DEGREE PROJECT IN MECHANICAL ENGINEERING,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2019

A Gimbal Stabilizer

Self-stabilizing platform for holding objects horizontally stable

MAKSIMS SVJATOHA

BEHNAM YOSEF NEZHAD ARYA

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

A Gimbal Stabilizer

Self-stabilizing platform for holding objects horizontally stable

MAKSIMS SVJATOHA
BEHNAM YOSEF NEZHAD ARYA

Bachelor's Thesis at ITM
Supervisor: Anders Gabrielsson, Syntronic AB
Examiner: Nihad Subasic

TRITA-ITM-EX 2019:43

Abstract

This bachelor thesis is about the design and construction of a self-stabilizing platform. The purpose of this system is to balance objects placed upon the platform by keeping the platform level regardless of how the mechanism itself is rotated. Its uses include stabilization of sensors, cameras and vehicle cockpits.

The prototype was constructed using 3D printing and basic machining. It uses two DC motors, an inertial measurement unit, an Arduino Uno microcontroller and a motor driver. The inertial measurement unit acts as an accelerometer and gyroscope, it measures the change in position and angle relative to its starting position. The controller algorithm processes the sensor signal and calculates an appropriate output signal. This output signal regulates the two DC motors in such a way that compensates for any angle changes in the platform.

This project is based on, and is the continuation of the work by J. Larsson, titled "Gimbal stabilizer for cockpit bases of terrain vehicle or combat boat: A proof of concept". The task is to develop it to a functioning physical prototype and implement a control system which is fast, responsive and precise.

The controller tuning process involved a trial and error approach, using binary search between parameters that give a performance that is too slow and a performance that is too fast and unstable. A satisfactory performance was achieved and the platform could effectively stabilize objects that weigh 400 grams at its center, 200 grams at its edges and 100 grams at its corners. This takes 100 milliseconds on average. Besides bearing loads, the platform could also compensate for sudden forced angle changes and any tilting of the mechanism the platform is attached to.

Keywords: mechatronics, stabilizer, self-stabilizing, feedback control, PID, Arduino, inertial stabilization.

Referat

En självstabiliserande plattform med lastbärande förmåga

Det här kandidatexamensarbetet handlar om konstruktionen och framtagningen av en prototyp för en självstabiliserande plattform med ändamålet att hålla ett objekt horisontellt stabil. Det innebär att oberonde av hur mekanismen vrids eller rör sig kommer plattformen alltid vara parallell med marken. Appliceringsområden för plattformen kan vara att stabilisera kameror och sensorer, samt att hålla förarkabinen i en båt horisontellt stabil oavsett hur vattnet runtomkring rör sig.

Prototypen består av en 3D-printad mekanism med två likströmsmotorer, en tröghetssensor, Arduino och motordrivare. Tröghetssensorn fungerar som en accelerometer och gyroskop - den avläser hur mycket plattformen ändrar sitt läge och vinkel relativt de ursprungliga. Data från tröghetssensorn bearbetas i en Arduino Uno-mikrokontroller med hjälp av reglerteknik, där en så kallad PID-regulator beräknar utsignal beroende på insignal. Arduinon skickar sedan utsignalen till motordrivaren som reglerar likströmsmotorerna för att kompensera för eventuella vinkeländringar.

Detta projekt är ett utvecklingsarbete av ett tidigare masterexamensarbete av J. Larsson, med titeln ’ ’Gimbal stabilizer for cockpit bases of terrain vehicle or combat boat: A proof of concept”. Målet är att gå från ett koncept till en praktisk och verklighetsbaserad prototyp, samt att designa en regulator med god precision, stabilitet och responstid.

För att bestämma hur regulatoralgoritmen ska fungera användes binärsökning för att hitta vilka P, I och D-värden PID-regulatorn ska ha. Det innebär att man ständigt tar ett medelvärde av en för långsam regulator och en som är för snabb och aggressiv tills önskad prestanda uppnås.

Den slutgiltiga prestandan ansågs vara tillräcklig och plattformen kunde effektivt stabilisera 400 gram i mitten på den, 200 gram på dess sidor och 100 gram på dess hörn på ungefär 100 millisekunder. Vidare kunde den kompensera för plötsliga påtvingade vinkeländringar och för lutning hos mekanismen som den sitter fast i.

Nyckelord: mekatronik, stabilisering, plattform, självstabiliserande, återkoppling, reglerteknik, PID, Arduino.

Acknowledgements

This Degree Project in Mechatronics, First Cycle was carried out during the spring semester 2019 and was submitted to Syntronic AB and KTH Royal Institute of Technology for the Degree of Bachelor of Science in Mechanical Engineering.

Our thanks and appreciation go to our supervisor Anders Gabrielsson who gave us a rewarding challenge and support throughout the entire period of this project.

We are grateful for the valuable lectures and excellent guidance and knowledge provided by our examiner Nihad Subasic.

The assistance provided by Staffan Qvarnström and Thomas Östberg with machining, finding suitable components and general help with the construction was invaluable and instrumental to achieving a functioning prototype.

We would also like to thank our teaching assistants Seshagopalan Thorapalli Muralidharan and Sresht Iyer for tremendously valuable assistance in this project.

Stockholm, May 2019

Maksims Svjatoha
Behnam Yosef Nezhad Arya

List of Abbreviations

CAD	Computer-aided design
DC	Direct Current
DoF	Degree of Freedom
IMU	Inertial Measurement Unit
ISP	Inertially Stabilized Platform
NC	Numerical Control
PID	Proportional-integral-derivative
PLA	Polylactic acid
SISO	Single Input, Single Output

Contents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Self-stabilizing mechanisms are mechanisms that try to remain stationary with respect to an external frame of reference, e.g. the Earth, even if their local frame of reference is moving in relation to it. Practical examples of this include balancing a pole inside a moving train or a camera on a tilting table, the train and the table being examples of moving local reference frames. This is called inertial stabilization.

The uses may vary and examples include consumer applications such as self-stabilizing cameras, spoons that compensate for impaired motor skills and food and drink trays that ease the strain on the holder. Other examples include military applications such as stabilizing cockpits of combat boats and terrain vehicles. In this case this mechanism will be a platform. These are often referred to as Inertially Stabilized Platforms [ISP][1]. .

1.2 Purpose

The purpose of this study is to build upon previous work on an original prototype self-stabilizing platform by J. Larsson and to verify any previous conclusions, if relevant. The end goal is to have a standalone, functioning physical prototype, which can be scaled and integrated into different systems that require inertial stabilization.

Despite sound theoretical models, the previous work did not produce a working physical model. This was mostly due to mechanics-related issues and problems during the construction of the model itself. The main task was therefore to examine the previous model and either improve it or suggest a different solution altogether.

The main research questions are as follows:

- How to rework the proof of concept from the master thesis by J. Larsson into a stable, practically functioning prototype?
- How to design and implement a control system which is both fast, responsive and precise?

The bonus research questions are:

- How much mass can a platform effectively stabilize?
- Suggest some area of application.

1.3 Scope

The end goal is to have reasonable physical model with self-stabilizing properties. Due to skill, knowledge and time constraints the scope will be limited to satisfying these demands on a basic level, with any additional time available being put into optimization of the model. More advanced parts of the project will use open source material, such as Arduino codes and libraries.

The model should be scalable to both larger and smaller applications, but the physical prototype used in this project will likely have more modest dimensions. Yaw is not considered important to control and the nature of the system makes it impossible to compensate for linear displacement, nor is it within the scope of this project.

This project is a bachelor's thesis in cooperation with Syntronic AB.

1.4 Method

The suggested model should have three layers - one for the roll motion, one for the pitch motion and a third, outer layer to keep the first two layers in place. The inner layer should be the inertially stable plate itself and the two other layers should be designed in such a way that the system stability is maximized, giving increased control over it. The final layer should be robust and rigid in order to support the rest of the ISPs mass and resist any forces try to affect it. See Figure 1.1 for an early concept model.

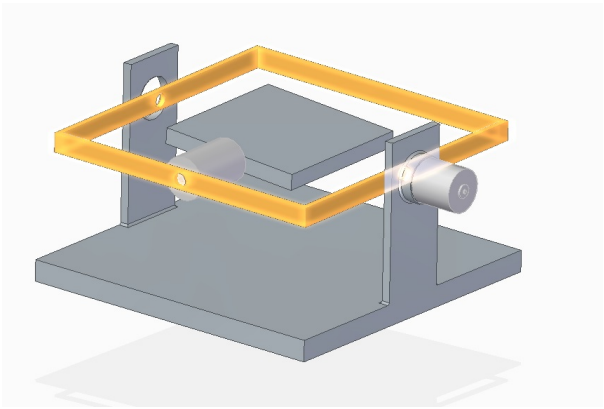


Figure 1.1: The plate in the middle, attached to the frame (marked in yellow) via a DC motor. The outer motor is attached to a stationary platform. Designed in Solid Edge.

1.5 Related Projects

This project is based on, and is the continuation of the work of J.Larsson, titled "Gimbal stabilizer for cockpit bases of terrain vehicle or combat boat - A proof of concept" [2]. A CAD model of the ISP can be seen in Figure 1.2.

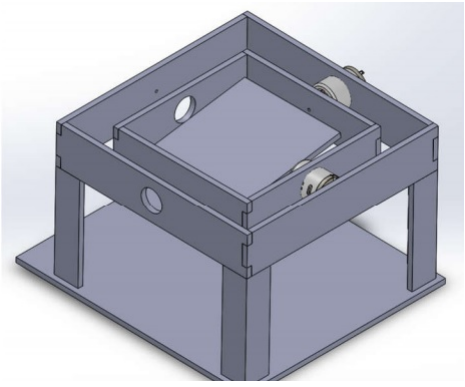


Figure 1.2: CAD model of previous work by J. Larsson [2].

While the theoretical work done by the preceding project is sound, other projects have also been examined to compare and examine possible alternative model and controller designs.

These mainly include ”Self-stabilizing platform - ZTäBiLAjZöR” by A. Karlsson and J. Cressell [3], and ”The Stabilizing Spoon - Self-stabilizing utensil to help people with impaired motor skills” by J. Abrahamsson and J. Danmo [4]. See figures 1.3 and 1.4, respectively.

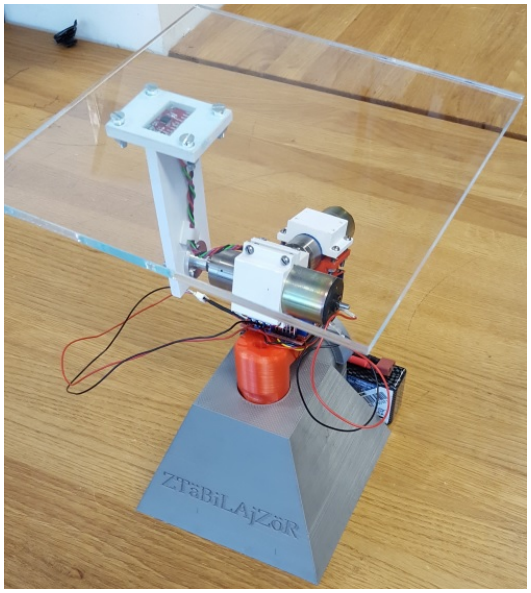


Figure 1.3: The ZTäBiLAjZöR by A. Karlsson and J. Cressell [3]

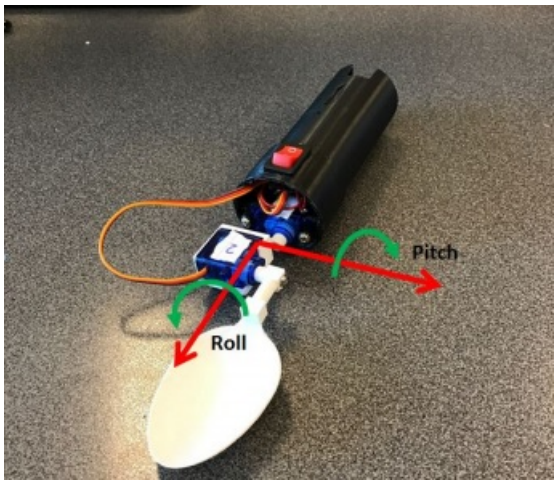


Figure 1.4: The Stabilizing Spoon by J. Abrahamsson and J. Danmo [4]

”Ball and Plate PID Control with 6 degrees of freedom Stewart Platform” by San José State University was examined [5]. There, two axes of the platform are used as the output of a PID controller. The controller uses a resistive touch panel mounted on the platform as input. A sensor is used to measure the ball position. See figure 1.5.

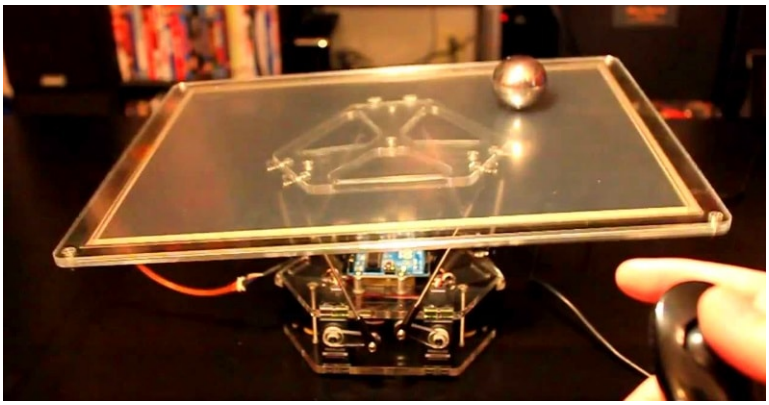


Figure 1.5: Ball and Plate PID Control With 6 DoF Stewart Platform by SJSU [5]

During the later stages of the project, additional inspiration was gathered from ”Self-Balancing Platform - How to compensate for imbalance with feedback from an IMU” by J. Nordlöf and P.Lagusson [6].

Chapter 2

Theory

This chapter explains the theory necessary to understand the problem.

The main principle behind ISPs is the use of an Inertial Measurement Unit [IMU] together with control theory to regulate the angle of these platforms in relation to an external frame of reference. The IMU detects changes in angle in relation to the external reference frame by using a combination of a gyroscope and an accelerometer to detect changes in three-dimensional linear and angular movement, meaning a difference in distance and angle. This means the IMU is measuring a total of six degrees of freedom [DoFs] . This data is sent to a controller, which regulates two DC motors to compensate for any changes in angular position. Note that depending on the design of the platform, some of the six DoFs mentioned are impossible to control.

2.1 Mechanics

2.1.1 Angular Movement

The pitch and roll movements are angular movements, meaning they are best described by an angle and a distance from a point of origin (where the coordinates are equal to zero). These are known as polar coordinates.

These three parts - angle, distance from the origin, and the position of the origin is everything that is necessary to describe any motion in polar coordinates.

Within the context of this project, angular position, angular velocity and angular acceleration are necessary to consider. An objects angular position is its angle in relation to an arbitrary axis. It is measured in radians, but often converted into degrees in order to be more relatable. The angular velocity is the rate of change of that position and the angular acceleration is how fast the angular velocity is changing, known as the first and second derivatives of the angular position. See figure 2.1 for visual context.

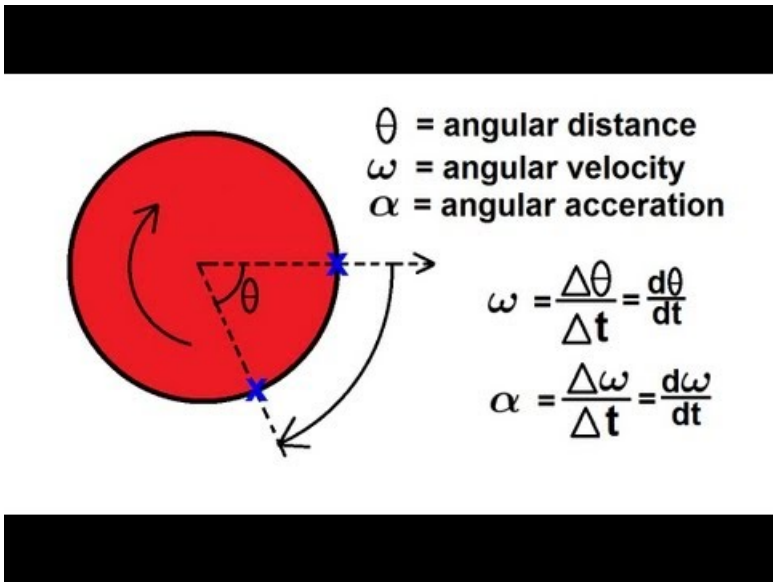


Figure 2.1: Angular motion, as explained by M. van Biezen [7]

2.1.2 Moments

An object receives angular acceleration around an axis due to a moment. It causes angular acceleration in the same way a force causes linear acceleration, essentially giving a moment the same function when it comes to angular movement, as a force has in linear movement.

2.1.3 Moments of Inertia

A moment of inertia is how much an object resists angular acceleration and is dependant on mass and the distribution of the mass in the object. The distribution of the mass of the object has a much greater effect on its mass than the value of the mass itself as seen by the general formulas [8]:

$$\sum_{i=1}^{n} mr^2 \tag{2.1}$$

$$\int_0^M r^2 dm \tag{2.2}$$

Where (2.1) is the discrete, and (2.2) is the continuous case.

2.2 Control Theory

In terms of control, the project presents two major challenges - the control of the pitch and roll axes and signal/noise filtering. The former will be done using two separate proportional-integral-derivative [PID] controllers. As each motor and corresponding PID controller is an independant system with a single input and output (called a SISO system in control theory), they can be approximately modelled as if they have no significant impact on each other, i.e. one motor does not have any major impact on the motion of the other motor. Noise filtering is done ideally using a Kalman filter, with a low pass filter as a possible, but inferior alternative [9][10].

2.2.1 PID Controller

Often preferred due to their simplicity, PID-controllers are the most widely used controllers to date [2]. Able to not only compensate for current errors, the integral and derivative parts can also be used to remember past states and predict future ones. The corresponding output is a sum of the three parts - P, I and D, and is described by equation 2.3 below.

$$u(t) = K_{\mathrm{p}}\varepsilon(t) + K_{\mathrm{i}}\int_0^t \varepsilon(\tau)d\tau + K_{\mathrm{d}}\frac{d\varepsilon(t)}{dt} \tag{2.3}$$

The "P" part of the PID controller is the proportional part grows proportionally to the error. The error is the difference between the desired and current value. The K value, or gain, for the P part changes how strong the relation is.

The I controller is the integrating part, which means its output is a sum of all error terms since it started receiving sensor input, meaning past errors influence current output. This behavior is ideal to sum up smaller errors to compensate for small errors, since the proportional output alone doesn't produce enough current to power the motors for small error terms. This accumulation of errors is also what allows it to effectively stabilize loads. This "remembering" of past errors can be problematic when the error is large. This is due to the I part still having an output when the desired angle is reached, since the sum of all errors isn't necessarily zero at that point. This leads to the platform overshooting the desired location.

The D part is proportional to the difference between past and current values, which gives a breaking effect before reaching the desired angle, generally reducing overshoot from the P and I parts.

In other words, higher values of K_{p} generally lead to a faster controller but also reduced stability. The integral term can eliminate even small errors in the output signal by accumulating them over time, but also impairs stability. Adding a derivative part can improve stability by adding a braking effect before the desired position is reached. In this project one PID controller will be used for each motor.

2.2.2 Kalman Filter

In control theory, there is a concept of an observer [11]. The observer is something that accounts for past inputs and outputs, as well as the mathematical model of the system, and makes an estimate of what the input is. This is useful if the input signal is inaccurate or otherwise unreliable. Described as an "optimal estimator" [9], the Kalman filter is considered the best linear estimator. Since the operating range of the system used within this project will likely stay close to the original, stable position, the system can be approximated as being linear - making the Kalman filter a primary choice for signal noise reduction.

2.3 Circuit diagram and component

Acting as the link between the programming in the Arduino and the mechanical model, the circuit powering the system will consist of a few main components and wires connecting them. Below is a short description and summary of the function of each major component.

2.3.1 DC Motors

DC motors are a type of actuator. In a circuit, actuators are components that act as outputs with the purpose of controlling a mechanical system, in contrast to sensors, which have the primary purpose of sending their measurements as inputs to other systems. Actuators and sensors are commonly interlinked, where sensor input into an intermediary such as a microcontroller results in some output to control the actuator.

When an input, in this case voltage, is supplied, the DC motor outputs an angular velocity proportional to the voltage, described mathematically as:

$$\omega = \frac{P}{M} \tag{2.4}$$

Where ω is the angular velocity in radians per second, P the power in watts and M the moment in newtonmeters. To get a clearer relation between ω and the supplied voltage, the following relation can be used:

$$P = UI \tag{2.5}$$

Where U is the voltage the motor receives and I the current flowing the motor. Combined, the two equations give:

$$\omega = \frac{UI}{M} \tag{2.6}$$

An important note is that the voltage that is applied to the motor is not equal to the voltage supplied by the power supply - there are losses along the way as described by Ohm's law, where the sum of all applied voltage and losses equal the supplied voltage [12].

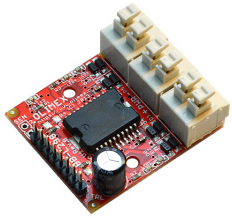
Motors used in this project are maxon A-max 22, Graphite Brushes with motortype 110147. See Appendix A.1.

2.3.2 Microcontroller

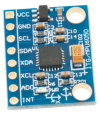
In order to have a larger degree of control of the input and output of the circuit, a microcontroller can be used, see figure 2.2 (a) for context. An Arduino Uno microcontroller was used in this project. The Arduino Uno is equipped with digital and analog input/output pins. The pins can be connected to sensors, and actuators such as motors and other components in the system. Signals from the sensors act as inputs. The Arduino is processing the data received from sensors and sends signals to activate actuators by using output pins, effectively acting as an intermediary between the two.

The Arduino can be programmed using the Arduino IDE and be uploaded to the microcontroller via a USB cable. Arduino is an open-source software and open source codes for a variety of purposes are widely available and will be used for this project [2].

(a) Arduino UNO



(b) Dual full-bridge driver L298



(c) IMU

Figure 2.2: Electronic components [13]

2.3.3 H-bridge

The DC motors must be able to spin in both directions. An H-bridge is an electronic circuit that makes it possible to switch the polarity of a voltage applied to a load. In the context of DC motors, H-bridges are often used to allow DC motors to run both forwards or backwards, rather than just a single direction. In this case there are

two motors. A dual full-bridge driver is needed to drive the motors with a reversal function. The H-bridge is pictured in figure 2.2 (b), and a schematic is shown in Appendix A.2.

2.3.4 Sensors

Sensors measure different physical quantities such as temperature, pressure, or as in this case, angular and linear position. The measured signal acts as output to be read or processed by another system or directly by a person.

The IMU, Inertial Measurement Unit used in this project is MPU-6050, which is a combined sensor with a 3-axis accelerometer, a 3-axis gyroscope and a processor that processes the raw data into motion and position information. The IMU is shown in figure 2.2 (c).

As discussed in section 2.2.2, the signal is rarely perfect or undisturbed and filtering is often required to be able to practically use the data given by the signal.

Chapter 3

Implementation

3.1 Manufacturing

The design was a redesign and optimization of the original model by J. Larsson [2]. This redesigned model had the advantage of having a well-balanced frame (see figure 3.1), although this model was later discarded in favor of an alternative swing-like design which is the current and final prototype. A CAD model of the final prototype is shown in figure 3.2.

This final prototype has several benefits. It has a lower center of mass that means better balance control. This swing-like model can be self-stabilizing mechanically. It takes less place, and is more serviceability.

The prototype was designed in Solid Edge, a Computer-Aided Design [CAD] program. Solid Edge was also used to numerically calculate the center of mass of the entire mechanism. It converted into a printable model via Cura which is a program designed to convert 3D models into so-called G-code. This G-code can be read by machines which utilize Numerical Control [NC] to function.

In this case, the NC machine was the 3D printer, model Ultimaker 2. This printer was used to manufacture the CAD models using melted polylactic acid [PLA] plastic , one vertical layer, or Z-layer, at a time. The height and width of these layers can be partially controlled via Cura for either increased accuracy or speed. The density of the model can also be controlled for speed, material cost and mass or moment of inertia specifications.

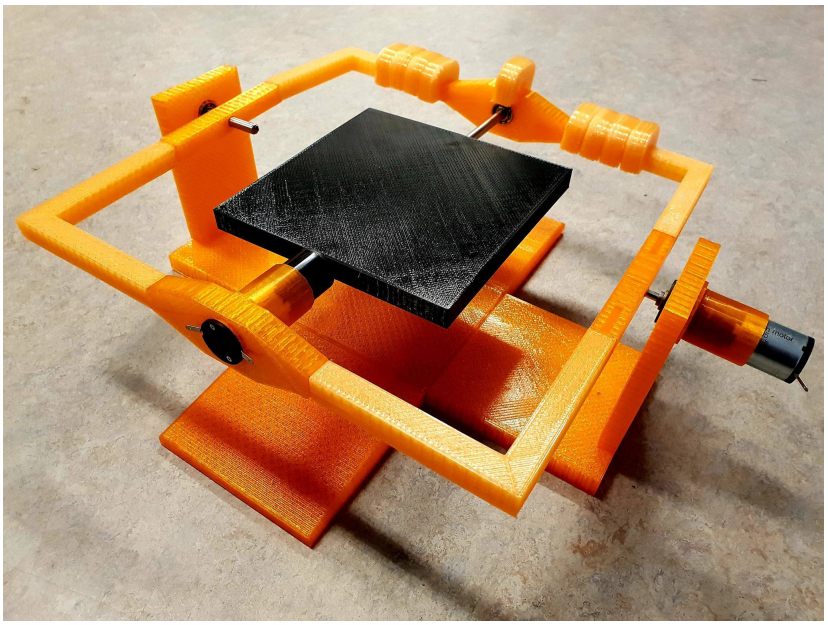


Figure 3.1: Redesigned concept prototype, with added weights on the far side compensating for the weight of the motor of the side opposite to them.

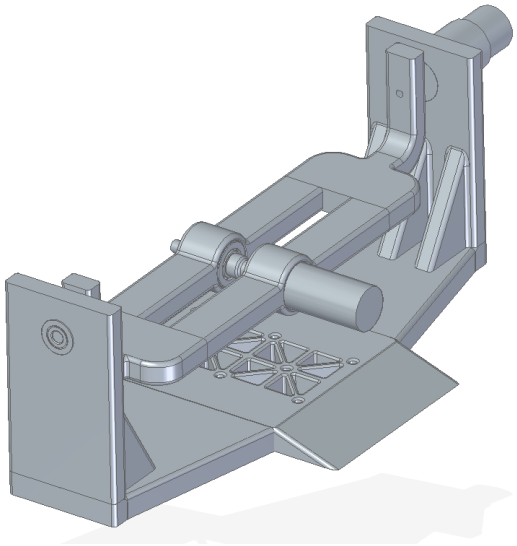


Figure 3.2: A CAD model of the final prototype. Designed in Solid Edge

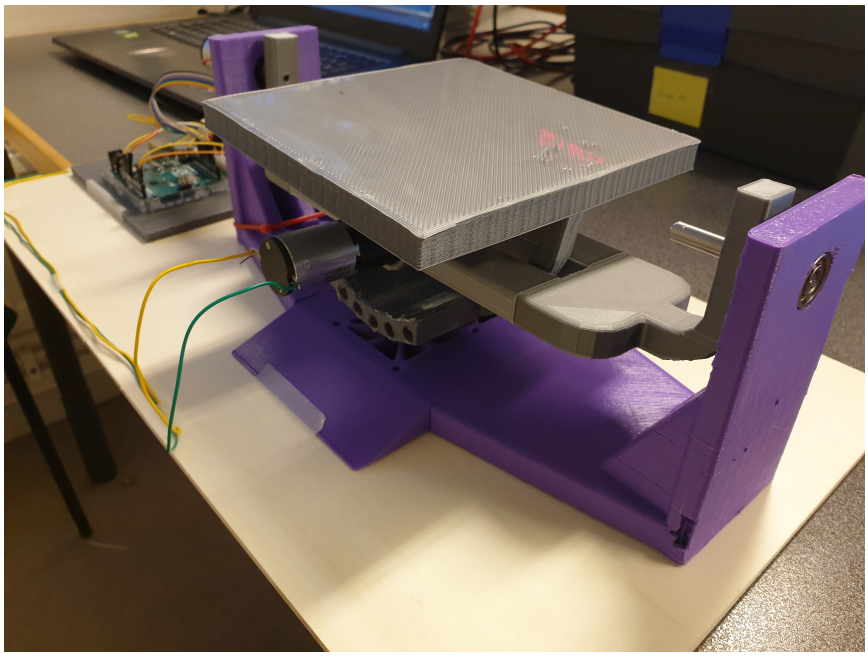


Figure 3.3: The final prototype printed with 3D printer.

The concept prototype required a base that i 30x30 centimeters. The Cura program only allowed models with a base up to 17x17 centimeters. To bypass this constraint a modular design was used, inspired by the Swedish company IKEA. Several pins are used to assembly parts together. See figure 3.4.

Due to this modular design the assembly was be done mostly by hand. Some basic tools, such as drills, hammers and tongs, were used as needed. Basic machining was required for the 3D-printed parts to fit better.

After initial testing, the concept prototype proved to be unstable due to the mass being distributed far away from the center of mass due to the outer frame. A temporary solution was to make the outer frame thinner, but this made it fragile and prone to breaking. Therefore, a new design was proposed, resulting in the current prototype. It contained fewer parts, the mass distribution was much more centered, the prototype was no longer prone to breaking. The center of mass was lowered as a positive and intentional side effect, further stabilizing the prototype.

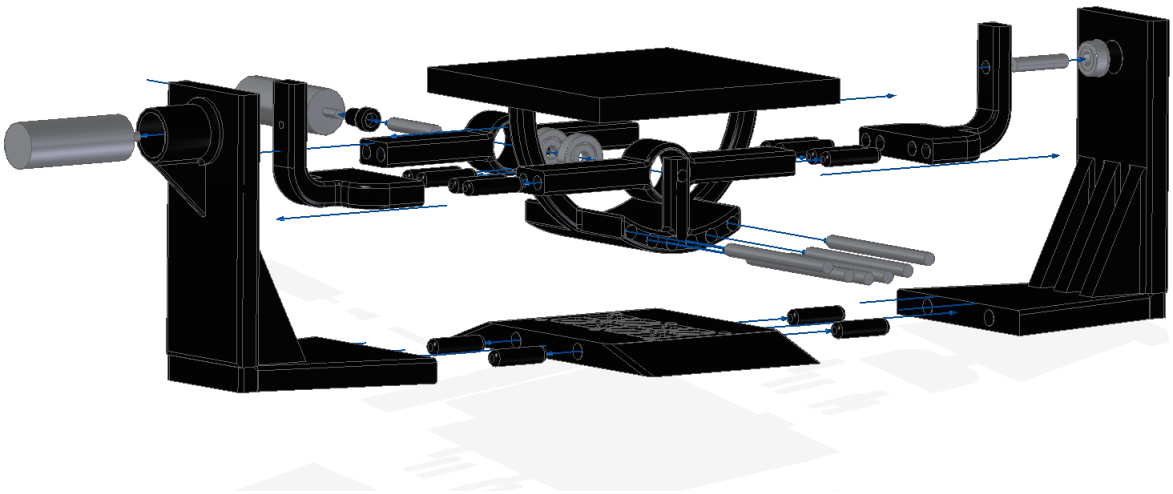


Figure 3.4: Exploded view of the prototype in Solid Edge.

3.2 Circuit diagram

To make the connections between the components easier to handle, a schematic was drawn. See figure 3.5.

The connections can be seen in table 3.1.

Table 3.1: Pin connections [14]

Connector at Arduino UNO	Connector at dual full-bridge driver
5V	CTRL$\langle 1 \rangle$, +5V
GND	CTRL$\langle 8 \rangle$, GND
DIGITAL$\langle 3 \rangle$, D3	CTRL$\langle 2 \rangle$, Enable$_A$
DIGITAL$\langle 5 \rangle$, D5	CTRL$\langle 3 \rangle$, IN1
DIGITAL$\langle 6 \rangle$, D6	CTRL$\langle 4 \rangle$, IN2
DIGITAL$\langle 7 \rangle$, D7	CTRL$\langle 5 \rangle$, Enable$_B$
DIGITAL$\langle 9 \rangle$, D9	CTRL$\langle 6 \rangle$, IN3
DIGITAL$\langle 10 \rangle$, D10	CTRL$\langle 7 \rangle$, IN4

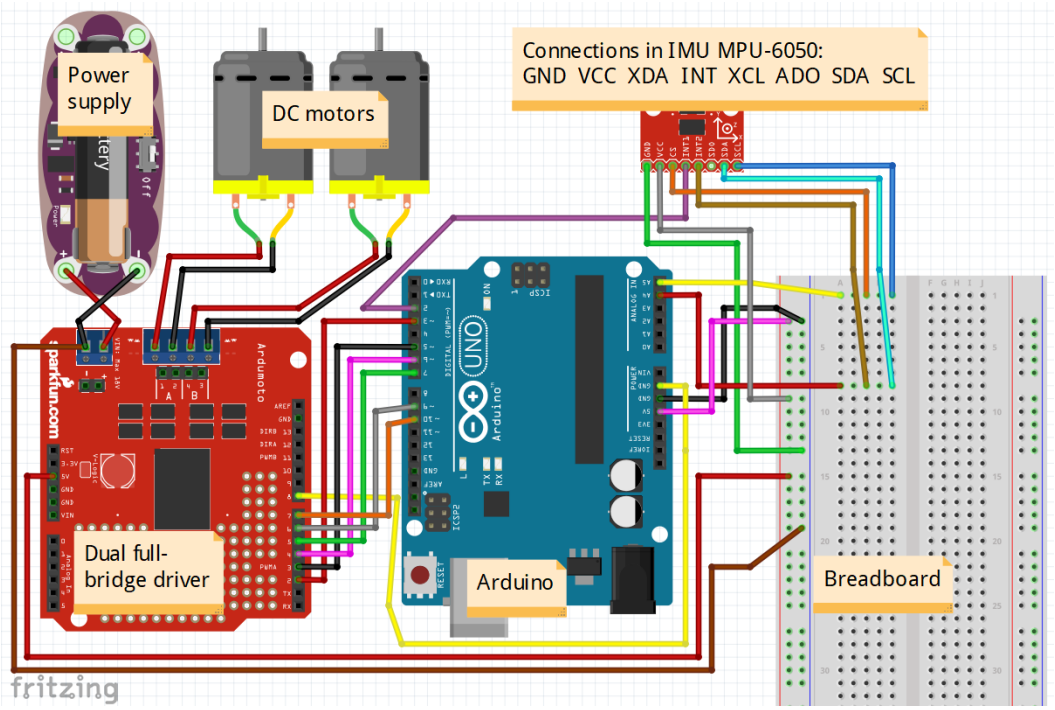


Figure 3.5: Circuitry schematic, made in fritzing

3.3 Control and Compensation

Following the theory, two PID controllers were used, one for each motor, to control the platform. A Kalman filter was used for signal filtering. In practice, all these are part of the Arduino programming and not separate parts of the circuit.

MATLAB and SimuLink were used together with a mathematical model of the prototype in an effort to simulate the system digitally and numerically produce good controller settings. This method was abandoned after a trial and error method combined with a binary search algorithm, proved more efficient. The binary search method consisted in testing the averages of values that are too low and too high until satisfactory performance was achieved.

Due to knowledge constraints and the relatively advanced nature of a Kalman filter on a Bachelor level project, an open source Arduino library was used for to simulate the Kalman filter. A low pass filter can also be used in an attempt to reduce high frequency measurement noise, to still produce somewhat more reliable measurements.

3.4 Testing

The tests conducted during the PID tuning process consisted of three parts. They designed to test the platforms ability to carry loads, compensate for tilt and disturbances. The final performance was difficult to predict during the construction phase of the project and therefore performance values from 1 to 5 were used (see appendix item 3), where 5 was a completely satisfactory performance with virtually no room for improvement and 1 was a performance that fulfilled none of the criteria. These values will also be called a performance score within this report.

Primarily, the compensation for disturbances was tested, since the test itself is quick and gives a general idea of the other two criteria. When the performance was good enough (generally above a performance value of 4), the other methods were used to fine tune the performance. The disturbance test itself was carried out by pressing down different parts of the platform for different amounts of time and observing the response, followed by an assessment of its performance value. This makes the test somewhat random or stochastic in nature, because the force of the pressure applied and the length of time that the pressure is applied varies each time. This make calculating more specific performance values such as rise time more difficult, and average values were therefore used.

The tuning process itself consisted of using a binary search algorithm to quickly reach the desired K values. This means that values that are known to be too low or too high from previous, random testing were used to start with and their averages were used instead until there is no longer a significant impact on the performance score. For example, the tuning of the P part consisted testing the values of 10 and 100 first. Being too low and too high, respectively, their average value of 55 was used instead. Often still being too high, the value of 32.5 would be used in the next iteration, and so forth. This was done for all three K-values, K_P, K_I and K_D, until the performance score was maximized.

Additional tuning was sometimes required after the binary search method was applied, and this was done using trial and error.

Chapter 4

Results and discussion

4.1 Physical model

The platform was successfully constructed according to expectations. Due to poor 3D printer tolerances in order to save time, some holes and rods had to be machined afterwards to fit better.

Due to the design of the model, the Y axis is naturally limited to rotating 15 degrees clockwise and counter-clockwise. The X axis had no such limitations and can in theory rotate freely around its axis.

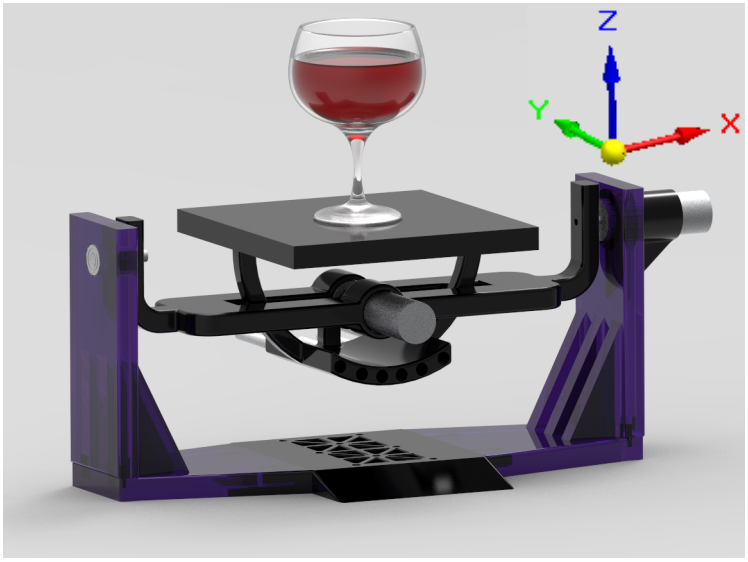


Figure 4.1: The prototype made in Solid Edge.

4.2 Performance

The final performance of the platform is deemed satisfactory, with further optimization of the model and the controller possible. The platform can effectively stabilize 400 grams at its center, 200 grams at the edges and close to 100 grams at the corners. Due to the somewhat random nature of the magnitude of the disturbances, a rise time equivalent was difficult to calculate. An average value was therefore recorded and ”less than” values were used for overshoot and settling time (see examples and further explanation below).

Results from the experiments where the disturbances and loads had varying magnitudes, the Y axis had an average rise time of approximately 120 milliseconds, an overshoot of less than 20 percent and a settling time of less than 400 milliseconds. The X axis had an average rise time of approximately 80 milliseconds, an overshoot of less than 100 percent and a settling time of less than 400 milliseconds. Three practical applications of the platform emerged from the testing and are as follows:

First, the platform can compensate for disturbances. A disturbance in this case is something that temporarily forces the platform to change angle. It can effectively get back to a preset angular position, although with minor overshooting and oscillatory behavior.

The oscillations have been successfully been removed by turning off the I part of the PID controller. This caused the platform to lose its ability to carry loads and to return to the preset angle due to the deviations or errors no longer accumulating over time. As the platforms purpose is to be able to stabilize objects, or loads, the oscillations caused by the I part being turned on part were therefore considered acceptable.

The platform can also carry loads. During testing, 100 gram steel weights were used, as well as a 500 millilitre PET plastic bottle. The bottle was filled with varying amounts of water to be able to vary the magnitude of the force applied.

During testing 12 volts and 2 amperes were applied to the motor driver. This resulted in spikes of current above the maximum continuous current limit of 0.681 amperes for each motor, but not the maximum stall current of 2.17 amperes. Since the platform was under load for short periods of time, it was considered safe.

The motors could produce enough torque to effectively stabilize 200 grams of applied load on the edges and approximately 400 grams at the center. At the corners tests were less conclusive, but approximately 100 grams could be effectively stabilized.

A third application of the platform is that it can keep its angle constant relative to the ground, even if its local frame of reference is moved. For example, if it is put on a table and the table is tilted, the platforms angle relative to the ground will remain constant until it reaches its physical limitations.

These three areas of application suggest that the platform is optimized specifically to keep objects placed on top of it horizontally stable. More specific areas of application include vehicle cockpit, camera and sensor stabilization and possibly to ease the serving of food and beverages.

4.3 PID tuning

During the tuning of the PID controller, the performance varied greatly. The process took several tuning and testing sessions and has been documented across several spreadsheets. See Appendix A.3. The key characteristics measured are response time, overshoot and how large the static error is. Ideally the response time is instant, with no overshoot or static error.

The process always followed the pattern of choosing a P value, followed by the I and D part, respectively. These roughly correspond to speed or response time for the P part, a reduction of the static error for the I part and a reduction of overshoot for the D part. The tuning of both axes was then done using manual binary search between values that are known to be too high and too low.

The swing-like design of the platform means that the controllers for the x and y axis had to be designed for different purposes. The x axis is mechanically self-stabilizing, as most other swings or pendulums, so it requires minimal assistance from the motor. Instead, the motor helps it remain stable when a load is applied and the controller was designed accordingly.

The y axis controller had to be designed for all three of the above applications. Optimal performance in all three was impossible, so a controller that had adequate performance in all three areas was considered satisfactory. The following PID settings perform satisfactory:

Table 4.1: PID setting

	K_p	K_I	K_D
Y axis	12.81	17.5	0.1
X axis	32.5	12.5	0.1

Chapter 5

Conclusions and future work

5.1 Design

The design was satisfactory and adequate on its own. The design allowed relatively free movement while being also having the desired rigid behavior. Improved design for various purposes or a better choice of construction materials possible.

5.2 Performance

The self-stabilizing properties of the platform were considered adequate and a proof that the design works and fulfills all the performance criteria of speed, stability and precision. The rise time values of 80 ms and 120 ms for the two axes can in this context be simplified to both be around 100 ms and presented as an approximate expected rise time value.

The inertia and center of mass of the system changes depending on what you put on the platform, changing the transfer function of the system. An algorithm that can better compensate for these changes would be preferable.

5.3 Optimization

Despite meeting and surpassing expectations, there is ample room for improvement of the design. The design of the entire platform could be made more inherently stable, e.g. by making the platform design more balanced even after the motors are attached or by making both axes mechanically self-stabilizing (e.g. with weights) so the controller design can focus more on the load bearing properties of the platform.

The use of a sound mathematical model (known as a transfer function) of the platform and motors can be used in Simulink to find good K values numerically rather than a trial and error approach.

Different motors can be used for different performance criteria. Motors that can generate a large amount of torque are preferable for bearing loads and motors with a high angular velocity are preferable for quick response times. Both are rarely possible due to the relation between a motors power, torque and angular velocity.

Another issues with this model is so-called sensor drift. This causes the desired position of the platform to change over time, causing a tilt as time passes. Using several inertial sensors or an external sensor like Ncoder sensor or Hall sensor that measures the angle of the platform can relate to zero position are therefore preferable.

Bibliography

[1] Inertially Stabilized Platform Technology, J.M. Hilkert, *CONCEPTS AND PRINCIPLES*
https://www.academia.edu/6332000/Inertially$_S$*tabilized*$_P$*latform*$_T$*echnology*, 2008.
*Retrieved*2019 − 03 − 15.

[2] J. Larsson, *Gimbal stabilizer for cockpit bases of terrain vehicle or combat boat - A proof of concept*,
http://www.diva-portal.org/smash/get/diva2:1225371/FULLTEXT01.pdf, 2018.
Retrieved 2019-02-01.

[3] A. Karlsson and J. Cressell, *Self-stabilizing platform - ZTäBiLAjZöR*,
http://www.diva-portal.org/smash/get/diva2:957123/FULLTEXT01.pdf, 2016.
Retrieved 2019-02-15.

[4] J. Abrahamsson and J. Danmo,
The Stabilizing Spoon - Self-stabilizing utensil to help people with impaired motor skills,
http://www.diva-portal.org/smash/get/diva2:1200521/FULLTEXT01.pdf, 2017.
Retrieved 2019-02-15.

[5] Stewart platform
https://siamagazin.com/ball-and-plate-pid-control-with-6-dof-stewart-platform/, 2019-02-13. Retrieved 2019-02-15.

[6] J. Nordlöf and P.Lagusson, *Self-Balancing Platform - How to compensate for imbalance with feedback from an IMU* 2015
http://www.diva-portal.org/smash/get/diva2:916229/FULLTEXT01.pdf?fbclid=IwAR29RlzTSc1rL
z-J9gybtW90yLLvuDZcGw, 2015. Retrieved 2019-05-01.

[7] Michel Van Biezen
https://www.youtube.com/watch?v=3lp7fyqNlu8. Retrieved 2019-04-01.

[8] hyperphysics
http://hyperphysics.phy-astr.gsu.edu/hbase/mi.html. Retrieved 2019-04-25.

[9] Monash University
http://biorobotics.ri.cmu.edu/papers/sbp$_p$*apers/integrated*3/*kleeman*
$_k$*alman*$_b$*asics.pdf*.*Retrieved*2019 − 04 − 15.

[10] Signal Processing Stack Exchange
https://dsp.stackexchange.com/questions/25518/digital-low-pass-filter-vs-kalman-filter. Retrieved 2019-04-25.

[11] MIT OpenCourseWare
https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-011-introduction-to-communication-control-and-signal-processing-spring-2010/readings/MIT6$_0$11S10$_c$*hap*06.*pdf*.*Retrieved*2019 − 04 − 25.

[12] ElectronicsTutorials
https://www.electronics-tutorials.ws/dccircuits/dcp$_2$.*html**Retrieved*2019 − 05 − 31

[13] Electrokit
https://www.electrokit.com. Retrieved 2019-02-16.

[14] Arduino
https://www.arduino.cc/en/Guide/Introduction. Retrieved 2019-02-13.

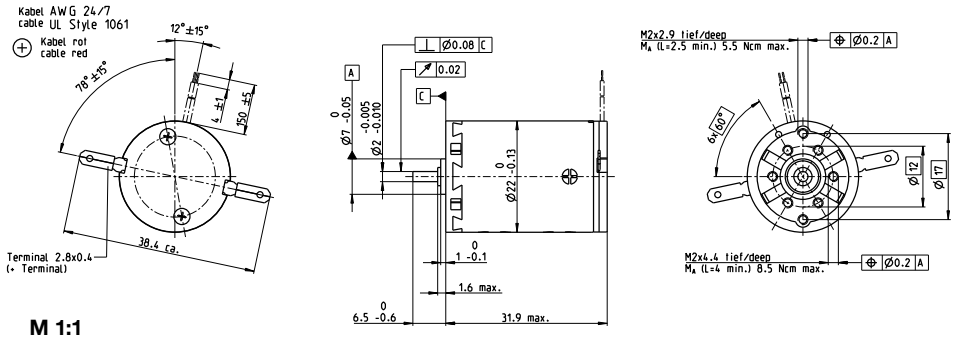
Appendix A

Appendix

Appendix A1, A2, A3 and A4 attached in the following pages.

A.1 Data sheet maxon

A-max 22 ∅22 mm, Graphite Brushes, 6 Watt



M 1:1

Stock program
Standard program
Special program (on request)

Part Numbers													
with terminals	110143	110145	110146	**110147**	110148	110149	110150	**110151**	110152	110153	110154	110155	
with cables	139840	353017	199807	320206	323856	108828	199424	202921	267433	325492	313302	353019	
Motor Data													
Values at nominal voltage													
1 Nominal voltage	V	6	9	9	12	12	15	18	24	24	36	48	48
2 No load speed	rpm	9240	9690	8500	10200	9170	10000	9770	10500	8480	9630	9110	8210
3 No load current	mA	83.1	57.9	49.6	45.8	40.5	36	29	23.7	18.4	14.2	9.99	8.84
4 Nominal speed	rpm	6240	6530	5350	7060	6000	6890	6600	7380	5270	6420	5840	4940
5 Nominal torque (max. continuous torque)	mNm	5.91	6.88	7.04	6.96	6.95	6.93	6.92	6.9	6.97	6.86	6.75	6.86
6 Nominal current (max. continuous current)	A	1.08	0.859	0.77	0.681	0.613	0.534	0.432	0.347	0.283	0.21	0.147	0.135
7 Stall torque	mNm	19.4	22.1	19.8	23.7	20.9	22.9	22	23.7	18.9	21.1	19.2	17.6
8 Stall current	A	3.29	2.59	2.04	2.17	1.72	1.65	1.29	1.12	0.721	0.606	0.393	0.325
9 Max. efficiency	%	67	70	69	72	70	72	72	73	70	72	71	70
Characteristics													
10 Terminal resistance	Ω	1.82	3.48	4.42	5.53	6.96	9.09	14	21.5	33.3	59.4	122	148
11 Terminal inductance	mH	0.106	0.223	0.288	0.363	0.445	0.585	0.891	1.37	2.1	3.69	7.3	8.97
12 Torque constant	mNm/A	5.9	8.55	9.73	10.9	12.1	13.9	17.1	21.2	26.2	34.8	48.9	54.3
13 Speed constant	rpm/V	1620	1120	981	875	790	689	558	450	364	274	195	176
14 Speed / torque gradient	rpm/mNm	500	454	446	444	455	452	457	456	461	468	487	479
15 Mechanical time constant	ms	20.9	20.2	20.1	19.9	19.9	19.9	19.7	19.7	19.8	19.7	19.9	19.8
16 Rotor inertia	gcm²	4	4.25	4.3	4.29	4.19	4.2	4.13	4.13	4.09	4.02	3.9	3.94

Specifications

Thermal data

17	Thermal resistance housing-ambient	20 K/W
18	Thermal resistance winding-housing	6.0 K/W
19	Thermal time constant winding	10.2 s
20	Thermal time constant motor	314 s
21	Ambient temperature	-30…+85°C
22	Max. winding temperature	+125°C

Mechanical data (sleeve bearings)

23	Max. speed	9800 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.012 mm
26	Max. axial load (dynamic)	1 N
27	Max. force for press fits (static)	80 N
28	Max. radial load, 5 mm from flange	2.8 N

Mechanical data (ball bearings)

23	Max. speed	9800 rpm
24	Axial play	0.05 - 0.15 mm
25	Radial play	0.025 mm
26	Max. axial load (dynamic)	3.3 N
27	Max. force for press fits (static)	45 N
28	Max. radial load, 5 mm from flange	12.3 N

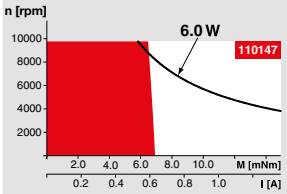
Other specifications

29	Number of pole pairs	1
30	Number of commutator segments	9
31	Weight of motor	54 g

Values listed in the table are nominal.
Explanation of the figures on page 64.

Option
Ball bearings in place of sleeve bearings

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

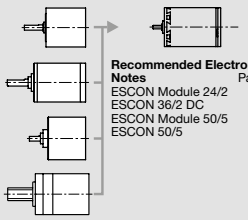
Overview on page 28–36

Planetary Gearhead
∅22 mm
0.1 - 0.6 Nm
Page 323/324

Planetary Gearhead
∅22 mm
0.5 - 2.0 Nm
Page 325/327

Spur Gearhead
∅24 mm
0.1 Nm
Page 331

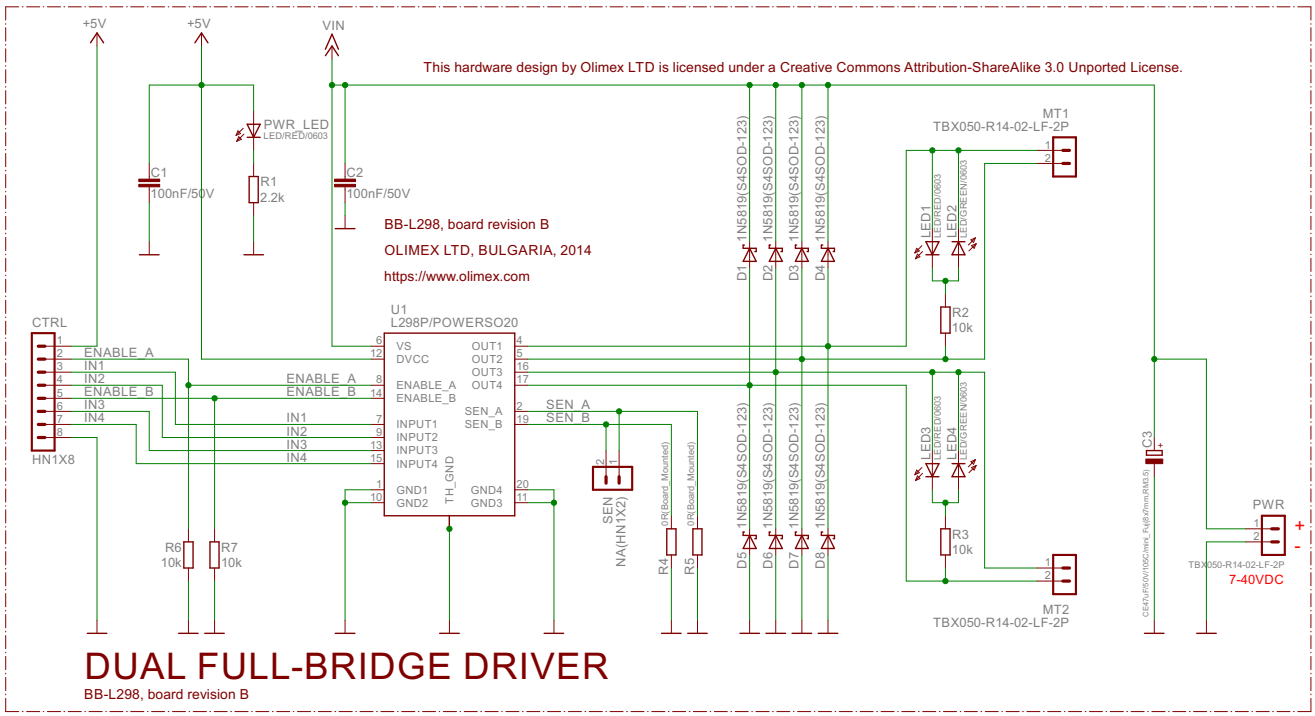
Spindle Drive
∅22 mm
Page 364/365



Recommended Electronics:

Notes	Page 30
ESCON Module 24/2	426
ESCON 36/2 DC	426
ESCON Module 50/5	427
ESCON 50/5	428

A.2 Schematic Olimex BB-L298



A.3 PID test

Trial session 1:	Settings: 12V, 700mA				
Iteration number	P value	I value	D value	Performance	Weight added (Y/N for yes/no)
1	3,75	0	0	3	N
2	3,5	1	0	3,5	N
3	3,5	1	0	3,75	Y
4	3,5	1	1	2,75	Y
5	3,5	1	0,05	3,75	Y
6	3,5	1	0,05	3,25	N
7	3,5	1	0,2	4	Y
8	3,5	1	10	1	Y
9	3,5	1	5	1,25	Y
10	3,5	1	0,5	3,25	Y
11	3,5	1	0,25	4,25	Y
12	3,5	5	0,25	4,5	Y
13	3,5	10	0,25	3,25	Y
14	3,5	4	0,25	3,75	Y
15	3,5	6	0,25	3,5	Y
16	3,5	5	0,4	4	Y
17	3,5	5	0,01	3,5	Y
18	2,5	5	0,4	4	Y

Trial session 2:	Settings: 12V, 700mA						
Iteration numbe	***P value***	***I value***	***D value***	***Performance***	***Weight added (Y/N for yes/no)***	***Iteration comments***	***General comments***
1	1	0	0	2,5	Y	Slow, unresponsive. Check code.	
2	10	0	0	2	Y	Overly aggressive.	
3	5	0	0	3,25	Y	Surprisingly responsive. Prone to static error.	Reaching maximum output destabilizes system.
4	6	0	0	3,5	Y	Slight improvement over iteration 4. Needs an integral action.	
5	6	1	0	3,75	Y	Slight improvement over iteration 5. Prone to winding.	
6	6	2	0	4	Y	Less prone to winding, unless disturbance is sustained.	
7	7	2	0		Y		

Trial session 1, y axis:	Settings: 12V, 700mA		Goal: Find satisfactory K values for the Y-axis.				
Iteration number	***P value***	***I value***	***D value***	***Performance***	***Weight added (Y/N for yes/no)***	***Iteration comments***	***General comments***
1	1	0	0	2,5	Y	Slow response, large static error	
2	10	0	0	3,25	Y	Fast response, large oscillations	
3	10	1	0	2,75	Y	Extreme oscillations, barely stable. Non-zero static error.	
4	10	10	0	2,5	Y	Still barely stable and heavy oscillations.	
5	10	100	0	1,5	Y	Near instant destabilisation from disturbances. Unstable system.	
6	10	50	0	2,25	Y	Slow destabilisation from disturbances. Unstable system.	
7	10	25	0	2	Y	As above, but faster destabilisation.	
8	10	20	0	2,5	Y	Closer to stability but still unstable.	
9	10	5	0	2,5	Y	Also close to stability, but unstable.	
10	10	1	1	3,5	Y	Greatly improved performance. Basic ability to counterbalance loads.	Reaching its physical limits makes the platform less stable
11	10	2	1	3,75	Y	Even better performance. Prone to static errors.	
12	10	3	1	3,5	Y		
13	10	4	1	3,25	Y		
14	10	4	2	3	Y	Aggressive behavior.	
15	20	5	1	3,75	Y	Prone to static error, somewhat oscillative.	
16	20	4	1	4	Y	Good stabilizing ability. Somewhat prone to static error.	
17	25	3,5	1	3,75	Y		
18	20	4	1	1	Y	Instant destabilization. Try to find new settings.	This is due to a repositioning on the IMU.
19	10	10	1	4	Y	Good performance, but can't handle larger loads.	
20	15	15	1	4,25	Y	Very good performance, can handle heavier loads.	Be careful when removing loads. Sudden removal causes instability.
21	15	15	0,05	4,5	Y	Reduces sensitivity to sudden changes in angle.	
22	15	15	0,005	4,75	Y	Reduced oscillations. Close to optimal behavior.	Further optimization of K values possible, but these are satisfactory.
Trial session 1, x axis:	Settings: 12V, 700mA		Goal: Find satisfactory K values for the X axis. This should prove easier since the swing design of the X axis makes it return to its stable position.				
Iteration number	***P value***	***I value***	***D value***	***Performance***	***Weight added (Y/N for yes/no)***	***Iteration comments***	***General comments***
1	10	0	0	3	Y	Slow response. Somewhat choppy movements.	
2	100	0	0	1,5	Y	Near-instant destabilization. Attempt binary search between 10 and 100.	
3	55	0	0	3,25	Y	Very good response time, somewhat oscillatory behavior.	
4	32,5	0	0	3,5	Y	Even better response time, oscillatory behavior with low amplitude.	Can get stuck when physical limits are reached.
5	43,75	0	0	3,5	Y	Better response time, stronger oscillations. Try averaging two previous iterations	The above point can have cause significant winding.
6	38,125	0	0	3,75	Y	Satisfactory P value.	
7	38,125	10	0	3,25	Y	Strong oscillations.	
8	38,125	10	0,005	4	Y	Satisfactory performance, although somewhat prone to getting stuck.	This is a model limitation, not a controller limitation.
Trial session 1, y axis:	Settings: 12V, 800mA		Goal: Find good conservative and aggressive K values, one axis at a time and implement. Attempt use both axes when finished. K values listed as (aggressive; conservative)				
Iteration number	***P value***	***I value***	***D value***	***Performance***	***Weight added (Y/N for yes/no)***	***Iteration comments***	***General comments***
1	20; 5	0,2; 15	1; 0,005	3,25	Y	Prone to sudden increases in output.	
2	20; 5	0,2; 15	0,5; 0,005	3,25	Y	Prone to sudden increases in output.	
3	20; 5	0,2; 15	0,25; 0,005	3,25	Y	Prone to sudden increases in output.	
4	10; 5	0,2; 15	0,25; 0,005	3,25	Y	Prone to sudden increases in output.	
5	10; 5	0,1; 15	0,125; 0,005	3,5	Y	Somewhat less sudden increases in output.	
6	10; 5	0; 15	0,125; 0,005	3,5	Y	Good performance, but no ability to stabilize loads.	
7	10; 5	1; 15	0,125; 0,005	3,25	Y	More sudden increases in output.	
							End of trial. Redesign model and start anew in new file.

Trial session 1, y axis:	Settings: 12V, 1000mA		Goal: Find satisfactory K values for the Y-axis. Note: avoid applying peak current for longer periods of time. All values rounded to 2 decimals, except D values.			
Iteration number	***P value***	***I value***	***D value***	***Performance***	***Iteration comments***	***Additions notes and general comments***
1	10	0	0	3	Somewhat slow. Significant overshoot. Static error present.	Will constitute the baseline "satisfactory" performance (BSP) of 3,00
2	100	0	0	1	Extremely oscillatory behavior.	Binary search between 10 and 100.
3	55	0	0	2	Initially stable, but very low stability margins. Lower P value.	
4	32,5	0	0	2,25	Somewhat higher stability margins. Lower P value.	
5	21,25	0	0	2,75	Close to BSP. Significant oscillations.	
6	15,625	0	0	3,25	Somewhat better than BSP. Somewhat reduced oscillations.	
7	12,8125	0	0	3,5	Satisfactory P value. Round to 12,81 and tune I part.	
8	12,81	10	0	3,75	Basic load bearing properties. Binary search between 10 and 20.	Slowly diverges from setpoint when bearing load.
9	12,81	20	0	4	Satisfactory. See if I = 15 improves performance.	
10	12,81	15	0	3,75	Somewhat worse load bearing properties.	
11	12,81	17,5	0	4,25	Very satisfactory performance. Move on to D part.	
12	12,81	17,5	0,001	4,25	Results inconclusive. Attempt 0,0001 next iteration.	D values are known to be low from previous tries. Attempt D=0,001.
13	12,81	17,5	0,0001	3,5	Unresponsive. Attempt 0,01.	
14	12,81	17,5	0,01	4,5	Impressive overall performance. Somewhat slow and oscillatory.	Attempt D = 0,1. Expected behavior is worse than current.
15	12,81	17,5	0,1	4,75	Close to ideal. Cannot handle large or sudden disturbances.	Sensor drift remains a problem, but is outside the scope of this project.
Rise time: 172ms	Overshoot: 19,5%	Settling time: <400ms		Note: Due to the varying magnitude of loads and disturbances, it is difficult to give a good estimate. This is a sample measurement.		

Trial session 1, x axis:	Settings: 12V, 1000mA		Goal: Find satisfactory K values for the Y-axis. Note: avoid applying peak current for longer periods of time. All values rounded to 2 decimals, except D values.			
Iteration number	***P value***	***I value***	***D value***	***Performance***	***Iteration comments***	***Additions notes and general comments***
1	10	0	0	3	Good basic performance, try binary search between 10 and 100.	
2	100	0	0	3,5	Very good resistance to disturbances, but should be less aggressive.	
3	55	0	0	3,5	Still aggressive, oscillatory behavior.	
4	32,5	0	0	3,75	Impressive P value performance. Proceed with I part.	
5	32,5	10	0	3,75	Results inconclussive. Load bearing capabilities insufficient.	Try binary search between 10 and 20.
6	32,5	20	0	3,25	Unstable behavior at heavy loads and disturbances.	
7	32,5	15	0	3,75	Load bearing slightly better than iteration 5. Current draw is a issue.	X axis diffucult to stabilize. Try the average of 10 and 15 for the I part.
8	32,5	12,5	0	4	Likely maximized load bearing capability. Proceed to D part.	Try D value from Y axis?
9	32,5	12,5	0,1	4,5	Very good overall performance. No further tuning necessary.	
Rise time: 85ms	Overshoot: 91,2%	Settling time <400ms				

Trial session 1, both axes:

General testing with tilting, disturbances and loads looks promising. Reaction times can be improved and current draw is an issue when stabilizing heavy loads or loads at the corners. Lack of friction on plate also an issue.

A.4 The Arduino codes

```
/*
 * Behnam Yosef Nezhad Arya
 * Maksims Svjatoha
 * KTH Degree Project in Mechatronics, First Cycle, 15 hp
 * 2019-05-05
 */

#include <MPU6050_tockn.h>
#include <Wire.h>
#include <PID_v1.h>
#include <Kalman.h>
#include <LCD03.h>

MPU6050 mpu6050(Wire);                    //IMU

long timer = 0;                           //IMU timer variable

//____________________________________________________________________________

//Declare variables
double x = 0;                             //IMU
double y = 0;                             //IMU
double ux = 0;                            //PID controller output x
double uy = 0;                            //PID controller output y
double kx = 0;                            //Kalman filtered angle x
double ky = 0;                            //Kalman filtered angle y
int writePin_y;                           //Will contain one of two possible values to enable y axis motor inversion
int writePin_x;                           //Will contain one of two possible values to enable x axis motor inversion
int SampleTime = 10;                      //PID sample time

//____________________________________________________________________________

//Set up Kalman Filter
```

```
32
33 Kalman kalmanX; // Create the Kalman instances
34 Kalman kalmanY;
35
36 //
     ______________________________________________________________________

37 //motor driver definitions, cleanup of unnecessary variables
       might be required
38
39 // set pin numbers:
40 const int LED  = 13;
41 const int Enable_A =  3;     // A low-to-high transition on
     the STEP input sequences the translator and advances the
     motor one increment
42 const int IN1 = 5;           // Direction of rotation
43 const int IN2 = 6;           // Mode of operation: Active/
     Sleep
44 const int Enable_B = 7;      // Enable/Disable the Driver
     operation
45 const int IN3 = 9;           // Reset when active turns off
     all of the FET outputs
46 const int IN4 = 10;          // Microstep Select
47 const int Threshold = 20;
48
49 #define IN1_L    digitalWrite(IN1, LOW);
50 #define IN1_H    digitalWrite(IN1, HIGH);
51 #define IN2_L    digitalWrite(IN2, LOW);
52 #define IN2_H    digitalWrite(IN2, HIGH);
53 #define IN3_L    digitalWrite(IN3, LOW);
54 #define IN3_H    digitalWrite(IN3, HIGH);
55 #define IN4_L    digitalWrite(IN4, LOW);
56 #define IN4_H    digitalWrite(IN4, HIGH);
57
58 //motor definitions, end
59 //
     ______________________________________________________________________

60
61 //
     ______________________________________________________________________

62 //PID definitions, start
63
```

```
64
65 //Define Variables we'll be connecting to
66 double Setpointy, Inputy, Outputy;
67 double Setpointx, Inputx, Outputx;
68
69 //Define the aggressive and conservative Tuning Parameters
70 double consKpy=12.81, consKiy=17.50, consKdy=0.1; //Last
       known good configuration: 15, 15, 0.005
71 double consKpx=32.5, consKix=12.50, consKdx=0.1;  //Last
       known good configuration: 38.125, 10.00, 0.005
72
73 //Specify the links and initial tuning parameters
74 PID PIDy(&Inputy, &Outputy, &Setpointy, consKpy, consKiy,
       consKdy, DIRECT);
75 PID PIDx(&Inputx, &Outputx, &Setpointx, consKpx, consKix,
       consKdx, DIRECT);
76
77 //PID definitions, end
78 //
     ____________________________________________________________________________

79
80 void setup() {
81   Serial.begin(9600);                    //IMU
82   Wire.begin();                          //IMU
83   mpu6050.begin();                       //IMU
84   mpu6050.calcGyroOffsets(true);         //IMU
85
86   // set the digital pin as output:      //Motor driver
87   pinMode(LED, OUTPUT);                  //Motor driver
88   pinMode(Enable_A, OUTPUT);             //Motor driver
89   pinMode(Enable_B, OUTPUT);             //Motor driver
90   pinMode(IN1, OUTPUT);                  //Motor driver
91   pinMode(IN2, OUTPUT);                  //Motor driver
92   pinMode(IN3, OUTPUT);                  //Motor driver
93   pinMode(IN4, OUTPUT);                  //Motor driver
94
95   //Set the state
96   digitalWrite(LED, LOW);                //Motor driver
97   digitalWrite(Enable_A, HIGH);          //Motor driver
98   digitalWrite(Enable_B, HIGH);          //Motor driver
99   digitalWrite(IN1, LOW);                //Motor driver
100  digitalWrite(IN2, LOW);                //Motor driver
101  digitalWrite(IN3, LOW);                //Motor driver
```

```
102   digitalWrite(IN4, LOW);                //Motor driver
103
104   //initialize the variables we're linked to
105   Inputy = ky;                           //PID
106   Inputx = kx;                           //PID
107   Setpointy = 0;                         //PID
108   Setpointx = 0;                         //PID
109
110   //turn the PID on                      //PID
111   PIDy.SetMode(AUTOMATIC);               //PID
112   PIDx.SetMode(AUTOMATIC);               //PID
113
114   //Changes output to include negative values (standard is
        0-255).
115
116   PIDy.SetOutputLimits(-255,255);        //PID
117   PIDx.SetOutputLimits(-255,255);        //PID
118
119   x = mpu6050.getGyroAngleX();           //IMU
120   y = mpu6050.getGyroAngleY();           //IMU
121
122   kalmanX.setAngle(x); // Set starting angle
123   kalmanY.setAngle(y);
124
125   PIDy.SetSampleTime(SampleTime);
126   PIDx.SetSampleTime(SampleTime);
127
128 }
129
130 void loop() {
131
132 //Reading the sensor data (gyroscope data around x and y
      axis).
133
134   mpu6050.update();
135
136   x = mpu6050.getGyroAngleX();           //IMU
137   y = mpu6050.getGyroAngleY();           //IMU
138
139
140 //Applying a Kalman filter to raw signal data.
141
142 double dt = (double)(micros() - timer) / 1000000; //
      Calculate delta time
```

```
143 timer = micros();
144
145 double gyroXrate = x / 131.0; // Convert to deg/s
146 double gyroYrate = y / 131.0; // Convert to deg/s
147
148 kx = kalmanX.getAngle(x, gyroXrate, dt); //Kalman filtered
      gyro signal, x axis.
149 ky = kalmanY.getAngle(y, gyroYrate, dt); //Kalman filtered
      gyro signal, y axis.
150
151 //PID tuning. Possible to implement different K values for
      large angular deviations.
152
153 PIDy.SetTunings(consKpy, consKiy, consKdy);
154 PIDx.SetTunings(consKpx, consKix, consKdx);
155
156 //The PID computation:
157
158 Inputy = ky;
159 Inputx = kx;
160
161 PIDy.Compute(); //Comment out to turn off Y axis
      stabilization.
162 PIDx.Compute();    //Comment out to turn off X axis
      stabilization.
163
164 uy = Outputy;
165 ux = Outputx;
166
167 //Handling of negative values. To reverse motor polarity,
      the voltage must be applied on a different pin.
168
169 if (uy < 0) {writePin_y = 10;}
170 if (uy > 0) {writePin_y = 9;}
171 if (ux < 0) {writePin_x = 6;}
172 if (ux > 0) {writePin_x = 5;}
173
174 //Only positive bit values between 0 and 255 are accepted.
      Once the pin is chosen, the bit signal must be positive.
175
176 if (uy < 0) {uy = -uy;}
177 if (ux < 0) {ux = -ux;}
178
179
```

```
180 analogWrite(writePin_y, uy);
181 analogWrite(writePin_x, ux);
182
183 Serial.print("x␣:␣");Serial.print(x);Serial.print(",␣");
      //IMU
184 Serial.print("kx␣:␣");Serial.print(kx);Serial.print(",␣");
      //IMU
185 Serial.print("y␣:␣");Serial.print(y);Serial.print(",␣");
      //IMU
186 Serial.print("ky␣:␣");Serial.print(ky);Serial.print(",␣");
      //IMU
187 Serial.print("ux␣:␣");Serial.print(ux);Serial.print(",␣");
      //IMU
188 Serial.print("uy␣:␣");Serial.print(uy);Serial.print(",␣");
      //IMU
189 Serial.print('\n');
      //IMU
190
191 }
```

TRITA -ITM-EX 2019:43