simulink provides flexibility and organization in building complex system models, improving the readability, maintainability, and overall manageability of a control or system development project.

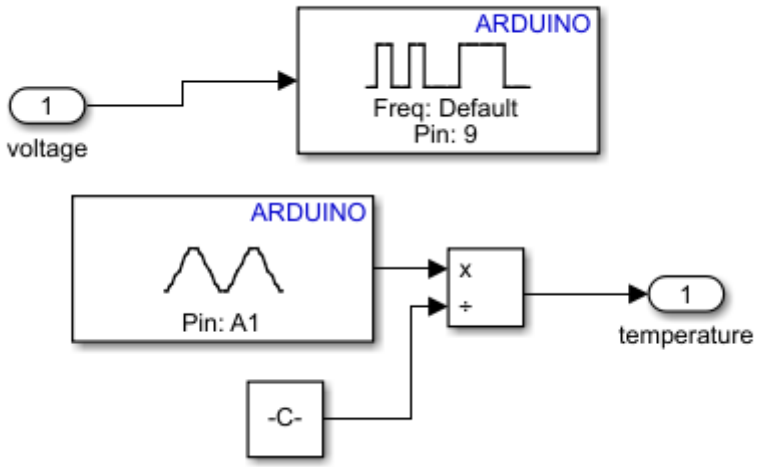

Figure 2. Block I/O system inside block subsystem

In the implementation of the "Subsystem" block in MATLAB Simulink for temperature measurement using the LM35 sensor on Arduino, Figure 2 is designed with two main connections. First, the block input is connected to PWM pin 9 on the Arduino. These pins can be used to control external devices or modules, such as heating elements, according to application requirements. PWM input allows the use of pulse signals with adjustable pulse widths for control purposes.

Next, in the "Subsystem" block, there is a temperature measurement process using the LM35 sensor. This sensor produces an analog voltage that is linearly correlated with the surrounding temperature. This process includes reading the voltage value from the LM35, which can then be interpreted as a temperature in degrees Celsius. The results of this temperature measurement will be the output of the "Subsystem" block.

Additionally, the output of the "Subsystem" block is connected to analog pin 1 on the Arduino. By connecting the temperature measurement output to the analog pin, the temperature data produced by the LM35 sensor can be read by the Arduino microcontroller and then processed or displayed according to application needs. The use of the “Subsystem” block helps in organizing and separating the temperature measurement logic from the control and output elements in the Simulink model, improving the readability and ease of maintenance of the model.

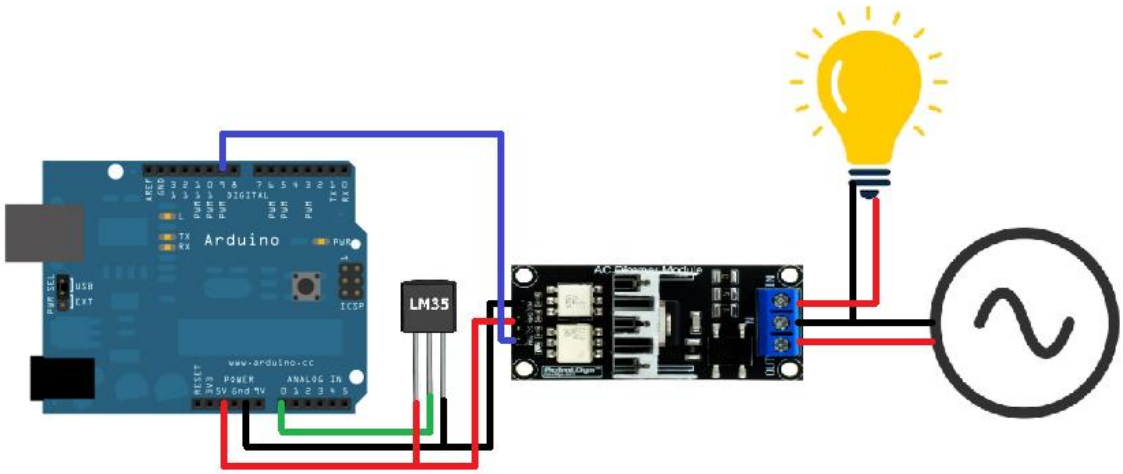

Figure 3. Wiring Hardware

The configuration in Figure 3 illustrates the use of an Arduino connected to the LM35 temperature sensor via analog pin A0. The LM35 temperature sensor produces an analog output that corresponds to the temperature of the surrounding environment. The Arduino, via pin A0, reads this analog value and can be used to monitor the temperature in that environment. Apart from that, there is a RobotDyn dimmer device connected to PWM pin 9 on the Arduino. The use of this PWM pin allows control of the brightness level of the dimmer for the light source connected to it.

The output from the RobotDyn dimmer is connected to a 220V power source, indicating that the dimmer can regulate the flow of electrical power to the lights connected to it. This allows controlling the brightness level of the lamp based on the signal sent by the Arduino via the PWM pin. Finally, a lamp with a focused beam connected to the power source will provide focused light and its brightness can be adjusted according to the signal from the dimmer.

Overall, this configuration creates a system where the Arduino uses the LM35 temperature sensor to monitor the temperature and regulate the brightness of the focusing lamp via the RobotDyn dimmer using the PWM signal from pin 9. This system can be implemented, for example, to create optimal lighting conditions based on the ambient temperature at a certain area.

This configuration involves using the output of the PID block on the Arduino to control the RobotDyn dimmer via PWM pin 9. The PID block plays a role in generating an optimal control signal based on the difference between the temperature setpoint (32 degrees Celsius) and the actual temperature value obtained from the LM35 sensor connected to the pin analog A0 on Arduino. With feedback from the LM35 temperature sensor, the PID block continues to produce control signals that vary according to changes in environmental temperature.

The influence of the PID output value on the RobotDyn dimmer via the PWM signal 9 allows adjustment of the lamp brightness level with a dynamic response to temperature changes. When the actual temperature value deviates from the setpoint, the PID block automatically generates a control signal that guides the dimmer to adjust the lamp brightness.

Thus, changes in environmental temperature will create controlled and responsive changes in the light brightness level.

Therefore, this configuration forms a system that runs automatically to reach and maintain a set point temperature of 32 degrees Celsius. The PWM output from the dimmer will change dynamically, influenced by the PID output value which is determined by the comparison between the setpoint and the actual temperature value from the LM35 sensor. The system is designed to achieve stability at a specified temperature and provide lighting appropriate to changing environmental conditions.

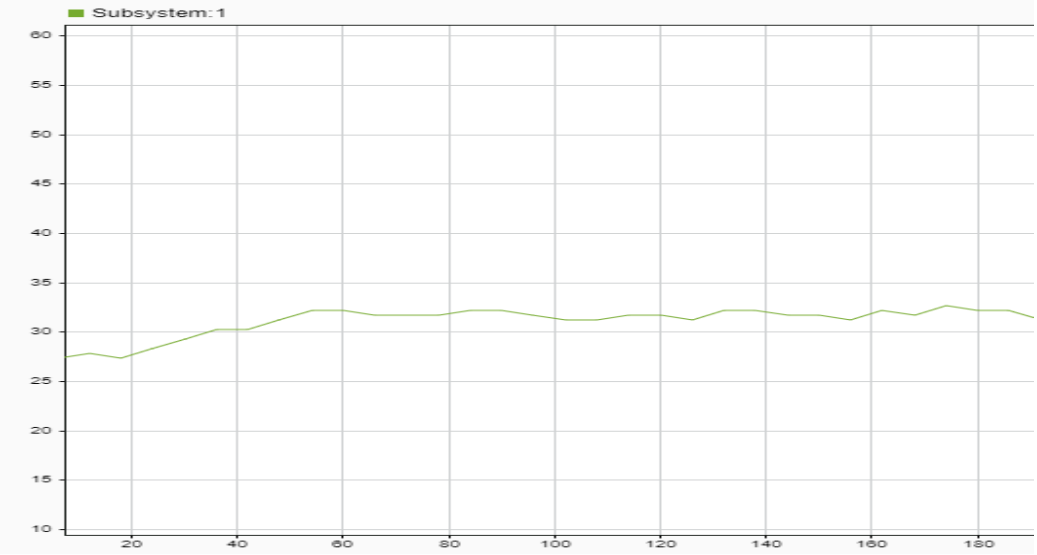

Figure 4. Temperature vs time with P=0.1, I=0.01

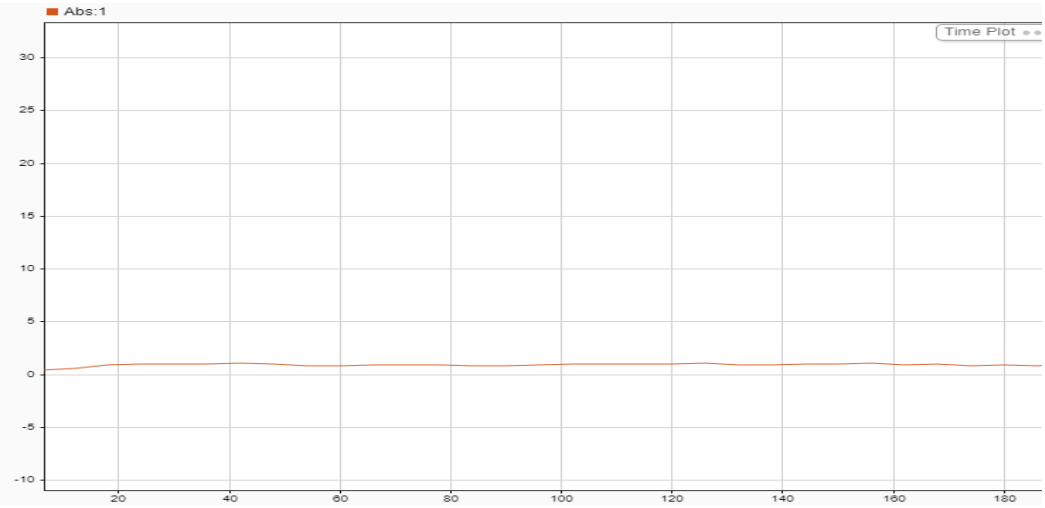

Figure 5. Pulse with modulation vs time with P=0.1, I=0.01

The PID graph in Figure 4 reflects the responsive and stable dynamics of temperature regulation. At the start, at 0 seconds, the graph shows a starting temperature of 27 degrees

Celsius. As time goes by, you can see a gradual increase in temperature until it reaches the set point at the 60th second, namely a temperature of 32 degrees Celsius.

After reaching the setpoint temperature, the graph shows that the PID system is successful in keeping the temperature at the desired level. However, special attention should be paid to the small changes visible after the 60th second. Although the main temperature is stable at 32 degrees Celsius, the graph shows small fluctuations with values going up and down by around 1 degree Celsius. These fluctuations can be caused by environmental changes or small variability in processes.

With parameter values P of 0.1, I of 0.01, and D of 0, you can detail how each PID component contributes to temperature control in the system. Proportional provides a contribution proportional to the difference between the setpoint and the actual value of temperature, Integral responds to the cumulative error over time, while Derivative produces a response to the rate of change of the error.

A relatively small P value of 0.1, the proportional impact of temperature changes will be quite moderate. This can help prevent an overly aggressive system response to small temperature changes. The also small I value of 0.01 indicates that integral control will respond to errors slowly, helping to eliminate steady-state errors in the system. Although D is set to 0, meaning there is no direct influence on the response to the rate of change of the error, this can be useful to prevent overshoot or excessive reaction to temperature changes.

With the PID parameters that have been set in Figure 5, producing a PWM value with a range of around 1 in the time range from 0 to the 180th second shows that the system effectively controls the brightness level of the lamp based on the detected temperature changes. A small PWM value, as in this case, indicates that the control signal provided to the dimmer on pin 9 of the Arduino is relatively low, creating a fairly moderate brightness setting.

In the initial time range from 0 to the 60th second, the PID graph shows an increase in temperature from 27 to 32 degrees Celsius. During this period, the PID system generates a PWM value that may increase gradually to increase the brightness level of the lamp and reach a level that corresponds to changes in temperature.

After reaching the setpoint at the 60th second, the PWM value can remain relatively stable, reflecting the PID system's ability to maintain the temperature at the desired level. The presence of fluctuations of approximately 1 in the time range from 0 to 180 seconds can be caused by environmental factors or small variability in the system, but this moderate response indicates that the system is generally successful in controlling temperature effectively.

Overall, the relatively small PWM value indicates that the PID system operates with stability and produces fairly smooth control over the lamp brightness level along with detected temperature changes.

This PID graph reflects good control performance, able to respond quickly to temperature changes and maintain stability at set point. Even though there were small fluctuations, the temperature values remained within the acceptable range, indicating that the PID system succeeded in controlling the temperature effectively and was responsive to possible environmental changes.

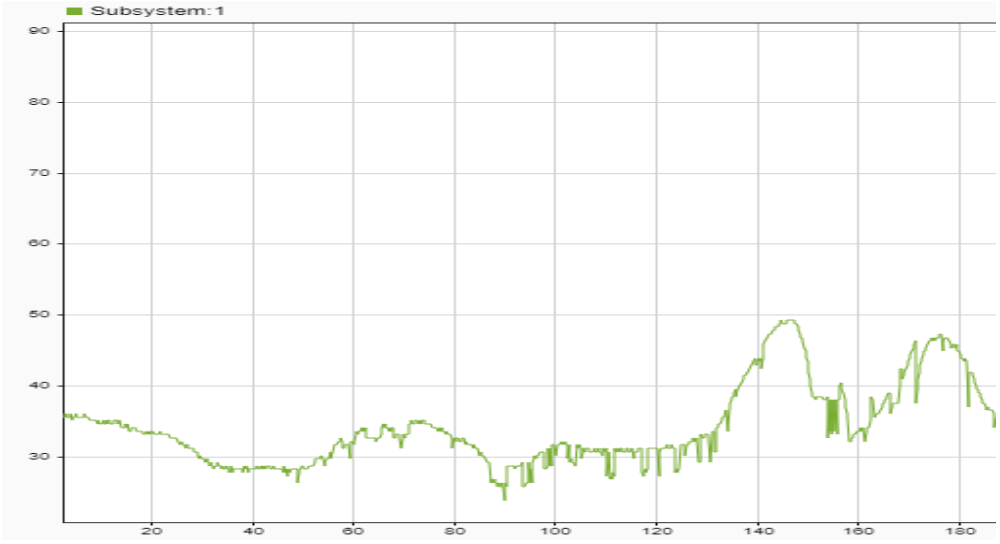

Figure 6. Temperature vs time with P=0.7, I=0.01

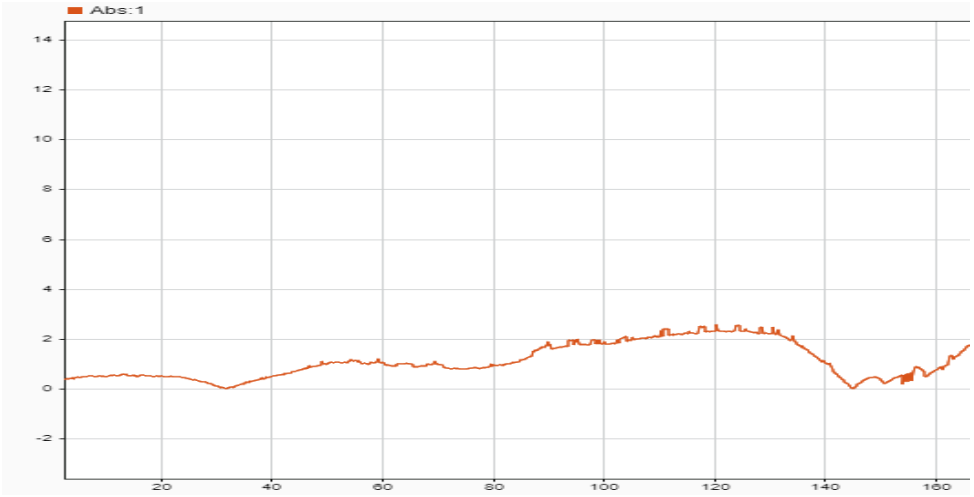

Figure 7. Pulse with modulation vs time P=0.7, I=0.01

parameters are set in Figure 6, the values P=0.7, I=0.01, and D=0, the resulting graph shows interesting dynamics in temperature control. This system is designed to provide a more aggressive response to temperature changes and exhibits different characteristics compared to previous configurations.

The graph starts with an initial temperature of around 36 degrees Celsius. In the 40th second, the temperature value was successfully reduced to 32 degrees Celsius, showing the PID system's fast and effective response to temperature changes. During the period 40 seconds to 50 seconds, the temperature remained stable at the setpoint value of 32 degrees Celsius.

Then, at the 70th second, the temperature rose again to 36 degrees Celsius, and the PID system was able to maintain this temperature stably until the 120th second. At the 150th second, the temperature suddenly increased significantly to reach 50 degrees Celsius, indicating a potential overshoot or excessive response of the system to sudden and large temperature changes.

The next period showed quite significant temperature fluctuations, reaching the highest value of 47 degrees Celsius in the 170th second and dropping to 35 degrees Celsius in the 180th second. These fluctuations may be caused by a larger increase in proportional contributions.

With the PID parameters set at P=0.7, I=0.01, and D=0, the results of the PWM values produced by the system show dynamic characteristics and are responsive to temperature changes. In Figure 7, at the start of the experiment ranging from the first to the 20th second, the system produces a PWM value of around 0.9, reflecting a fairly active response to relatively small temperature changes. At the 30th second, the PWM value decreases to 0, indicating an attempt by the system to adjust the brightness level or turn off the lights in response to the temperature drop. At the 50th second, the PWM value again rises to 1, indicating the system's adaptation to a significant increase in temperature.

It is important to note that at the 80th second, the PWM value reaches a stability of around 1, indicating that the system successfully achieved the desired brightness level and was able to maintain it throughout that period. At 110 seconds, the PWM value increases to 2, reflecting a more aggressive response to temperature changes and increased brightness levels. However, at 145 seconds, the PWM value drops back to 0, perhaps in response to the temperature drop or as an effort to maintain stability. Finally, at 180 seconds, the PWM value rises again to 2, indicating a more intense response to temperature or changes in environmental conditions.

These results reflect the complexity of PID system dynamics in controlling lamp brightness levels based on temperature changes, and emphasize the importance of adjusting PID parameters to achieve an optimal balance between fast response and maintaining system stability.

Overall, the PID configuration with P=0.7, I=0.01, and D=0 produces a more dynamic response to temperature changes, but also presents challenges in maintaining system stability, especially when sudden and significant temperature changes occur. Further adjustments to the PID parameters may be necessary to improve system performance and reduce fluctuations that occur.

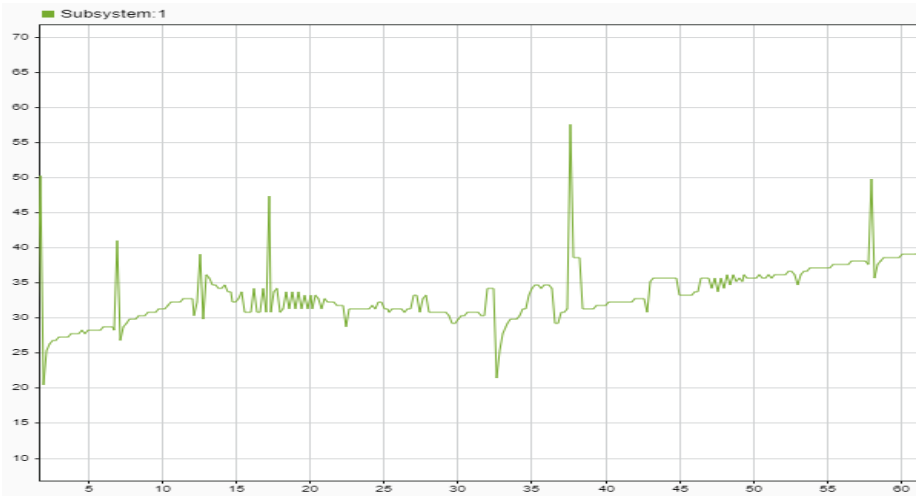

Figure 8. Temperature vs time with P=1, I=0.1

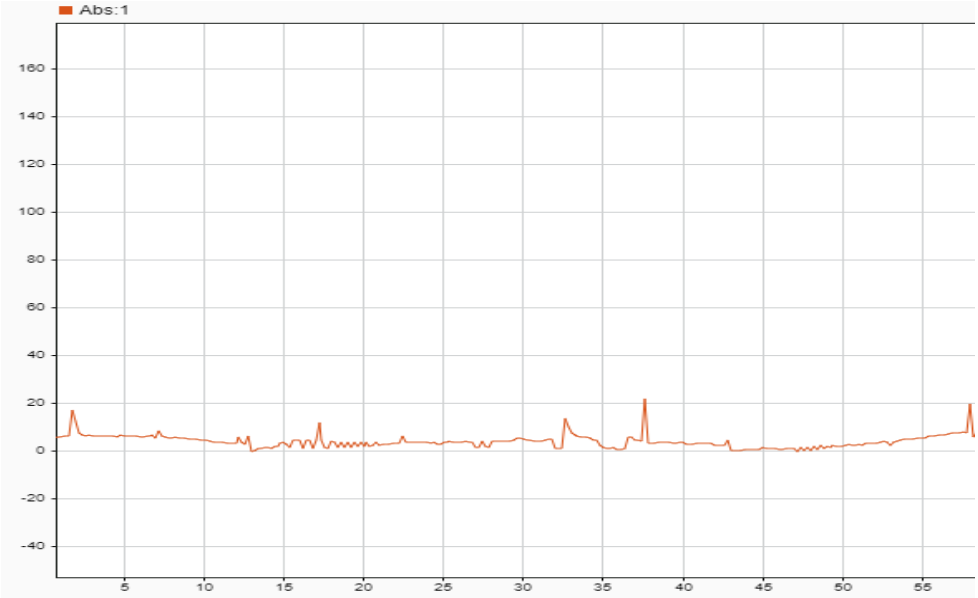

Figure 9. Pulse with modulation vs time P=1, I=0.1

By changing the PID parameters in Figure 8 to P=1, I=0.1, and D=0, the system shows a more active response to temperature changes. Initially, at 0 seconds, the temperature starts from 27 degrees Celsius. After 10 seconds, the temperature rose significantly to 32 degrees Celsius, indicating a fast response from the PID system. However, after reaching the set point at the 10th second, the temperature continued to fluctuate.

In the 13th second, the temperature rose higher to 35 degrees Celsius, but fell back to 32 degrees Celsius in the 15th second. Next, repeated up and down variations occurred, with the temperature reaching 34 degrees Celsius in the 16th second and dropping back to 32 degrees Celsius in the next second. This up and down event continued until the 45th second.

After the 45th second, the graph shows that the temperature starts to increase significantly. In the 47th second, the temperature rose even higher, reaching 39 degrees Celsius. This temperature increase lasted until the 60th second, indicating that the system was attempting to maintain the temperature at a higher level after the previous period of fluctuation.

The fluctuations in the PWM value of the PID in Figure 9 from 0 to 20 indicate that the PID system responds with varying intensity to temperature fluctuations. Large changes in PWM values can reflect a rapid and aggressive response to temperature changes, but too large fluctuations can also indicate the system's difficulty in achieving stability.

In the initial to 10th second, when the temperature rises from 27 to 32 degrees Celsius, the PWM value may experience a significant increase. Then, fluctuations in the PWM value are visible when the temperature experiences up-down variations in the 13th to 45th second range.

As the temperature begins to increase significantly after the 45th second, reaching a peak at the 60th second, fluctuations in the PWM value may reflect the system's attempt to adjust the lamp brightness level according to the larger temperature change.

Too large fluctuations in PWM values can result in noticeable changes in brightness levels and may be undesirable in applications requiring stable control. In this case, further adjustments to the PID parameters or implementation of an anti-windup algorithm may be necessary to reduce fluctuations and improve system stability.

The dynamic temperature response changes and fluctuations seen in the graphs reflect a higher sensitivity to changes in PID parameters, especially with proportional and integral increases. PID parameter settings need to be observed and adjusted so that the system can provide a more stable response and in accordance with the desired setpoint

4. Conclusion

Programming the PID to maintain temperature stability on the Arduino Uno using MATLAB Simulink provides a number of significant advantages in controlling and maintaining temperature in a system. First, the PID approach provides a scalable solution for various temperature control scenarios. Using MATLAB Simulink, users can easily adjust PID parameters such as P, I, and D to optimize control performance according to system characteristics and specific needs.

The three graphs that have been discussed depict the dynamics of a temperature control system using different PID parameters. In the first graph, with parameters P=0.1, I=0.01, and D=0, the system shows a moderate response to temperature changes with controlled fluctuations, reaching setpoint at 60 seconds and maintaining it stably.

The second graph, with changes in parameters P=0.7, I=0.01, and D=0, shows a more dynamic and complex response. The system was able to cope with temperature fluctuations initially but showed challenges in maintaining stability, especially when sudden temperature changes occurred.

The third graph, with parameters P=1, I=0.1, and D=0, produces a very active response to temperature changes, with larger fluctuations in the PWM level. The system shows the ability to overcome fluctuations, but there is significant instability in the 45th to 60th second range.

Overall, changes in PID parameters affect system characteristics, with higher P values tending to produce more active responses and larger fluctuations. However, adjustment of these parameters must be done carefully to achieve a balance between fast response and maintaining system stability. The first graph shows a good balance between responsiveness and stability, while the second and third graphs show the challenges of maintaining stability in certain situations.

References

[1] T. Xin, Z. Zhenjing, S. Guoxin, D. Li dan C. Jingyu, “Modeling and Simulation of Layered Water Tank Based on MATLAB/SIMULINK,” dalam Journal of Physics: Conference Series, 2022.

[2] M. K. Mohammed dan M. A. Amer, “Parameters Estimation Tests of Induction Machine Using Matlab/Simulink,” dalam Journal of Physics: Conference Series, 2021.

[3] A. Mazin, G. Anwar, N. Abdulazez dan A. Ammar, "Design and Implementation of Line Follower Arduino Mobile Robot Using Matlab Simulink Toolbox," Iraqi Journal for Electrical And Electronic Engineering, vol. 17, no. 2, pp. 11-16, 2021.

[4] E. B. Said dan A. Younes, "Implementation of a Novel MPPT Tactic for PV System Applications on MATLAB/Simulink and Proteus-Based Arduino Board Environments," International Journal of Photoenergy, 2021.

[5] T. A. Hussein dan I. S. Ibrahim, "Implementation of Selective Harmonics Elimination for Single Phase Inverter using Arduino and Simulink MATLAB model," Tikrit Journal of Engineering Sciences, vol. 27, no. 3, pp. 31-37, 2020.

[6] R. M. Yousif, B. Noorulden, B. Oguz dan H. M. Alaa, "A New Novel Optimization Techniques Implemented on the AVR Control System using MATLAB-SIMULINK," International Journal of Advanced Science and Technology, vol. 29, no. 5, 2020.

[7] K. S. Amul, K. G. Abhishek, R. J. Rishabh dan K. G. Sarvesh, "Modelling and Simulation of Photo voltaic System Using Matlab/Simulink," dalam Journal of Physics: Conference Series, 2020.

[8] M. Khairudin, H. Wijaya dan a. Muslikhin, "Converter matlab fuzzy inference to arduino Csystem," dalam Journal of Physics: Conference Series, 2020.

[9] M. Mohcin, E. Rachid, S. Amal dan H. Laamari, "Real-Time Implementation Of A New Efficient Algorithm For Source Separation Using Matlab & Arduino Due," International Journal of Scientific & Technology Research, vol. 9, no. 4, pp. 531-535, 2020.

[10] L. Abdul, Z. A. Afif, A. W. Hendro, R. Robbi dan T. Elsayed, "Motor DC PID System Regulator for Mini Conveyor Drive Based-on Matlab," Journal of Robotics and Control, 2020.

[11] W. S. Salam, H. S. Dina dan N. A. Fatin, "Simulation model of PID for DC-DC converter by using MATLAB," International Journal of Electrical and Computer Engineering, vol. 11, no. 5, pp. 3791-3797, 2021.

[12] A. A. Buraq, K. O. Manal, A. E. Yaser dan W. S. Salam, "Simulation model of ACO, FLC and PID controller for TCP/AQM wireless networks by using MATLAB/Simulink," International Journal of Electrical and Computer Engineering, vol. 13, no. 3, 2023.

[13] W. Huarong dan Y. Qiyan, "Research on PID control of electric traction DC motor speed based on MATLAB," dalam Journal of Physics: Conference Series, 2022.

[14] D. Amine, "An Arduino-based Low-Cost Hardware for Temperature Control," WSEAS Transactions on Systems, pp. 54-66, 2021.

[15] N. Seiichi, "Arduino-based PID Control of Temperature in Closed Space by Pulse Width Modulation of AC Voltage," International Journal of Computer and Systems Engineering, 2021.