Design of a light, crash-worthy, self-driving bike

Bachelor's thesis

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of electrical engineering

Ayman Alzein, Mohammad Kanjo, Wiggo Klapp,
Jonatan Ornung, Jacob Rutgersson, Valdemar Samuelsson

May 24, 2023

Design of a light, crash-worthy self-driving bike

Supervisor: Jonas Sjöberg, Professor, Electrical Engineering
Examiner: Jonas Fredriksson, Professor, Electrical Engineering

Bachelor's Thesis 2023
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Chalmers logotype.
Typeset in LATEX
Gothenburg, Sweden 2023

Abstract

The focus of this thesis is to further develop a previously built plastic bike with the goal to have it become self-driving. In comparison to a conventional metal bike, the flexible nature of the plastic bike poses a challenge when it comes to the control aspect to achieve self-driving. A number of structural changes was therefore made to make the bike more rigid and better adapted for a control system. The size of the bike was reduced and play in the joints was decreased. These structural changes reduced the flexibility, however it was not completely removed. Therefore a transfer function for the flexibility of the plastic bike, specifically the fork, was developed. The fork was modeled as a spring with parameters k,b and J. A number of tests where developed to find these parameters. In addition to the transfer function for the flexibility, a transfer function of the complete bike as well as both a P and a PD-regulator were developed. The parameters k_p and k_d were tuned and the behaviour of the bike could then be simulated. Real world testing with the physical bike was conducted. The thesis concludes with recommendations for further improvement of the bike's design and control system.

Acknowledgements

During the project we have received invaluable help from several people. We would especially like to thank our supervisor Jonas Sjöberg for his guidance and assistance throughout the project.

We would also like to express our gratitude towards our examiner Jonas Fredriksson for taking the time to read our reports and give us valuable feedback.

Also, we deeply appreciate the assistance from Ossian Eriksson and his help to overcome several obstacles we encountered.

Finally, we want to express our sincere appreciation for everyone involved in the Autobike-projects in the "bike lab". The cooperation and exchange of knowledge made the project both easier and more exciting.

Nomenclatures

The next list describes several nomenclatures that will be later used within the body of the document

CAD Computer Aided Design

CSV Comma-seperated values

ESC Electronic Speed Controller

Head tube The head tube is the part of a bike's tubular frame where the front fork steering tube is mounted. This is the orange mount in the front where also the tubes making up the frame is mounted.

IMU Inertial Measurement Unit

Labview Laboratory Virtual Instrumentation Engineering Workbench. See Appendix 6.3.1

LHP Left half plane

Matlab Matrix laboratory. See Appendix 6.3.2

Pcontroller proportional controller

PDcontroller Proportional-Derivative controller

PIDcontroller Proportional-Integral-Derivative controller

PWM Pulse Width Modulation

RHP Right half plane

Simulink Simulation and link

Contents

1 Introduction

1.1 Background

Self-driving cars are quickly becoming a reality, and with them come significant advancements in transportation technology. These vehicles have the potential to revolutionize the way we move around, offering increased safety and efficiency on the roads. However, before self-driving cars can become a common sight in traffic, they must be thoroughly tested to ensure they meet official safety standards. That's where vehicles used in testing self-driving cars come in. These specially designed vehicles allow researchers and engineers to develop and refine self-driving technology in a controlled environment, helping to pave the way for safer and more efficient roads.

In the AutoBike research project by Chalmers, self-driving bicycles are being used in testing of self-driving cars and their automatic brake systems. Project partners are Volvo Cars, Veoneer, Autoliv, AstaZero and Mälardalen University. The project started in 2019 with a master thesis (Erdinç. U., 2019) and consists of several metal bicycles that are continuously improved.

These bicycles drives a pre-defined route that may include swerving, while carrying a dummy. This is to make it as similar to a real bicyclist as possible, to test the car's sensors and systems. A problem when testing the cars at higher speeds with these metal self-driving bicycles are their heavy weight and hard metallic parts. These pose a safety risk for the person in the car, and also risks damaging the car. Therefore a bicycle constructed of lighter materials while still having self-driving capabilities could be a potential solution to the problem.

1.1.1 Plastic bike

During 2022 a lightweight bike made out of acrylic tubes and 3D-printed PLA-parts was developed (Choucha, L., Smajic, E., 2021). Figure 1 shows a picture of this bike. The bike lacked any self-driving capability. There are aspects that are different with a flexible plastic bike compared to for example Chalmers existing rigid self-driving metal bikes (Strömberg, A., Leikvik, T., 2022). One aspect is that the frame and specifically the fork of a plastic bike is more flexible than a metal bike. Higher flexibility poses a problem when it comes to the self-driving control system, where the bicycle has to respond to steering inputs in a fast manner. The use of a flexible material may cause a slow unstable system, taking a long time to respond to an input signal. Achieving self-driving for a flexible bicycle may require adjusting the control system to compensate for higher flexibility, as simply reducing flexibility may not be enough.

Figure 1: Original plastic bike (Choucha, L., Smajic, E., 2021)

1.1.2 Bicycle parts

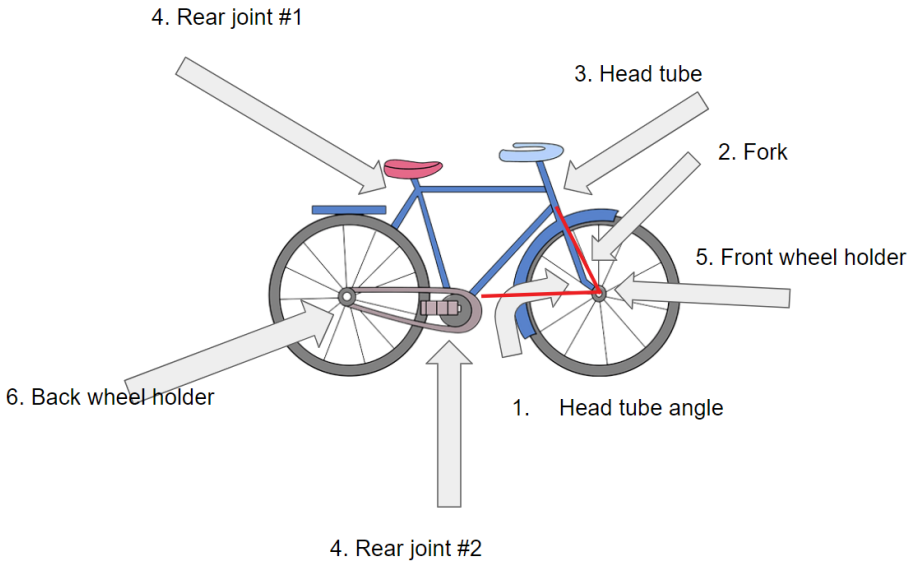

Figure 2: Different parts of the bike

1.1.3 Portable self-driving hardware

There have been several iterations of self-driving metal bikes previously developed. Some of these bikes used self-driving hardware that was fully integrated with the bikes (Erdinç. U., 2019) while there was one hardware system developed to be a universal mounting system with the purpose to be able to quickly move the self-driving hardware in between bikes (Strömberg, A., Leikvik, T., 2022). This universal mounting system was utilized in this project. The portable self-driving hardware consisted of actuators, sensors, controllers and a myRIO acting as the main computer of the system. Figure 3 shows a picture of the plastic bike equipped with the portable self-driving hardware.

Figure 3: Plastic bike fitted with the portable mounted hardware

1.1.4 Sensors and actuators

There are multiple sensors included in the self-driving hardware. The IMU (Inertial Measurement Unit) use a sensors to provide relevant information on the force and angular rate of the bike in multiple axes. The IMU provides a variety of orientation related input data which is crucial for the control of the bike. A GPS is used to track the positioning of the bike.

The portable self-driving hardware contains two actuators. A forward DC hub motor mounted in a bike wheel. The purpose of this motor is to propel the bike forward. The second actuator is the steering motor responsible for the steering of the bike. This motor is connected with a belt to the bikes steering rod.

1.1.5 Control system

Figure 4 is a picture of a block diagram illustrating the closed loop system of a flexible bike. The input of the complete system is a reference roll angle and the output is the actual roll angle. The input of $G_{Flex}(s)$ is the reference steering angle, and the output of $G_{Flex}(s)$ is the actual steering angle. A difference between this system compared to a metal bike is the existence of $G_{Flex}(s)$. The flexibility of a metal bike is often small enough to be negligible. When designing a control system for a metal bike there is no need to take the flexibility into account, whereas in the case with the plastic bike it is necessary.

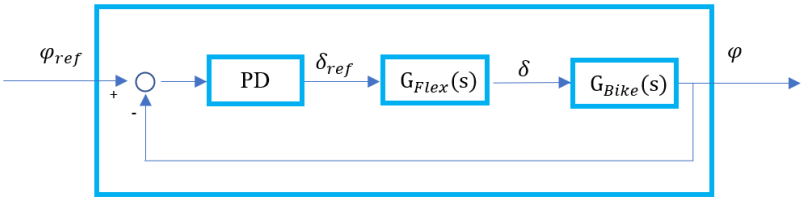

Figure 4: Feedback control system with a PD controller. G_{Bike} is the bike's transfer function. G_{Flex} is the fork's flexibility, δ is the steering angle and φ is the roll angle.

The controller of the bike calculates a new steering angle based on the roll angle (the angle of tilt from side to side) error from target value. It is therefore crucial that the steering angle can be applied in a quick and exact way. The flexible nature of the plastic bike, and specifically the fork creates an angular difference between reference steering angle and the wheel angle. For metal bikes, this angular difference is often so low that it is negligible. But for a flexible plastic bike this angular difference can cause problems when it comes to balancing the bike. A bike is naturally an unstable system. This is due to the fact that from a control system view, there are poles in the right half plane (RHP) (Lennartsson, B., 2000).

1.2 Problem statement

The goal for this project is to develop methods for reducing the bike's flexibility, mount hardware components, create an accurate model of its remaining flexible behavior and design a controller that takes these factors into account.

1.3 Contributions

There are several structural improvements made to the plastic bike, including improving mounting techniques. Section 2.1 describes the methods used to accomplish the bikes structural improvements. In section 2.1.2, experiments to evaluate the improvements on structural design of the plastic bike were developed.

A new controller has been designed and three experiments were developed to be able to determine parameters needed for this model. Section 2.2.3 goes into details on these experiments which result in a model of the flexibility.

2 Method

2.1 Mechanical Design Process

The improvement process of the bicycle consists of small, incremental improvements in order to make the project as time and cost efficient as possible. Each small step were easy to implement and could be evaluated directly.

- The length of the tubes was reduced to make the size of the bike more similar to a regular bike.
- Duct-tape were used at the tip of the tubes to fill the existing gap between the tubes and the 3d-printed joints.
- The motor wheel was moved to the back of the bicycle instead of the front to reduce the moment of inertia affecting the fork.
- Both holders for the front wheel were re-printed with small modifications to improve tolerances and reduce movement.
- Holes were drilled in the back wheel holder to fit the larger mounting screws of the motor wheel.
- Plastic rings were printed to be pressed between the bearing and the bearing slot to reduce play in the fork.
- Handlebars were designed to make it easier to handle the bike when it wasn't driving and to give the appearance of a regular bike.
- Nylon rods were pressed into the fork-tubes to further improve rigidity.
- Washers were added at the front wheel to reduce pressure on the printed mount at a given force applied by the scew-nut. This also made it possible to further tighten the screw-nuts.

The improvements where validated through the tests (see sections 2.1.2 and 2.2.3)

A new head tube and fork clamp was designed to further improve the bikes rigidity but in the end there was no time to print or install these new designs.

2.1.1 Electronics Mounting Process

When mounting the portable electronics box with the included "universal mounting system" some changes were made. The mounting system for the electronics was shortened to fit along the acrylic tubes and between the 3d-printed joints. A timing belt pulley was mounted on the bike's steering rod with a conical clamping bushing. These two parts were sponsored to the project by Norelem (2023).

During testing it was discovered that the acrylic plate clamped to the head tube moved a small amount when the steering motor made quick turns. It was also discovered that the electronic box was rocking a small amount during testing. This made the bike hard to control due to the IMU being very sensitive. The rocking was due to the inherent flexibility of the plastic mount for the box. To counteract this, as can be seen in in figure 20, a thin piece of plywood was fitted between the rear joint and the electronics box.

2.1.2 Validation Procedures

To validate the improvements in rigidity five different experiments was developed. These were used in combination with tests for flexibility parameters to show the improvements.

Experiment 1, the bicycle was laid onto a table with just the steering axle and the fork sticking out. The head tube was laid static onto the edge of the table and the movement of the fork caused by play were measured as a distance between the ground and the bottom of the fork . For the minimum distance the forks were affected by only gravity itself. The maximum distance were measured when a small, human force directed upwards affected the fork until the play was gone and the fork started flexing.

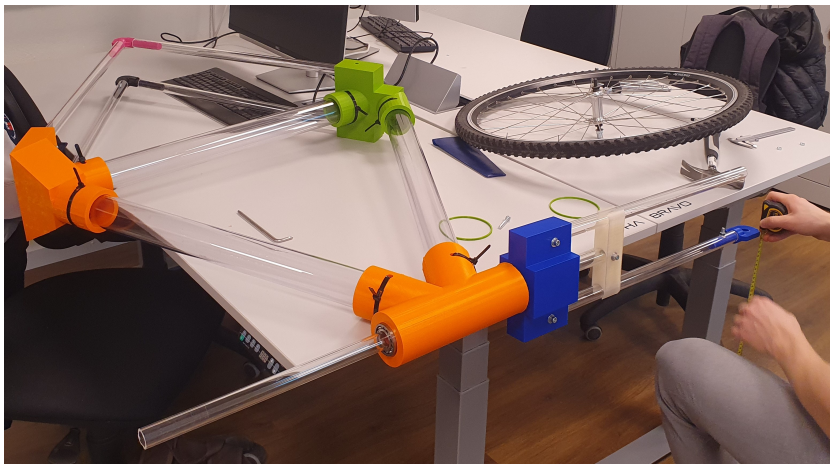

Figure 5: Experiment 1

Experiment 2, the bicycle were moved further out from the table and the movement between the tubes mounted to the head tube were also taken into account. The measurements were made in the same way as experiment one.

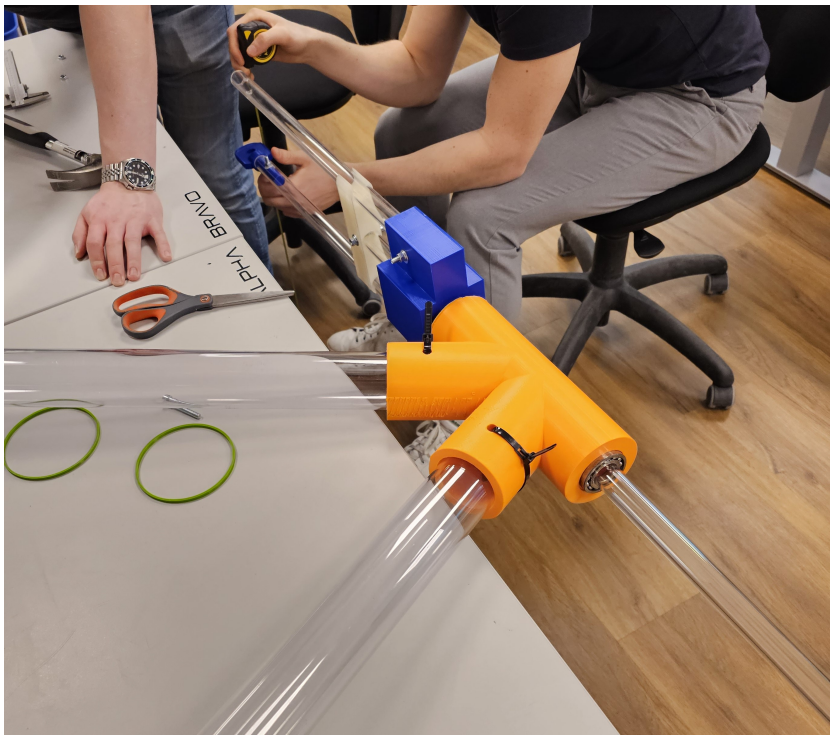

Figure 6: Experiment 2

Experiment 3, the bicycle were moved even further out from the whole table until the whole frame were exposed to gravity. The rear joints were held against the table and a small force lifted the head tube as much as the play allowed. The measurements of the heights was taken at the head tube, in the vertex between the holes for the frame.

Figure 7: Experiment 3

Experiment 4 and **Experiment 5**, measurements were instead taken at the back of the bicycle. The back of the frame were laid static onto the table and for Experiment 4, the distance between the ground and inner part of the wheel were taken. The minimum distance were measured when only gravity affected the wheel and for the maximum distance a small, human force were applied until all the play was gone and the frame started flexing. The only difference between the experiments is the measurement point, in Experiment 5, the distance were measured from the outer part of the wheel instead of the inner.

Figure 8: Experiment 4

Figure 9: Experiment 5

2.2 Bike Dynamics Analysis

In order to do simulations and analysis on the plastic bike, the dynamics had to be modeled in terms of transfer functions. The transfer functions could be obtained in two ways:

- Derive it from scratch.
- Take an existing transfer function for a metal ridge bike and multiply it with another transfer function that describes the flexibility of the fork.

Given the time constraints of the project, the second option was chosen for its efficiency while still capturing the key dynamics of the plastic bike.

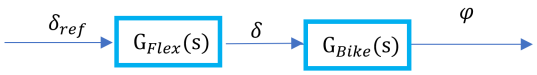

Figure 10: An overview of the plastic bike's transfer functions. δ_{ref} is the reference steering at the hand-bar angle, δ is the true steering angle at the front wheel and φ is the bike's roll angle

2.2.1 The transfer function for a metal bike $G_{Bike}(s)$

The transfer function of a rigid metal bike can be expressed as a mathematical relationship between the input signal, which is the steering angle denoted by δ, and the output signal, which is the roll angle denoted by φ. The bike moves at a constant speed, v, which is a fixed parameter in the transfer function. (K. J. Åstrom., R. E. Klein,. A. Lennartsson, 2005).

$$G_{Bike}(s) \approx \frac{av}{bh} \frac{s + \frac{v}{a}}{s^2 - \frac{g}{h}} \tag{1}$$

Where:

- a is the horizontal distance between the mass center and the rear wheel axle
- b is the distance between the wheels (wheel-base).
- h is the height of the mass center.
- g is the gravitation constant.

2.2.2 The transfer function for flexibility $G_{Flex}(s)$

Since the fork is flexible, the transfer function of the flexibility can be derived as follows: The fork is assumed to behave as a torsion spring. The torque built into the spring using the eqaution for a damped torsion oscillator is $T_{spring} = k\theta + b\dot{\theta} + I\ddot{\theta}$.
Where:

- k is the spring constant.
- Θ is the angle difference.
- δ is the steering angle
- δ_{ref} is the reference steering angle
- b is the damping constant in the spring.
- I is the moment of inertia of the spring.

The inertia of the fork is assumed to be negligible compared to the inertia of the front wheel which gives the equation $T_{spring} = k\theta + b\dot{\theta}$
To derive the transfer function of the fork that describes the difference between the angle of the handlebar and the true angle of the wheel, the torque applied to the spring by the wheel $T_{spring} = J\dot{\omega}$ where J is the moment of inertia of the front wheel around the steering axis.

$$J\dot{\omega} = k\theta + b\dot{\theta}$$

$$J\dot{\omega} = k(\delta_{ref} - k\delta) + b(\dot{\delta}_{ref} - b\dot{\delta})$$

$$J\dot{\omega} = k\delta_{ref} + b\dot{\delta}_{ref} - k\delta - b\dot{\delta}$$

$$J\ddot{\delta} = k\delta_{ref} + b\dot{\delta}_{ref} - k\delta - b\dot{\delta}$$

$$\{Laplace\} \Rightarrow Js\delta = k\delta_{ref} + bs\delta_{ref} - k\delta - bs\delta$$

$$\delta = \frac{bs + k}{Js^2 + bs + k}\delta ref$$

$$G_{flex} = \frac{\delta}{\delta ref}$$

Therefore

$$G_{flex} = \frac{bs + k}{Js^2 + bs + k} \tag{2}$$

2.2.3 Finding fork flexibility parameters

The transfer function that models the flexibility of the plastic bike 2 contains three unknown parameters, as illustrated in Figure 11. In order to obtain values for these parameters, three experiments were designed.

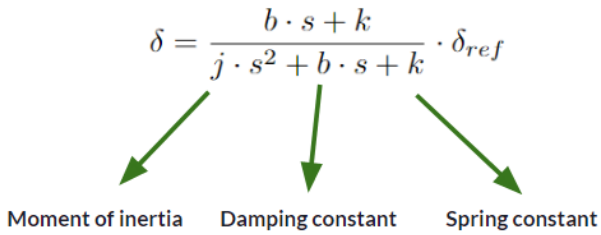

Figure 11: G_{flex} parameters

Experiment 6: Spring constant k

The following procedure was used to determine the spring constant k.

1. First, the bike's front wheel was lifted off the ground to ensure that the fork was vertical. This was important because any misalignment could result in inaccurate measurements.
2. Next, the handlebars were locked onto the bike's frame using clamps. This ensured that the handlebars would be unable to rotate even when a torque was applied. This was important because any rotation of the handlebars would have resulted in a change in the angle of the wire attached to the wheel, making it difficult to measure the angle accurately.
3. A wire was then tied to the front edge of the wheel, at the very front of the bike. The other side of the wire was connected to a known mass, which was released to run over a rod. As the mass traveled over the rod, it pulled the wire and caused the front wheel to rotate.
4. The resulting angle φ was measured.

According to the angular form of Hooke's law: $k = \frac{T}{\theta} = \frac{m \cdot g \cdot r}{\theta}$, Where r is the lever, i.e the front wheel's radius. Since the other values in the equation are known, the value of the spring constant k can be calculated.

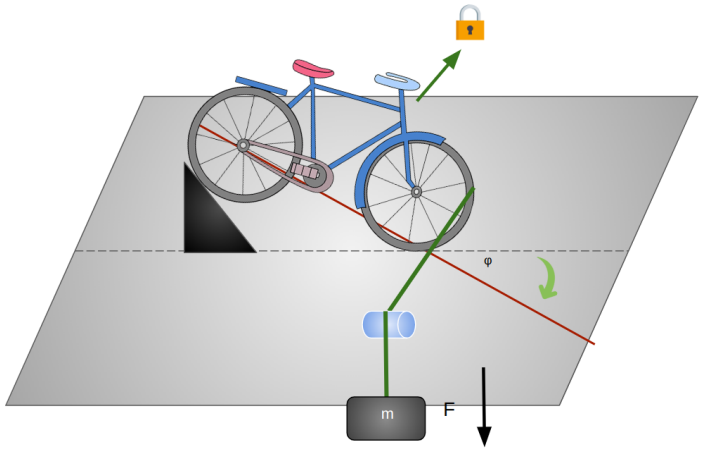

Figure 12: Spring constant experiment

Important things to consider when performing this experiment

- The wire attached to the wheel should be Non-elastic to minimize the effect on the wheel's rotation.
- The wire should be attached to the front edge of the wheel as precisely as possible to ensure that the measurements are consistent and accurate.
- The mass used in the experiment should be large enough to cause a visible rotation of the wheel, but not so heavy that it causes damage to the bike or the equipment.

Experiment 7: Moment of inertia J

The following procedure was used to determine the moment of interia.

1. First, the bike's front wheel was lifted off the ground to ensure that the fork was vertical and not affected by friction from the ground. This was an important step because any friction could have affected the accuracy of the measurements.
2. Next, a wire was wrapped around the fork's rod under the handlebars, and the other side of the wire was connected to a known mass. This allowed the mass to be used to create a rotational force on the fork and wheel assembly.
3. A video was then recorded from above the fork's rotating axis. The mass was released and was run over a rod, resulting in the fork and wheel assembly to rotate. By analyzing the video, the angular acceleration was calculated.

Analyzing the video, the angular acceleration was then calculated. Newton's second law was used to obtain J.

$$T = \alpha \cdot J \qquad J = \frac{T}{\alpha} \qquad J = \frac{m \cdot g \cdot r}{\alpha}, \text{ where r is the fork rod's radius.}$$

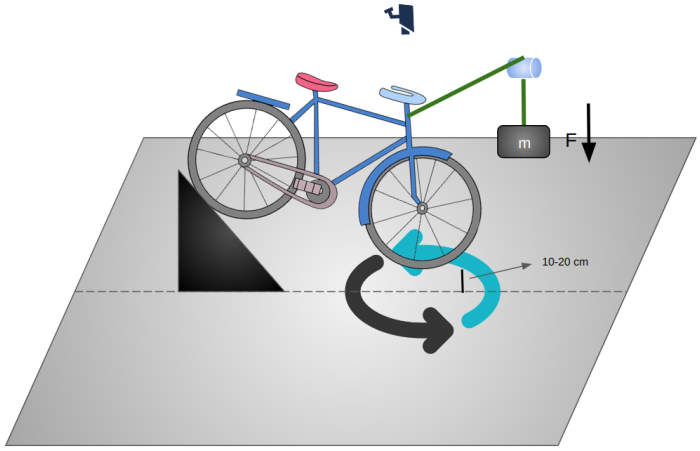

Figure 13: Moment of inertia experiment

Important things to consider when performing this experiment

- The mass used in the experiment should be large enough to create a visible rotation of the fork and wheel assembly, but not so heavy that it causes damage to the bike or the equipment.
- The video recording should be taken from directly above the fork's rotating axis to accurately measure the angular acceleration.

Experiment 8: Damping coefficient b

The following procedure was used to determine the damping coefficient.

1. The bike's front wheel was lifted off the ground to ensure that the fork was vertical and not affected by friction from the ground. This was important because any friction could affect the accuracy of the measurements.
2. The handlebars were locked onto the bike's frame using clamps to prevent any rotation, even when a torque was applied. This ensured that the bike remained stable during the experiment.
3. A laser pointer was mounted on the upper side of the front wheel, pointing in the forward direction of the bike. This allowed the motion of the front wheel to be recorded.
4. The bike was placed pointing directly at a wall. This allowed the motion of the laser pointer dot on the wall to be recorded and analyzed.
5. An impulse was applied to excite the wheel, which allowed the wheel to start oscillating under its natural frequency.
6. The motion of the laser pointer dot on the wall was recorded using a high frame per second camera. This allowed the horizontal coordinates of the motion to be collected with a high degree of accuracy.
7. A Python script 1 was used to collect the motion's horizontal coordinates in a CSV-file. This data was then used to calculate the damping coefficient of the fork.
8. With the use of Matlab, a plot of the horizontal coordinates from the CSV-file was created. The plot was used to analyze the damping of the fork and determine the damping coefficient b.

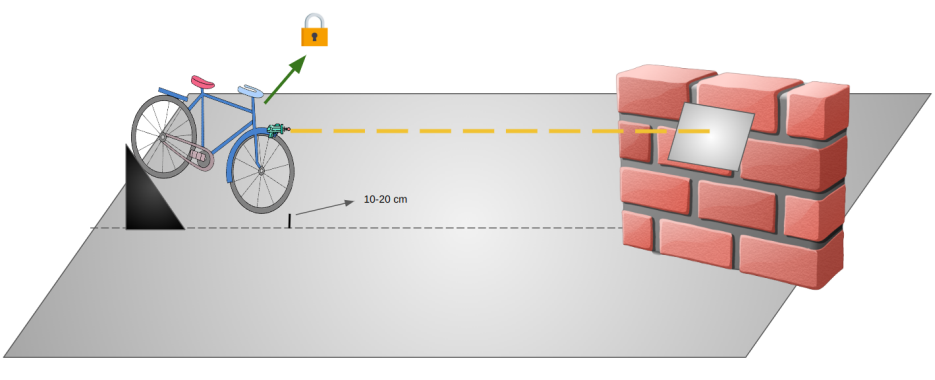

Figure 14: Damping constant experiment

Important things to consider when performing this experiment

- The handlebars should be locked onto the bike's frame using clamps to prevent any rotation even when a torque is applied. This ensures that the bike remains stable during the experiment.
- The laser pointer used to record the motion of the front wheel should be securely mounted on the upper side of the wheel, pointing in the forward direction of the bike. This ensures that the motion is accurately recorded.
- An impulse should be applied to excite the wheel, which allows the wheel to start oscillating under its natural frequency. The amplitude of the oscillations should be kept small enough to ensure that the system remains within its linear range.
- The motion of the laser pointer dot on the wall should be recorded using a high frame per second camera. This allows the horizontal coordinates of the motion to be collected with a high degree of accuracy.

To accurately analyze the motion of a laser point in a video, a Python script (Github, 2018) was adapted using the OpenCV computer vision library . The script incorporates various tracking algorithms to detect and follow the laser pointer dot as it moves across the frame.

The script is designed to be user-friendly, with the option for the user to draw a bounding box around the laser point. This bounding box is used to track the position of the laser point and record it in a data array. The script runs until the user clicks on the Escape button, at which point the data is saved to a CSV file for further analysis.

Overall, this script provides a reliable and customizable tool for analyzing the motion of laser points in video data. The script is availible in Appendix 6.1.

The spring's damping factor b is assumed to have an exponential damping characteristic. The general equation for an exponentially damped sinusoid:

$$\theta(t) = Ae^{-\alpha t}cos(\omega t + \phi)$$

$$\alpha = b/2j$$

$$\omega = \sqrt{k/j - (b/2j)^2}$$

Such that the magnitude θ is equal to $\theta(t) = Ae^{-\frac{b}{2jt}}cos(\sqrt{k/j - (b/2j)^2}t + \phi)$

Since all the parameters in the exponential damping equation were known except for b, various b values were tested until the plot of the exponential damping equation became similar to the data in the CSV file.

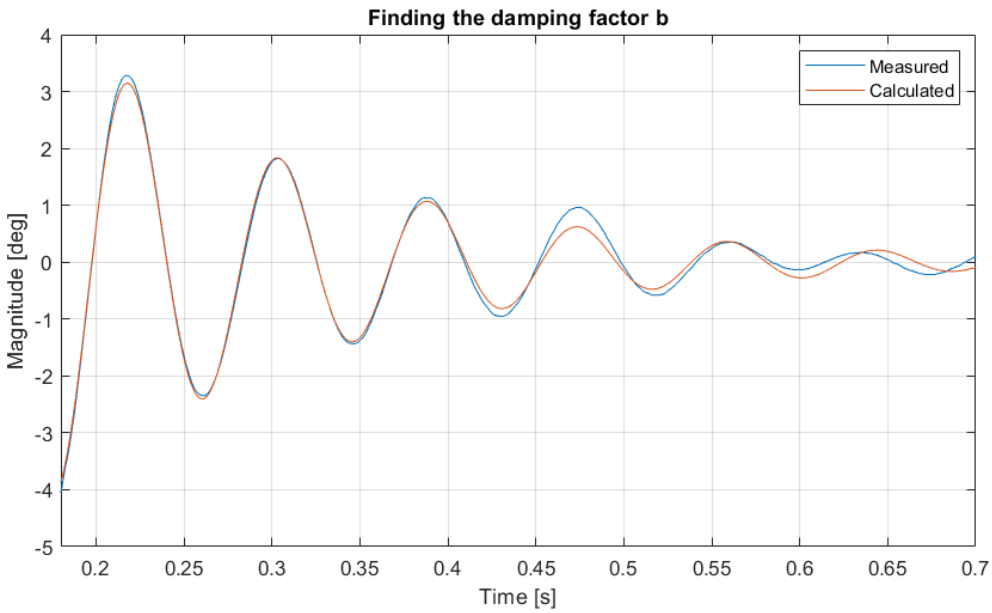

Figure 15: A comparison between the measured data from the laser pointer x-coordinates and the expected response based on an exponential damping equation. The b-value had been tuned until the graphs matched.

2.3 Control System Design Approach

The control system of the bike consisted of two cascade-connected controllers: an outer trajectory controller and an inner balancing controller. The trajectory controller was responsible for controlling the path of the bike, while the balancing controller maintained the bike's balance.

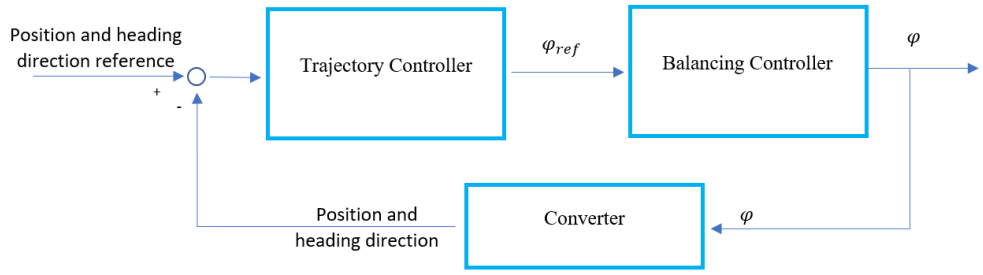

Figure 16: An overview of the bike control

Figure 16 provides an overview of the bike control system. The balancing controller used a feedback control system with a PD controller to keep the true roll angle φ close to the reference roll angle φ_{ref}.

Optimally, the balancing controller should be a feedback PID controller, so that the true roll angle would be close to the reference. However, since the balancing controller was inside an outer trajectory feedback controller there was no need to have the integration part, I, because the outer loop handles steady-state errors.

The fork flexibility added a pole pair with a high imaginary component to the pole-zero map. Having a derivative part in the controller moved these flexibility poles further away in the pole-zero map in the vertical direction. Therefore a P controller was developed. A PD controller was however investigated later on in the project.

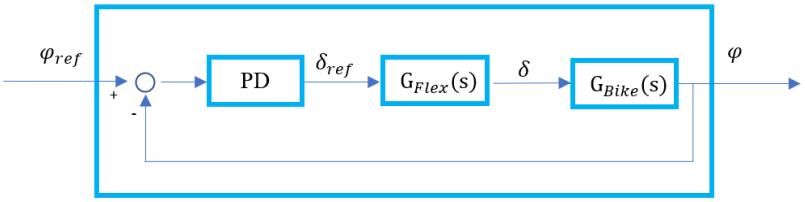

Figure 17: Exploded view of the balancing controller shown in Figure 16. A feedback control system with a PD controller. G_{Bike} is the bike's transfer function. G_{Flex} is the fork's flexibility, δ is the steering angle and φ is the roll angle.

2.3.1 P controller

In the case of a stiff bike i.e. $G_{flex} = 1$, the open-loop system 18 only has one pole in the right half-plane (RHP). Despite this the system was still controllable meaning that the system can achieve self-driving. In the closed-loop system, the open-loop pole was combined with one more pole pair that depended on the value of the proportional coefficient k_p. As k_p was increased to a specific value, the added pole-pair moved from the RHP to the left half-plane (LHP), and the system became stable. Thus, the k_p value was only lower bounded. However, too large k_p resulted in amplified noise which negatively affected the system.

In the case of a plastic bike with added flexibility as a transfer function, the closed-loop system had one more pole-pair than in the metal case, which also depended on the value of k_p. As k_p was lower than a specific value, the new pole-pair moved to the LHP. This meant that the k_p value was both upper and lower-bounded. The k_p value was lower bounded due to the bike's dynamics and upper bounded due to the added flexibility.

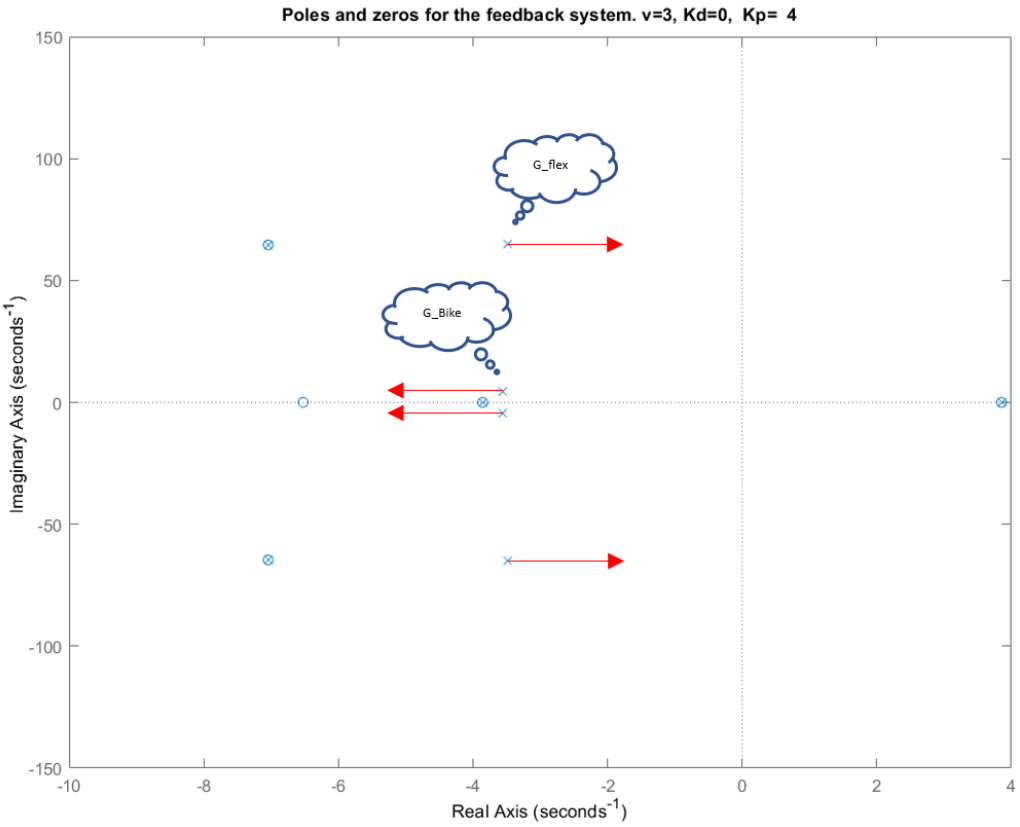

Figure 18: Shows the poles and zeros for the feedback system. i.e $\frac{k_p \cdot G_{flex} \cdot G_{Bike}}{(1+k_p \cdot G_{flex} \cdot G_{Bike})}$. By increasing k_p the poles will move in the red arrows direction and vice-versa.

To determine the optimal k_p value for the balancing controller for a specific velocity, a Matlab function (Function A 2) was constructed. This function took the transfer functions parameters G_{Flex} and G_{Bike} and returned an interval of k_p values that guaranteed system stability. Since the poles mostly moved horizontally in the pole-zero coordinate system when changing k_p, another Matlab function (Function B 2) was constructed that took the k_p interval and returned the optimal k_p value that ensured that the poles were located as far as possible from the imaginary axis (y-axis).

The results for this section can be found in the corresponding section 4.1

2.3.2 PD controller

For the design of the PD controller, the Ziegler–Nichols method was utilized ("Ziegler–Nichols method", 2023). A prerequisite to use this method was that k_p's upper bound as well as the oscillation period time were both known. At k_p's upper bound (k_pmax) the system became marginally stable. This was due to the flexibility poles being situated close to the imaginary axis in the pole-zero map. This caused an oscillating system.

From the previously mentioned Matlab function that found the k_p interval, k_pmax can be denoted as k_u. The period time of the oscillation can be denoted as T_u. T_u was obtained by calculating the natural frequency of the damped motion using $\omega = \sqrt{k/j - (b/2j)^2}$ and therefore $T_u = \frac{2\pi}{\omega}$.

When k_u and T_u are known, a Ziegler table was utilized to obtain the PD-controller parameters. The table is shown below.

Control Type	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	–	–	–	–
PI	$0.45K_u$	$0.8\bar{3}T_u$	–	$0.54K_u/T_u$	–
PD	$0.8K_u$	–	$0.125T_u$	–	$0.10K_uT_u$

Figure 19: Ziegler table ("Ziegler–Nichols method", 2023)

The results for this section can be found in the corresponding section 4.2

2.3.3 Signal delay

The procedure that happens when the bike´s roll angle is changed

1. The bike's roll angle is changed
2. The IMU sensor detects the roll angle
3. MyRio will process this information and send a PWM signal to the ESC
4. ESC will drive the the steering motor
5. When the steering motor is rotating, the encoder detect the handlebars rotation

The considered delay happens between step 2-3. The following procedure was used to determine the delay in the system.

1. The hardware was first mounted onto the bike.
2. Next, the system was turned on.
3. The group then attempted to lean the bike to the right and left multiple times.
4. During these trials, the motions of the handlebar motors were recorded using LabView and saved as a CSV file.
5. The resulting data is now available for analysis.

Important things to consider when performing this experiment

1. While leaning the bike, maintain a consistent and controlled motion to ensure reliable data collection.
2. Let the handlebars be as free as possible to rotate when the bike leans.

The results for this section can be found in the corresponding section 4.4

3 Mechanical design results and analysis

As previously stated the goal was to reduce both play and flexibility in the bike to make it suitable to steer using a control-system. Several tests showed that the bike was improved a substantial amount compared to the previous version of the bike.

It has also been concluded where the hardware components is mounted, and consequently where the center of gravity is located.

3.1 Reduction of play in bicycle

The table shows the results from the experiments procedures described in section 2.1.2, comparing the measurements before and after the improvements.

The table shows a reduction of play in all experiments, which in total resulted in a substantial improvement of the bikes rigidity.

Experiment nr	Before improvements	After improvements
Experiment 1	20 mm	5 mm
Experiment 2	100 mm	12 mm
Experiment 3	50 mm	4 mm
Experiment 4	40 mm	4 mm
Experiment 5	30 mm	3 mm

Table 1: Results from the experiment procedures from section 2.1.2.

3.1.1 Reduction of play analysis

The experiment procedures can be found in section 2.1.2 and 2.2.3.

The improvements in Experiment 1 was likely due to the 3D-printed rings used to eliminate play between the head tube and ball-bearings. Better fastening from using duct tape and reduced length of the forks was also a factor.
As experiment 2 evaluates both the play tested in experiment 1, and the play between the head tube and the frame, it can be deduced that movement between head tube and the frame was the most substantial factor for movement range in this experiment. Because of how the test is constructed, using a small force applied by a human, human error can not be eliminated in that the tubes might flex from the applied force when testing the play. Therefore it is assumed that a portion of the measured 12 mm play is due to flex in the tubes.

The improvements that led to the Experiment 3 result were probably the shortened pipes and the improved fastening of the tubes. Experiment 4 and 5 were similar. These improvements is likely due to the new rear wheel mount and the improved fastening of the tubes.

3.2 Reduction of flexibility in the bike's fork

Table 2 presents the results of the experiments conducted to determine the unknown parameters k, b, and J in the transfer function representing the flexibility of the bike. The experiments were performed twice, and significant improvements were observed between the two sets of results.

Specifically, the value of k increased substantially from 11.296 to 90, representing an improvement of approximately 8 times. The moment of inertia remained constant throughout the experiments. This indicates that the improvements made increased the rigidity, without increasing moment of inertia. Additionally, the damping ratio increased from 0.092 to 0.3, about a three times improvement.

Overall, the results show that both the damping ratio and the value of k increased, which suggests a reduction in the flexibility of the plastic bike. These improvements are important in ensuring that the bike is less flexible then it was before.

Experiment nr	Before improvements	First iteration	Second iteration
Experiment 6	$k = 11.296$	$k = 54.0488$	$k = 90$
Experiment 7	$J = 0.0213$	$J = 0.0213$	$J = 0.0213$
Experiment 8	$b = 0.092$	$b = 0.092$	$b = 0.3$

Table 2: The parameters of the flexibility before doing any improvements, after doing the first iteration of improvements and the final result after the second iteration of improvements. The methods for deriving these parameters can be found in section 2.2.3

3.2.1 Flexibility analysis

In the first iteration of experiment 6,7 and 8. The main improvement is believed to be the new front wheel mounts. This reduced its ability for bending deformation leaving mostly torsional deformation when steering. Shortening of pipes, better fastening of pipes to the joints likely also had minor effects. For the second iteration of tests, the improvements can mostly be derived from the tightening the nuts at the front wheel and the nylon rods. It was discovered that while transporting and rolling the bike, the wheel nuts gradually loosened. Consequently periodic checks and torquing of the wheel nuts were introduced as a precautionary measure.

Because of the large improvement between the first and second iteration, even though rather small changes on the bike, it is suggested that when testing the first time, the wheel nut may not been tightened satisfactory. If the wheel would have been tightened before the first tests, the results might have differed. Though, the end result after all the improvements would still be the same.

3.3 Bike measurements and mounting of electronics

Measurement	Before improvements	After improvements
Wheelbase (b)	158 cm	118,5 cm
Weight	-	22 kg
h	75 cm	66 cm
a	90 cm	46 cm

Table 3: Bike measurements and parameter changes

Before any improvements the bike was relatively large compared to an ordinary bike, with a wheelbase of 158 cm. After cutting the pipes, the wheelbase was 118,5 cm, similar to the self driving metal bike which has a wheelbase of 114 cm.

There was also changes in parameters that is used in the simulation of the bike. The bikes total weight became $22kg$. Variable h which is the height of mass center was reduced from 75 cm to 66 cm. Variable a is the horizontal distance between the mass center and the rear wheel axle and was reduced from 90 cm to 46 cm. This is both due to reducing size and the mounting location of the electronic hardware.

The electronics were mounted similar to how they were mounted on the metal bike, with some modifications. The steering motor was mounted in the same way, a bracket which is fastened to the head tube using hose clamps. The electronics box was mounted on top of the frame, rather than in the middle of the frame. This was both because it did not fit in the frame without modifications to the box, and a high mass center, a, is preferred to make the control system more responsive. The electronics box mounting bracket was cut off by 3 cm to move the box further backwards, to make room for the handlebars.

A simple but important modification was to switch the place of the wheels. The heavier wheel with the hub motor was moved to the back to reduce the moment of inertia in the front wheel and forks. This made the forks flex less and enabled the bike to steer faster.

Figure 20: The finished bike.

In addition to our modifications, a router (RUT955) and two antennas were mounted to the bike by other students, to enable wireless configuration of the bike.

Figure 21: Router and antennas.

A new head tube and fork clamp was designed to further improve the bikes rigidity. These parts were never printed or installed. The head tube was designed with a larger head tube angle. This was to counteract both torsional and vertical flex of the fork. The front was designed to be flat and incorporate holes that align with the steering motor mounting plate.

The fork clamp was designed shorter to be able to shorten the fork tubes even further, and thinner with chamfered edges to be able to freely turn without hitting the frame, considering the reduced head tube angle.

4 Control System Design Results and Analysis

The main goal of the control system design was to develop a controller that was capable enough to allow the bike to achieve self-driving. Self-driving of the bike was never achieved. There were however some smaller incremental improvements which were achieved. This section provides the key results of the control system design.

4.1 P controller

The procedures for conducting these simulations is outlined in section 2.3.1

As mentioned previously, the velocity of the bike was critical for the control system and the self-driving aspect. Figure 22 shows the different velocities and the k_p value intervals associated with each velocity. Low velocities required very high k_p values, However, higher velocities resulted in lower k_p values, that fulfilled the stability condition, which is having the poles at the LHP.

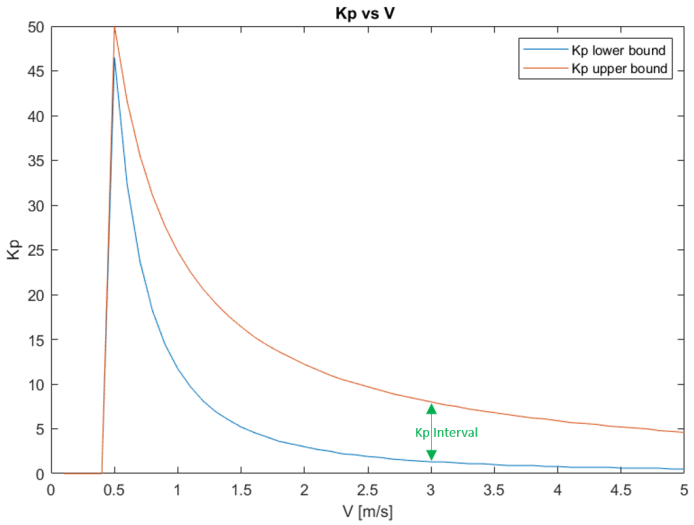

Figure 22: The possible k_p values for different driving velocities. At v = 3 m/s the k_p should be within $[1.3, 8.0]$ to ensure stability. At a low speed such as 0.3 m/s there are no k_p values that achieve stability.

Using the functions A and B 2. The k_p interval for a stable system for v = 3 m/s was determined to be [1.3,8.0] and the optimal k_p value 4.

4.1.1 P controller result analysis

Figure 23 below shows the step response of the system at velocity $v = 3\ m/s$ when the input signal was a unit step. Multiple values for the controller parameter, k_p, were chosen from the interval and plotted in the same figure to compare them.

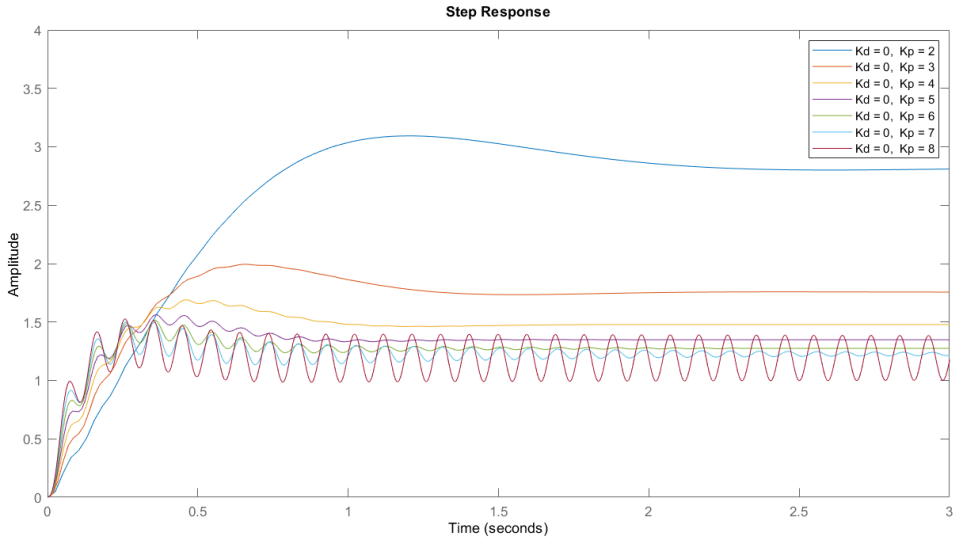

Figure 23: Step response of the feedback system. i.e $\frac{k_p \cdot G_{flex} \cdot G_{Bike}}{(1+k_p \cdot G_{flex} \cdot G_{Bike})}$ using multiple values of $k_p \in [1.3, 8.0]$ and k_d equal to 0.

At $k_p = 8$ the system became almost perfectly oscillating (marginally stable) which indicated that the G_{flex} poles started to intercept with the imaginary axis in the pole-zero map on its way to the RHP. For higher k_p the oscillation was amplified and the system became unstable.

It was also noticeable that a low k_p led to a higher steady-state error with a smooth output signal while a higher k_p led to a lower steady-state error but with large oscillations.

Since all values in the k_p interval guaranteed a stable system, determining the optimal value of k_p was challenging. The function B 2 may not have been the optimal method to find the optimal k_p.

4.2 PD controller

The procedures for conducting these simulations are outlined in section 2.3.2

Adding a derivative part to the controller made the poles move vertically away from the real axis. This led to a higher frequency oscillation in the system response. Since the $G_{flex}(s)$ has a pole pair with a large imaginary component, this will make it difficult to integrate a derivative part into the controller.

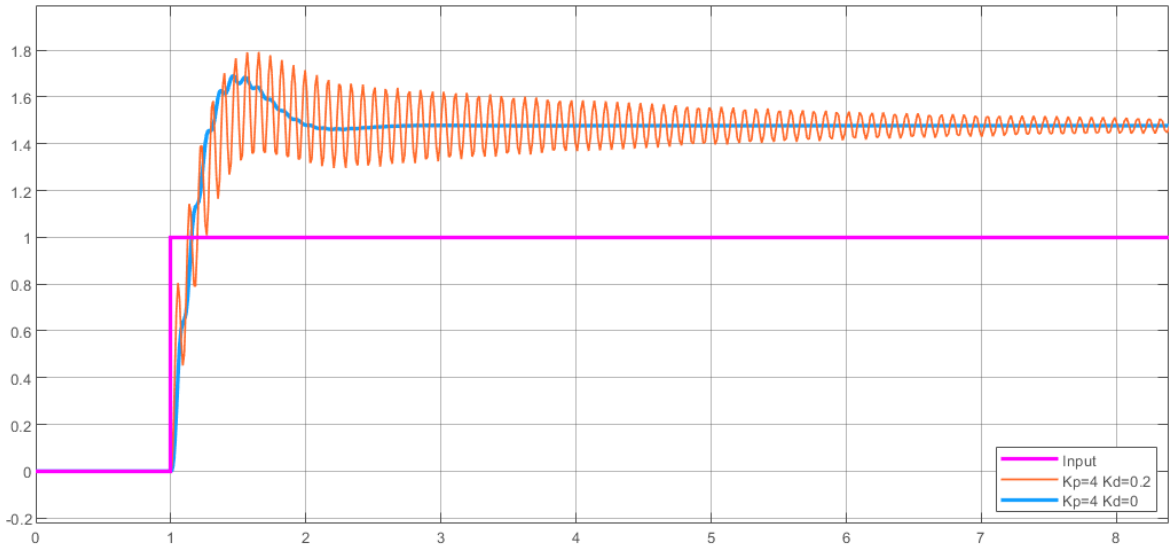

Figure 24: A simulation of the roll angle shows a comparison between a P and PD controllers to show how a little k_d value affects the system.

In order to use the ziegler table 19 the K_u and T_u were calculated to:

$$K_u = 6.4$$
$$T_u = 0.95s$$

Once K_u and T_u were calculated. The ziegler table 19 was used to obtain PD-controller parameters. The results were:

$$K_p = 6.4$$
$$K_d = 0.07$$

4.2.1 The PD controller result analysis

Determining the k_p and k_d values for a stable system can be challenging. Therefore, to determine the values, the Ziegler–Nichols method was used. It is however hard to know if this way is the most suitable to determine the control parameters, or if there is a better way. Adjusting the P-controller with $k_p = 4$ in the following section 4.1 resulted in good control and close to no oscillations. On the other hand, the PD controller with $k_p = 6.8$ and $k_d = 0.07$ was a bit faster and had a lower steady-state error but a lot of oscillation occurs.

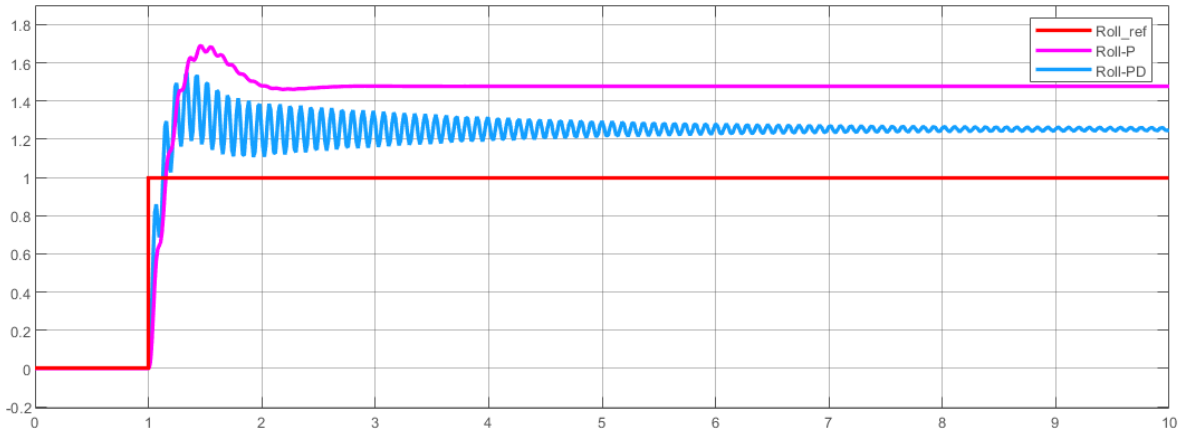

Figure 25: A simulation of the roll angle using two different controllers, P and PD that have been designed.

4.3 Flexibility's affect on the reference angle δ_{ref}

The input signal for the flexibility transfer function $G_{flex}(s)$, denoted as the reference steering angle δ_{ref}, is transformed into the output signal denoted as δ (The true steer angle at the front wheel). In Figure 26, a comparison between the input and output signals demonstrates the impact of flexibility on the bike's performance while driving the bike.

Due to this flexibility, the bike's front wheel does not precisely follow the reference angle as shown in Figure 26, which may result in instability during operation.

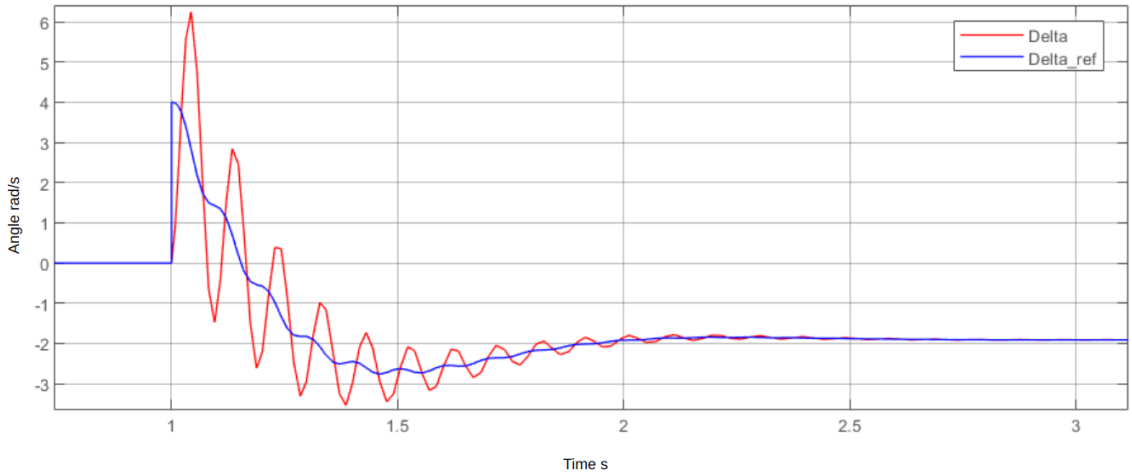

Figure 26: A simulation shows δ_{ref} as an input signal to the G_{flex} that represents the reference steering angle while δ represents the true steering angle on the front wheel.

The discrepancy between the angle δ and the reference angle δ_{ref}, as illustrated in Figure 26, can be related to the amplitude amplification at this particular frequency as shown in Figure 27. As mentioned in section 2.2.2, the natural frequency of the transfer function for flexibility can be determined using the following equation:

$$\omega = \sqrt{\frac{k}{j} - \left(\frac{b}{2j}\right)^2}$$

By incorporating the values for the second iteration of improvements from Table 2, the natural frequency of the fork is calculated to $\omega \approx 65 rad/s$. This frequency creates the notch in the blue graph in Figure 27.

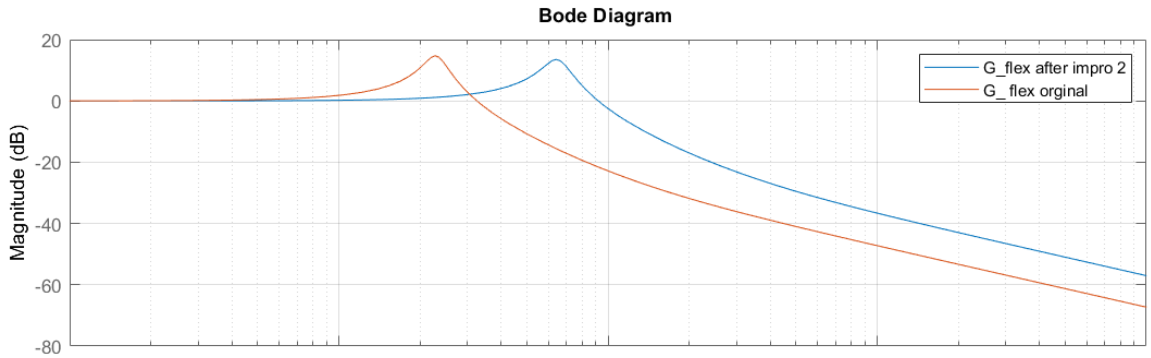

Figure 27: The blue graph shows the bode diagram of G_{flex} after the second iteration of improvements. Its resonance frequency is 65 rad/s. While the orange one shows the bode diagram of G_{flex} before improvements and its resonance frequency was 14 rad/s. This confirms that the improvements made the fork much stiffer.

4.3.1 Result analysis of the flexibility

In order to eliminate the signal's oscillation amplification that appears in Figure 26, some physical structure changes to the fork has to be done. This can be done by changing the material or geometry of the bike, making it stiffer. A stiffer bike is likely to bring with it a bode diagram moved further to the right, resulting in decreased amplification of the oscillation and enables a faster turning rate of the bike. This causes the system to become more robust.

4.4 Signal delay

The experimental procedures for conducting this experiment are outlined in section 2.3.3. Figure 28 displays the Pulse Width Modulation (PWM) signal (depicted in blue), Additionally, the graph presents the Y-axis acceleration data obtained from the Inertial Measurement Unit (IMU) sensor (shown in yellow). The calculated delay between these signals is approximately 30-40 ms.

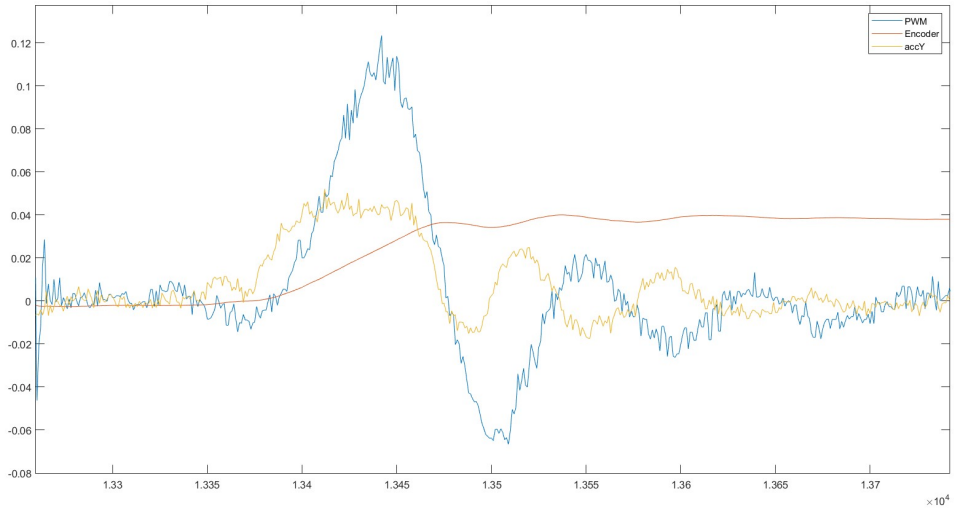

Figure 28: The data shown here has been logged by the real controller. Blue PWM, yellow Y-axis acceleration and red handlebars' angle.

4.4.1 Signal delay result analysis

The system's required response time is inversely proportional to the bike's velocity. This means lower velocity needs faster response time. Whether a 30-40 ms delay is enough to cause a problem depends on the velocity of the bike but likely it does not cause any problems at most velocities.

4.5 Real world tests

Both a P and a PD-controller was designed and tuned based on the simulations. Both were then tested in a real world test on the actual plastic bike. The tests were performed by letting a person run alongside the bike while holding it. When the desired speed was reached the person briefly released the bike for it to control itself.

4.5.1 Primary real world test - P-controller

When testing the bike with a P-controller, the LabVIEW code (Github, Autobike) only included the balancing controller with a roll reference equal to zero as well as a complementary filter that estimated the system's states such as speed, acceleration, roll angle, etc.

This test was performed after the first iteration of structural improvements which limited the k_p interval that was calculated using Function A described in 2. The k_p interval was [1.3, 2.4] for a velocity of 3 m/s. According to the simulation, the bike should be able to achieve self-driving. However, in this real-world test it did not.

4.5.2 Secondary real-world test - PD-controller

This test was performed after the second iteration of structural improvements. As the fork was now stiffer with a higher value for the spring constant k, it was now possible to increase the controller parameters k_p and k_d. They were now set to 6.4 and 0.07 respectively. The LabVIEW code used in this test was almost the same as in the primary running test, however, the complementary filter had been replaced by a Kalman filter to obtain more accurate state estimations. The Kalman filter construction was not a part of this project. It was created and implemented in LabVIEW by other students in another project. Once again the simulation suggested that the bike would be able to achieve self-driving, but this was not the case in the real-world testing.

4.5.3 Real world test result analysis

Using a low k_d equal to (0 or 0.07) and $k_p \in [1.3, 8]$ the system reacted slowly. This was due to the flexibility of the fork, preventing us to increase the controller parameter to make the controller react faster.

A general problem with these tests was that the bike needed speed to be able to achieve self-driving. The problem occurred when the person released the bike while running. It is almost impossible to release the bike without affecting the roll angle (pushing it side-ways) which resulted in a disturbance being introduced to the system in every test.

The head tube angle for the plastic bike was smaller than one of a regular bike. The smaller the head tube angle is, the more power is required from the steering motor to rotate the handlebars because rotating the fork leads to changing the height of the center of mass. This angle leads to increased rotational force on the fork on top of the calculated one from inertia. The angle also makes the bike's center of mass oscillate vertically because of the fork acting as a bending spring. This motion was not modeled in the simulation and might be one of the factors causing the bike not to behave in the real world as it does in the simulation.

After the tests were conducted it can be concluded that the bike can balance for a very short time. As soon as a disturbance occurs the bike starts to oscillate and fall which means that the system is not robust, and the bike does not achieve self-driving.

5 Discussion and further development

The real system was not stable in our running tests. The primary reason for the instability was a slow system caused by the limitations that the flexibility created, specifically the fork. Although we modeled the system as a transfer function and tuned the controllers based on that model. The running test showed that the flexible bike did not behave as the simulations. This suggests that our model may not have accurately captured all the relevant dynamics of the flexible bike.

The results suggest that using a P or PD controller may not have been the most effective approach for our application. A different control algorithm may have been more suitable to handle the bike's flexibility. Using a derivative term in the controller may not be a good idea in the case of a flexible bike because the faster response time caused oscillations without any noticeable improvement in stability.

The system exhibited around a 30-40 ms delay, which can be considered relatively low. However, the delay might still have contributed to the observed instability, especially when combined with the bike's flexibility. This delay could make it challenging for the system to react quickly enough to maintain stability when traveling at lower velocities. Further investigation would be needed to determine how delay influenced the overall system performance. The same hardware were mounted and tested on a metal bike and the delay did not affect the stability. Therefore the delay was considered a cause to instability.

The current head tube angle in the bike design is relatively small, resulting in significant up and down oscillations from bumps during operation. These oscillations can introduce disturbances that may negatively affect the system's stability. By decreasing the fork angle, the bike's susceptibility to such disturbances could be reduced, potentially leading to a more stable and better-performing system. We did not model the bumping up and down explicitly, which might have contributed to the discrepancy between our modeling results and the observed real-world performance.

Future research should investigate the effect of the vertical oscillations on the IMU sensor data and system performance more thoroughly. By incorporating the oscillating dynamics into the model, the control system can be designed to better account for and mitigate the disturbances caused by the bike's movement. Reducing the oscillations, either by modifying the fork angle or through other design improvements, could result in more reliable sensor data and ultimately lead to better overall system performance. Additionally, it would be valuable to explore alternative sensor technologies that may be less susceptible to the impact of oscillations and other disturbances.

In future development of the bike, several improvements can be considered to enhance the bike's performance and stability. Firstly, the joints can be 3D-printed with tighter tolerances for the tubes, which would reduce any undesired movement and increase the rigidity of the overall structure. Secondly, mounting holes could be designed to match the screw diameter, allowing the screws to thread the plastic and lock securely in place. This would provide a more robust connection between components and reduce the likelihood of unintended movements or vibrations. It is also important to remember to periodically check and tighten the wheel-nuts for the front wheel.

To counteract the rocking and swinging of the electronics box a better solution to fix the box to the frame or rear joint needs to be designed. A suggestion is that the "universal mount" is redesigned in a way that incorporates the flat surface of the rear joint. It also needs to clamp harder onto the rod. To fix the problem with the steering motor fastening and flexing of forks both torsionally and vertically, it is suggested that the redesigned head tube should be printed and used.

Things to consider to reduce the flexibility of the fork
Another potential suggestion for further development would be to explore the use of a stiffer plastic or metal material for the fork. By selecting a material or redesign the fork with less flexibility, the fork's contribution to the system's instability could be minimized. This would simplify the modeling process. Investigating various materials and their properties could help identify the best option for achieving the desired balance between flexibility, durability, and weight.

The graph in left side of Figure 29 illustrates that as the damping constant b increases, the notch in the diagram becomes smaller, indicating a lower signal amplification as shown in Figure 26. The graph on the right side of the figure demonstrates that increasing the spring constant k results in a higher notch frequency for the system, which means the further away to the right side the notch is located. This brings with it the ability to control the system at a faster rate.

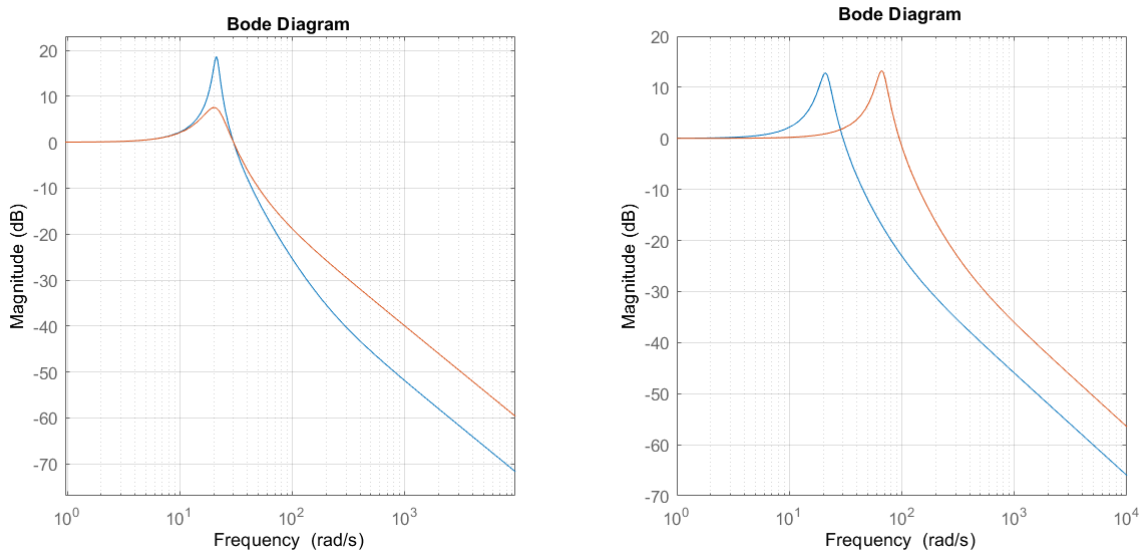

Figure 29: Shows how the bode diagram of the flexibility transfer function changes when k and b increases or decreases. The left graph shows that by increasing the value for b the blue graph will flatten out to behave more as the orange one. On the other hand, the Figure to the right shows that increasing the k value of the blue graph push it to the right to act more as the orange one.

References

[1] Åström, K. J., Klein, R. E., Lennartsson, A. (2005). *Bicycle dynamics and control: adapted bicycles for education and research,* , IEEE Control Systems Magazine, vol. 25, no. 4, pp. 26–47. doi: https://doi.org/10.1109/MCS.2005.1499389

[2] Choucha, L., Smajic, E. (2021) *Design of light, crash-worthy, self-driving bike.* (Bachelor's Thesis). Gothenburg: Department of electrical engineering, Chalmers. Availible: https://hdl.handle.net/20.500.12380/304403

[3] Guanzheng, W., Sjöberg, J. (2022). Lateral Control of a Self-driving Bike. *International Conference on Vehicle Electronics and Safety, ICVES 2022*, doi: https://doi.org/10.1109/ICVES56941.2022.9986548

[4] Catia. (2019) *Catia V5* (Version 6R2019)[Windows program] Dassault Systems

[5] Cura. (2023) *Cura*(Version 5.2.2)[Windows program] Ultimaker

[6] Lennartsson, B. (2000). *Reglerteknikens grunder (4th ed.)*. Lund: Studentlitteratur AB.

[7] Strömberg, A., Leikvik, T. (2022) *Construction of universal mount and design of new hardware layout.* (Bachelor's Thesis). Gothenburg: Department of electrical engineering, Chalmers. Availible: https://hdl.handle.net/20.500.12380/305444

[8] Erdinç, U. (2019) *Modelling, Validation and Control of an Autonomous Bicycle.* (Master's Thesis). Gothenburg: Department of electrical engineering, Chalmers. Availible: https://hdl.handle.net/20.500.12380/257311

[9] Nakhmani, A. (2020). Kalman Filter. In *Modern Control: State-Space Analysis and Design Methods (1st Edition)*. New York: McGraw Hill. Retrieved 2023-05-05 from https://www.accessengineeringlibrary.com/content/book/9781260459241/chapter/chapter10

[10] *Klämbussning Taperlock*. Downloaded 2023-05-08. https://norelem.se/sv/Produkt%C3%B6versikt/System-och-komponenter-f%C3%B6r-maskin--och-anl%C3%A4ggningskonstruktion/23000/Sp%C3%A4nnsatser-axelnav/Koniska-sp%C3%A4nnbussningar/Kl%C3%A4mbussning-Taperlock-typ-1108-gr%C3%A5j%C3%A4rn/p/23200-0382522

[11] *Kuggremsskiva*. Downloaded 2023-05-15. https://norelem.se/sv/Produkt%C3%B6versikt/System-och-komponenter-f%C3%B6r-maskin--och-anl%C3%A4ggningskonstruktion/22000/Kuggremmar-kuggremskivor/Kuggremskivor-profil-HTD-5M-f%C3%B6r-montering-med-koniska-sp%C3%A4nnbussningar/Kuggremskiva-st%C3%A5l-fosfaterad/p/22005-0515040

[12] Github *OpenCV-Object-Tracking* Retrieved 2023-03-05 from https://github.com/ehsangazar/OpenCV-Object-Tracking

[13] Github *Autobike* Retrieved 2023-04-05 from https://github.com/Autobike/Autobike

[14] Ziegler–Nichols method, Wikipedia, last updated 2023-02-19, Retrieved 2023-05-10 from https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method#cite_note-1

6 Appendix

6.1 Python script to follow the laser dot in experiment 8

```
import cv2
import sys
import numpy as np
import pandas as pd

(major_ver, minor_ver, subminor_ver) = (cv2.__version__).split('.')

if __name__ == '__main__' :

    # Set up tracker.
    # Instead of MIL, you can also uses
    tracker_types = ['BOOSTING', 'MIL','KCF', 'TLD', 'MEDIANFLOW', 'CSRT', 'MOSSE']
    tracker_type = tracker_types[5]

    if int(minor_ver) < 3:
        tracker = cv2.Tracker_create(tracker_type)
    else:
        if tracker_type == 'BOOSTING':
            tracker = cv2.TrackerBoosting_create()
        if tracker_type == 'MIL':
            tracker = cv2.TrackerMIL_create()
        if tracker_type == 'KCF':
            tracker = cv2.TrackerKCF_create()
        if tracker_type == 'TLD':
            tracker = cv2.TrackerTLD_create()
        if tracker_type == 'MEDIANFLOW':
            tracker = cv2.TrackerMedianFlow_create()
        if tracker_type == 'CSRT':
            tracker = cv2.TrackerCSRT_create()
        if tracker_type == 'MOSSE':
            tracker = cv2.TrackerMOSSE_create()

    # Read video
    video = cv2.VideoCapture("./videos/video.mp4")

    # Exit if video not opened.
    if not video.isOpened():
        print("Could not open video")
        sys.exit()

    # Read first frame.
    ok, frame = video.read()
    if not ok:
        print('Cannot read video file')
        sys.exit()

    # Define an initial bounding box
    bbox = (287, 23, 86, 320)

    # Uncomment the line below to select a different bounding box
    bbox = cv2.selectROI(frame, False)

    # Initialize tracker with first frame and bounding box
    ok = tracker.init(frame, bbox)

    data    = []
    counter   = 0

    while True:
```

```
        # Read a new frame
        ok, frame = video.read()
        if not ok:
            break

        # Start timer
        timer = cv2.getTickCount()

        # Update tracker
        ok, bbox = tracker.update(frame)

        # Calculate Frames per second (FPS)
        fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer);

        # Draw bounding box and save the data into an array
        if ok:
            p1 = (int(bbox[0]), int(bbox[1]))
            p2 = (int(bbox[0] + bbox[2]), int(bbox[1] + bbox[3]))
            middle_point_x = p1[0] + ((p2[0] - p1[0])/2)
            data.append(middle_point_x)
            counter += 1
            if counter == 480: break

            cv2.rectangle(frame, p1, p2, (255,0,0), 2, 1)
        else :
            # Tracking failure
            cv2.putText(frame, "Tracking failure detected", (100,80), cv2.
    FONT_HERSHEY_SIMPLEX, 0.75,(0,0,255),2)

        # Display tracker type on frame
        cv2.putText(frame, tracker_type + " Tracker", (100,20), cv2.
    FONT_HERSHEY_SIMPLEX, 0.75, (50,170,50),2);

        # Display FPS on frame
        cv2.putText(frame, "FPS : " + str(int(fps)), (100,50), cv2.
    FONT_HERSHEY_SIMPLEX, 0.75, (50,170,50), 2);

        # Display result
        cv2.imshow("Tracking", frame)

        # Exit if ESC pressed and create the CSV file
        k = cv2.waitKey(1) & 0xff
        if k == 27 :
            data = np.array(data)
            x = np.linspace(0, 1, len(data))
            y = np.array(data)
            df = pd.DataFrame({'x': x, 'y': y})
            df.to_csv("data.csv", index=False)
```

Listing 1: The Python script provided is used to extract the motion of a dot from a video and save the data in a CSV file

6.2 Matlab code

The function A starts at the row 24. The function B starts at the row 61.

```
clc
clear
% Dimensions
a = 0.46;   % the horizental distace between the mass center and the rear wheel axle
b = 1.185;  % wheel-base
h = 0.66;   % mass center's height
% Other parameters
v = 3;      % the driving speed in [m/s]
g = 9.81;   % the gravitation constant
% Flexibiltiy
j   = 0.0213;       % the monment of inertial of the front wheel
b_damping   = 0.3;  % forks damping constant
k   = 90;           % forks constant

% the transfer function of a bike with negligable fork's flexibility
G_bike = ((a*v)/(b*h))*tf([1 v/a],[1 0 -g/h]);
% the transfer function of the fork's flexibility
G_flex  = tf([b_damping k],[j b_damping k]);

% An example of how to use the functions
Kp_interval = find_kp_interval(G_bike,G_flex)
Optimal_Kp = opt_kp(G_bike,G_flex,Kp_interval)

%% Find kp interval (Function A)
% The function check for which kp values the system is stable by looking
% at the number of pole in the rhp. as soon as we get more than one pole
% in the rhp the system will be conciderd as unstable.
% Kp values are tested between 0.1 upp to 50

function kp_interval = find_kp_interval(G_bike, G_flex)
    N=500;
    kp_itr=zeros(N,1);
    kp_interval=zeros(1,2);
    for n=1:N
        kp=0.1*n;
        Plant= (kp*G_flex*G_bike)/(1+kp*G_flex*G_bike);
        p=pzmap(Plant);
        flag=0;
        for i=1:size(p)
            if real(p(i))<0 % if stable
                flag=flag+1;
                if flag == 7
                    kp_itr(n)=kp;
                end
            end
        end
    end
    init=0;
    for i=1:size(kp_itr)
        if kp_itr(i)>0
            if init==0
                kp_interval(1)=kp_itr(i);
                init=1;
            end
        end
        if i>2 && kp_itr(i)==0 && kp_itr(i-1)>0
                kp_interval(2)=kp_itr(i-1);
        end
    end
end
```

```
%% Optimum kp value (Function B)
% Since the poles moves almost horizentally, this function test the kp values
% in the "kp_interval" to find the optimal value where all the pole that
% affected by Kp located at a line parallel to the imaginary axis.
% In other word we are optimazing for the largest shortest distance between
% the feedback poles and the imaginary axis.
% Note: This function return the optimum Kp or at least granteed that the
% feedback poles are located at least at the same distance as the G_bike's
% open loop's pole that already exist in the RHP

function optkp = opt_kp(G_bike,G_flex,kp_interval)
    optkp=0;
    kp=kp_interval(1);
    while true
       Plant=(kp*G_flex*G_bike)/(1+kp*G_flex*G_bike);
       p=pzmap(Plant);
       shortest1=10000;
       for i=1:size(p)
          if abs(real(p(i))) < shortest1
              shortest1 = abs(real(p(i)));
          end
       end
       kp=kp+0.01;
       Plant=(kp*G_flex*G_bike)/(1+kp*G_flex*G_bike);
       p=pzmap(Plant);
       shortest2=10000;
       for i=1:size(p)
          if abs(real(p(i))) < shortest2
              shortest2= abs(real(p(i)));
          end
       end
      if shortest2 < shortest1
          optkp=kp-0.01;
          disp(['The poles are located at a line prallell to y-axis at distance of  ',
     ...
              num2str(shortest1), ' and the optimal kp is ',  num2str(optkp)])
          break
      end
      if kp>kp_interval(2)
          disp('The optimal Kp is located outside the given Kp interval')
          break
      end
    end
end
```

Listing 2: The matlab code includes the function A and function B

6.3 Software programs

6.3.1 LabVIEW

LabVIEW is a software program that stands for Laboratory Virtual Instrument Engineering Workbench. It is a graphical programming language that is utilized by National Instruments for various industrial automation applications and instrument control. Engineers find it easier to program applications in LabVIEW by simply dragging and dropping graphical icons. These software components are represented as graphical icons within LabVIEW, including data structures, functions, and loops.

With the assistance of the LabVIEW application, engineers can create and customize their own applications without requiring extensive programming experience. Moreover, LabVIEW offers comprehensive tools such as built-in support for common hardware devices like actuators and sensors. In summary, LabVIEW is a powerful software program that is widely used for developing and customizing applications in various industrial sectors. Labview was used to implement the controller into the bike, and to read data from the bike.

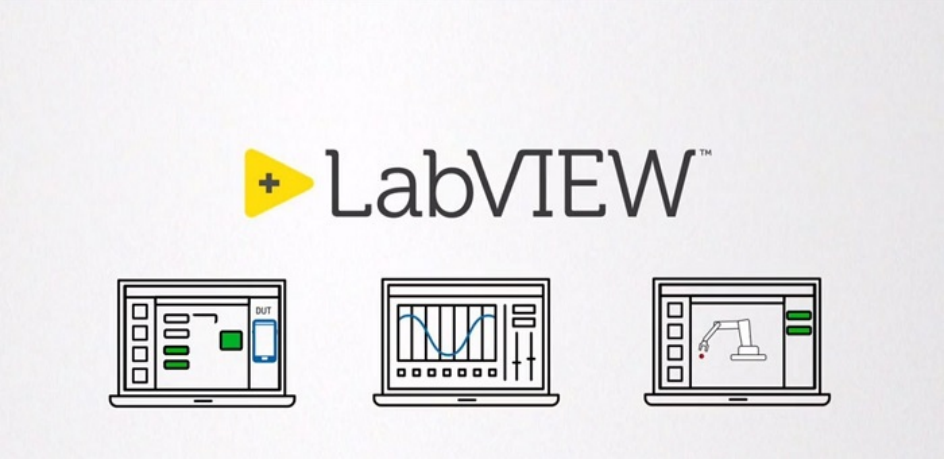

Figure 30: LabView

6.3.2 Matlab/Simulink

Matlab is a versatile high-level programming language designed for the manipulation of numerical data. Simulink, on the other hand, is a powerful graphical programming software used for modeling complex systems and analyzing dynamic systems. Simulink offers a graphical interface and pre-built blocks that simplify the creation of complex models. These tools are widely used by engineers and scientists to develop and test advanced technologies, making them an indispensable part of the research and engineering community. Matlab and Simulink was used to simulate how changes affected the bike dynamics and to identify suitable parameters for the controller. These simulations were used to predict the behavior of the bike with different controllers and bike geometries.

Figure 31: Matlab-Simulink

Figure 32: Timing belt pulley, article 22005 [11]

Figure 33: Clamping bushing, article 23200 [11]