SELF-BALANCING ROBOT

B. Bojkow, T. Djalilov, K. Teymurov, D. Loef, Sz. Letkiewicz, V. Kulić Golja
Group 11

Abstract

Despite the vast number of commercially available self-balancing vehicles such as scooters and segways, the process of designing and controlling an unstable system as such is greatly underappreciated. This paper presents the process of systematically design a robot that is able to balance itself on a single wheel. A physical model that describes the the mechanical behaviour of the system was developed along with an open loop analysis to demonstrate the system's instability. By linearizing the system, an appropriate PID controller was initiated to stabilize the system. A closed loop step response along with physical observations were then made to investigate the systems stability.

Keywords— PID control, Feedback system, Transfer function, State space

1 Introduction

A self-balancing robot is a robot which balances itself on a wheel so that it prevents itself from falling. Closed-loop feedback control adjusts the motors, according to real-time data from onboard motion sensors, to compensate for tilting motion in order to keep the robot upright. This paper address the problem by describing the design of a self-balancing robot, as well as the controller it uses to balance. The theoretical aspects and problem definition are shown in sections 2 and 3. The systematic design approach used to develop the robot is presented in Section 4. Section 5 centres around modelling the behaviour of the system that is then used to design the controller in Section 6. Section 7 consists of discussion and conclusions of the project.

2 Problem definition

The primary objective is to create an unstable robot that balances itself using a real-time implementation of the PID-Control algorithm. The process of modelling and implementing the PID-controler will apply only to systems attempting to balance a mass upwards in a vertical position, similarly to an inverted pendulum. Furthermore, it is universally agreed that simulation and software models are not exact representation of physical systems and their behaviours. Therefore, the project will involve the use of a model and physical measurement. The following design requirements have been made to ease the recreation of this project.

- Single wheel
- System must only drive forward and backwards
- Robot should not exceed 100 € budget
- Rigid bodies should be 3D printed
- Robot should not be larger than 30x30x30cm

The feedback loop will be based on measuring the body's angle of tilt. This will require filtration and signal analysis from the motion sensor. The analysed data will serve as the input for the PID controller, that will determine the magnitude of opposite torque needed to balance the wheel.

3 Design description

A self-balancing robot can be depicted by two main components, the wheel, and the body. The wheel allows the robot to drive forwards and backwards, prevents any sideways movement. On the other hand, the body of the robot serves as a housing system for the onboard electronics, and must be mounted to the wheel's axis of rotation. Consider Fig. 1, showing a render of the designed robot.

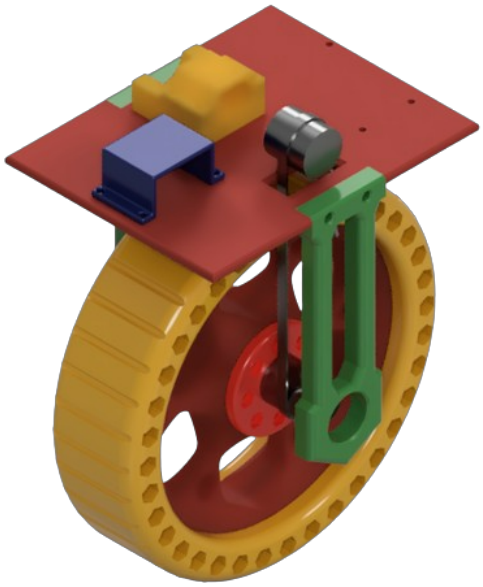


Figure 1: Schematic of body component with dimensions

The DC motor responsible for rotating the wheel is located on the body of the robot. Using a belt and 2 tooth pulleys, the motor's axis of rotation is connected to the wheel's. Consider the following specifics of the robot:

3.1 Wheel

The wheel was designed and printed using a 3D printer, to allow for more control over the design, keep the costs low and as a lack of better alternatives. As a result, the dimensions of the wheel were restricted to the size of the printing bead. Consider the schematic shown in Appendix 1. The inner wheel shown in the figure above was printed using 0.3kg of Polyethylene terephthalate glycol (PETG), which is a stiff enough material to firmly hold bolts and other metal components. On the other hand, The outer wheel consists of 0.75kg 95A Thermoplastic Polyurethane (TPU), which is a much softer ma-

terial that prevents more slipping when in contact with a surface such as the floor. The wheel has a radius of 9 cm from the axis of rotation.

3.2 Body and Electronics:

The electronics for this system consist of an accelerometer/gyroscope unit, which serves to determine the orientation of the body. A motor, used to rotate the wheel. A battery and driver to power the motor, and lastly an Arduino UNO to control the system. The Adafruit NXP Precision 9DoF was chosen as the motion sensors (IMU) of choice due to its low price point and access Arduino integration. To choose the motor, the worst case scenario, in which the body is 90 °from the top, was considered. With an assumed mass 700g of the body, it was determined that a 2N/m rated motor was more than enough. To accompany the motor, a strong enough battery of 7.4V lipo battery was chosen. The full bill of materials can be seen in table 1.
Once the electronic components have been determined, a body to house the and attach the body to the wheel was designed for the 3D printer. The schematic of the body can be seen in Appendix 2. Consider Fig. 2, showing the assembled electronics compartment.

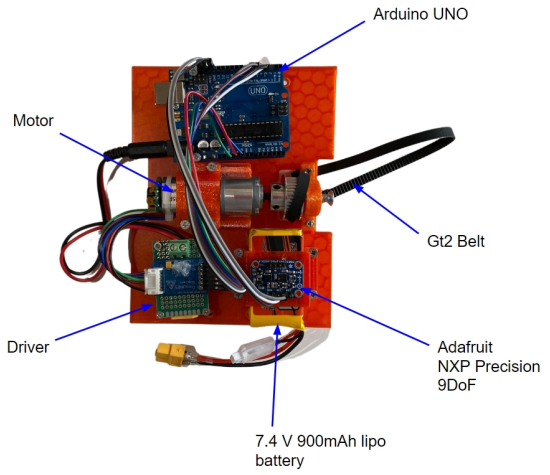


Figure 2: Assembly of electronics on the body of robot

With the body and wheel complete, the assembled robot resembles the render shown in Fig. 1.

4 Model description

4.1 Physical model

To model the system, a simplification is made by reducing the problem to a planer system [4]. A physical model of the system can be seen in Fig. 3. This system closely resembles an inverted pendulum, if certain assumptions are made. Firstly, The wheel experiences no slipping when in contact with the ground, meaning $x(t) = R\theta_w$. Secondly, the torque experienced by the wheel is equal to the torque experienced by the body, hence the system experiences no mechanical or electrical losses. Lastly, all bodies are considered as rigid [6].

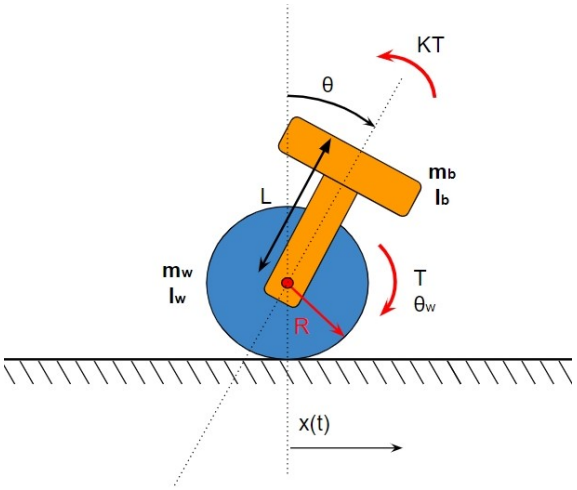


Figure 3: Physical behaviour of robot model

The characteristics illustrated in Fig. 3 can be seen in table 4.1, where mass is m, length L, radius R and inertia I. θ represents the angle and T the torque. Due to the belt and different gear ratios between the wheel and motor, proportional constant K is implemented.

	Wheel (w)	Body (b)
Mass m (kg)	0.586	0.386
length R and L (m)	0.090	0.110
Inertia I (kgm^2)	0.05050	0.00086

Table 1: Design parameters

The mass of the components was determined using a balance scale, whereas the length of the components was measured using a ruler. To determine the inertia of the body, the component was placed 15 °from its equilibrium position to allow for small angle approximation [9]. The body was then released and allowed to oscillate around the wheel's axis of rotation. The period was measured and used in equation 1. The same methodology was used to determine the inertia of the wheel, except a string was used to suspend the wheel, and its length was used as L.

$$I = \frac{T_{period}^2 mgL}{4\pi^2} \tag{1}$$

The dynamics of the system can be modelled using Lagrangian and Hamiltonian dynamic [8][5], show in in equation 2

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} = Q_j \tag{2}$$

As a result, the total kinetic energy of the body can be described [7] [8] by equation 3.

$$T = \frac{1}{2}\dot{x}^2\left[m_w + m_b + \frac{I_w}{R^2}\right] + \frac{1}{2}\dot{\theta}^2[I_b + mL^2] \tag{3}$$

The constraint forces are Q_1 for θ_w and Q_2 for θ are determined in equations 4 and 5.

$$Q_1 = T \tag{4}$$

$$Q_2 = m_b g L sin(\theta) - T \tag{5}$$

Taking into account the assumption that no slipping occurs, and the above-mentioned equations the system can be described as in equations 6 and 7.

$$(C_1)\ddot{x} + (C_2 cos(\theta))\ddot{\theta} = T + (C_6 sin(\theta))\dot{\theta}^2 \quad (6)$$

$$(C_3 cos(\theta))\ddot{x} + (C_4)\ddot{\theta} = C_5 sin(\theta) - KT \quad (7)$$

The system constants can be seen in table 4.1

C_1	$(m_w + m_b)R + \frac{I_w}{R}$
C_2	$m_b LR$
C_3	$m_b L$
C_4	$I_r + m_b L^2$
C_5	$m_b Lg$
C_6	$m_w LR$

Table 2: System constants

4.2 Open loop analysis

By implementing the kinematics equations above, and open loop analysis can be simulated for a step input (torque). to simulate the input torque, an initial tilt angle has been set to 15 °($\pi/12$). The constant K has also been set to 1, which assumes that there is no interference from the gear ratios. Four outputs can be presented, the position of the wheel, x describes how far the wheel would have travelled. The wheel's velocity $\dot{x}$, the body's angle of tilt θ and the angular velocity of the body $\dot{\theta}$. Consider the open loop response shown in Fig. 6. It can be seen that the system is unstable.

In the position time graph, the position can be seen increasing exponentially, with small ripples. These oscillations become more apparent in the velocity time graph, showing that the system loses stability after approximately 0.5 seconds. As expected, the tilt angle decreases rapidly with time. After 5 seconds, the tilt angle has reached 130 radians, indicating that the system cannot afford to have a large delay in the PID controller. The instability of the system is once again seen in the angular velocity graph.

4.3 linearization

The system can be linearized using at the operation point, which occurs at the equilibrium, when $\theta = 0$ °[7]. The system equations can then be rewritten into the following, equations 8 and 9.

$$(C_1)\ddot{x} + (C_2)\ddot{\theta} = T \quad (8)$$

$$(C_3)\ddot{x} + (C_4)\ddot{\theta} = (C_5)\theta - KT \quad (9)$$

To obtain the transfer functions of the linearized system equations, the Laplace transform must be taken, assuming no initial conditions. Consider the input state space below;

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -C_2C_5/(C_4C_1 - C_2C_5) & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -C_2C_5/(C_4C_1 - C_2C_5) & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ (KC_2 + C_4)/(C_1C_4 - C_2C_3) \\ 0 \\ KC_2 + C_4)/(C_1C_4 - C_2C_3) \end{bmatrix} T$$

and output;

$$\begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} T$$

4.4 Transfer function

The transfer function can be modelled using the system components. Consider Fig. 4, indicating the role of the electronic components in the system.

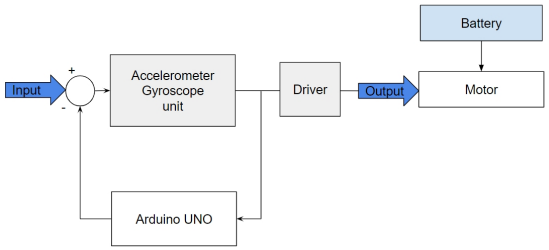


Figure 4: System transfer function in terms of electrical components

Using the determined state space, and the values determined in table4.1, the transfer function of the system can be determined for the four outputs [1]. A value of 1 was used as the constant (K) Consider the following table 3.

$\frac{X(s)}{T(s)}$	$\frac{s^2(C_4+KC_2)-C_5}{s^4(C_1C_4-C_2C_3)-s^2(C_1C_5)}$	$\frac{0.0094(s-6.673)(s+6.673)}{s^2(0.0034s^2-0.2702)}$
$\frac{\dot{X}(s)}{T(s)}$	$\frac{2(s^2(C_4+KC_2)-C_5)}{s^4(C_1C_4-C_2C_3)-s^2(C_1C_5)}$	$\frac{0.0094s(s-6.673)(s+6.673)}{s^2(0.0034s^2-0.2702)}$
$\frac{\theta(s)}{T(s)}$	$\frac{-s^2(C_3+KC_1)-C_5}{s^4(C_1C_4-C_2C_3)-s^2(C_1C_5)}$	$\frac{-s^2 0.6911}{s^2(0.0034s^2-0.2702)}$
$\frac{\dot{\theta}(s)}{T(s)}$	$\frac{-s^3(C_3+KC_1)-C_5}{s^4(C_1C_4-C_2C_3)-s^3(C_1C_5)}$	$\frac{-s^2 0.6911}{s^2(0.0034s^2-0.2702)}$

Table 3: Transfer functions of system

The zeros and poles of the system are denoted in the transfer function's numerator and denominator. A positive pole (8.879), is to be seen. According to control theory, this confirms that the system is unstable. However, the system is said to be proper, meaning it has a greater number of poles than the number of zeros. With this information, this system is physically realizable and can be controlled. Consider the table 4 showing the zeros and poles of the system.

	$\frac{X(s)}{T(s)}$	$\frac{\dot{X}(s)}{T(s)}$	$\frac{\theta(s)}{T(s)}$	$\frac{\dot{\theta}(s)}{T(s)}$
Zeros	± 6.673	$\pm 6.673, 0$	0	0
Poles	$\pm 8.879, 0$	$\pm 8.879, 0$	$\pm 8.879, 0$	$\pm 8.879, 0$

Table 4: Transfer functions of system

With the linearized transfer function, a comparison between the linearized and non-linear step response is made in Fig.5. The plot illustrates that the difference between the two data sets increases, however it can be said that until five seconds, the linearization doesn't experience a very large difference, therefore the approximation holds.

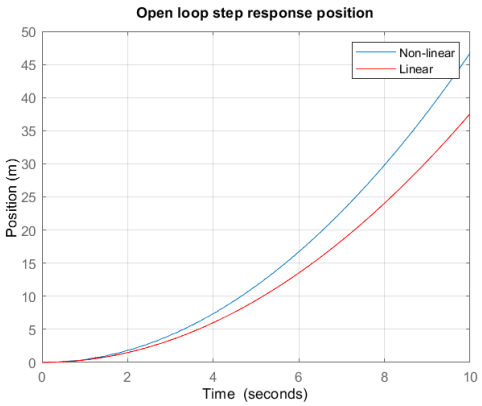


Figure 5: Step response of linear and non-linear open loop

5 Controller Design

5.1 Controller parameters

A PID controller used is this system consists of three gain values, proportional integral derivative. the transfer function of a PID controller is given by the following equation 10 [3].

$$T_c(s) = k_p\left[1 + 1/T_i s + T_d s\right] \tag{10}$$

The Ziegler–Nichols II method [3] was used to determine the gain matrix values needed to control the robot. This method can be considered a physical approach, as the built robot is required. Alternatively, the pole placement method could be used to determine the gain matrix theoretically. The gain matrix consists of the following values.

- The proportional response K_p error multiplier
- The integral response T_i sums the error term over time
- The derivative response T_d factor of error

For the Ziegler–Nichols II method, all PID parameters are set to zero and the proportional gain K_p is increased until the system begins to oscillate. The desired response is a fast yet unstable oscillation. The control parameters are then calculated on the basis of critical value of gain K_u and the resulting period of sustained oscillations, Pu which are described by the following equations 5.1;

$$K_p = 0.6K_u$$
$$T_i = 0.5P_u$$
$$T_d = 0.125P_u$$

Next, the integral response was increased until the system stops to oscillate. However, the integral term reduces the steady state error, but increases overshoot. A large overshoot would cause the system to tilt in the opposite direction to the one that has just been corrected. The overshoot can be reduced using the derivative part, which yields higher gain with stability, yet making the system, sensitive to noise. Using this method, the parameters were calculated to be $K_p = 7.5$, $T_i = 0.165$, and $T_d = 0.5$.

5.2 Signal analysis

The motion sensor (IMU), used in these devices, relies on calibration and filtration of the signal to present accurate data. This process is included in the programming of the sensor, known as the "Sensor fusion algorithm" [2]. The mathematical model behind the IMU consists of a 3-axis gyroscope and 3-axis accelerometer, which are fused together using a Madgwick filter. The filter works as follows; firstly, the gyroscope and accelerometer measurements are obtained from the sensor and normalized. Secondly, the orientation increment from the accelerometers measurement is computed using a gradient step. The same is done of the gyroscope measurements, using numerical integration. Lastly, the calculations are fused to obtain the estimated attitude. This signal analysis is repeated by the IMU, for every time instant, and produces a filtered signal that is sent to the Arduino.

5.3 Behaviour of system

The torque induced on the body is assumed to be proportional to the torque exerted by the wheel, but in the opposite direction. This difference causes an error that the system attempts to reduce, which increases the settling time of the system. To account for this difference, the constant of the torque K has been optimized to a value of 1.4. A closed loop step response for a tilt angle of $\pi/12$ of the system can be simulated using equation 10, Consider Fig.7

Taking the position/time step response, the linear behaviour of the system can be seen until approximately 4m, supporting the small angle approximation. The step response also experiences no overshoot, before settling at 6m. The settling occurs at approximately 12 seconds, meaning the robot will move 6 meters forward in 12 second before coming to a standstill. The velocity time graph indicates that the system will reach a velocity of 1.8m/s when subject to a $\pi/12$ tilt, before removing. It can be seen that the system experiences another increase in velocity around the 6 seconds mark, which is coherent with the angle tilt and angular velocity graph. This signifies that the system is likely to overcompensate, causing an overshoot. In addition, both the tilt angle and angular velocity graphs tend to a value of 0 suggesting that the system will come to a standstill, after approximately 12 seconds.

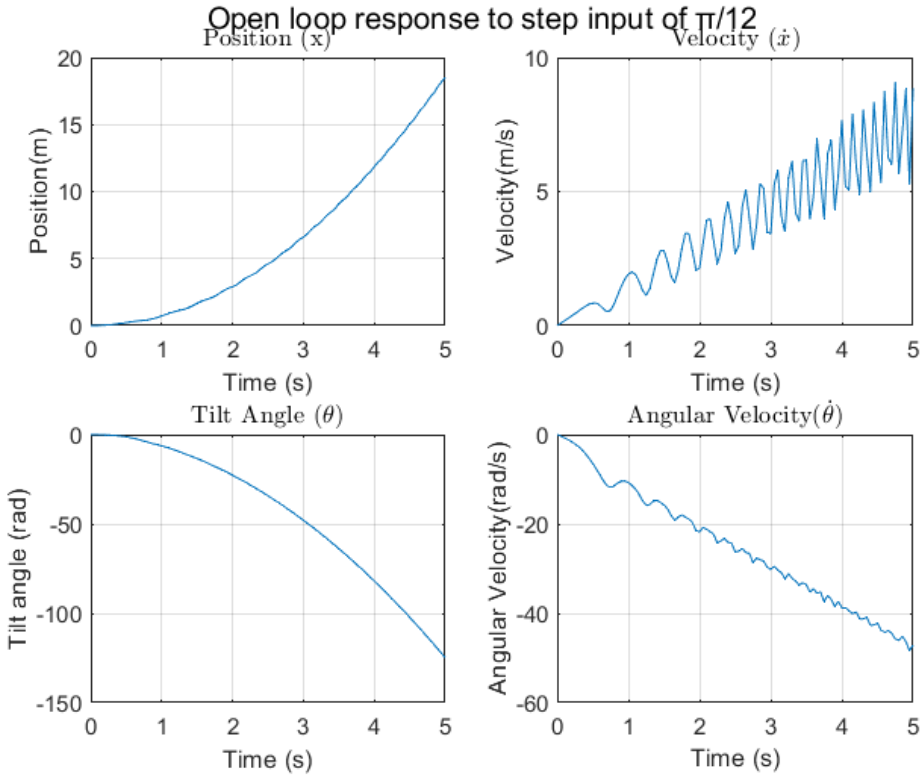


Figure 6: Open loop response

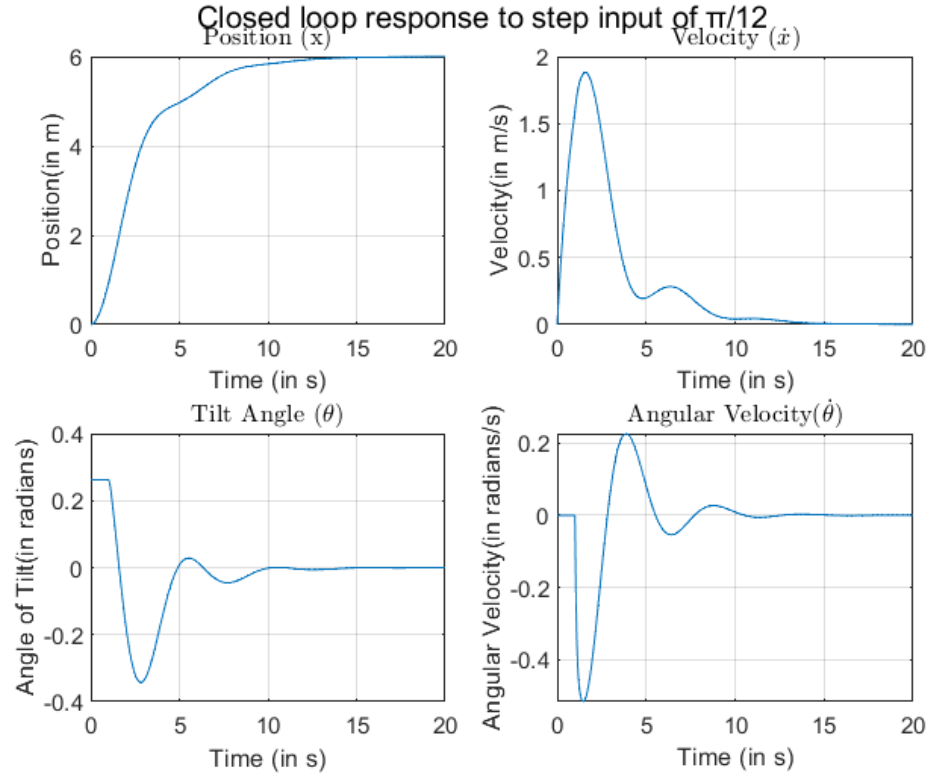


Figure 7: closed loop response

6 Conclusion

To summarize, this paper described the process used to design, model and programme a self-balancing robot. Analysing the behaviour of the system demonstrates the viability of the model. Although, it can be concluded that the model represents the system to some degree of accuracy, there are some limitations. The main drawback is related to the design of the model. As it has been mentioned, the system is one-dimensional. Even though, the robot can stabilize itself while moving forward and backwards, it can tilt to the side, which will result in falling. The model does not account for this, yet it is a common occurrence in the physical behaviour of the robot. To prevent this issue from occurring in the future, the wheel could be printed with a wider base, to limit this effect even more. Alternitavley, the robot could be mounted onto two rails and allowed to only tilt in one direct. Another option would be to implement a method of controlling the side to side swivel and include this behaviour in the model. Another, assumption made in the model, that has a more significant influence than presumed, is the impact of slipping and different surface types. The robot can be seen to preform as expected on softer surfaces such as carpets, whereas it struggles to balance on hard ground. An improvement would be to use a different material for the wheel, use a rail system, allowing the robot to move without any slippage. Regarding the controller parameters, calculated using Ziegler–Nichols II, an improvement for the future would be to determine the parameters theoretically using the pole-zero placement method, and compare the results. Ziegler–Nichols II, is superior in terms of practical application as it account for errors and assumptions that the model omits, however it would be interesting to see if the theoretical PID controller would also work in practice.

Overall, the usage and the analysis of the input signal turned out to be successful. However a problem arose at larger angles, due to a design error. As expressed beforehand, the required motor for this application is a 2N/m motor, however the chosen motor is rated for only 0.4N/m torque. To overcome this issue, a different motor should be used in the design. This was seen in experimental tests, where the motor was not powerful enough to recover the top plate from even around 50° of tilt. Besides buying a more powerful battery and retuning the controller, a design improvement would also be to rectify the belt slipping issue. Instead of relying on the belt, a motor used on drones should be purchased and mounted at the centre of the wheel.

7 List of symbols

Symbol	Name	Unit
€	Euro	-
R	Radius of wheel	m
θ	Tilt angle	radians
θ_w	Angle of wheel	radians
x	Distance	m
L	Length of body	m
m_b	Mass of body	kg
m_w	Mass of wheel	kg
I_b	Inertia of body	kgm^2
I_w	Inertia of wheel	kgm^2
K	Torque constant	N/m
T	Torque	N/m
g	Gravitational constant	$\mathrm{m/s}^2$
s	Laplace transform output	-
$T_c(s)$	PID transfer function	-
k_p	Proportional gain	-
T_i	Integral gain	s
T_d	Differential gain	s
K_u	Critical gain value	-
P_u	Period of oscillations	s

References

[1] Control Tutorials for MATLAB and Simulink - Inverted Pendulum: System Modeling.

[2] How to Fuse Motion Sensor Data into AHRS Orientation (Euler/Quaternions).

[3] dr. H.K. Hemmes, prof. dr. S.G. Lemay, ir. I.C.W.T.A. van Veldhoven, dr. J.W.J. Verschuur, dr. ir. H. Wormeester, and prof. dr. H.J. Zwart. *Modelling, Analysis and Control of Dynamic Systems*. v7.0 edition, September 2022.

[4] Byung Kim and Bong Park. Robust control for the segway with unknown control coefficient and model uncertainties. *Sensors*, 16:1000, 06 2016.

[5] Jerry B. Marion and Stephen T. Thornton. *Classical dynamics of particles and systems*. Saunders College Pub, Fort Worth, 4th ed edition, 1995.

[6] M. Muhammad, S. Buyamin, M.N. Ahmad, and S.W. Nawawi. Dynamic modeling and analysis of a two-wheeled inverted pendulum robot. In *2011 Third International Conference on Computational Intelligence, Modelling Simulation*, pages 159–164, 2011.

[7] Palak Purohit, Poojan Modi, and Udit Vyas. Kinematic control of 2-wheeled segway, 09 2021.

[8] Stephen Thornton. *Classical dynamics of particles and systems*.

[9] Paul Allen Tipler and Gene Mosca. *Physics for scientists and engineers: with modern physics*. W.H. Freeman and Company, New York, NY Basingstoke, sixth edition, [extended version, ch. 1-41, r] edition, 2008.

8 Appendix A

Components	Quantity	Price (Euro)
Arduino UNO	1	4.05
FXOS8700+FXAS21002 (Adafruit NXP Precision 9DoF)	1	14.95
PmodHB5 by Diligent	1	15.99
2S 900mAh lipo battery by SUNPADOW	1	22.99
Motor IG220053X00085R by Diligent	1	5.00
Protoboard	1	0.10
M3 bolts and nuts	20	2.16
M4 bolts and nuts	4	0.56
f686zz bearing	1	2.95
608 bearings	6	4.50
M8 nuts	6	1.90
M8 threaded shaft 18 cm	1	3.70
Gt 20 tooth pulleys	2	3.49
Gt2 belt 23 cm	1	2.95
TPU 95A filament	0.75kg	8.00
PETG filament	0.30kg	3.00
	Total:	96.29

Table 5: List of components used in robot

8.1 Reflection

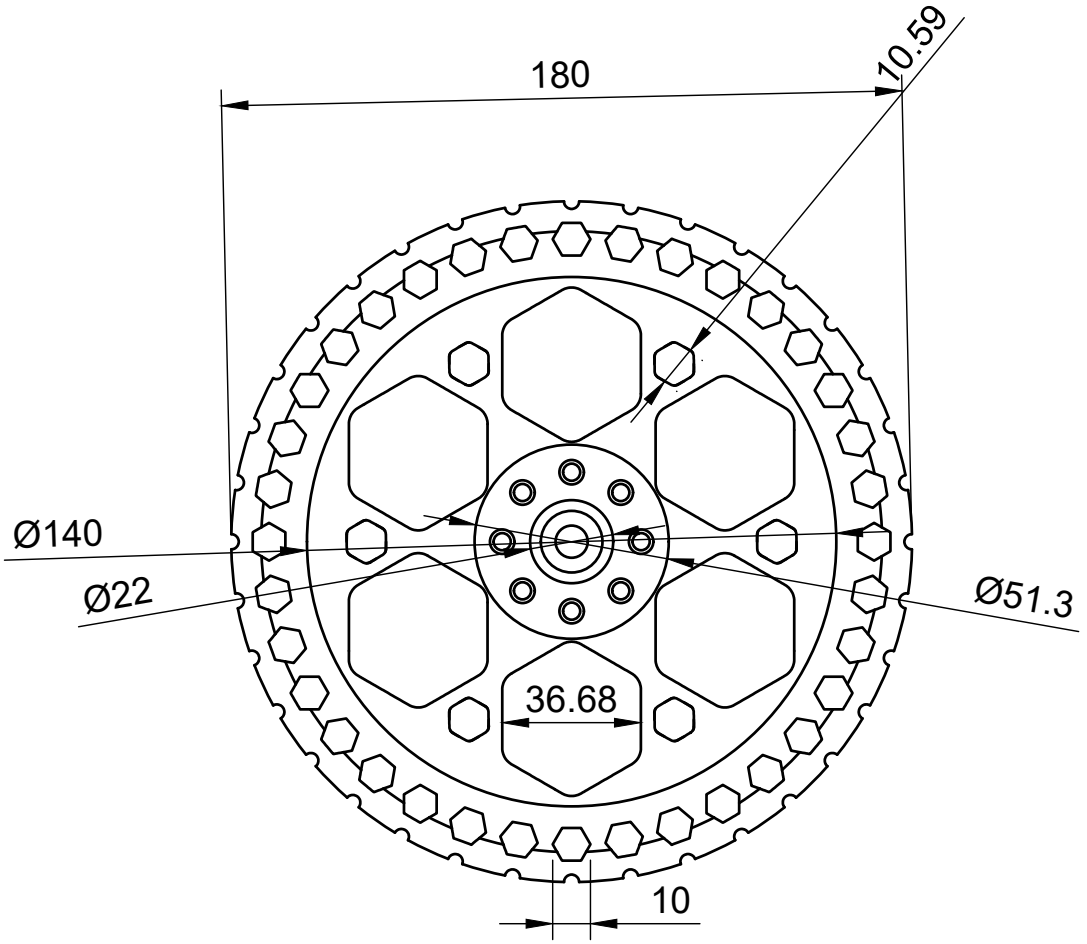
As a group, we worked together in module 4, so we had some initial insight into our collective strengths and weaknesses. In the module, we wanted to focus on dividing the work more evenly, yet remain with a similar work ethic of playing to individual strengths. This proved to be quite challenging, as certain tasks required significantly more work than others, causing frustration in some team members. Looking back at the project plan, we have achieved what we set out for yet some of the issues along the way could have been prevented. For example, we started the project by printing the parts well in advance, but unfortunately, we underestimated the amount of work that is needed to program a PID controller using an Arduino. To prevent this issue, our future time planning should account for exams and also leave even more time for adjustments. Similar to module 4, Brunon Bojków and Viktor Kulić Golja took the lead on designing the robot. Brunon Bojków has exceeded our group's expectations with the design of the wheel, as he spent additional care to perfect it.

Quote from group member Brunon Bojków: In my opinion, we screwed up with task division. Everyone is better at different tasks, and on that we based our task division, which led to a big disproportion in workload. Some tasks just take much less time than others. Also most of the time, not all members were present during meetings which makes communication harder and teamwork less enjoyable.

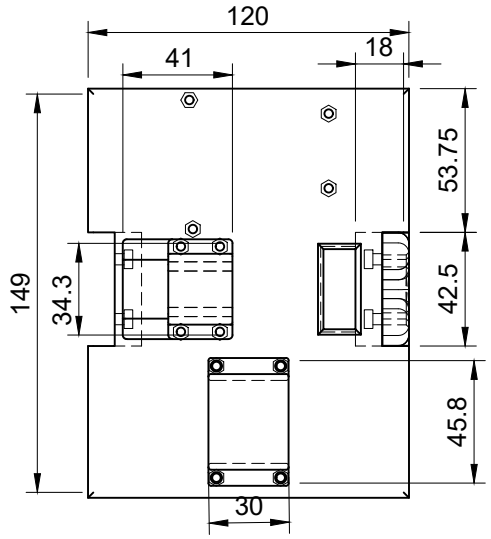
8.2 Point division

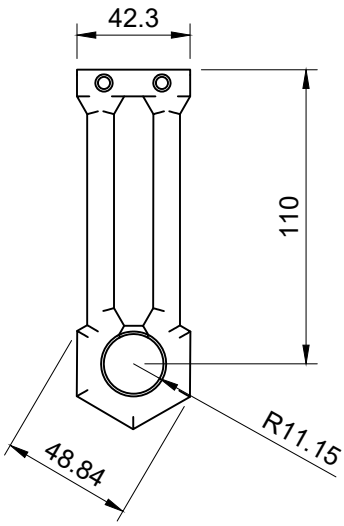
Point giver		Point reciver					
		Viktor	Kanan	Timur	Brunon	Szczepan	Daniel
	Viktor		0.35	0.45	0.6	0.5	0.1
	Kanan	0.575		0.375	0.575	0.375	0.1
	Timur	0.6	0.4		0.5	0.5	0
	Brunon	0.8	0.4	0.4		0.4	0
	Szczepan	0.7	0.2	0.3	0.8		0
	Daniel	0.5	0.2	0.2	0.5	0.6	
	Total	3.175	1.55	1.725	2.975	2.375	0.2

Figure 8: Point division



Dept.	Technical reference	Created by Brunon Bojków 08/11/2022	Approved by	
		Document type	Document status	
		Title Wheel	DWG No.	
			Rev.	Date of issue / Sheet 1/1





Dept.	Technical reference	Created by Brunon Bojków 08/11/2022	Approved by	
		Document type	Document status	
		Title Top plate Connector pole	DWG No.	
			Rev. / Date of issue	Sheet 1/1