ENEL 869 – Report

Ball height controller

Sithija Anjana Wijesinghe – 200506111

Apr-2025

Contents

Introduction

In engineering and automation, feedback control is a fundamental concept and plays a critical role in making systems stable and reliable. Feedback control enables a system to automatically correct itself in response to changes. There are two main components of a feedback-controlled system, the mechanism that is being regulated and the feedback received from the system.

System overview

This project uses a stream of air to control the altitude of a floating ball in a tube. The ball is placed in a tube with a fan attached to the top running at a constant speed. At the bottom of the tube, a controllable door allows the opening to be opened to different levels to control the stream of air. By controlling the stream of air, the height that the ball would float is regulated. An IR distance sensor is installed at the bottom of the tube, this is used as feedback to the system to control the system.

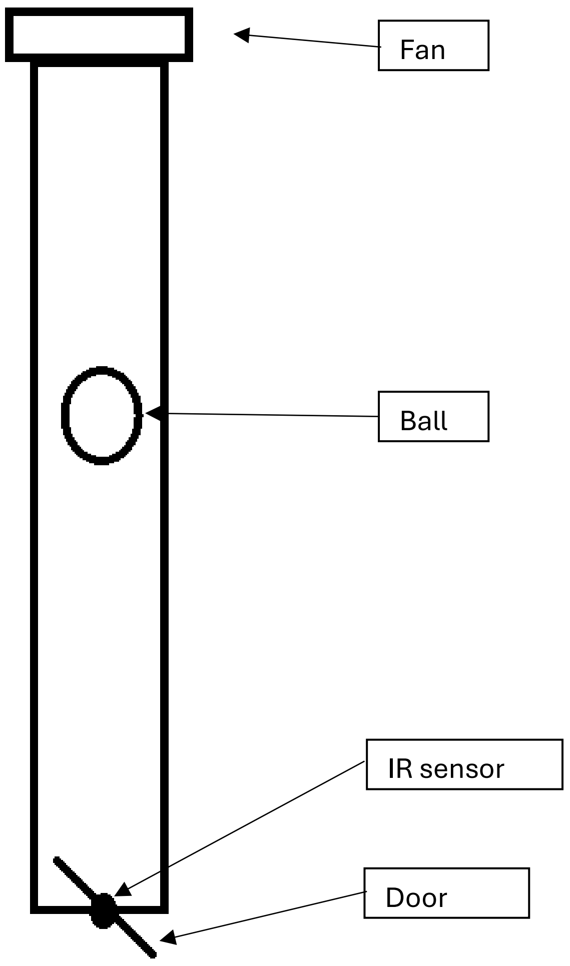


Fig. 1 - Hardware diagram

Requirements

- The system shall control the height of the floating ball by regulating the stream of flowing air.
- The system shall use feedback from the IR sensor to determine the height of the floating ball.
- The system should be able to maintain the height of the ball within 5cm of the given height.

Useful scenarios

The concept being explored is controlling the air flow. This can be made into a great tool by scaling up to simulate drone altitude control in different wind conditions. The simulation could be used to test stability and control algorithms before testing out in the real-world.

How it would work,

- The drone would be placed in the tube
- The drone would start flying
- The air control system would change the air flow to suit different scenarios and how the flight control system works in those situations will be observed.

Functional components

- Tube enclosure with door attached to the bottom and fan to the top.
- Servo motor connected to the door to control the level of the opening.
- IR sensor attached to the bottom of the tube to determine the distance to the floating ball.
- MCBSTM32EXL as the controller
- PID control system software to regulate the air flow by controlling the servo to change the level of the opening using the data from the IR sensor as the feedback.

Design alternatives

The fan can be replaced with a variable speed control fan. This would enable two types of systems to be designed,

1. The door at the bottom of the will be removed leaving just an opening. Controlling the fan speed can now be used to control the air flow and this the height of the floating ball.
2. Both the door and fan speed are controlled. Given that we have two modes of controlling the air flow, we could control the height of the ball with better precision.

Methodology

The ball height controller was successfully implemented using a STM32F103ZG microcontroller to control the system using PID (Proportional-Integral-Derivative) control.

The hardware diagram (Fig.1) shows the hardware of the device and circuit diagram section captures the electrical connections. The control loop sections shows the control diagram for the system, the software diagram section shows how the software functions, and the configurations sections captures the main configurations applied to the microcontroller. The implementation section goes over the steps carried out during the project.

Circuit diagram

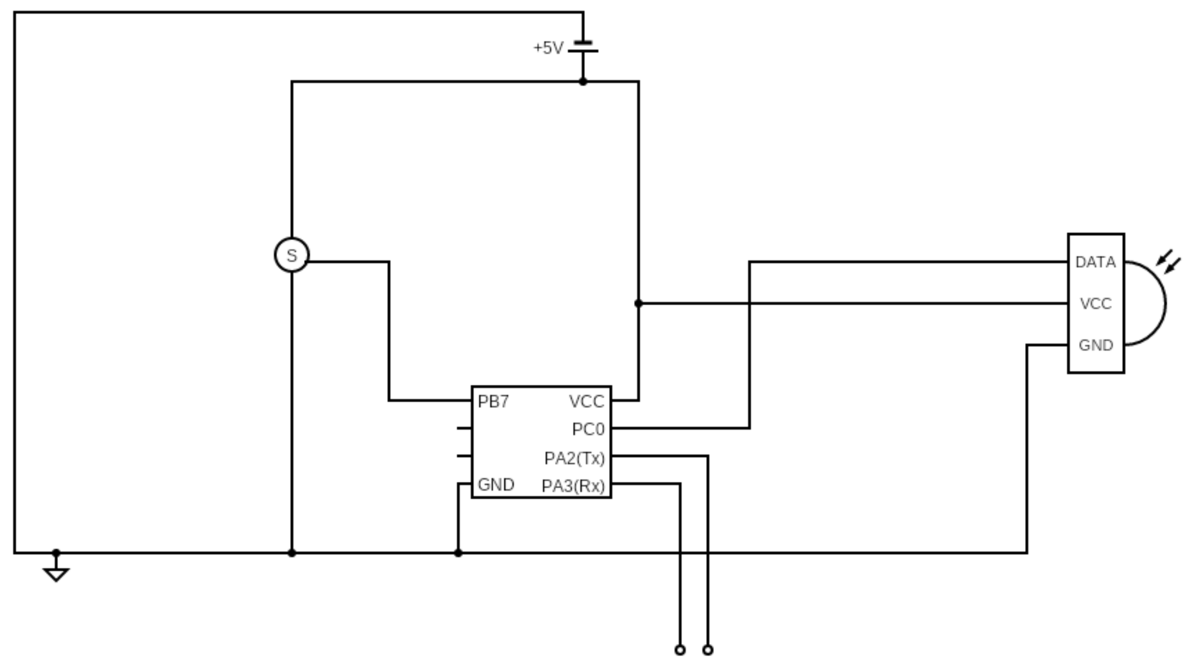


Fig. 2 - Circuit diagram

The circuit diagram of the system is captured in Fig. 2. The IR sensor was connected to the STM32F103ZG at pin PC0 as an analog input. The servo motor was connected to pin PB7 as a PWM output. Pins PA2 and PA3 were used as transmitter and receiver pins for USART2 to send and receive serial data.

Control loop

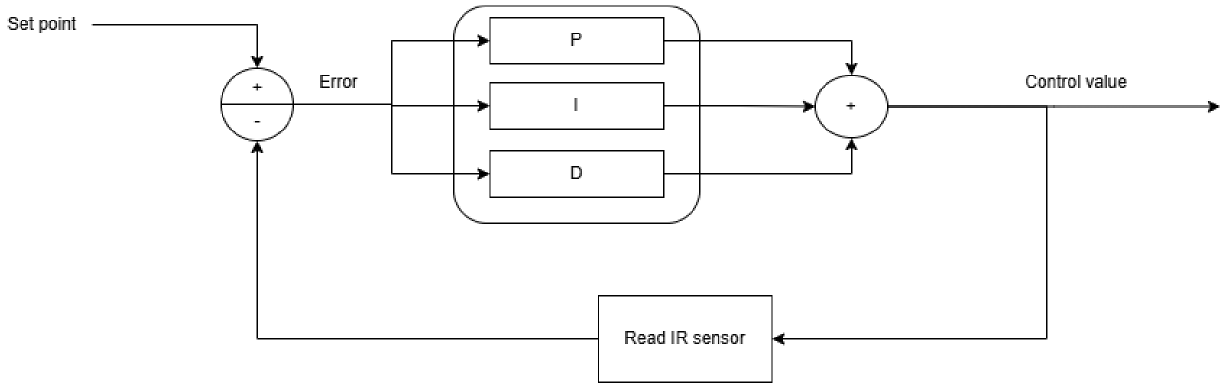
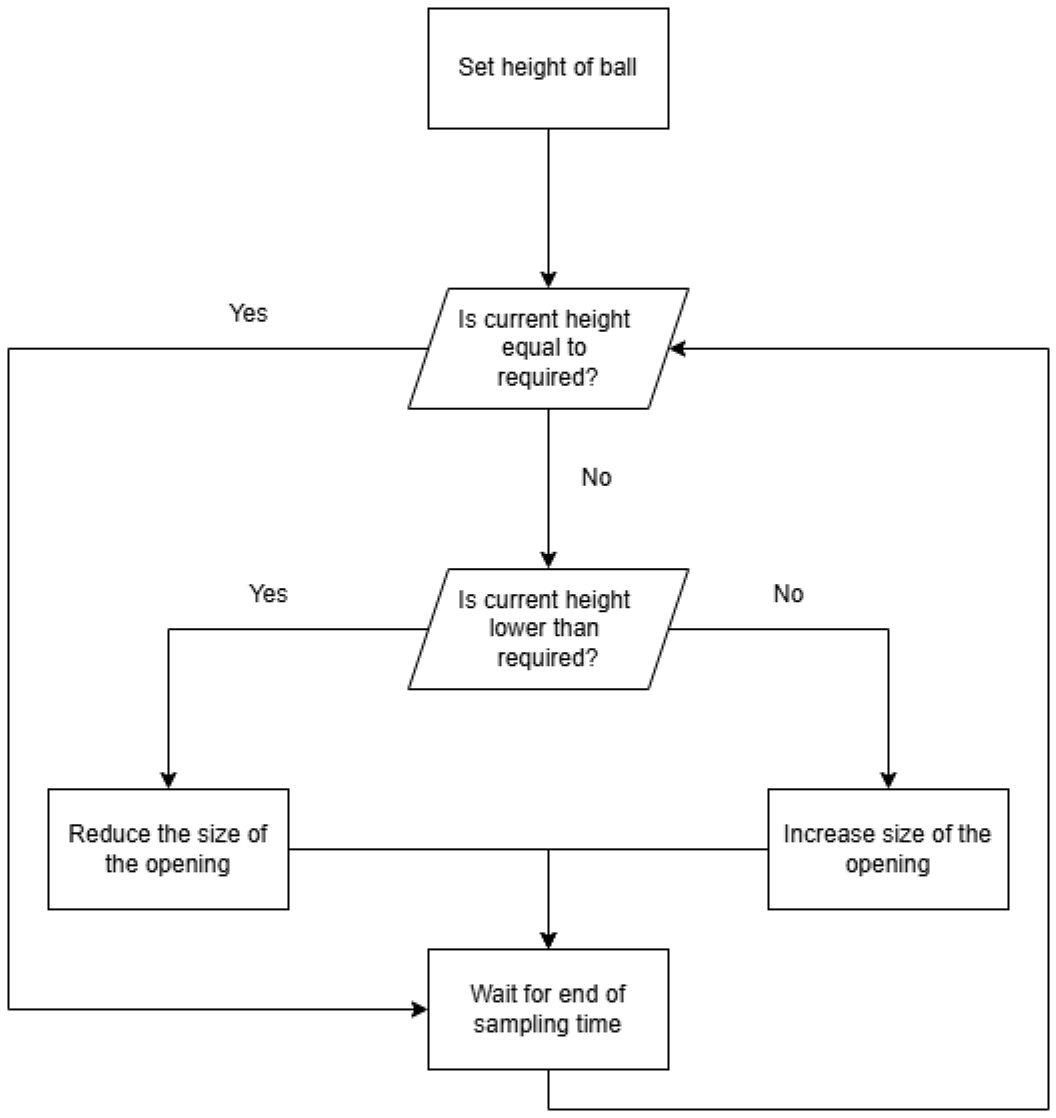


Fig. 3 – Control loop

A PID (Proportional – Integral – Derivative) controller was implemented to control the ball height. Fig. 3 shows a representation of the PID controller.

Using the feedback from the IR sensor, the error between the required point and the current value is calculated which is fed into the PID calculator. The output of the PID calculation is used to adjust the servo motor.

Software diagram



Set height of ball
Yes
Is current height equal to required?
No
Yes
Is current height lower than required?
No
Reduce the size of the opening
Increase size of the opening
Wait for end of sampling time

Fig. 4 - Software diagram

Fig. 4 shows the high level software diagram. As the required height of the ball is given (input using serial communication), the IR sensor is used to read the current height of the ball. By calculating the difference between the required and the current height, it can be determined if the ball needs to be raised or lowered. If the current height is lower, then the ball needs to be raised and the opening size is increased. If the current height is higher, then the ball needs to be lowered and the opening size is decreased. The low level software diagram for the implementation is shown in *Appendix – Low level software diagram* and *Appendix – PID value limiter* and *Appendix – servo value limiter* capture the limiter in place.

C was used to program the STM32F103ZG using the Keil microvision IDE.

Configurations

The following configurations were applied to the STM32F103ZG

Type	Configuration
Timer 1	Prescaler 72, ARR 0
Timer 4	Prescaler 36, ARR 40000
PB7 – Servo controller (PWM)	Output 50MHz, alternate function output push-pull
PC0 – IR input (ADC)	Floating input
USART2 – CLI	PA2 as Tx and PA3 as Rx
ADC1	Scan mode
USART2 – Rx interrupt	NVIC->ISER[1] : NVIC_ISER_SETENA_6
ADC1 - interrupt	NVIC->ISER[0] : NVIC_ISER_SETENA_18

Other:

Tube number 2 was used initially and then switched to tube number 1 as the servo motor in tube 2 failed.

Implementation

The hardware and the electrical connections of the device were already in place at the start of the project.

The first step was to configure and test out the servo motor. The STM32 was programmed to set the pin connected to the servo motor as a PWM output. The servo required a period of 20ms therefore an ARR of 40,000 was used with a prescaler of 36 for the timer attached to the relevant pin, which in this case was timer 4 with PB7. By visual observation, the highest and the lowest values the servo required were determined. A software limiter was implemented as to not go beyond these values. This is to ensure that the servo is not damaged by turning it over the maximum it can handle safely while giving enough range for the software to work in. Software block diagram for the servo value limiter is shown in Appendix's Servo value limiter section.

The second step was to setup a timer that can be used to configure the sampling time. Timer 1 was used for this feature and was run at the maximum speed of 72MHz.

The third step was to setup the serial output. This would enable to visualize the running values of the system. The serial interface was initially setup to transmit data. This was carried out using USART2 with pins PA2 and PA3. A baudrate of 115200 was used for communication.

The fourth step was to setup the IR sensor. The pin PC0 was setup as an analog floating point input and ADC1 was configured to convert the analog input received by the PC0 to a discrete value that can be used. The serial communication was used to print the values onto a terminal to observe them. An analysis was performed to understand how the IR sensor responds to the change in height of the ball inside the tube (explained in experimentation and observations section). ADC interrupts were turned on and configured to keep converting and reading values when called.

The fifth step was to implement version tracking. Git post-commit hook was used to capture the git hash. This is used as the version for that build.

The sixth step was to improve the serial communication and add features that can be used to manually control the value of the servo motor. Methods to change the level of the servo motor and setting other required values using serial communication was implemented. Interrupts with serial communication was also enabled so that the system can function while paying attention to any user input.

The seventh step was to implement a PID control system. There are few different version of PID control (similar equations with minor differences between each other). Three different versions were implemented and the equation given below represents the final implementation which performed best.

$$P = Kp * \varepsilon_{input}(t)$$

$$\varepsilon_{integral}(t) = \varepsilon_{integral}(t-1) + \left(\varepsilon_{input}(t) * r\right)$$

$$I = Ki * \varepsilon_{integral}(t)$$

$$D = Kd * \frac{\varepsilon_{input}(t) - \varepsilon_{input}(t-1)}{r}$$

$$PID = P + I + D$$

Legend for the equation above,

PID: PID output value

P: proportional value

Kp: proportional constant

I: integral value

Ki: integral constant

D: Derivative value

Kd: derivative constant

ε: error

r: sampling frequency

A PID limiter was implemented (software block diagram shown in Appendix's PID limiter section) to limit the maximum amount the PID can affect the system to regulate the output calculated.

The eighth step was to implement moving filters for the IR sensor values. It was noted that there is noise in the IR sensor values at times. In order to overcome this a two moving filters were implemented, a moving mean average filter and a moving median average filter. With experimentation the moving median average filter performed better and was included in the final system (more details in the experimentation and observations section). A buffer was implemented to which the IR values were added. For the mean filter, the mean of this buffer was calculated and for the median filter, the buffer was sorted by value and the middle value was selected.

The final step was to tune the PID controller to control the height of the ball. This was done by setting the P constant first and tuning it to make the controller be able to get the ball to oscillate around the required level. Next I constant was tuned to assist in the system being able to stop the ball at a specific level but this can adds some noise that will cause the ball the level when its close to the required level and to get the level closer to the required level the D constant finetuned.

Features implemented

- Set PWM value to control servo motor
    - Hard limit the value to a given range
- Read the analog input from the IR sensor and convert to digital using the ADC
    - Use interrupts to read the IR value
- Configure USART2 to create a functional serial communication
    - Print values to the serial interface
    - Read values from the serial interface

    - Use interrupts to interrupt the program when a key is pressed
- Moving average filter
    - Moving mean average filter
    - Moving median average filter
- PID controller
- Versioning
    - Using git post commit hook to obtain git hash and use it as the version

Issues faced during implementation

The Ulink-Me device is used to connect the development board to a PC to flash and debug the device. The Ulink-Me device was not detected by Keil. Troubleshooting the issues led to the observation that the cable used to connect the Ulink-Me to the PC only contained connections for power and ground and not data (only 2 connections of the 4 in USB-A was connected. The fix was to switch the cable and a cable with all 4 connections were used which fixed the issue.

Came across the error "00008000H - 00008447H" when flashing the program onto the development board. The fix was to adjust the settings under utilities of options for the target in Keil.

Came across the issue ".\Objects\project.axf: error: L6031U: Could not open scatter description file .\Objects\project.sct: No such file or directory". The fix was to enable "use memory layout" from target dialog in Keil.

Came across the issue "Not enough information to list image symbols" when compiling the program. The C files were not added to the relevant directories and adding them fixed it.

The servo motor malfunctioned during the PID tuning phase when tube 2 was used. This wasn't a software bug and it was a hardware malfunction. Therefore the tube was switched to tube 1 and the PID controller was retuned for it.

Experimentation and observations

IR values for height analysis

In order to configure the system, it is required to understand how the feedback functions (what values it can take where). For this, the IR values seen for different heights in cm of the ball was noted. This was done by using a program to read 50 IR values once a key was pressed. The ball was moved from 2cm to 62cm taking readings at every 2cm inside the tube. The Fig. 5 shows the resulting plot for the median of the 50 samples per height.

NOTE: the analysis was done with tube 2 while the final implementation was done with tube 1 as the servo on tube 2 malfunctioned.

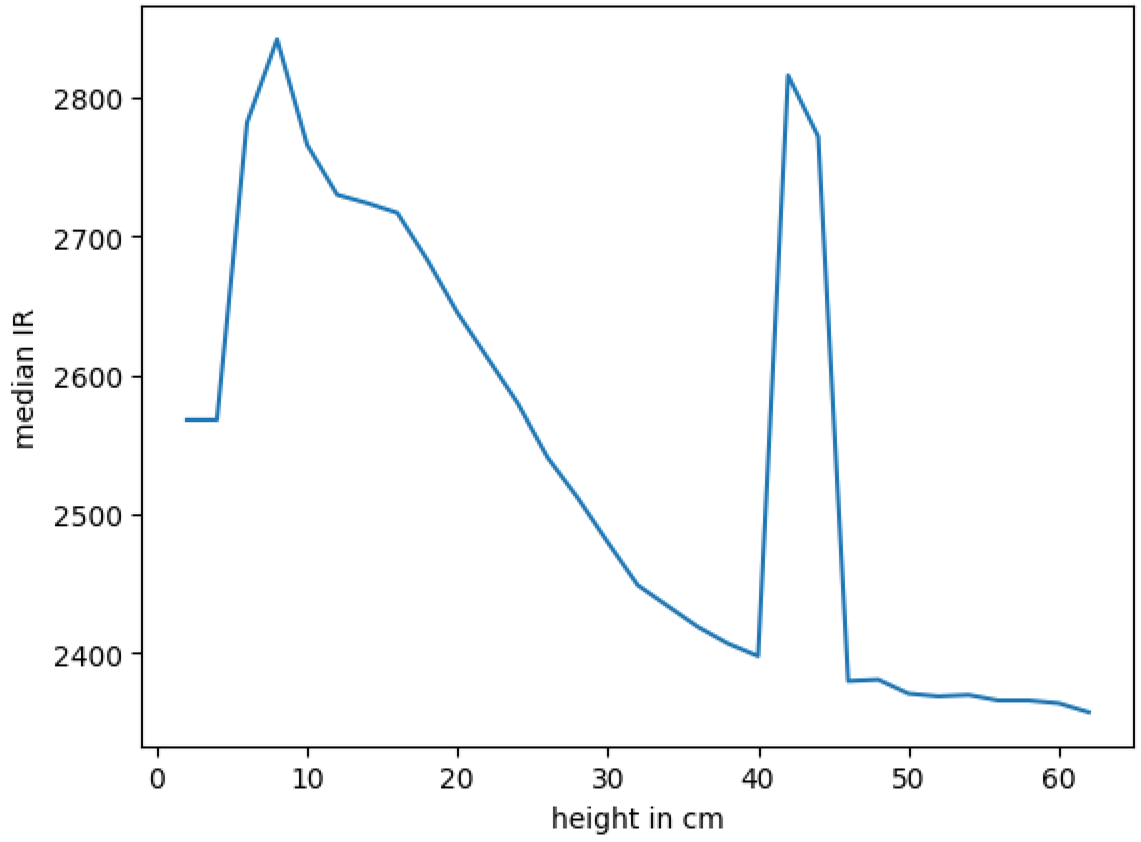


Fig. 5 - Median IR value seen for different heights of the ball

A sharp increase in IR values is seen up until 10 cm where it peaks to about 2900. Then the IR values drop with increasing height until 40 cm. There is a sharp increase in IR values after this point causing a large peak of about 2850 and it drops back down at about 46 cm. Form here on the IR values gradually decrease with the lowest at about 2300.

The mode and max plots for the same dataset followed the same pattern as the plot for the median as shown above. The min value plot for the sample however was seen to have a completely different pattern as shown below.

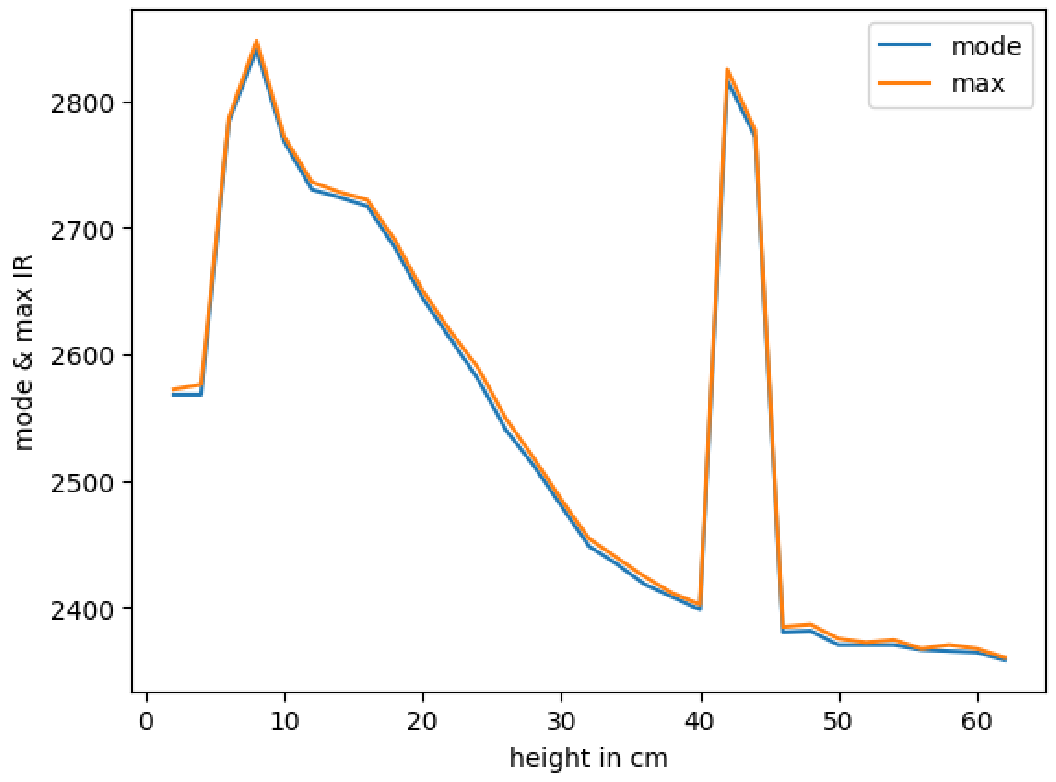


Fig. 6 - Mode and maximum IR values seen for different heights of the ball

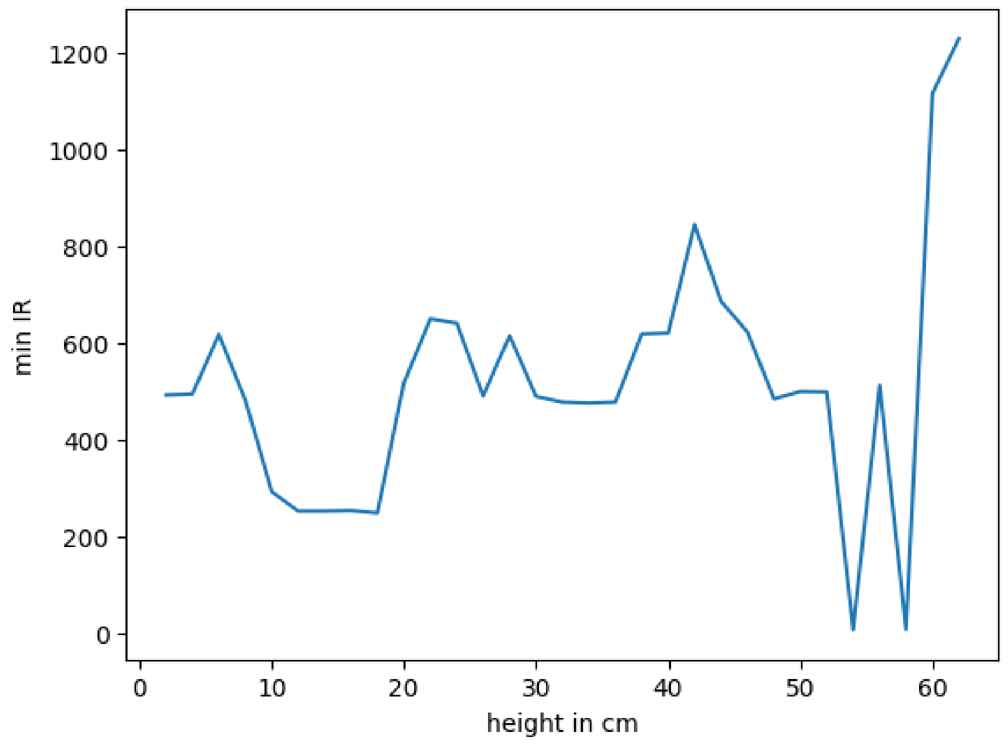


Fig. 7 - Minimum IR values seen for different heights of the ball

By observing the plots, it can be seen that the median, mode, and maximum provides a better representation of the height of the ball than the minimum. The hypothesis for the noisy minimum values is that it is caused by noise where and the noise causes the IR value to drop significantly. The decision was made to use the test mean and median values using a moving filter so that the best value that represents the height of the ball would be used.

Two moving filters were implemented, a mean moving filter and a median moving filter. These were tested and the response of the ball height controller was observed. As the first step the buffer size was configured. It was noted that a buffer size of 5 gave the best and most stable results for both approaches as decreasing it didn’t show any improvement to using the raw IR values without a filter while increasing it caused more noise to added to the system making it harder to control. Therefore the buffer was set to 5.

Comparing the median to the mean moving filter, it was observed that the median provided a more stable output that the mean. The hypothesis is that the mean would be affected more by noise (low IR values that were noted earlier) than the median filter.

By analyzing the plot further, it can be determined that the goldilocks zone to control the ball is between 10 cm and 40 cm. Lower than 10 cm is challenging because of the drastic change in the very small range. As there is a peak in the readings between 40 cm and 50 cm, controlling the ball in this area is again going to challenging.

Therefore, a median filter with a buffer of 5 was used primarily and the ball was to be controlled only withing the range of 10 cm and 40 cm.

Ball height control

Once the ball height control system was implemented using PID, the control system was finetuned by optimizing the following parameters:

1. P multiplier
2. I multiplier
3. D multiplier
4. Sampling time
5. Max change for PID per iteration

There were two sets of parameters that worked well for this system.

First set performed best at required height (IR value) 1750 while it was able to work relatively well for values between 1950 and 1700:

1. P = 2
2. I = 0.1
3. D = 1
4. Sampling time = 50ms
5. PID limit = 200

Second set performed best at required height (IR value) 1750 while it was also able to work relatively well for values between 1800 and 1700.

1. P = 1
2. I = 0.1
3. D = 0.5
4. Sampling time = 100ms
5. PID limit = 50

The second set was set to have a slower response so it was able to keep the ball in the required height with less oscillations but it had a hard time bringing the ball to the right height therefore had trouble.

The best heights the ball were taken using the first set of parameters as they were overall more stable than the second set. The images below show the results from this.

The system was able to hold the ball within ±2 cm at its best and was seen to be able to hold the ball within ±3.5cm most of the time.

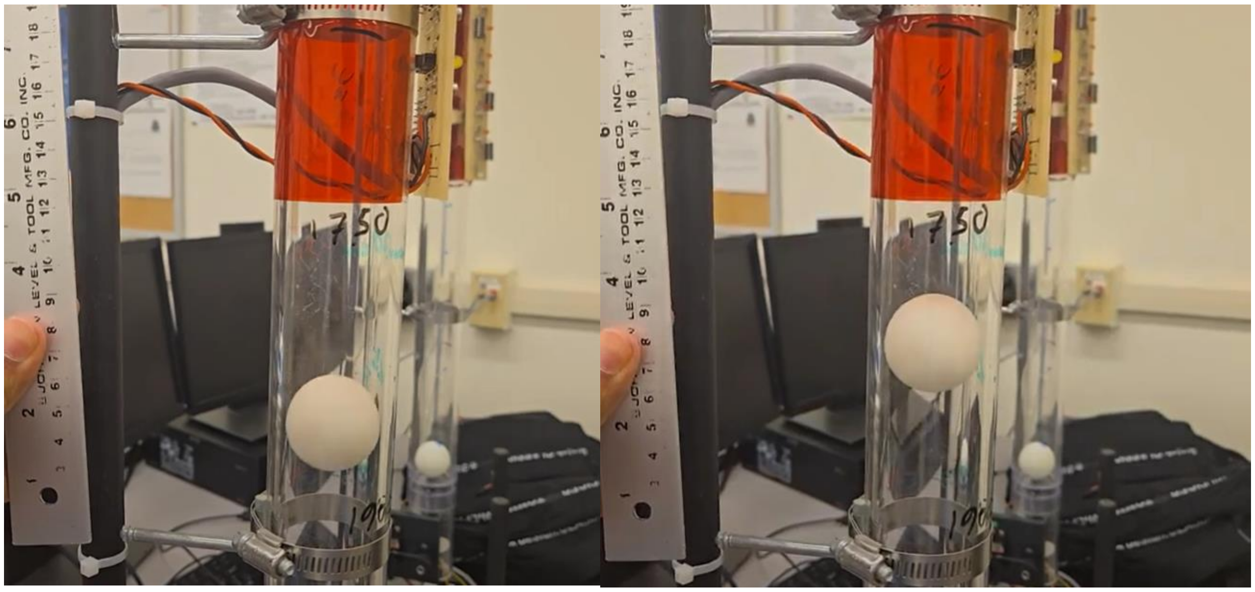


Fig. 8 - Best case: Left low level (at 4 cm on the ruler) and right high level (at 8 cm on the ruler)

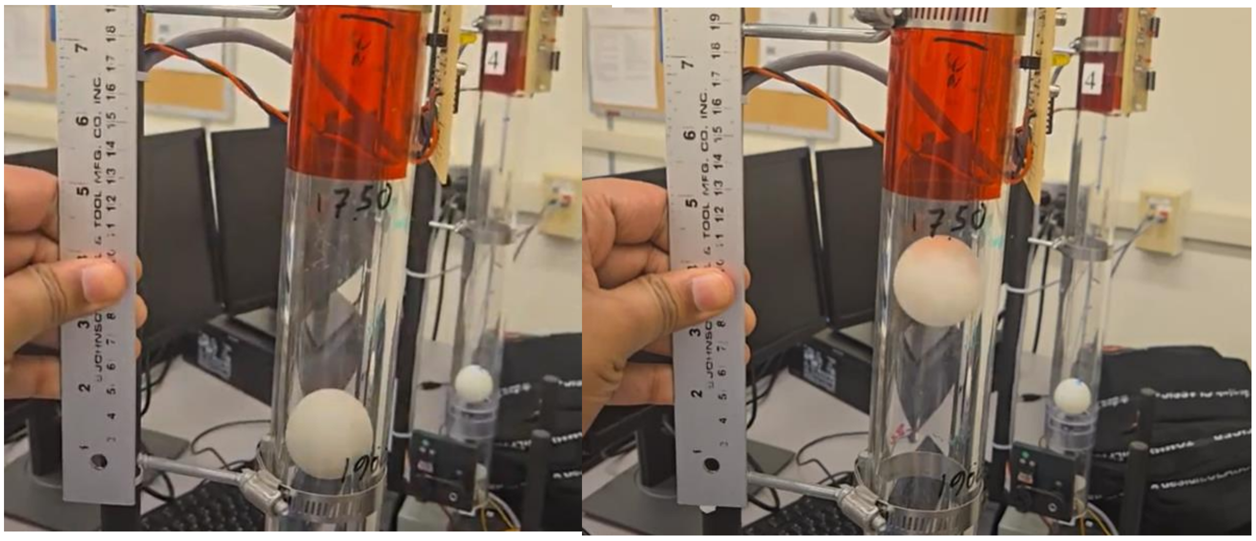


Fig. 9 - Regular case: Left low point (at 3cm on the ruler) and right high point (at 10cm on the ruler)

Visual observation

It was observed that the ball cannot be controlled using PID or manually at heights less than 10 cm as the value was very close to the ball making it hard to hold the ball in a given position.

Also, holding the ball in place using PID or manually at heights over 40 cm was challenging except for a few select positions (it is unclear why the ball can be held at some points but not all in this range). At locations higher than 40 cm, the ball tends to accelerate at a higher rate with a very small change to the servo value making it challenging to control.

Due to the aerodynamics inside the tube, a slow response rate made it harder for the controller to control ball and therefore a higher response rate was preferred. This was because it was seen that the ball tended to move a lot for one level change in the value at some points and move only a little at other points. Because the height changes of the ball was high at some points and low at others for the same change in the value, a slower response rate will not be able to respond to the changes on time to move the ball towards the required level. Also, very high response rates also caused issues as the PID controller kept overcorrecting and not allowing time for the ball to settle down.

Command line interface interaction

A fully functional CLI interaction system was developed. This contains the functionality to control all the required aspects of the system by inputting the relevant data.

The system has three distinct modes, closed loop (PID controlled and fully automatic), open loop (control the servo motor manually using the CLI), and debug mode (where key values are set).

On startup, the system will boot into the open loop mode and print out the available functionalities onto the screen. The running values of the system will always be printed out onto the screen if in open or closed loop modes.

Open loop mode

Available features:

Key	**Action**
i (lowercase i)	Increase servo value by 1
I (uppercase I)	Increase servo value by 10
d (lowercase d)	Decrease servo value by 1
D (uppercase D)	Decrease servo value by 10
q (lowercase q)	Enter debug mode
e (lowercase e)	Enter PID mode

Pressing any other key will not have any effect.

The running values of the system will be printed out (updated).

Debug mode

Available features:

Key	**Action**
r (lowercase r)	Do nothing and go back to open loop mode
i (lowercase i)	Increase servo value by 1
I (uppercase I)	Increase servo value by 10
d (lowercase d)	Decrease servo value by 1
D (uppercase D)	Decrease servo value by 10
h (lowercase h)	Set maximum value the servo can take (Max servo limit)
l (lowercase l)	Set minimum value the servo can take (Min servo limit)
s (lowercase s)	Set current value of the servo
1 (number 1)	Set P multiplier in PID
2 (number 2)	Set I multiplier in PID
3 (number 3)	Set D multiplier in PID
a (lowercase a)	Set required height the ball should float in (Required IR value)
w (lowercase w)	Set sampling time
z (lowercase z)	Set PID limit (maximum change the PID can have each iteration)

Only one function can used at a given time. Once that is done, the system will go back to open loop mode. I.e. if the debug menu is entered and then “a” is pressed, the system will request a new height that should be achieved and once that is given and “enter” is pressed, the system will update the variables and go back into open loop mode. If another change is also required, then the debug menu should be entered again for it.

The running values of the system will be printed out (updated).

Closed loop mode

The system will starting calculating the PID value with the given sampling rate and other variables set. The servo value will be adjusted accordingly and the system will attempt to hold the ball at the given height.

Pressing any key while in this mode will put the system into open loop mode.

Conclusion

The project investigated the implementation of a controller to control the height of a ball in a tube by regulating the airflow. The airflow was regulated by changing the level of the opening of the valve. A PID controller was implemented to use the feedback from an IR sensor to control the level of the ball. A fully functional serial interface was implemented to be able to configure every required aspect of the system.

The project used a STM32F103ZG microcontroller on a mcbstm32exl development board. C was used for bare metal programming using the Keil microvision IDE.

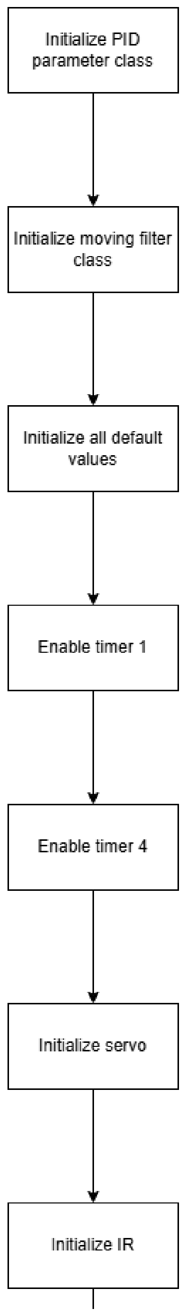
The project was successfully completed by holding the level of the ball within 2 cm at the best and 3.5 cm regularly from the given height. This exceeds the goal set during the proposal of 5 cm.

Appendix

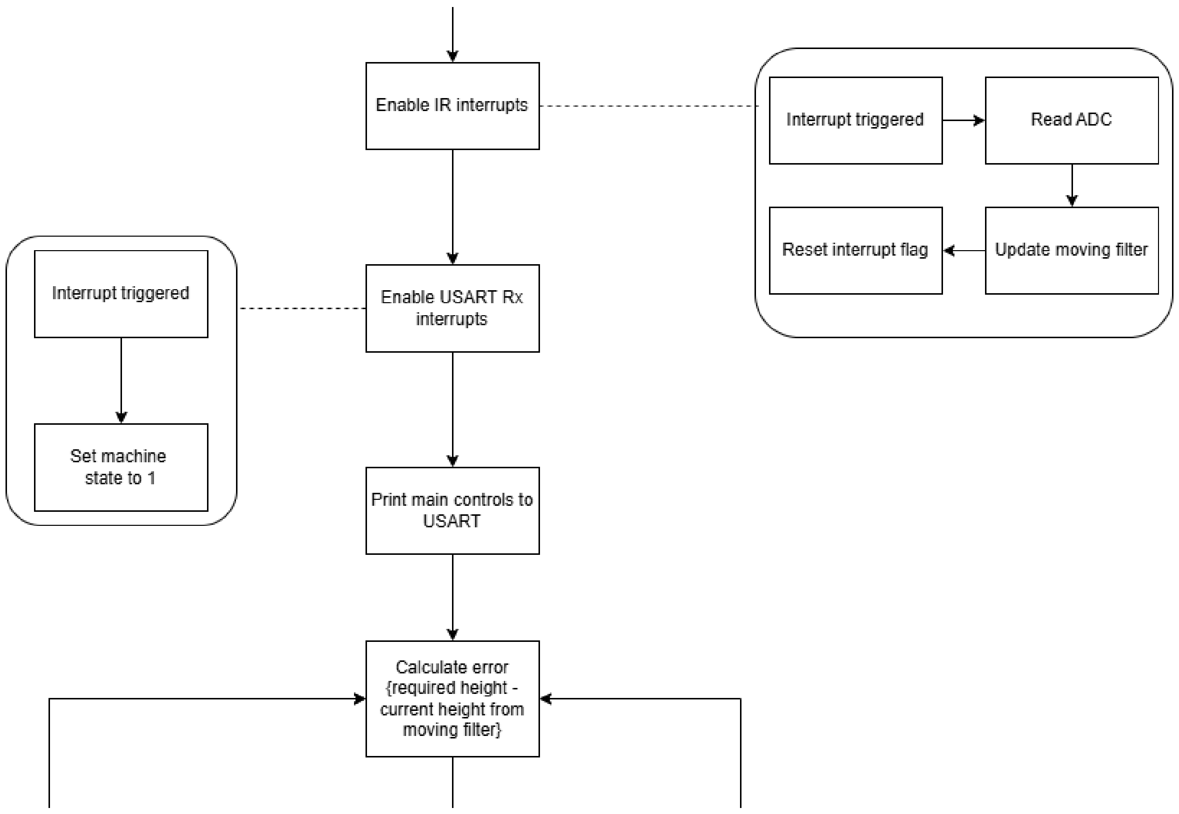
Low level software diagram

The below images show the low level software diagram for the program.

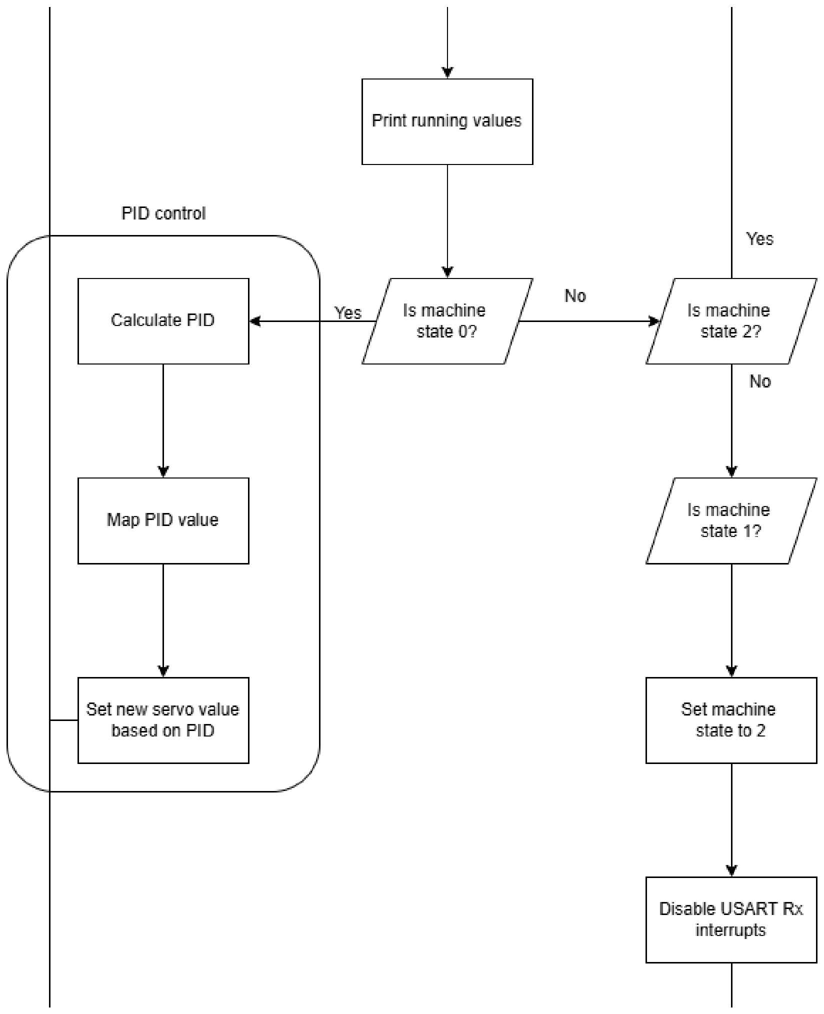
NOTE: the diagram is separated into parts as the whole image does not fit as is.


Initialize PID parameter class
Initialize moving filter class
Initialize all default values
Enable timer 1
Enable timer 4
Initialize servo
Initialize IR

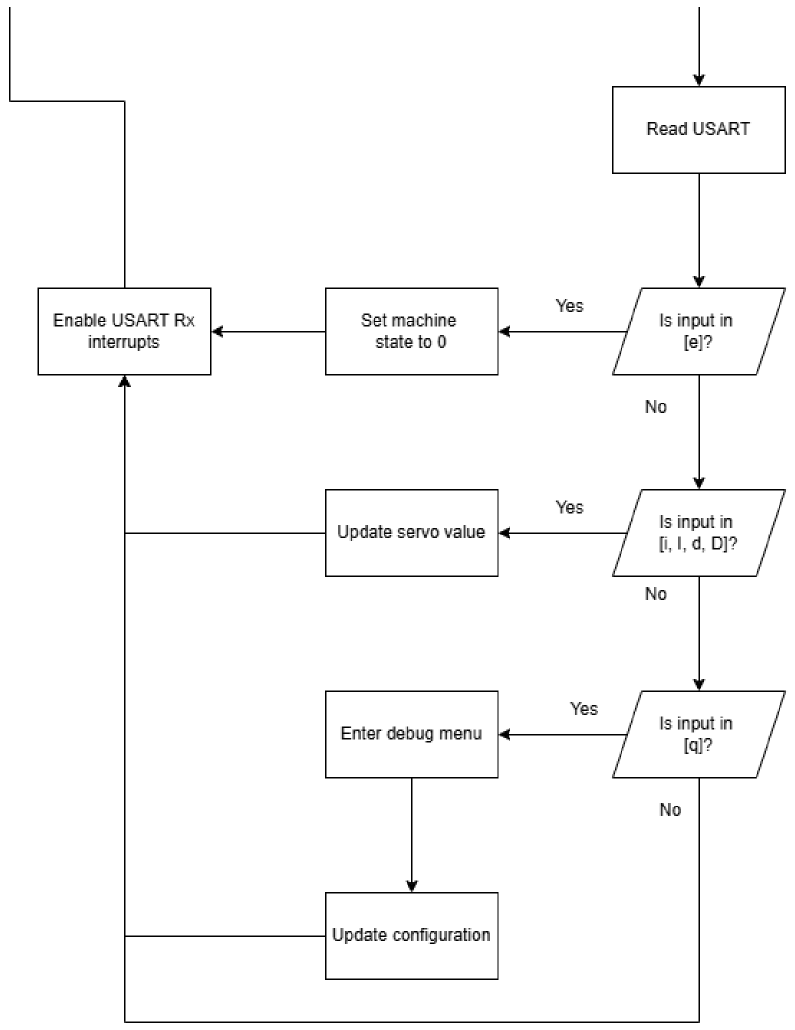
Appendix Fig 1 – Low level software diagram - part 1



Appendix Fig 1 – Low level software diagram - part 2

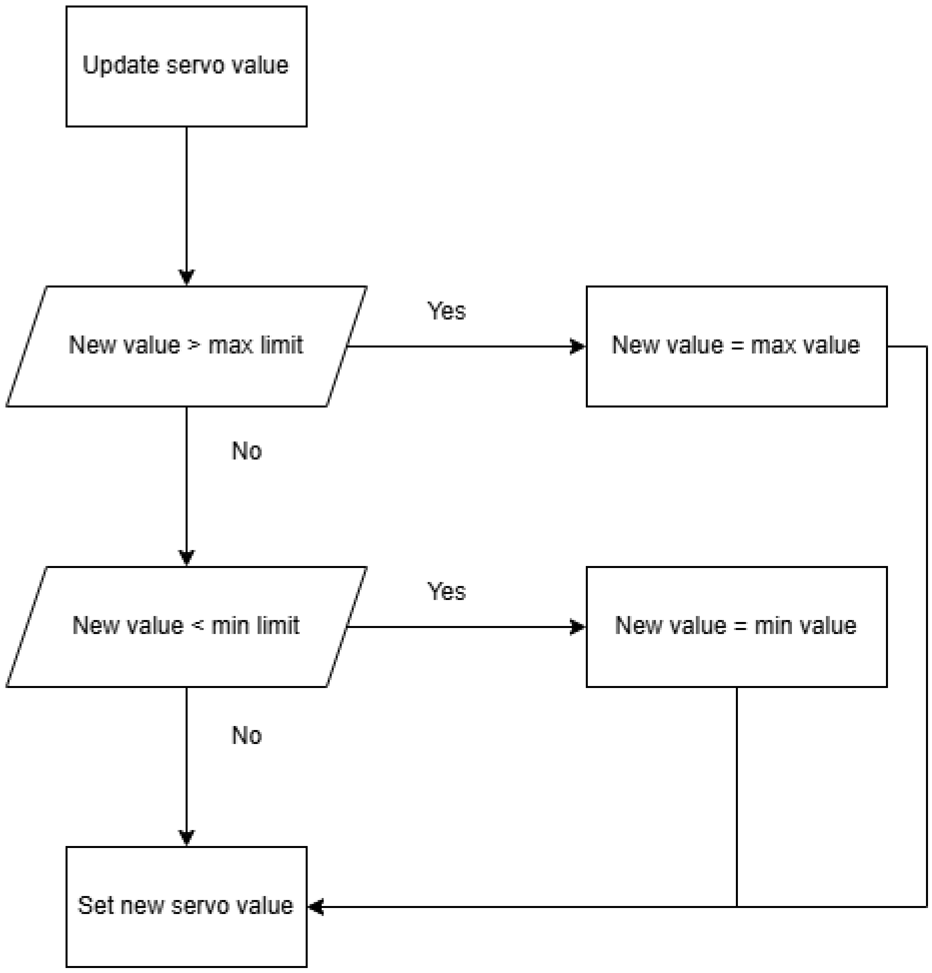


Appendix Fig 1 – Low level software diagram – part 3



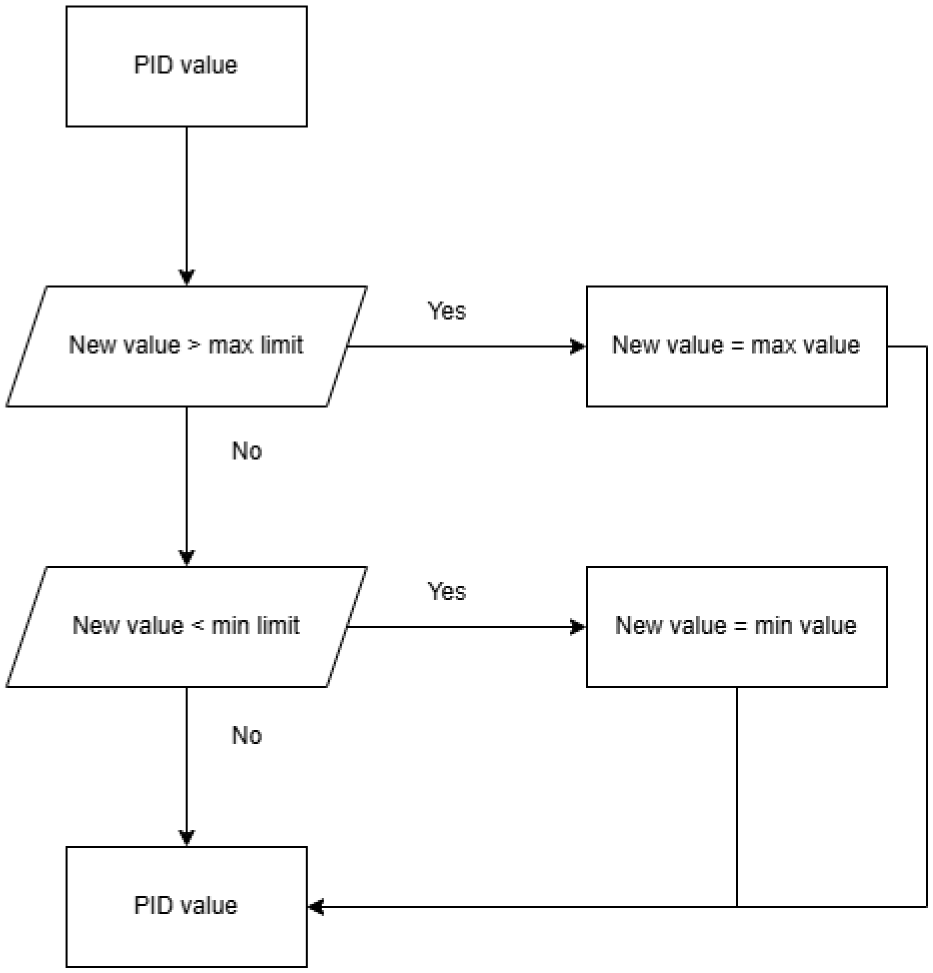
Appendix Fig 1 – Low level software diagram – part 4

Servo value limiter



Appendix Fig. 2 – Servo value limiter software diagram.

PID value limiter



Appendix Fig. 3 – PID limiter software diagram.