

**Table of Contents**

[Introduction 3](#_Toc133508790)

[Automatic control theory 3](#_Toc133508791)

[Application and scope 3](#_Toc133508792)

[Simplifications and assumptions 4](#_Toc133508793)

[Control Theory 4](#_Toc133508794)

[PID Controller 4](#_Toc133508795)

[Proportional control 5](#_Toc133508796)

[Integrating control 5](#_Toc133508797)

[Derivative control 5](#_Toc133508798)

[System Design 6](#_Toc133508799)

[Requirements 6](#_Toc133508800)

[Hardware 6](#_Toc133508801)

[1. Frame 6](#_Toc133508802)

[2. Arduino 7](#_Toc133508803)

[3. Servo Motor 7](#_Toc133508804)

[4. Sensor(HC-SR04) 8](#_Toc133508805)

[Controller design 8](#_Toc133508806)

[Implementation 8](#_Toc133508807)

[Electrical circuit 9](#_Toc133508808)

[Arduino programming 10](#_Toc133508809)

[Section-1 10](#_Toc133508810)

[Section-2 10](#_Toc133508811)

[Section-3 11](#_Toc133508812)

[Section-4 11](#_Toc133508813)

[Section-5 12](#_Toc133508814)

[Result 12](#_Toc133508815)

[1. Do It Yourself (DIY) Adaptation 12](#_Toc133508816)

[DIY materials and adaptations used in the project 12](#_Toc133508817)

[Measurement taken of the DIY frame 13](#_Toc133508818)

[2. Tuning Values 13](#_Toc133508819)

[Future Improvement 14](#_Toc133508820)

[Conclusion 15](#_Toc133508821)

[Reference List 16](#_Toc133508822)

**Introduction**

Mathematically speaking, control theory is a branch of study that deals with dynamical systems. To achieve the desired reaction and outcome of a system, or to stabilize an unstable system, is the primary goal of building a control model. Additionally, control theory will be applied to an engineering discipline, notably mechatronics, in this project.

The Centrifugal Governor was invented during the Industrial Revolution to regulate fuel injection and hence the speed of a steam engine, marking the beginning of the widespread usage of automatic controls. However, James Clerk Maxwell didn't establish the notion of automatic control until the second part of the 19th century. As a result of theoretical and technological improvements, control theory is now applicable to a wide range of domains and specialties. As a result, the engineering discipline finds considerable value in the study of control theory. This thesis's goal is to show how a theoretical concept gets materialized in a physical model and how variations and flaws in the implementation lead to distinct physical responses.

## **Automatic control theory**

Although the subfields of automated control theory are diverse, the underlying theory and analytical approaches are rather stable. The automatic control design method chosen, however, may change depending on the specifics of the work at hand (Glad and Ljung, 2000). The project makes use of a proportional-integral-derivative controller, or PID controller for short. The proportionate component mitigates the impact of disruptions without wholly cancelling them out. However, while the essential element mitigates disturbance effects, it may introduce instability into the system. The derivative controller element does boost stability margins, but it does so at the expense of a rise in measurement error (Goodwin, Graebe and Salgado, 2001).

## **Application and scope**

Ball balancing platform will be used to implement, monitor, and analyze the aforementioned control design approach for the sake of study. The goal is to keep a ball from rolling off a flat surface by compensating for factors like wind and other external forces.

In addition to traditional building materials, three distinct pieces of equipment will be used to bring the edifice to life, generate data, and obtain the appropriate motion output. In the thesis, the platform is a resistive touch panel that relays information about the ball's location to a microcontroller. A pair of servo motors are controlled by a microprocessor, which in turn determines the ball's speed and acceleration based on the input data. In response to a control signal from a microcontroller, servo motors cause an angular displacement by altering their position. The study will make use of the information gathered from both the physical model and the simulation of the system. In order to test, we will manually perturb the ball's position and then observe the effect on the system's steady-state stability. This thesis attempts to address the following issues:

* Can a ball balancing platform be built with sufficient performance using only linear control techniques?
* What, if anything, accounts for the discrepancies between theoretical models and experimental results?
* To what extent do practical results differ from theoretical predictions when attempting to stabilize a system that is inherently unstable or only partially stable?

## **Simplifications and assumptions**

To obtain the equations of motion for a ball on a platform, the following simplifications must be assumed:

The ball is completely spherical and homogeneous

There is no friction between the ball and the platform

The ball does not experience any upward translation with respect to the platform.

# **Control Theory**

As aforementioned, a PID controller will be utilized to control the system. In this section its different parts will be explained in detail.

## **PID Controller**

Given a feedback loop mechanism, the PID, or proportional-integral-derivative, controller will automatically apply corrections. The PID allows for the continuous monitoring and stabilisation of control-modulated systems. In the controller, the error e(t) is what is being measured. where the deviation from a reference point is used to calculate the inaccuracy. However, the controller's individual components can be built in any order (P, PI, PD, etc.). As we'll see below, equation describes the regulated output signal:

## **Proportional control**

In order to decrease steady-state error, a proportional controller boosts the system's proportional gain. When a proportional gain, denoted by a constant called KP, is introduced, the proportional response is modified so that the steady-state error is directly proportional to KP (Margalith and Mergler, 1982). In addition, the equation for the proportional term, or signal output, is as follows:

*UP* (*t*) = *KP e*(*t*)

## **Integrating control**

The integral controller scales in inverse proportion to the error's magnitude and, ideally, its finite duration in time. The cumulative error over a finite time period corrects the offset introduced into the system in the past. As a result, the inaccuracy at steady state disappears (Bennet, 1993). However, KI, the integrated control, may worsen the system's response by, say, crippling it in terms of response and leading to transitory or oscillatory behavior. Equation describes the signal produced by the integral term:

*uI* (*t*) = *KI*

## **Derivative control**

The slope of the time series of error responses can be used to identify the derivative controller. The derivative gain, KD, is then multiplied by the error's slope (Tan, Liu and Turk, 2011). The equation defining the derivative term is:

# **System Design**

## **Requirements**

Development of the previously illustrated system, in Fig. 3.1, is restricted by a series of requirements, both on mechanical construction and performance. These requirements are based on previous attempts to construct a similar system [6]. Following are the requirement necessary for the system design.

* Allow for the platform to incline 20 degrees in every direction
* Minimize rotational play around the z-axis of the platform
* Settling time (5%) < 3s (Ts < 3)
* Overshoot < 5% (M < 0.05)
* Static error for a step input < 5 mm

# **Hardware**

## **Frame**

The frame of this project is entirely made of cork sheet. First we came up with a design consisting of a single lever bridge/ramp. The ball will balance on the bridge and one end of the bridge is connected to our servo motor through a moveable arm. In the other end we placed our ultrasonic sensor to collect our feedback. The Arduino will gather necessary feedback from the sensor and give signals to the motor to compensate the changes to keep the ball balanced.

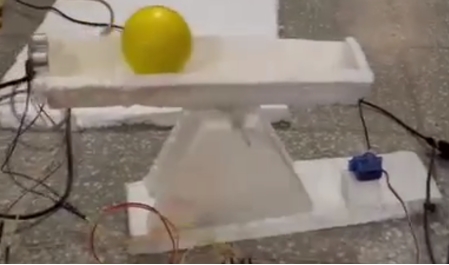


Fig 1. DIY Frame Implemented for this project

## **Arduino**

The Arduino is used in this project as the main micro-controller. It reads the inputs from the ultrasonic sensor to the measure the position of the ball in our bridge. Then it performs the PID calculations to compute the control signals that are required to keep the ball balanced on the specified position. The PID algorithm takes the error signal as input and produces the control signal as output, which is then used to drive the servo motor to adjust the position of the ball (Arduino, 2015).



Fig 2. Arduino Uno

## **Servo Motor**

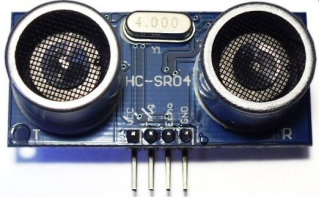
The servo motor's purpose in a ball balancing PID controller is to provide precise position control of the platform on which the ball rests to maintain the ball's equilibrium at a specific position, allowing the system to balance the ball in real-time.



Fig 3. Servo Motor

## **Sensor(HC-SR04)**

The purpose of an ultrasonic sensor in this project is to provide feedback on the position of the ball, which is used by the PID controller to calculate the error signal and make real-time adjustments to maintain the ball's balance (Straete et al., 1998).

Fig 4. HC-SR04 Sensor

## **Controller design**

The system is only moderately stable due to the presence of a double pole at the starting point. When a feedback loop is provided, the system's step response is completely dominated by non-decaying oscillatory components. KP, KI, and KD determine the necessary control output to provide a stable system response. Iteratively establishing values for Kp, Ki, and Kd in MATLAB ensures that all system criteria are met, leading to system stability. The static error, in theory, is controlled automatically by the system because of the built-in integrator.

As a result, the PID controller may not need an integrating component.

# **Implementation**

The desired reaction and actual output of the systems differ greatly between theoretical and actual physical execution. In order to get the intended response, it is necessary to incorporate and adjust for previously ignored characteristics and simplifications. Here we detail the approach taken while writing the code for the physical system model. It is important to remember that the procedure is identical for all axes of rotation.

## **Electrical circuit**

As follows below in Fig. 2 is the electrical circuit and the respective components. Besides wiring the components are the Arduino Uno microprocessor, two servo motors, a resistive touch panel, a small breadboard and a voltage step up component. Complete list of components in Table 1.1. The Arduino and touch panel is powered through USB and the servo motors from an external power source stepped up to 6V. The capacitors are used to prevent temporary power deficit and assure continuity in the power supply to the servo motors.

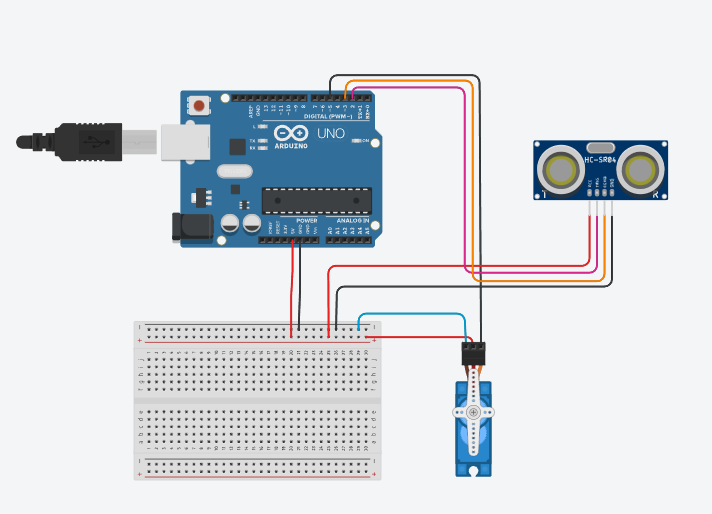
Fig 5. Schematic Diagram produced in TinkerCAD

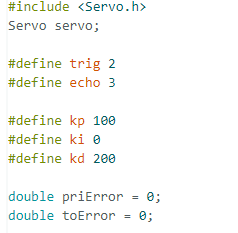
Table 1. Component List

|  |  |
| --- | --- |
| Qty | Component Name |
| 1 | Arduino Uno |
| 1 | Breadboard |
| 1 | HC-SR04 |
| 1 | Servo Motor |

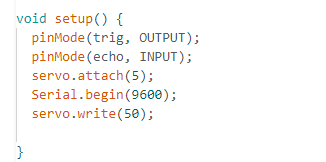
## **Arduino programming**

The Arduino IDE facilitates the usage of existent Arduino library functions and programmes, as well as those located in public GitHub repositories. These libraries allow for the automated processing, execution, and application of tasks like filtering, PID control, and servo manoeuvring. However, the functions rely on predetermined coding parameters for the system to determine the appropriate action. Here I break down the ideas and Arduino code utilised in this project.

## **Section-1**

The first line includes the Servo.h library, which provides functions to control servo motors. Servo servo; creates an instance of the Servo class. The #define statements define two constants, trig and echo, which correspond to the trigger and echo pins of the ultrasonic sensor. The #define statements also define the tuning constants for the PID controller: kp, ki, and kd. These constants determine how the controller responds to errors in the system. priError and toError are variables used to calculate the error in the PID controller.

## **Section-2**

This function sets the initial conditions for the program. pinMode(trig, OUTPUT) and pinMode(echo, INPUT) set the trigger and echo pins of the ultrasonic sensor as output and input pins, respectively. servo.attach(5) attaches the servo motor to pin 5. Serial.begin(9600) sets up serial communication at a baud rate of 9600, which can be used for debugging purposes. Finally, servo.write (50) sets the initial position of the servo motor to 50 degrees.

## **Section-3**

This function is the main loop of the program. PID() is called repeatedly to calculate the PID value and adjust the position of the servo motor.

## **Section-4**

This function calculates the distance measured by the ultrasonic sensor. It uses the pulseIn function to measure the time it takes for the ultrasonic signal to bounce back from an object and calculates the distance in centimeters. Here the total length of the moving arm is 29cm we have divided it in half and stored the value in ‘long’ variable

## **Section-5**

This function calculates the PID value using the distance the ultrasonic sensor measures. setP is the desired setpoint for the system, and error is the difference between the setpoint and the current distance. Pvalue, Ivalue, and Dvalue are the proportional, integral, and derivative components of the PID controller, respectively. PIDvalue is the sum of these components. priError is set to error for use in the next iteration, and toError is accumulated to calculate the integral term. Serial.println(PIDvalue) prints the PID value to the serial monitor for debugging purposes. map converts the PID value to a range that can be written to the servo.

# **Result**

## **Do It Yourself (DIY) Adaptation**

### **DIY materials and adaptations used in the project**

1. Cork Sheet
2. Glue Gun
3. Metal Stick
4. Pen Shells
5. Foam Ball

### **Measurement taken of the DIY frame**

* At first an 11-inch prism shape hollow bridge with one side open was constructed with cork sheet.
* Then, we cut out a big chunk of cork sheet as a base of our whole framework.
* Then we needed to construct a base that will hold out pivot point. We used 2 trapezoidal shaped cork sheet cut outs to hold our pivot point.
* Then we used the glue gun to put together all the separate parts and got our structure.

## **2. Tuning Values**

KP, KI, and KD are three parameters used in PID (Proportional-Integral-Derivative) control systems that affect the behavior of the system.

KP, or the proportional gain, determines how much the output of the system responds to changes in the error signal. A higher value of KP causes a stronger response to the error signal, but it can also lead to overshooting and oscillations.

KI, or the integral gain, determines how much the controller responds to errors over time. It helps to reduce the steady-state error in the system, but a high KI value can also lead to overshoot and instability.

KD, or the derivative gain, determines how much the controller responds to changes in the error over time. It helps to reduce overshoot and oscillations in the system, but too high a value can cause instability and make the system more sensitive to noise.

The overall behavior of the system is determined by the balance between these three factors. The proportional gain makes the system responsive to the current error, the integral gain makes the system responsive to past errors, and the derivative gain makes the system responsive to future changes in the error. Finding the optimal values of KP, KI, and KD requires careful tuning to balance the system's response to these three factors, as well as the specific requirements of the system being controlled.

To begin tuning the PID controller, we set ki and kd to zero and focus on tuning kp, which controls the proportional term of the controller and is crucial for system stability. We gradually increase kp until the system starts to oscillate, then reduce it to a level where the oscillations dampen out as quickly as possible. Once kp is set, we move on to tuning ki, which is responsible for reducing steady-state errors. We start with a small value of ki and gradually increase it until we see an overshoot, then decrease it to minimize the overshoot. Finally, we tune kd, which is responsible for reducing overshoot and damping out oscillations. We start with a small value of kd and gradually increase it until we see a reduction in overshoot and oscillations, but decrease it if the system becomes unstable. It's important to experiment and make small adjustments to the values of kp, kd, and ki until the desired performance is achieved, as the optimal values can vary greatly depending on the specific system. Simulation software or mathematical models can also help determine initial values for the tuning process.

# **Future Improvement**

1. **Increase ramp length:** Adding more time to the ramp could be a good way to improve the current hardware control system project. As a result, the ball's range would increase, making the system more difficult to master. A stronger motor is needed to guide the ball up the longer ramp, and the control algorithm may need to be tweaked to accommodate the larger distance the ball must go.
2. **Use a smoother surface:** The floor could be made smoother for the ball to roll more easily, for example. With less resistance between the ball and the floor, balancing would be less of a challenge. This would necessitate the installation of a new platform or surface, as well as modifications to the system to accommodate its unique properties.
3. **Replace sonar sensor with fuzzy sensors:** It has also been suggested that fuzzy sensors be used in place of sonar ones. Fuzzy sensors can provide more precise readings since they are more adaptive to the ball's changing position and motion. To incorporate the new sensors into the system, more programming and hardware adjustments would be required.
4. **Alternative to servo motor**: A other kind of motor could be utilized in place of the servo one now installed in the system. It's possible that various motor types might require various control tactics due to their unique qualities and capacities. This would necessitate swapping in a new motor and modifying the control algorithm to suit the new motor's specifications.

# **Conclusion**

Some observations gained over the course of building a fully functional ball balancing platform may provide light on the aforementioned subject. Any system that is either partially stable or unstable by nature will always require state adjustment. A ball balancing platform uses servo motors to implement the desired state change, which is an inclination of the platform relative to the ball. Therefore, the modification must be managed. Although these steps can be taken manually, in the context of control theory, the controller—PD, for example—would take them.

However, a controller relies on an input entirely. The position of the ball provides the input in the case of the balancing platform. In other systems, the input may be something like temperature, pressure, velocity, etc (Cheok and Loh, 1987).

The essential factors to think about while designing a stabilizing system are the constraints imposed by the ability to modify states, the controller architecture, and the input. Therefore, the system's performance is contingent upon, yet constrained by, the speed and accuracy with which input data can be gathered, as well as the constraints of the controller. These restrictions are realized in a ball balancing platform by the maximum allowable angular velocity for the servo motors. Consider the analogue touch panel's positional accuracy, the capabilities of the Arduino, and the PD controller architecture.

The PD controller's performance is highly sensitive to the gain levels (Kp and Kd). The Kd gain term, modifies the output's rate of change, decreasing overshoot and making the system more stable and responsive during transients. Alternatively, as KP is increased, the system responds proportionally more quickly.

# **Reference List**

Glad, T. and Ljung, L., 2000. *Control theory*. CRC press.

Goodwin, G.C., Graebe, S.F. and Salgado, M.E., 2001. *Control system design* (Vol. 240). Upper Saddle River: Prentice Hall.

Margalith, A. and Mergler, H.W., 1982. Optimum setting for proportional controller. *IEEE Transactions on Industrial Electronics*, (2), pp.165-175.

Tan, J., Liu, K. and Turk, G., 2011. Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications*, *31*(4), pp.34-44.

Bennett, S., 1993. Development of the PID controller. *IEEE Control Systems Magazine*, *13*(6), pp.58-62.

Arduino, S.A., 2015. Arduino. *Arduino LLC*, *372*.

Van de Straete, H.J., Degezelle, P., De Schutter, J. and Belmans, R.J., 1998. Servo motor selection criterion for mechatronic applications. *IEEE/ASME Transactions on mechatronics*, *3*(1), pp.43-50.

Morgan, E.J., 2014. HC-SR04 ultrasonic sensor. *Nov*.

Cheok, K. and Loh, N., 1987. A ball balancing demonstration of optimal and disturbance-accomodating control. *IEEE Control Systems Magazine*, *7*(1), pp.54-57.