Control loop Research

Biblical and theoretical research in controlling and managing systems heavily influenced by external factors.

Contents

Introduction

There exist many different control loops, each with their own specific implementation which warrants a specific use case. This means that there can be more than one implementation which is effective, or quite the opposite; one implementation which is specifically made for the problem at hand.

Research questions

Main Question:
Which controller logic is the best for our application?

Aiding questions:

1. What are control loops?
2. Which common control loops are there and what are their characteristics?
3. Which common feedback controllers can we appoint?

Aiding practical activities:

1. Practical application of control loop design into 'Dynamic Load Scaling' program

1. What are control loops?

A control loop is a common building block in equivalently named ‘Control Systems’. These control systems are either present in day-to-day life, or in specific industrial applications. They are in place to guard and control certain processes, often doing this by adjusting certain process variables like temperature.

A control loop consists of a process sensor, a controller function and the final control element. This control element is what ultimately adjusts the process variable for that control loop to its desired set-point.

(Wikipedia, Control Loop, n.d.)

1.1 Which specific parts can we appoint in a control loop?

Like beforementioned, there are a few parts of which a control loop is built from. These parts can all have different characteristics, which should be taken into account when building a control system.

In other words, we can call these ‘parts’, the building blocks of control systems. And are often found in many a control system.

- Plant

The plant is another name for the system which should be controlled. Plant is a remnant from other industries where control loops and systems have been used too. Many of these industries involve some form of factory or industrial process, often located on/in a physical plant. Hence the name.

- Controller

The controller is the logic which decides, according to the data fed to it, how the system will respond to that data. It decides how the data is handled, and thus how severe a system reacts to certain data.

In this document, various controllers will be highlighted, showing their function.

- Sensor/feedback line

Like previously mentioned, control systems are not a new invention coming along with software design. Many, if not most, control loops exist partly in the physical world too. To create a proper feedback to the beginning of the control loop, a sensor is used. This provides the controller with the feedback data of the type the sensor reads.

In our application this is not necessary, we don’t need a dedicated sensor. Although we do measure something programmatically, that is our ‘sensor’. Sometimes this is still included in diagrams, even though there is no physical sensor.

- Data conversion

Often neglected in traditional articles but found on more practical forums and discussion threads is the act of converting/preconditioning data after it has left the controller and is on its way to the plant.

The plant sometimes is built without a control system in mind. In combination with the chosen process value, this could mean that a controller output cannot directly drive the plant. For this, conversion techniques exist which are all unique to their implementation. These are often not mentioned or showed and assumed to be part of the plant.

Even though its use is rarely seen, it is included in this research since it clarifies a lot about how controllers are used. They are not there to output an ideal parameter; they are there to control the plant. Ideal parameters can also be created according to a controller by converting it properly.

2. Which common control loops are there and what are their characteristics?

In control loops we can appoint two different types of control loops, the so called “open-loop” and “closed-loop” control loops. They are sometimes also referred to as feedforward (open) and feedback (closed).

2.1 Open-loop control

In open-loop control systems, the control action done from the controller has no knowledge of what the system currently outputs. Stated differently, open-loop controllers have no knowledge about how the system is currently doing.

The controller knows what to output to reach a specific setpoint, since this has been pre-calculated by the operators. Cruise control for example, can be calculated to some extent. If the pedal pressed down 5cm, means 50mph; we can create a datasheet of all possible inputs and corresponding throttle positions. The controller then doesn’t have to know how the system is doing, since 5cm means 50mph.

Great examples for this also involve final control elements which induce periods of time, like some thermostats. Instead of heating until the setpoint is reached, they heat for a select period. If in this period of time the temperature goes over the setpoint, the control system does not take note and continues heating until the timer has run out.

2.2 Closed-loop control

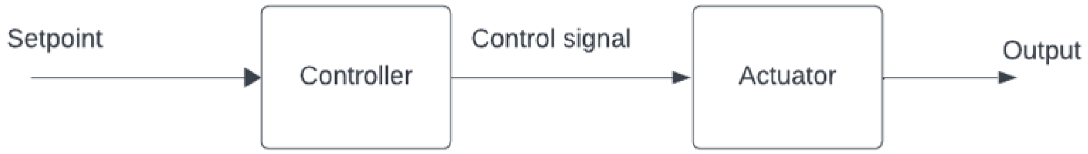
In closed-loop control system, the control action done from the controller relies directly on the process output value to adjust the process variable. This means that when the setpoint is reached, and the process output confirms this; the system will stop/alter the way it is influencing the process.

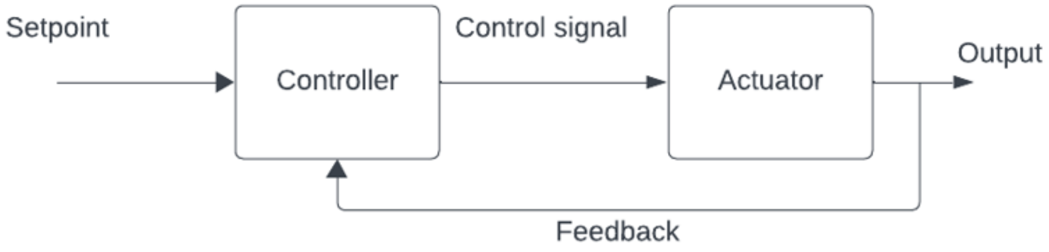
To achieve this, feedback is given back to the controller about how the system is doing. This can be done programmatically or using sensors to gather certain data about the system.

If we take the same thermostat analogy, this means that when the setpoint is reached; we immediately stop heating until the process output deviates from our setpoint again.
Note that many thermostats do not work like this, though it provides a great example in contrast to the open-loop control loop.

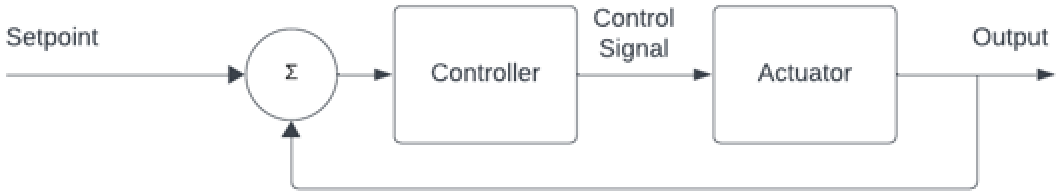
2.3 Simplified visualization

To visualize how open and closed loop systems differ from each other, we can best use a simplified diagram to show this.





Though, often an error is calculated instead of feeding directly back into the controller. A more common feedback diagram is the following.


Setpoint
Σ
Controller
Control Signal
Actuator
Output

2.4 Open vs. Closed loop

Now that open and closed loop controls are discussed, alongside their characteristics. If we take a look at what we discussed; we can clearly see and appoint some advantages of a closed-loop system.

- Instability

A closed loop system can respond to the instability induced by incalculable factors. A great example is cruise control. Math can account for the wind drag, and friction in an open loop configuration. But the road and its bumps cannot be calculated, which induces disturbances in the control loop.

In our example, this will lead to the car sitting just below the target speed.

(Wikipedia, Classical control theory, n.d.)

- Robustness

Due to open loop systems being heavily pre-planned and pre-calculated, it naturally requires a very precise model of the final environment it will operate in. If there exist even the slightest impurities in this model, the result will be calculated with a slight error. Resulting in a control loop which behaves differently than expected.

In a closed loop design, the controller is adjusted in such a way that it can respond to previously mentioned disturbances. This means that if the disturbances are induced by the math involved to design the closed loop, it can respond to these too and filter them out.

These are two main reasons why considering a closed loop vs open loop approach can really affect how a plant is performing. There are others, but they boil down to these two core ideologies.

2.5 Negative and Positive feedback

The job of a control system is to control a certain process. In technology, we often see it control certain processes to make it stable. Like a stepper motor for example. But in the nature around us, we can also find control loops. Though not as deliberately tuned, we can appoint them.

A cup of tea, cooling down to room temperature is a great example. This is a control loop, with the ambient room temperature being the setpoint; and thermodynamics the controller to gradually cool the cup of tea.

However, in this example; the room does not warm up to add more heat to the tea again. There is some nuance, but we can assume that the room cannot be used to warm our tea up again.

This is an example of a negative feedback loop, where the state of the plant constantly deteriorates; and is kept in balance by a negatively acting force.

(Feedback Loops, n.d.)

An example for a positive feedback loop, is population with a fixed birth percentage. If we observe bacteria for example, the following is a very common occurrence.

When the population is low, and has not met its limit, it rapidly expands until it reaches a point where due to external factors; the growth is not exponential anymore and caps out. These external factors for bacteria, is often food, since they need little other things. This experiment is often described in ecology classrooms to describe population theory. But we can also use it to illustrate control loops.

This is a great example of a positive feedback loop, where the state of the plant is kept at an equilibrium by a positive growing force. The equilibrium is reached by adding, not subtracting.

(Feedback Loops, n.d.)

2.6 First and Second order systems

Up until this point, it was assumed that controllers only adjust one process variable. While in reality there are many cases where more parameters are changed to control a singular output.

In other words, a first order system has one degree of freedom; while a second order system has two degrees of freedom. (Porwal, 2021)

There do exist even higher order systems, however; if a system with a higher order than one has a characteristic dominant first order implementation inside of it. We can consider this higher order system as a first order system. (S, 2021)

3. Which common feedback controllers are there and what are their characteristics?

Aside from controller loop differences, we can also find many differences in the way these control loops approach and try to solve differences induced by external factors. There are many, but we limited ourselves to these three common variants.

3.1 Bang-Bang Controller (on-off)

A Bang-Bang controller (also known as on-off controller) is a controller which switches its output to a binary value. In other words, either 0 or 1; on or off. This form of a controller is one of the simplest around.

The controller introduces all basic parts like a setpoint, error and, albeit a very simplified, controller output. The output is determined by checking if the error is either positive, or equal to/lower than 0. If positive, the output becomes 1 (on). If negative, the output becomes 0 (off).

(Wikipedia, Bang-Bang control, n.d.)

3.2 PID

PID is a very well known and often used controller due to its easy tunability and its effectiveness. A PID controller in nature, is a combination of three processes which all influence the measured error margin in a specific way. The name “PID”, is an abbreviation of these three processes, Proportional, Integral and Derivative (controller). While the proportional part itself can be used as a feedback controller, the integral and derivative part are usually accompanying it to solve problems the proportional part cannot reach the setpoint by itself.

(Wikipedia, Proportional Integral Derivative controller, n.d.)

3.2.1 Proportional

A proportional controller (P – controller), is a linear feedback control system, which is of the "closed-loop" type. A proportional controller tries to control a system by introducing a correction. This system correction has a proportional relationship with the error of the control system. The error is presented as the difference between the setpoint, and the output/measure variable.

As such, we can mathematically represent a proportional controller like so.

$$P_{\text{out}} = K_p\, e(t)$$

Where:
Pout = Output of the P controller
Kp = Proportional gain
e(t) = error at given time

(Wikipedia, Proportional Integral Derivative controller, n.d.)

3.2.2 Integral

An integral (I) controller is added to a proportional controller when the proportional gain cannot bring an error down to absolute zero. We call this a residual error, which 'resides' after our proportional module failed to eliminate this completely.

This integral correction value is measured in accordance with passed time. Over time, the integral calculates an additional control effect based on the historic cumulative error over the passed time. If the error ultimately is down to 0; the integral will stop growing and remain the same. The proportional error and effect will go down; but this is in turn compensated for by the, now steady, integral.

As such, we can mathematically represent an integral controller like so.

$$c(t) = \frac{1}{T_i} \int e(t)dt$$

Where:
C(t) = Output of the I controller
1 over Ti = The integral time
dt = Indicates that the function is being integrated in respect to a time variable t

(Zak, n.d.)

3.2.3 Derivative

A derivative controller differs in the way it operates compared to the P and I controllers. These two latter named controllers are both feedback controllers, requiring input from the process to calculate the next control variable. A derivative controller is a feedforward controller, and does not need this control output. As such, a “D(erivative)-only controller” does not exist effectively, whereas P- and I-only controllers do exist. (Zak, n.d.)

A derivative controller tries to predict the process outcome or conditions based on the change in the error calculated by the other controllers. It does this to try and prevent oscillations in the system which cause the error to never approach 0.

It calculated its output based on the amount of change over time. The bigger this difference (a good indication for massive oscillations), the more influential the controller output will be.

As such, we can mathematically represent a derivative controller like so.

$$c(t) = T_d \frac{de}{dt}$$

Where:
c(t) = Output of the D controller
Td = Derrivative Time Constant
De = The differential change in the error
Dt = The differential change in the time

(Zak, n.d.)

3.3 Heuristic

Heuristic problem solving is a collective name for everything that can be solved by practical means, though these means not giving any guarantee of being optimal or well suited to the problem compared to, better known, alternatives.

Heuristic approaches often can be done way quicker than following a set of rules associated with the problem. Especially if we take a look at feedback controllers, we can quickly gather that practically tuning a controller compared to calculating one can be considerably quicker.

More often than not however, without reinventing the wheel; it becomes increasingly difficult to take care of things like residual error margins with a heuristic approach, like we see the I(ntegral) controller do.

While heuristic approaches can be inferior to more known methods, it provides a faster and easier way to sketch out the system. In simple control loops, heuristic techniques can be applied quickly. The core principle of heuristic approaches is to keep it very simple. Often heuristics are linked to our psychological decision making, with it trying to embody the style of human thinking.

(Wikipedia, Heuristic, n.d.)

4. Practical application of control loop design into Dynamic Load Scaling program

To see how we could potentially use a control loop design in our project, we can analyse how and if certain configurations previously covered fit inside our model.

4.1 Introduction to the Dynamic Load Scaling program

To control the load induced on the internet by a camera livestream, we have created a program which automatically scales the quality of the outputted stream by influencing how the JPEG encoder encodes separate frames. We can influence this encoder by giving it a value of 0 to 100.

We approach the internet usage by calculating a throughput percentage of our network adapter according to a predefined threshold. This means that with an initial threshold of 2 mbit, 1 mbit gives us 50% allowed usage.

The following variables are available to us to build our control system around:

- Threshold
- Usage percentage
    - Sent/received bytes/s
- Encoder quality setting
- FPS (we can only control this)

4.2 Possible control loop design variants

If we take a look at the nature of this problem, and the available data. We can propose the following ideas to aid in designing our control loop.

These controllers were taken from previously found controllers, not all controllers are included since not all are possible in this use case.

4.2.1 Open-loop/Feedforward proportional controller

In an open loop configuration, we can control our loop based on pre-calculated data; this eliminates the need to do a feedback loop. However, we can only do this under very strict conditions; otherwise the system will spin out of control.

If we know the threshold induced by the configuration of the system, we can figure out which quality setting induces which amount of load. This way, we eliminate the need for a feedback signal since we know and calculated what our output is.

Ideally, this means that we should use an inverse model of our plant to calculate the ideal input parameters. (RICE, n.d.)

Usage of variables

Setpoint: Desired Usage percentage
Process Variable: Encoder quality setting

4.2.2 Closed-loop proportional controller

In a closed-loop configuration we can use the output of the system to directly scale our input. We can do this using a P controller.

Usage of variables

Setpoint: Desired Usage percentage
Measured variable for error: Current Usage percentage
Process Variable: Encoder quality setting

4.2.3 Closed-loop Heuristic controller

With a heuristic controller we can quickly make a working controller without the complexity of setting a proper gain or other variables which calculate a specific response.

Usage of variables

Setpoint: Desired Usage percentage
Measured variable for error: Current Usage percentage
Process Variable: Encoder quality setting

4.3 Proof of Concept of chosen designs

From the three possible scenarios we proposed in previous paragraph, we chose two to continue developing and turn into a POC. This has various reasons, which are described below.

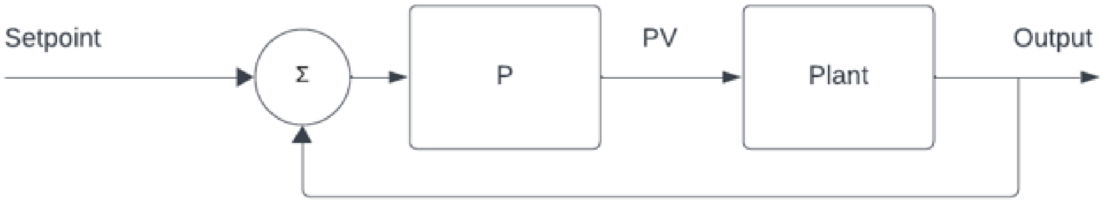
We chose to continue with method *4.2.2* and *4.2.3* which means an open-loop approach falls off the list of options. This is mainly because the use case of implementing a control system is to detect and react to disturbances which are unpredictable.

Unpredictable also means uncalculatable, and the only way an open-loop control system knowns anything; is if we can calculate it beforehand.

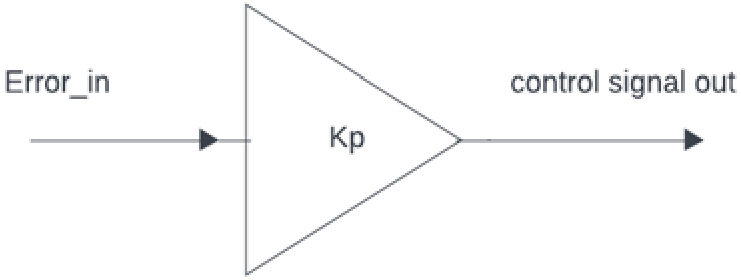
Since the other two methods do not rely on an open-loop approach, and thus do not share the same problem, we can implement them as a proof of concept.

4.4 Closed-loop Proportional controller

We can implement a proportional controller to control our process. We can display our control loop design like so.



The inside of the P Block looks like so.



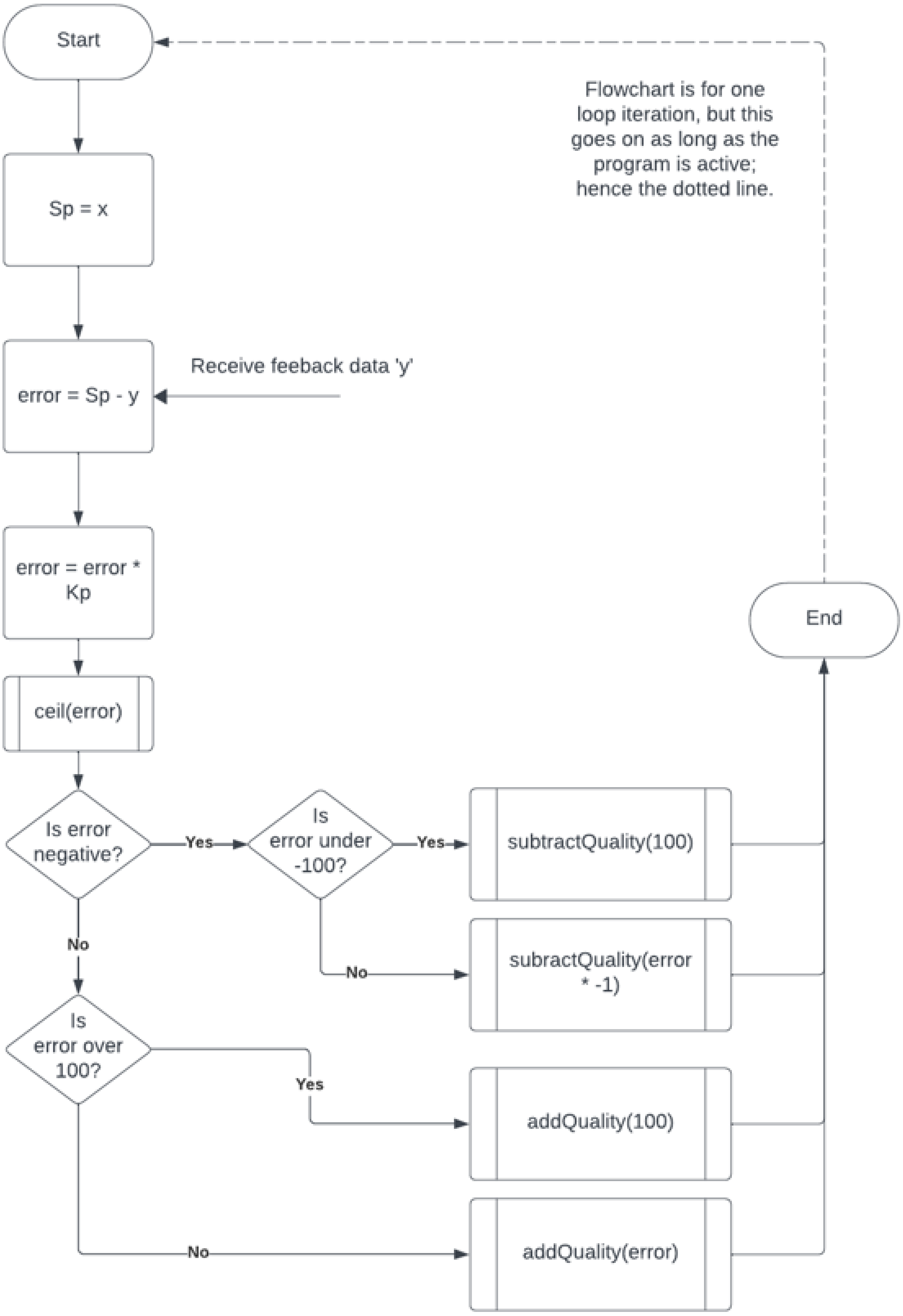
The only thing which rests us to do, is to tune for the optimal gain of the P controller. We are doing this by utilizing a variation of the 'Trial and Error' method; but simultaneously measure the output of our program to a file and graphing it to visually represent its characteristics.

Trial and error is a method of tuning which relies on guessing, and 'being content with' the output. Usually the proportional factor is the main control during tuning. If applicable, the other factors are used for refining. (James Bennet, PID Tuning via Classical Methods, n.d.)

This method was used since there were no I and D factors, which this tuning method barely uses anyways. Furthermore, due to the low input range of our plant, finetuning a controller to a very fine precision will not aid us.

4.4.1 Flowchart of P-controller driven System

To illustrate what happens when the system cycles one loop, we made a flowchart to show the logic involved.



4.4.2 Test setup

The setup for testing and tuning the P controller is very straightforward. If we take note of the adjusting usage percentage of the system (Data ‘y’ in the flowchart), we can plot this against our setpoint (Sp in the flowchart); and see how the controller behaves itself by approaching the setpoint.

We can save these system updates by a rate of 1 entry per second, which immediately also gives us a timeframe to track how long the controller takes to respond.

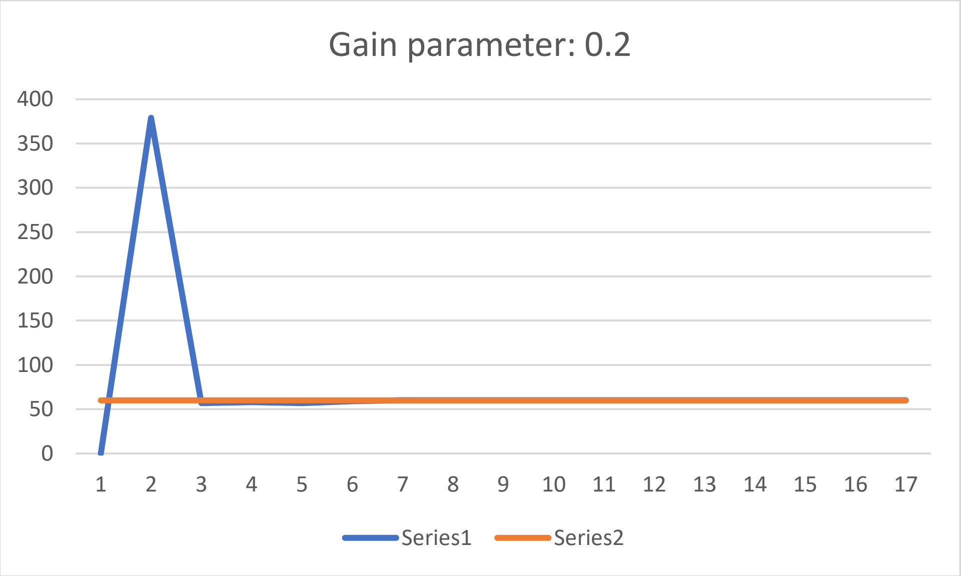
We don’t simulate these results, rather; they are collected from a real P controller which we integrated into our product. The code for this can be found in the code attached to this project.

4.4.3 Test results

All data is recorded in the excel file accompanied with this research. Only the most notable results are discussed here.

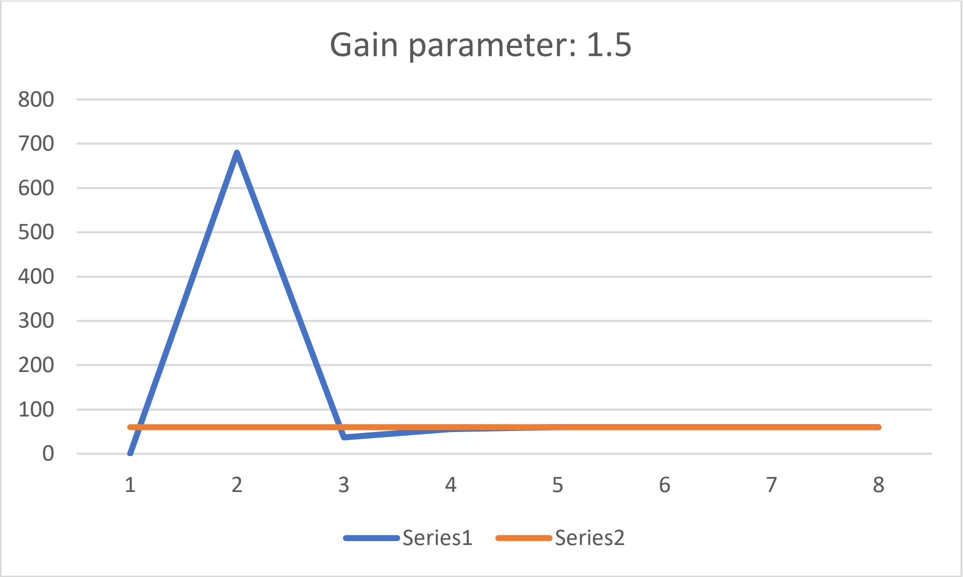
During testing some significant gain parameters arose as being quite effective.

At the lower end of the spectrum, 0.2 proved to be a very stable gain value. Only undershooting very lightly while after being extremely stable.



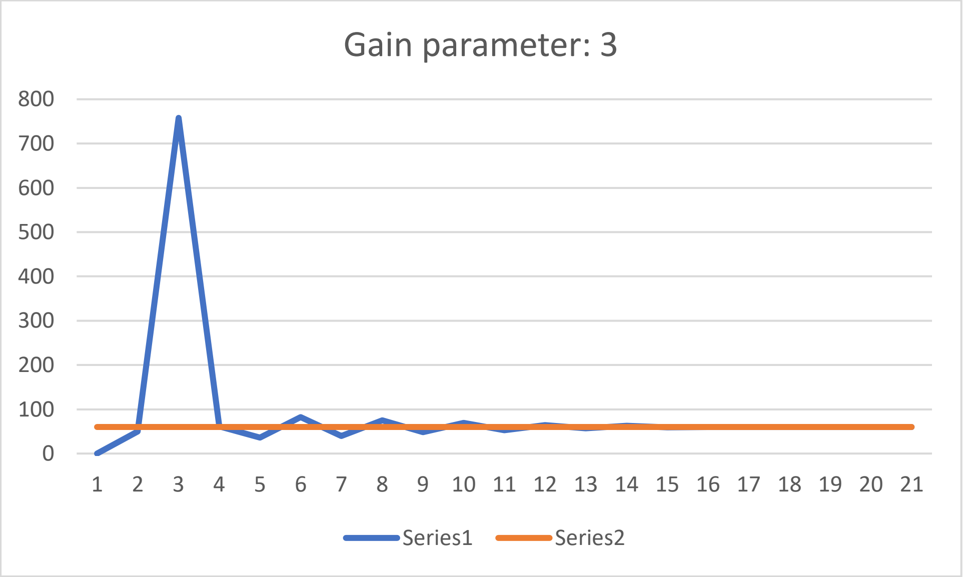
When we compare the raw data to a higher gain parameter, we see that 0.2 does take a while to completely settle on 60 as opposed to other values.

To illustrate this, we could look at a gain parameter of 1.5; which graphs out like this.



We can see here that a gain parameter of 1.5 needs way less time to settle at 60; but does so more 'violently'. This is in line with our research, which indicates that a higher gain results in sharper and heavier adjustments. This can be applied to the previous statement about a 0.2 gain taking more time as well.

We pushed the system to where it starts to oscillate to illustrate the point where the gain parameter gets too big. Light oscillation begins at a gain parameter of 2.5, but becomes too bad to use at 3.



4.4.4 *Optimal gain for use case*

From this data we can derive a gain value which is best for our application. Judging from speed alone, a gain value of 1.5 would be ideal. Since the faster the stream is stabilized, the better.
However, we must also think about the comfort of the user, who is viewing this stream in a first

person perspective. A rapidly changing quality might make this user nauseous, so opting for a slower gain value to damper quickly changing quality would be better.

Therefore, a gain value of 0.2 has the least overshoot compared to other tested values; and is slow enough whereas it would not induce nauseating effects.

4.4.5 Rounding and how it affects P controllers

As seen in the flowchart, or gathered from other sources in this document; our process variable, quality, can only be in integer form. The only solution to this, is to scale the output of the P controller accordingly to desired input of the plant. In our case, that means rounding a float; after which we cast it to an integer.

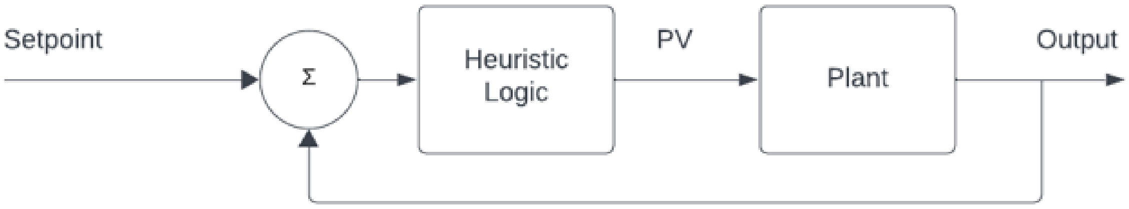
This is often done to accommodate for different Servo/motor constraints; which differ from manufacturer to manufacturer. Usually, they are mapped; though rounding provides the same effect if we have been working with percentiles all along.

The output functionality of the P controller will be affected, but more slight than it seems at first glance. A P controller will still be able to function fine with rounded values, the only part of the system which suffers is the plant. Since the plant can't adjust the process variable as accurately as it could potentially do with a float.

Increasing the max size of the process variable or casting its implementation to a float will aid the system, but clearly is not needed to make a P controller function properly.

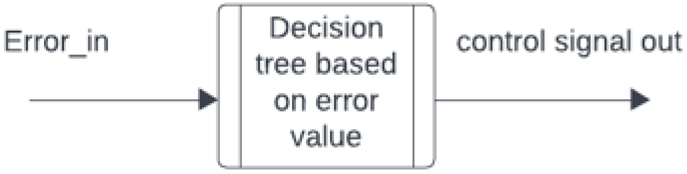
4.5 Closed-loop Heuristic controller

We can implement a heuristic controller to control our project, in a diagram; this looks like the following.



This looks very similar to the previously proposed P controller, but do note that while the surrounding architecture is the same; the way incoming error data is handled is now completely different.

Without making it unreadable, the heuristic logic controller can be displayed like so.

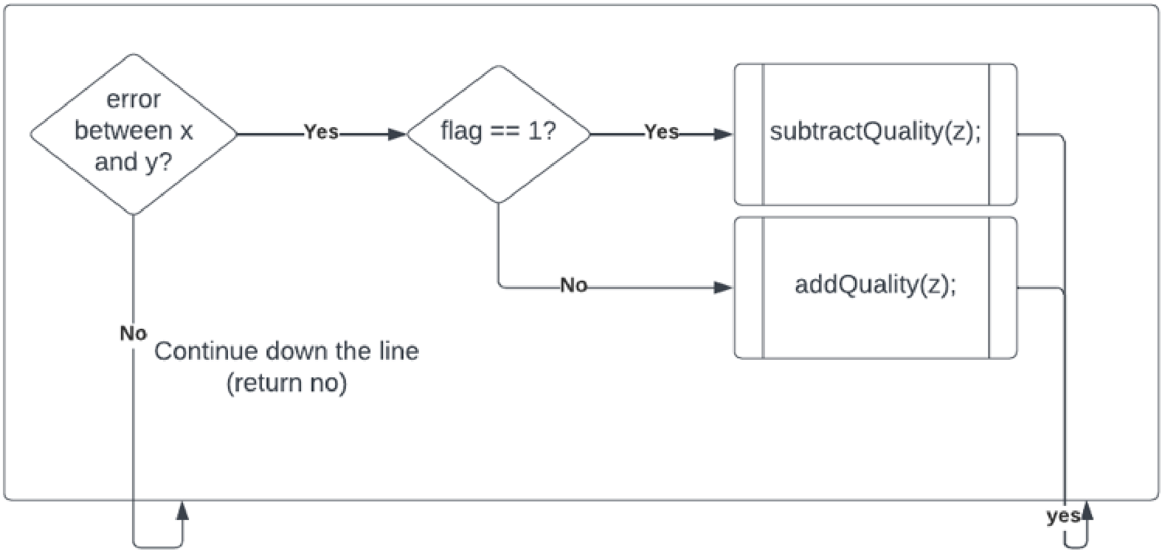


This decision tree relies heavily on if statements and therefore differs massively compared to our P controller in the way it handles data.

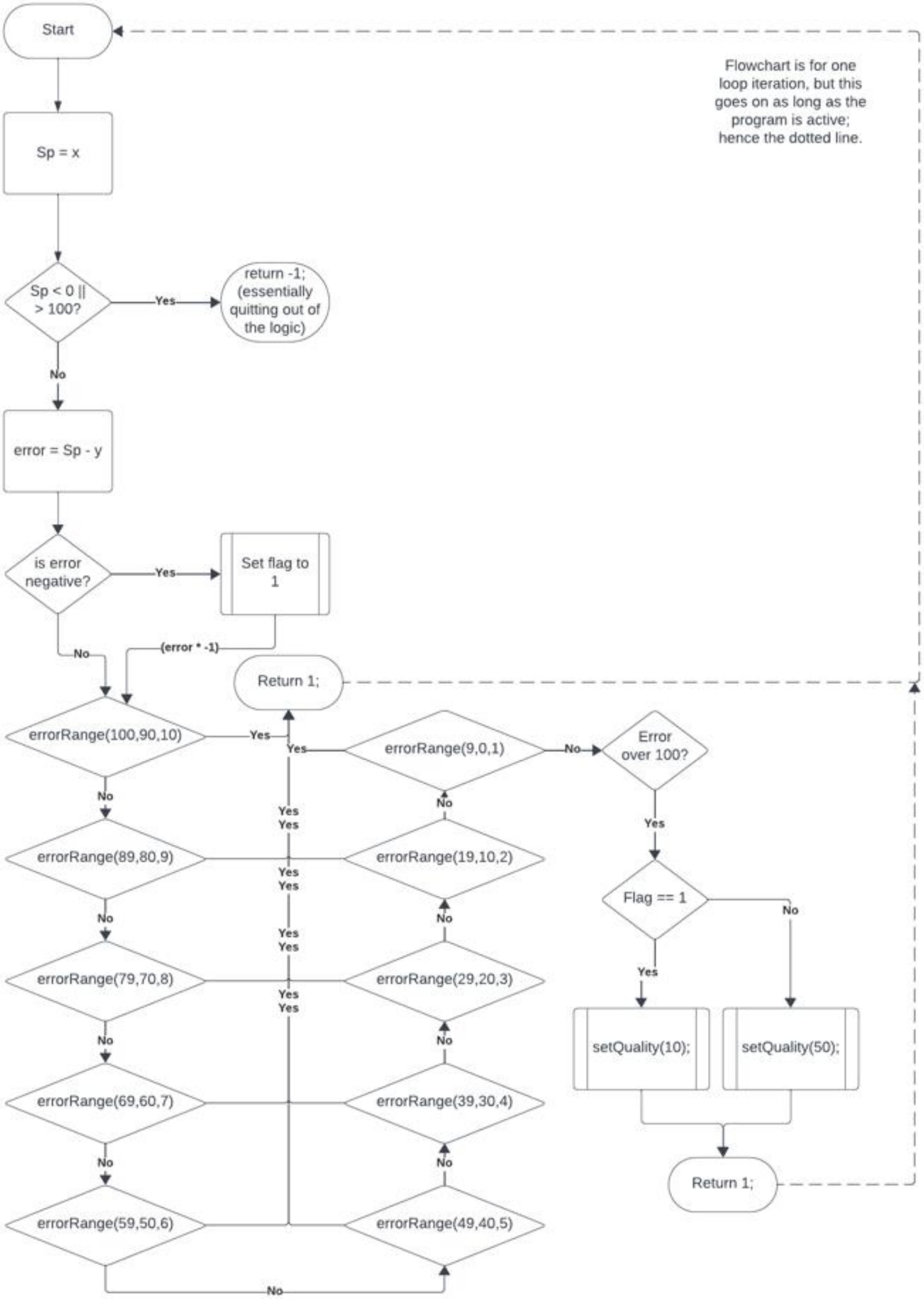
4.5.1 Flowchart of Heuristic controller driven system

To illustrate how our heuristic controller behaves, we can much better display this in a flowchart than in a block scheme.

To limit size and improve readability, the following logic is henceforth addressed as a function called 'errorRange(x,y,z)'. With the max min range being x and y respectively; and the correctional number being z.



This makes the complete flowchart look like follows.



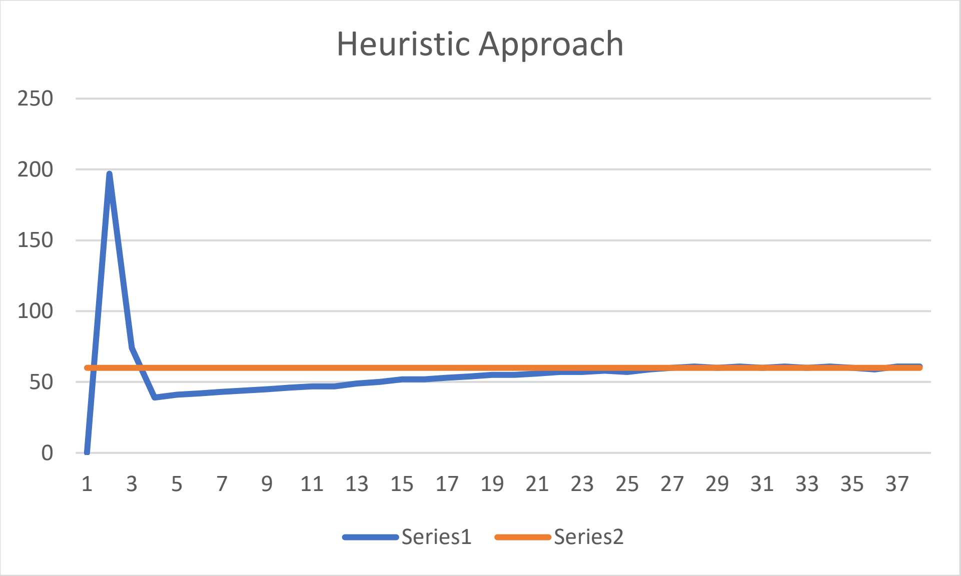
4.5.2 Test setup

The test setup for this heuristic controller relies on the same data to measure its response. We have a setpoint and a process output which reflects this setpoint. An error is calculated over this and the system is adjusted to the logic provided by the controller.

Naturally, the only thing that has changed is the type of controller, this is also why we can compare the test results from the P controller to the test results provided below. Since there has been no change to the system, except the controller logic.

We don't simulate these results, rather; they are collected from a real heuristic controller which we integrated into our product. The code for this can be found in the code attached to this project.

4.5.3 Test results



Taking a look at the test results for the heuristic controller, we can see that it is quite slow compared to the PID. Moreover, it oscillates continuously and reaches the setpoint only momentarily.

4.6 Ideal choice according to testing

After considering the test outcome of both controllers we can safely say some things about the characteristics of both these controllers. This also allows us to label which is better for our situation.

A heuristic approach, while seemingly easier, turns out to be less practical when compared to a more calculated approach.

We can see this clearly in the fact that our heuristic controller has the tendency to oscillate. This is not weird, considering we have hardcoded an oscillation into our controller; by having no state where nothing happens to the process variable. In other words, there is no situation where the process variable gets adjusted by 0; indicating a stable state/reaching the setpoint.

Comparing this to the P controller, it clearly has states where if a proper gain has been chosen it has reached a stable state; and where the interference with the plant is 0. Apart from that, if an improper gain has been chosen. It is very easy to adjust, compared to essentially re-writing the entire heuristic controller.

Concluding, we can confidently say that a P controller with a gain set to 0.2 is the superior choice of feedback controller for this situation.

5. Conclusion

To conclude our research, lets first look back at our main research question. We described it as follows:

Which controller logic is the best for our application?

In short, we did not only find that there is more to a control loop than just its controller; it was found that how this controller is setup in a control loop matters just as much. Furthermore, different controller logics were proposed, while some were even tested in a small-scale test to see how effective they were in our real use case.

From the data gathered from the tests, and the knowledge from the research; we can comfortably state that choosing from the control systems spoken about in this research.
A Closed-Loop system with a P Controller boasting a 0.2 as Kp (gain) Is the most effective setup for our application.

6. Bibliography

Feedback Loops. (n.d.). Retrieved from Carleton edu: https://serc.carleton.edu/introgeo/models/loops.html

James Bennet, A. B. (n.d.). *LibreTexts Engineering*. Retrieved from LibreTexts: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_(Woolf)/09%3A_Proportional-Integral-Derivative_(PID)_Control/9.03%3A_PID_Tuning_via_Classical_Methods#:~:text=The%20trial%20and%20error%20tuni

James Bennet, A. B. (n.d.). *PID Tuning via Classical Methods*. Retrieved January 21, 2024, from libretexts: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_(Woolf)/09%3A_Proportional-Integral-Derivative_(PID)_Control/9.03%3A_PID_Tuning_via_Classical_Methods#:~:text=The%20trial%20and%20error%20tuni

Porwal, S. (2021). *What is the difference between a first order and a second order control system?* Retrieved from quora: https://qr.ae/pK6Szh

RICE, U. (n.d.). *Feedback Control*. Retrieved January 21, 2024, from Clear Rice: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_(Woolf)/09%3A_Proportional-Integral-Derivative_(PID)_Control/9.03%3A_PID_Tuning_via_Classical_Methods#:~:text=The%20trial%20and%20error%20tuni

S, S. K. (2021). *What is the difference between a first order and a second order control system?* Retrieved from quora: https://qr.ae/pK6JRc

Wikipedia. (n.d.). *Bang-Bang control*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Bang%E2%80%93bang_control

Wikipedia. (n.d.). *Classical control theory*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Classical_control_theory

Wikipedia. (n.d.). *Control Loop*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Control_loop

Wikipedia. (n.d.). *Heuristic*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Heuristic

Wikipedia. (n.d.). *Proportional Integral Derivative controller*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Proportional%E2%80%93integral%E2%80%93derivative_controller

Zak, J. B. (n.d.). *P, I, D, PI, PD, and PID control*. Retrieved from libretexts: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Chemical_Process_Dynamics_and_Controls_(Woolf)/09%3A_Proportional-Integral-Derivative_(PID)_Control/9.02%3A_P%2C_I%2C_D%2C_PI%2C_PD%2C_and_PID_control

7. Additional sources

PID Loop Mapping/Setting Output Range. (2021, January 24). Arduino Forum. https://forum.arduino.cc/t/pid-loop-mapping-setting-output-range/692871/4

PID output to PWM input. (2012, November 18). Arduino Forum. https://forum.arduino.cc/t/pid-output-to-pwm-input/130128

Explained, D.-. P. (2020, July 23). *PID controller explained*. PID Explained. https://pidexplained.com/pid-controller-explained/

PID tuning | How to tune a PID controller - RealPars. (n.d.). https://www.realpars.com/blog/pid-tuning

Can PWM be used for PID controlling. . . (n.d.). Electrical Engineering Stack Exchange. https://electronics.stackexchange.com/questions/121081/can-pwm-be-used-for-pid-controlling

Motor Control using PWM and PID | Stratify Labs. (2013, October 15). Stratify Labs. https://blog.stratifylabs.dev/device/2013-10-15-Motor-Control-using-PWM-and-PID/