Ball and Plate
Project in TFY4190

S. S. Olderkjær, M. Trætli

April 2022

Abstract

In this experiment, a modified PID controller was implemented on an Arduino to control the position of a steel ball on a plate using a resistive touch screen to track position and two servo motors to move the plate.

Introduction

A proportional-integral-derivative(hence PID) controller is a control loop mechanism one may use to provide feedback to a control system in order to make said system stable. The controller calculates the error value between the desired position and the current position and applies a correction based on the proportional, integral and derivative terms of the error value. With this mechanism, it is possible to apply accurate and responsive corrections to a control function, and it is therefore used in numerous application.

Using this controller, we wanted to implement a control system which was able to balance a steel ball on a plate even when introducing an external force on the ball. For that to work, a sensor to track the position of the ball along with some actuators which changes the orientation of the plate were needed. By continuously updating the desired position, it is possible to make the ball follow patterns. Thus after making the system capable of balancing the ball and placing it back at the centre, we wanted to implemented more complex patterns such as the pattern of a square and a circle. Videos of the results can be found here.

Theory

A PID controller is a simple regulation system. The controller continuously computes the error value $e(t)$ as the difference between the desired setpoint $r(t)$ and a measured process variable $y(t)$ such that

$$e(t) = r(t) - y(t).$$

The goal of the controller is to minimize the error value over time by adjusting the control variable $u(t)$. The overall control function of a PID controller is generally described as

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}$$

where $e(t)$ is the error from setpoint at time t, K_p, K_i and K_d are all non-negative and denote the coefficients for the proportional, integral and derivative error respectively.

For the problem at hand, it is beneficial for the system to have greater response the further away the the ball is from the desired position. Especially since the ball at all times must stay inside the boundaries of the rather small plate. Therefore, a square proportional term is included. Thus the control function for the system is

$$u(t) = K_p e(t) + K_{p^2} e(t) \cdot |e(t)| + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}$$

where K_{p^2} is the coefficient for the squared proportional error. In practice the measurements are made discretely, so the integral and derivative error has to be found numerically. Between the time points of two measurements occuring at t' and t where $t' > t$, we applied the following first order approximations

$$\int_t^{t'} e(\tau)\, d\tau \approx e(t') \cdot (t' - t) \quad \text{and} \quad \frac{de(t)}{dt} \approx \frac{e(t') - e(t)}{t' - t}.$$

This yields a total integration error of

$$\int_0^{t_n} e(\tau)\, d\tau \approx \sum_{i=1}^{n} (t_i - t_{i-1}) e(t_i),$$

where t_i for $i = 0, 1, 2, ..., n$ are the times of measurement, and t_n is the current time. The constants K_i, K_d, K_p and K_{p^2} must be tuned by trial and error, though different tuning strategies exists.

Method

Components and Equipment

To track the position of the ball, a 5-wire resistive touch screen was used. Due to the touch screen being a large plate it would double as a platform for the ball to move on. Using 4 of the wires, this touch screen sets up a potential gradient across the screen and is able to read the position of an object along that gradient using the last wire. Thus, for measuring the postion in 2 dimensions, the gradient had to be set up in differently for the measurement of the x-position and y-position respectively. This plate is only able to take measurements if the object placed on it is heavy enough. Therefore, we would have to use a steel ball which was heavy enough to be accurately measured.

In order to correct the system based on the positional error on the touch screen, two servo motors were used to adjust the angle of the plate. Since the error and movement along the x-axis and y-axis respectively are independent, the PID controllers can be implemented separately for each servo motor. Since the two servo motors are not able to support the touch plate by themselves, we had to create a universal joint which held the majority of the weight, while the servos were given arms that connected to the edges of the plate.
To enable the sensor and actuators to communicate, a control unit is required. For this, an Arduino UNO suffices since it is simple to use and has voltage supplies for the servo motors as well as pins for reading and writing both analog and digital signals. The arduino also has a processing unit which allows it to calculate the PID output to the servos.

A 3D-printer was used to create the remaining static parts in the setup.

A list of the components that were used is shown in the list below. Links have been included for some of the parts.

- 5 wire resistive touch plate
- 2 Servo motors
- Miscellanious screws
- 2 x $10k\Omega$ and 5 x 220 Ω resistors
- LEDs in different colors
- Miscellanious wires
- Arduino uno
- Steel ball with diameter 1.5 inches
- 2 push buttons
- Miscellanious 3d printed parts

Experimental setup

The experimental setup that was used can be seen in figure 1, 2 and 3. A wooden base platform holds the parts together. In the middle of the base platform, a universal joint with a smaller platform on top was mounted. The universal joint allows the top platform to change its angle freely, while the top platform is used to hold the resisitive touch screen. Two holders for the servo motors were also mounted on the base platform, directly underneath the connection point between the servos and the platform. In order to make the static parts used in the setup, a 3D-printer was used. The STL-files for these parts can be found in the our GitHub repository, and editable versions can be found in our onshape folder. In addition, a chassis for the control unit was printed to protect the control unit and was mounted in a corner of the base platform. A breadboard was mounted on the base platform and was used for wiring the LEDs and push buttons, as well as supplying power to the servos. A circuit diagram of the setup is shown in figure 4. When everything was put in place, the wires and buttons were glued in place.

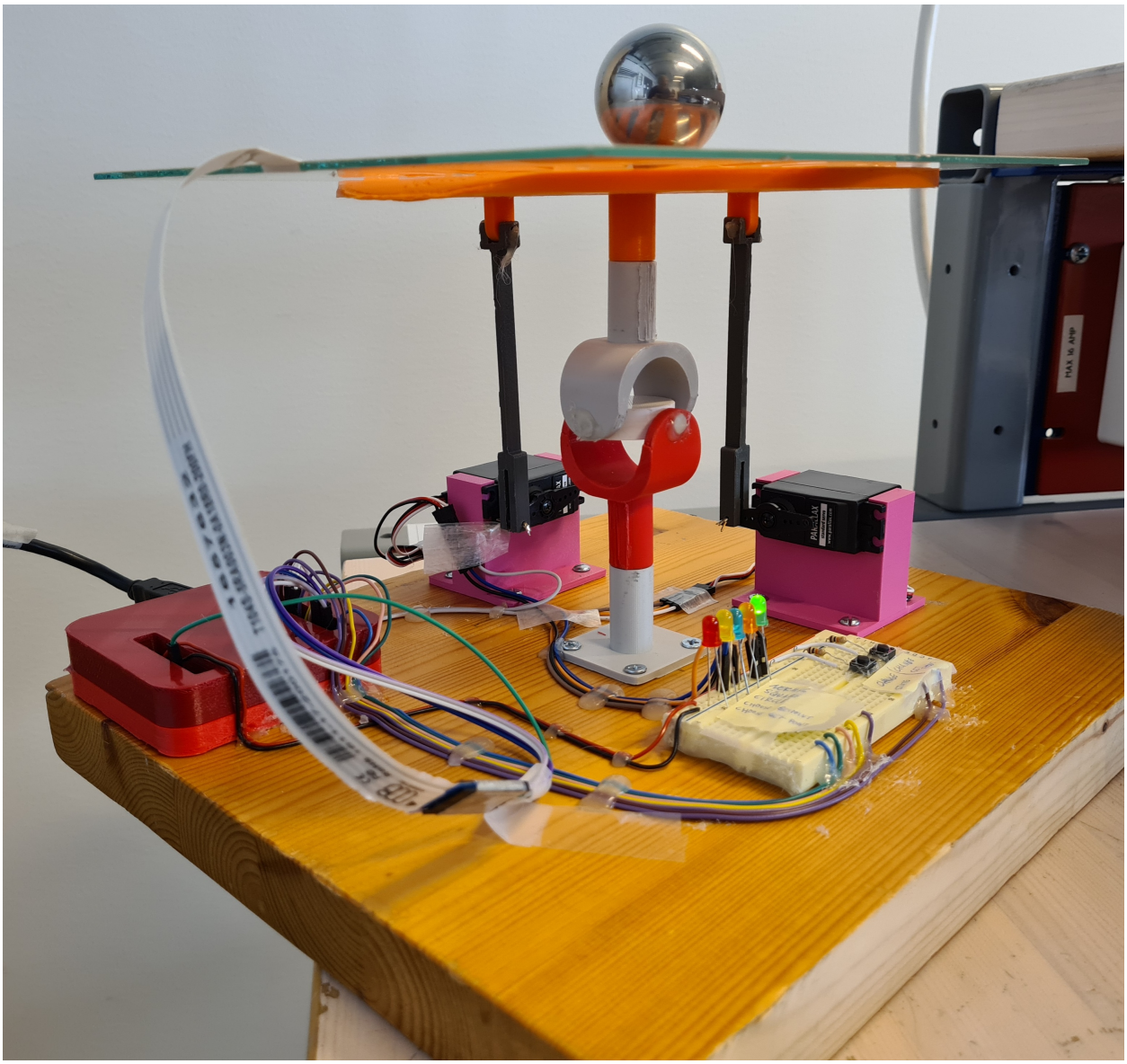

Figure 1: An overview of the whole setup. The breadboard placed in the lower left corner contains indication LEDs and buttons, the Arduino Uno is placed safely in its own red box in the bottom right corner. The ball is nicely balancing the ball in Normal mode.

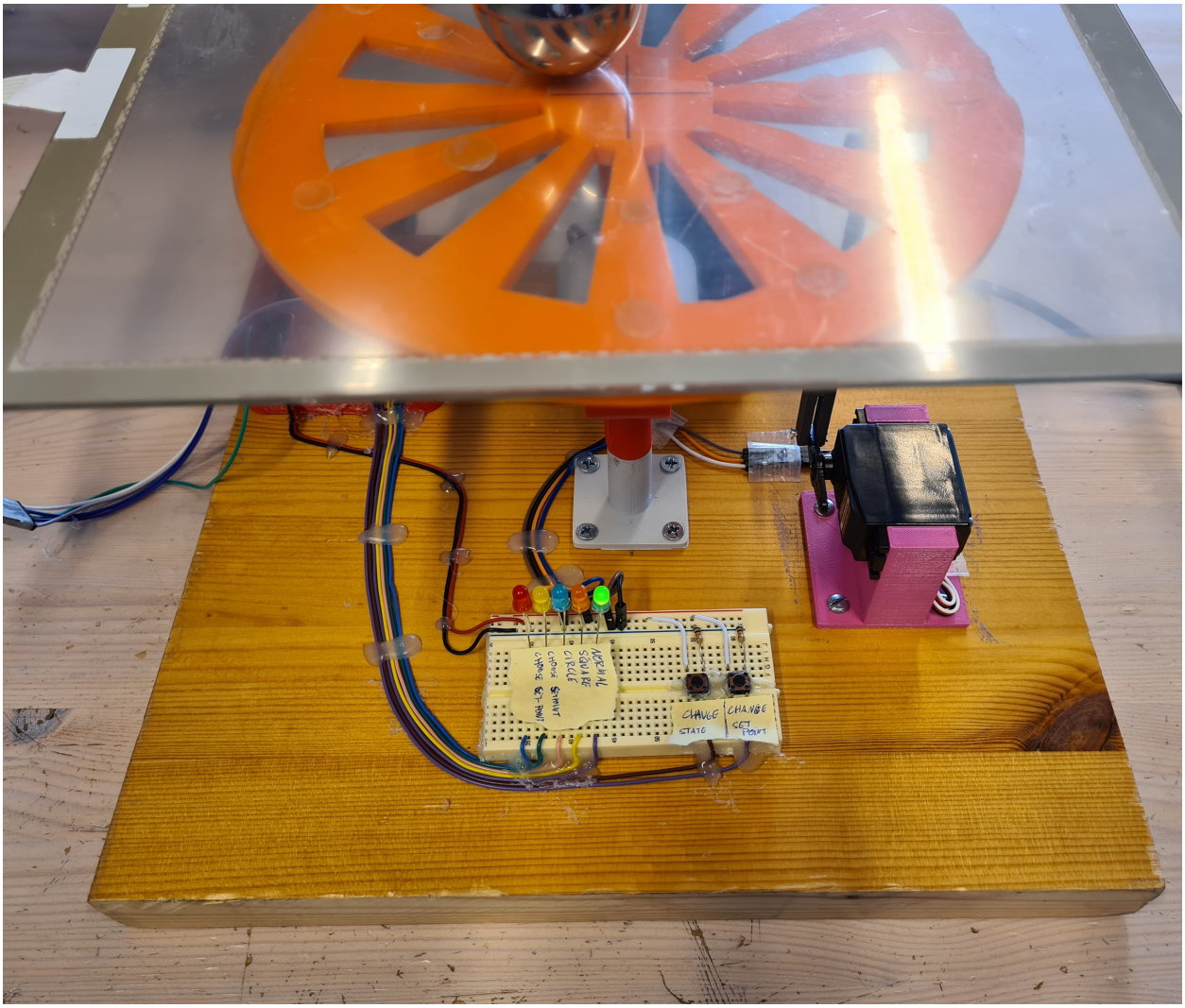

Figure 2: This picture shows the breadbord with the LEDs and buttons that were used in this project. The LEDs are marked with their purpose, and so are the buttons. From left we first have two LEDs (red and yellow) indicating a state of choosing a new setpoint, the next LED (blue) indicates Circle mode, the next LED (orange) indicates Square mode and the last LED (green) indicates Normal mode. The left button is for changing states, while the right button is for choosing a new set point.

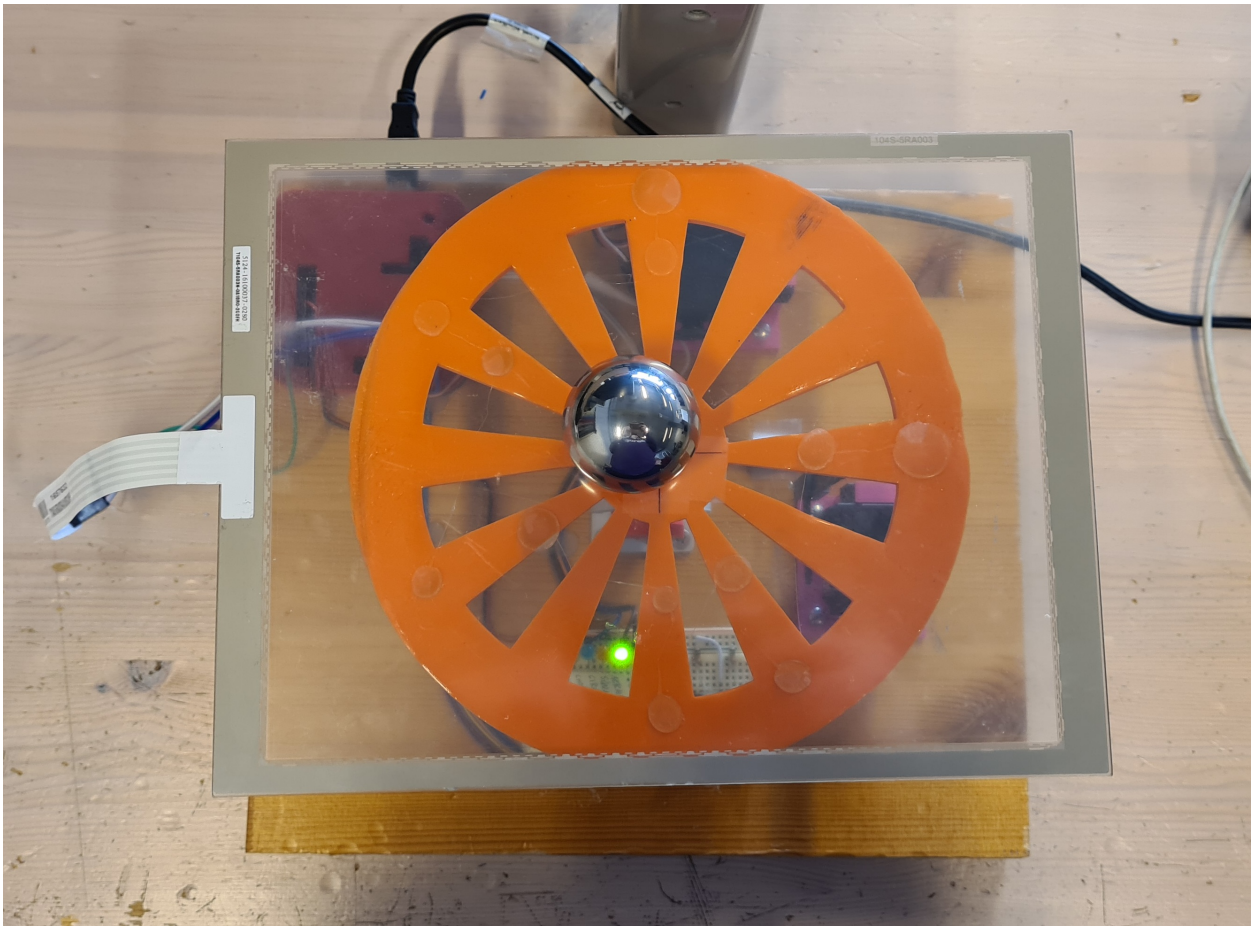

Figure 3: This picture shows the setup from above. Here one can see the PID-controller balancing the ball around the center of the resistive touchscreen.

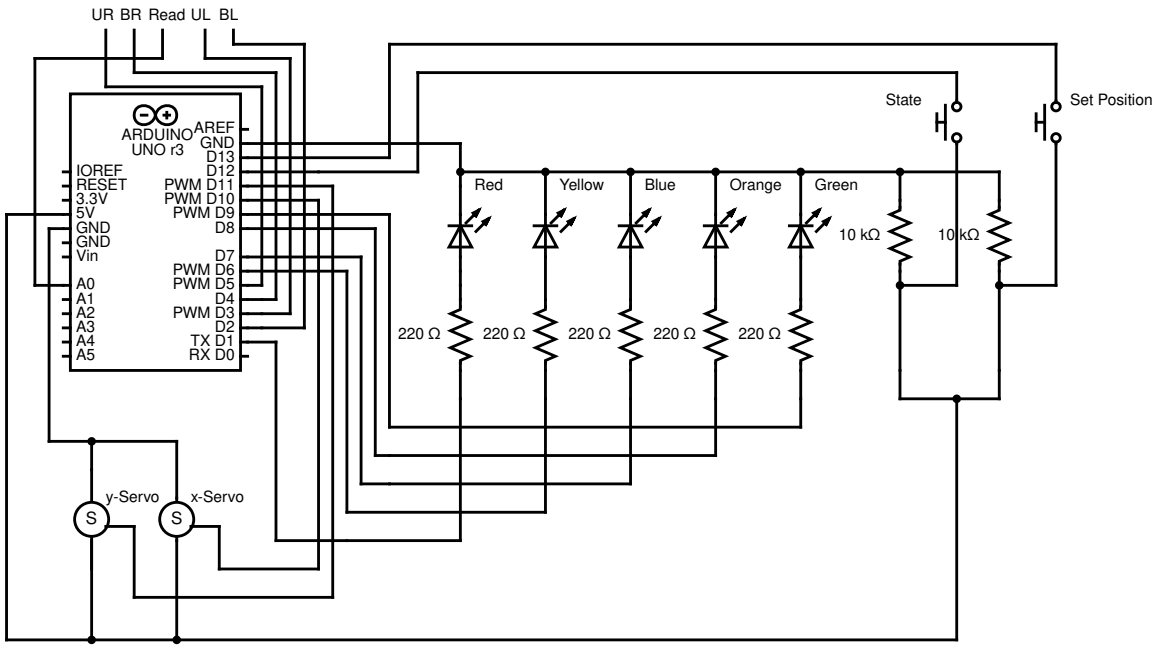

Figure 4: A circuit diagram for the experimental setup. UR, BR, Read, UL and BL refer to the pins of the resistive touch screen.

Code

The code for this project was written in C using Arduinos own IDE. A link to the code on GitHub can be found here. The code works as a state machine switching between normal, square and circle mode by pressing the state-switcher button. A state-diagram of the state machine is shown in figure 5. Every loop in a given state consists of reading the x and y position, followed by computing the PID outputs for the servos, and then moving the servos to the desired position. During these loops we also check for button input. Since the measurements given by the resistive touch screen has a lot of variance, we had to find a way to smooth the input signal. For this, we took multiple measurements in quick succesion and put the mean of those measurements as the desired position. In addition, since it is infeasible that the ball moves more than a couple centimeters at a time, if the mean of the measurements were wastly different to the previous position, we would cap the change at 2 cm. An explaination of the states can be found below.

When in normal mode, indicated by the green LED, the desired position will be constant in time and the system will attempt to postion the ball at rest in this position. When the ball has been sufficiently close to the desired position for a couple of seconds, it will trigger a stability mode which has smaller coefficients for the PID controller as well as a longer waiting time between PID computations. This allows for calmer adjustments to the position of the ball. The normal mode in practice can be seen on here. One may also press the change set point button to change this desired position manually by pressing the button, then moving and holding the ball at the new target position and press the button again. This is indicated by the yellow and red LEDS.
When in square mode, indicated by the orange LED, the desired position will be updated every 5 seconds. The desired position changes sequentially through the positions of the corners of a square centered at the centre of the plate with sides of length 4 cm. This can be seen in practice on here.

When in circle mode, indicated by the blue LED, the desired position will be updated every time the PID output is computed. The change in position is rather small, and the ball will move in a circular pattern around the center of the plate. This can be seen in practice on here.

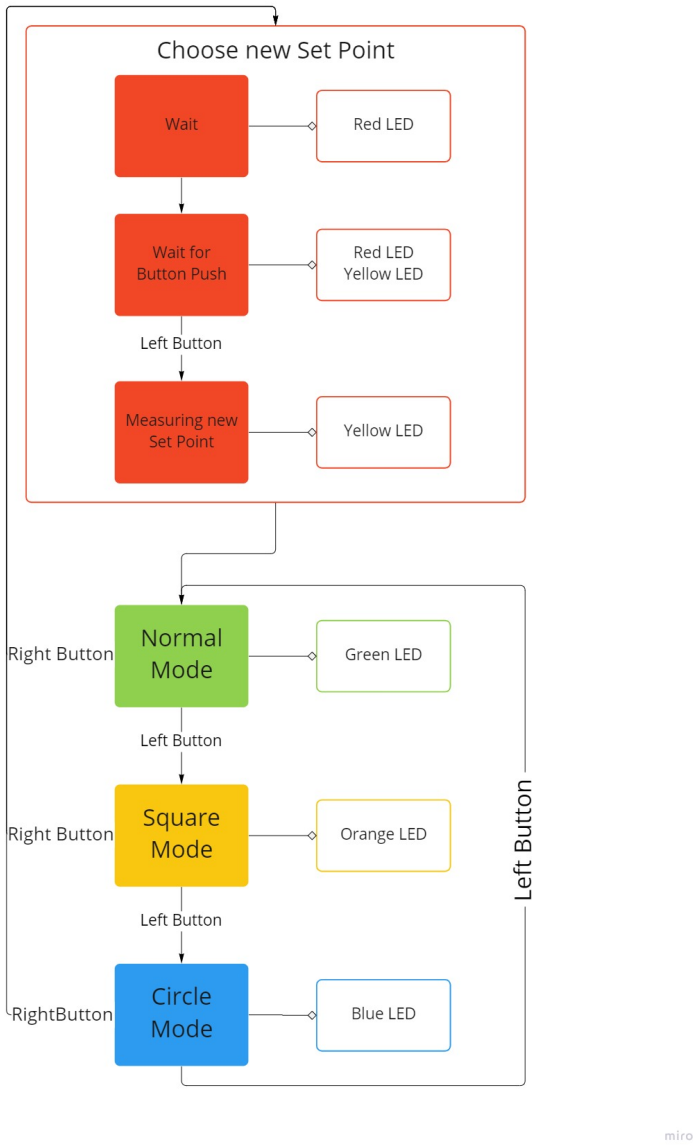

Figure 5: State diagram for the Ball and Plate project.

Calibration

Once the experimental setup was complete, we had to calibrate the input reading of the screen as well as the mapping from the PID-outputs to servo angles and the constants of the PID. The resistive touch screen ouputs an analog output, which when converted by the ADC on the arduino yields values ranging from 0 to 1023. We wanted to measure the position in centimeters. Thus for each axis we had to measure the minimum and maximum values that the resistive touch screen outputed, and then divide the physical length of the active area of the touch screen on the difference between the maximum and the minimum values. The mapping from PID-outputs to servo angles was found by trial and error. The coefficients for the PID was also found by trial and error, and varied based on which state the system was currently in.

Results

In normal mode, the system performed quite well. When left alone, the ball would be mostly at rest, only requiring minor adjustments every now and then due to the imperfections in the plate. This was particularly a problem when the target position was directly on top the center of the universal joint, as any small displacement from the center would cause the plate to change its resting angle slightly. When an external force was applied to the ball, the system would correct the position as long as the force that was applied would not knock the ball of the plate. The ball was placed at the corners with the target position in the middle of the plate, and the system would be able to catch the ball when it was let go off. This worked very well along one diagonal, but for the other diagonal it would overshoot it a little, causing a the ball to overshoot and spiral in towards the target position. The Normal state in practice can be seen here.

In square mode, the system was able to shift the position of the ball to mimic the square pattern, though with small deviances. When external forces were applied, the system would be able to account for this. The resulting square motion can be seen here.

In circle mode, the system was able to move the ball in a circular pattern. As the speed went up, the ball would move in a circle with a larger radius than intended. The resulting Ball and Plate circle motion can be seen here.

Discussion

The regulation system worked really well taking into account the loose fits in the construction, the resolution of the servo motors positions and the measurement errors in the resistive touchscreen.

The result of the loose fits was that if one makes the plate as straight as possible and place the ball on it, due to the weight of the ball it will roll in different directions depending on which side it is placed on. The direct cause of this is the change from pushing to pulling on the servo motors. This instability was mainly introduced by the universal joint, the servo motors and the joints between the servo motors and the plate. We designed and 3D-printed the parts this universal joint was made from, and the designs have room for improvement. To fix this instability one could either improve the designs, or buy one in the store. The loose fits in the servo motor was primarily caused by connection between the servo motor and the associated servo arm. Between the servo arms and the plate, we used 3D-printed rods with bend metal in the joints. The loose fits of the joints seemed to introduce the most of the instability, which totaled at a few degrees. A way of reducing the loose fits problems drastically is by using a design where the force directed at the rods connected to the plate always is a downward force. On the internet one can find multiple designs, often with three servo motors and no universal joint, that are designed in such a way.

The resolution of the position of servo motor were in whole degrees. A higher resolution of the servo motors would likely give a smoother and more stable movement. The resistive touchscreen suffered from faulty measurements. We observed the measurements by plotting the position while moving the ball around on the plate. Along the $y = x$ axis, the positions were really good, while along the $y = -x$-axis, the measurements suffered from frequent and large jumps in the position. To account for this, we used a mean of multiple observations and made a max steplength, as mentioned in the Code section. Even with this cleaning logic in place, there were some measurement errors along the $y = -x$-axis. This cause the ball to move somewhat wierd along the mentioned axis. A better cleaning logic or filtering would reduce this problem, and we would suggest trying a Kalman filter.

Conclusion

The build and PID-controller worked well. In normal mode the PID-controller managed to balance the ball around a chosen set point, the square movement worked as desired and the circular movement was smooth

and worked as desired. On the downside there were some undesired stuttering and unwanted behavior due to loose fits and measurement errors. Future work consists of trying different filtering methods, where a Kalman filter is suggested, and either make the fits thighter or redesigning to remove the problems arising from loose fits.