# Self-balancing cube

**Daniel Holm**[1] **Rasmus Kalvenes**[2] **Jonathan Vesterlund**[3] **Gustaf Waldén**[4]

Project Advisor: **Olle Kjellqvist**[5]

[1]da5838ho-s@student.lu.se [2]ra0820ka-s@student.lu.se [3]jo2583ve-s@student.lu.se
[4]gu6615wa-s@student.lu.se

***Abstract:*** The aim of this project was to make a cube balance on an edge, or even a corner if time allows it. This would be done with reaction wheels inside the structure. Besides this, the project also focused on working from a system design perspective, including 3D-printing and selecting electrical components. Using a PID controller with some additions, the group successfully balanced a single axis variant of the problem. A functioning LQG controller was also implemented in simulations of the problem. The learning experience was great. The most significant factors for achieving balance were sampling time, motor strength and pendulum length. A cube structure was created, but could not be made to balance due to time constraints and the factors mentioned.

## 1. Introduction

The goal of this project is to explore the possibility of making a cube balance on its edges by utilizing reaction wheels. Additionally, there are some key areas that were focused on in this project in order for the participants to gather additional experience. These are the following:

- Working with a overarching system perspective.
- Producing prototypes by means of 3D printing.
- Writing a functioning control algorithm tailored to the problem.
- Creating a working program for the entire system.

The project was divided into three subtasks where the first was to balance an inverted pendulum on a single axis setup, the second was to balance a cube on an edge and the third was to have the cube rise to an edge from rest.

### 1.1 Previous work

This project has been a replica of the work done by M. Gajamohan et. al[5]. Their work has shown that it is possible to make a cube balance using reaction wheels and it also a source of information in which the group could to some extent find relevant equations for calculations. However, it must be mentioned that a Google search also indicates Gajamohan et al. are not the only group to have done this kind of project. Other sources of inspiration are the many open source projects from ReM-RC[7] and this YouTube video[4].

### 1.2 Concept of reaction wheels

Reaction wheels are a motor-controlled variant of flywheels, i.e. free spinning wheels used to store rotational energy by conserving angular momentum. When the rotational speed of the wheel changes, the body its attached to will counter-rotate due to the conservation of angular momentum. Note that rotation of the body can only happen during torque, meaning that accelerating a still wheel has the same effect as braking an already spinning wheel. By attaching a motor to the fly wheel these torques can be controlled, creating a reaction wheel. One of the most common applications of reaction wheels is in spacecrafts and satellites, where multiple wheels are used to control and stabilize the orientation.

## 2. Physical setups

Besides reaction wheels, one for each axis to be balanced around, there are several other components. One motor to drive each reaction wheel and to control the motors an Arduino UNO R3[3] was used. In addition, an Adafruit Motor Shield v2.0[1] was used to control the motors. The shield has the capability to distribute current to the motors of the system from the battery. Combined with Adafruit's library[2], the shield uses pulse-width modulation (PWM) to control speed and direction of DC motors. This was needed since it allows for a higher voltage and stronger current than what the Arduino can supply. The microcontroller reads the measurements from an inertial measurement unit (IMU) which calculates the current angle of the system. The IMU used was the BNO055 from Bosch Sensortec[8] with the help of Adafruit's library[**IMU-lib**].

### 2.1 Single axis setup

In order to get a better understanding of reaction wheel balancing a single axis setup was designed and 3D-printed. This setup can be seen in Figure 1. The single axis setup was later modified and improved. The full assembly, including electronic components can be seen in figure 2. Despite it being a simple structure, it has the benefit of easy access to port. The microcontroller was not attached to the structure, it laid flat on the table. A prototype build like this allowed us to detect which components are sufficient for the end goal.

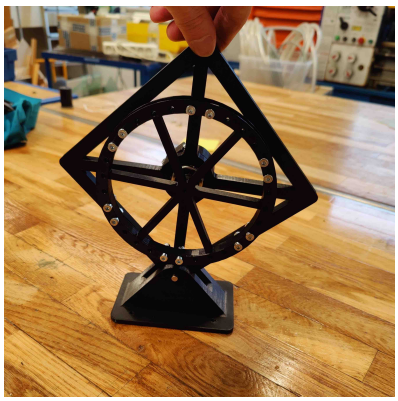

**Figure 1.** The first version of the single axis setup

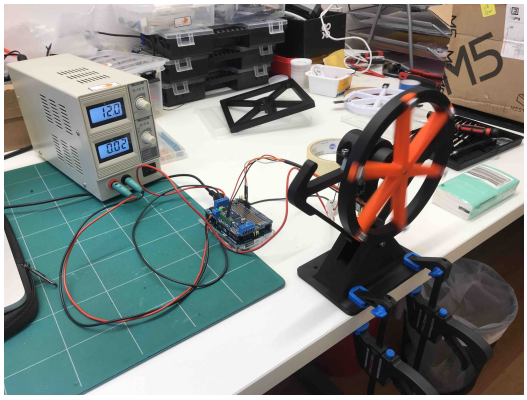

**Figure 2.** The second version of the single axis setup, with electrical components

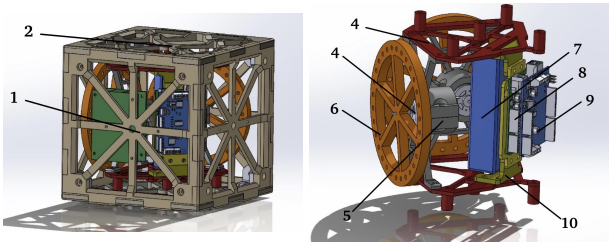


**Figure 3.** The components in the system numbered.

### 2.2 The cube

Designing the structure for the cube proved to be a lot more difficult and time consuming than expected. The intricate design demanded things to be decided based on assumptions, which later turned out to be incorrect. Some parts had to be redesigned multiple times. Some parts, e.g. four sides of the cube, were made identical to make the printing process less difficult. Other parts, engine mounts among them, were made symmetrical for the same reason. An advantage of having multiple identical parts is that if one part fails, it can potentially be replaced by another. While the replacement part is being printed, a side of the cube that is not critical when testing can be relocated into the critical position. The design of the cube can be seen in Figure 3 with the parts named in Table 1.

| Component | Number |
|---|---|
| Cube, side | 1 |
| Cube, top and bottom | 2 |
| Mounting plate | 3 |
| DC motor | 4 |
| Motor holder | 5 |
| Reaction wheel | 6 |
| Battery | 7 |
| Arduino | 8 |
| Arduino Shield | 9 |
| Electronics holder | 10 |

**Table 1.** The components in the system.

## 3. System dynamics

For both the cube and the single axis setup, the system dynamics can be analyzed in one plane only, provided the reaction wheel is parallel to that plane. In that case, the system is effectively an inverted pendulum constrained to one axis around a pivot point. When doing the mathematical analysis, we focused on the single axis setup as in Figure 4,

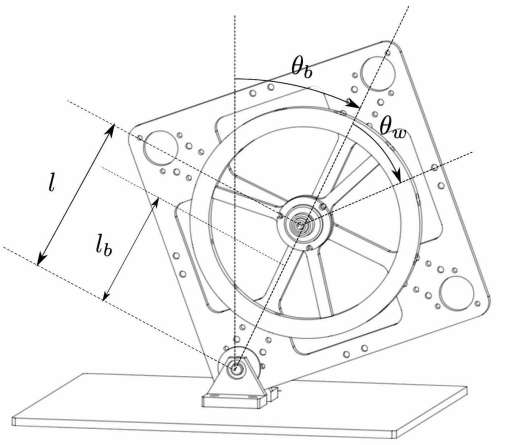


**Figure 4.** Diagram of single axis setup from [5].

where $\theta_b$ is the tilt angle of the pendulum body and $\theta_w$ is the angle of the wheel with respect to the body. The distance between the motor and the pivot point $O$ is denoted by $l$ whereas $l_b$ is the distance between the pendulum body's center of mass and $O$. Due to the symmetry of our prototype we assumed that $l = l_b$. We also assumed zero friction in the pivot point due to the use of a ball bearing. With these assumptions, we can modify the system of equations from [5] to get

$$\begin{aligned} \ddot{\theta}_b &= \frac{(m_b + m_w) g l \sin\theta_b - T_m + C_w \dot{\theta}_w}{I_b + m_w l^2}, \\ \ddot{\theta}_w &= \frac{(I_b + I_w + m_w l^2)\,(T_m - C_w \dot{\theta}_w)}{I_w (I_b + m_w l^2)} \\ &\quad - \frac{(m_b + m_w) g l \sin\theta_b}{I_b + m_w l^2}, \end{aligned} \tag{1}$$

where $m_b$ is the mass of the structure and the motor, $m_w$ is the mass of the reaction wheel, $T_m$ is the motor torque, $C_w$ is the dynamic friction coefficient of the wheel. $I_w$, $I_b$ are the

moments of inertia of the wheel around its rotational axis and the pendulum body around $O$ respectively.

### 3.1 Motor dynamics

The motors used were initially simple brushed DC motors whose relation between torque $T_m$ and current $i$ drawn by the motor can be described as

$$T_m = K_m i, \tag{2}$$

where $K_m$ is the torque constant. We can hence describe the acceleration of the reaction wheel as

$$\ddot{\theta}_w = \frac{K_m i - C_w \dot{\theta}_w}{I_w}. \tag{3}$$

### 3.2 State-space model

We can now combine (1) and (3) to get a full model that accounts for the current $i$. By using the approximation of small angles, $\sin\theta \approx \theta$, the system of equations becomes linear. The state variables are chosen as $\mathbf{x} = [\theta_b \ \dot{\theta}_b \ \dot{\theta}_w]^T$. Linearizing around $[0\ 0\ 0]^T$ gives

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + Bu, \\ \mathbf{y} &= C\mathbf{x}, \end{aligned} \tag{4}$$

where

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 & 0 \\ \frac{(m_b+m_w)gl}{I_b+m_w l^2} & 0 & \frac{C_w}{I_b+m_w l^2} \\ -\frac{(m_b+m_w)gl}{I_b+m_w l^2} & 0 & -\frac{C_w(I_b+I_w+m_w l^2)}{I_w(I_b+m_w l^2)} \end{bmatrix}, \\ B &= \begin{bmatrix} 0 \\ -\frac{K_m}{I_b+m_w l^2} \\ \frac{K_m(I_b+I_w+m_w l^2)}{I_w(I_b+m_w l^2)} \end{bmatrix} \text{ and } C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \end{aligned} \tag{5}$$

where the control signal $u$ is the current $i$ and the output signal $\mathbf{y}$ is both $\theta_b$ and $\dot{\theta}_b$. The goal is to control only $\theta_b$ and stabilize it around 0, but the angular velocity $\dot{\theta}_b$ will also be directly readable from sensor data.

The state-space model was then discretized using zero-order hold to get the discrete-time model

$$\begin{aligned} \mathbf{x}_{k+1} &= A_d\mathbf{x}_k + B_d u_k, \\ \mathbf{y}_k &= C\mathbf{x}_k, \end{aligned} \tag{6}$$

where $A_d$ and $B_d$ are the discrete-time counterparts to $A$ and $B$.

### 3.3 Parameter identification

To simulate the state-space model, the physical parameters had to be identified. $I_b$ was calculated with

$$I_b = \frac{s_f^2}{6} m_f + \frac{r_m^2}{2} m_m + (m_f + m_m) l^2, \tag{7}$$

where $s_f$ is the side length of the frame, $r_m$ is the radius of the motor, $m_f$ is the mass of the frame and $m_m$ is the mass of the motor. $I_w$ was calculated with

$$I_w = \frac{1}{2} m_w (r_{\text{outer}} + r_{\text{inner}})^2 \tag{8}$$

where $r_{\text{outer}}$ is the radius of the wheel and $r_{\text{inner}}$ is the radius to the inner edge of the wheel. The maximum torque of the motor can be found in [6]. The torque constant $K_m$ can also be calculated from the data sheet, using the equation

$$K_m = \frac{T_2 - T_1}{i_2 - i_1}, \tag{9}$$

where the torques $T_1, T_2$ and currents $i_1, i_2$ were chosen at two different points.

## 4. Controlling the process

Two control strategies were considered, PID-control and an observer-based controller using LQG.

### 4.1 PID-control

The PID controller continuously reads the current output signal $\theta_b(t)$ and compares it to the reference signal $r(t)$ to calculate the current error $e(t) = r(t) - \theta_b(t)$. The control signal $u(t)$ is then decided according to

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{d}{dt} e(t), \tag{10}$$

where the control parameters $K_P$, $K_I$ and $K_D$ are predetermined. Since the objective is to balance a pendulum, i.e., make $\theta_b = 0$. The reference will therefore be set to $r = 0$. Since the controller has to be discretized in the implementation the integral is replaced by a sum and the derivative by a difference quotient. If the output signal is read at time intervals of $\Delta t$ we get

$$\begin{aligned} u(t) = K_P e(t) + K_I \sum_{k=0}^{N} [e(k\Delta t)\Delta t] + K_D \frac{e(t) - e(t-\Delta t)}{\Delta t}, \\ t = N\Delta t, \end{aligned} \tag{11}$$

for the number of samples $N$. The sampling time $\Delta t$ can be baked into the parameters $K_I$ and $K_D$ leading to fewer calculations in the code implementation. For ease of reading we let

$$\begin{aligned} I(t) &= \sum_{k=0}^{N} [e(k\Delta t)\Delta t], \\ D(t) &= \frac{e(t) - e(t-\Delta t)}{\Delta t}, \\ t &= N\Delta t. \end{aligned} \tag{12}$$

### 4.2 Additions to the PID controller

The first addition is adjusting the reference signal based on the current error. If the reference is moved further away from the current output the error will be larger and the controller will have a stronger response. For an angle increment of $\Delta\theta$ the updated reference angle $r(t + \Delta t)$ will be

$$r(t+\Delta t) = \begin{cases} r(t) + \Delta\theta, & e(t) > 0. \\ r(t) - \Delta\theta, & e(t) < 0. \end{cases} \tag{13}$$

This will have the most significant impact on the integral part of the controller. The increment value should be relatively small, especially with a quick sample time, otherwise the controller may not be able to catch up. This can be beneficial if you want especially the integral part to grow quickly. We also introduce an integral saturation $I_{\max}$, making the sum $I(t)$ capped to $\pm I_{\max}$.

Another addition to the PID controller was motor feedback. This was achieved by having the control signal feedback into itself. With a discretized controller this becomes quite simple by introducing a fourth control parameter, $K_C$, and adding the term $K_C u(t - \Delta t)$ to equation (11), giving

$$u(t) = K_P e(t) + K_I I(t) + K_D D(t) + K_C u(t - \Delta t). \tag{14}$$

### 4.3 LQG control

First, we introduce independent disturbance and noise processes $\mathbf{w}_k$ and $\mathbf{v}_k$ into the discrete-time system from (6) to get

$$\begin{aligned} \mathbf{x}_{k+1} &= A_d \mathbf{x}_k + B_d u_k + \mathbf{w}_k, \\ \mathbf{y}_k &= C\mathbf{x}_k + \mathbf{v}_k. \end{aligned} \tag{15}$$

Let both processes be white Gaussian noise with covariance matrices $V$ and $W$ respectively. By introducing a Kalman filter and state feedback we form the control equations

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} &= A_d \hat{\mathbf{x}}_k + B_d u_k + L(\mathbf{y}_k - C\hat{\mathbf{x}}_k), \\ u_k &= K\hat{\mathbf{x}}_k, \end{aligned} \tag{16}$$

where $\hat{\mathbf{x}}_k$ is the estimated state vector. In order to use equation (16), both matrices $K$ and $L$ need to be determined. They were determined by solving the linear-quadratic-Gaussian (LQG) optimal control problem. The feedback gain $K$ is chosen such that it minimizes the cost function

$$J(u) = \sum_{k=1}^{\infty} \mathbf{x}_k^T Q \mathbf{x}_k + u_k^T R u_k, \tag{17}$$

for some weight matrices $Q$ and $R$. Both the feedback gain $K$ and the Kalman gain $L$ can be obtained by solving their respective algebraic Riccati equations. This was done in Matlab using the commands `dlqr` and `dlqe`, which determines the $K$ and $L$ matrices respectively. For `dlqr`, the weight matrices had to be chosen, which was done by experimenting with different penalties. For `dlqe`, the covariance matrix for the measurement noise could be estimated using measured data. Estimating the covariance matrix for the noise affecting the states is difficult, and therefore an identity matrix is used instead.

### 4.4 Motor limitations

Most of the components used were off-the-shelf components, that were available at the university. This was a limiting factor, especially in terms of motors. A weak motor that provides low torque, puts a restriction on the maximum recovery angle. By analyzing the system statically at some tilt angle $\alpha$, we can calculate the motor torque necessary to cancel out the torque $T_O$ in the pivot point due to the gravitational force on the system. The torque $T_O$ can be calculated as

$$T_O = (m_b^* + m_w) g l \sin\alpha, \tag{18}$$

where $m_b^*$ is the mass of the pendulum body and the motor being considered. This equation was used to estimate motor requirements depending on the mass of the motor and desired recovery angles. The torque $T_O$ can be seen as the bare minimum to recover from $\alpha$. In practice, much higher torque is required if the pendulum is to also accelerate towards the equilibrium. Note also that the pendulum will not be still when at this angle $\alpha$. It will have fallen there and thus have a non-zero angular velocity, requiring an even higher torque.

### 4.5 Sampling limitations

Apart from torque generation in the motors, the sampling time is also an important factor. In these types of unstable (open loop) systems a quick sampling rate is of critical importance in order to recover from disturbances before it is too late. Although a fast sampling rate is desired, it is limited by the hardware. The most limiting factor for the sampling time was the IMU, which outputs data at 100 Hz[8] i.e. a sample time of 10 ms.

## 5. Results

### 5.1 Selecting the new motor

Out of the available off-the-shelf motors the L1491210[6] from Micro Motors was selected. According to (18), it should momentarily hold a tilt angle of 4°, which was more than twice as powerful as the first motor used.

### 5.2 Physical parameters

The physical parameters were identified for the single axis setup. These values were used for the simulations and can be found in Table 2.

| Parameter | Value |
|---|---|
| $I_b$ | $2.1 \cdot 10^{-3}$ kg · m$^2$ |
| $I_w$ | $6.4 \cdot 10^{-3}$ kg · m$^2$ |
| $m_b$ | 0.170 kg |
| $m_w$ | 0.076 kg |
| $l$ | 0.093 m |
| $K_m$ | 0.5 Nm/A |
| $T_{\max}$ | 0.015 Nm |

**Table 2.** Parameters and their measured values.

The only parameter that was not identified was the friction coefficient, $C_w$. Instead, the same value as the one in [5] was used, $C_w = 5 \cdot 10^{-5}$ kg · m$^2$ · s$^{-1}$.

### 5.3 Measurements from the IMU

The structure was tilted slowly from side to side while the angle was measured at 10 ms intervals. The results can be seen in Figure 5.

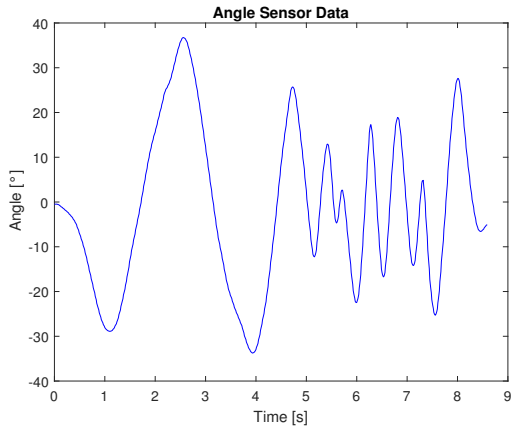


**Figure 5.** Angle measurements from the IMU.

The structure was then laid still while the angular velocity from the gyrometer was measured at 10 ms intervals. The results can be seen in Figure 6.

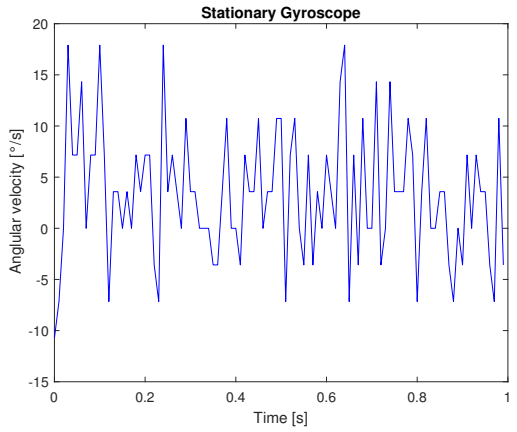


**Figure 6.** Angular velocity measurements from the IMU.

### 5.4 Simulations

The model in (4) and (5) was simulated using both a PID controller and a LQG controller. The simulation results can be seen in Figure 7 and Figure 8. For both simulations, a sample time of 15 ms and a pulse disturbance of 0.5° was used.

The parameter settings for the discrete PID controller can be seen in Table 3.

| Parameter | Value |
|---|---|
| $K_P$ | 3 |
| $K_I$ | 1 |
| $K_D$ | 0.3 |

**Table 3.** The chosen PID parameters.

The optimal gain matrices $K$ and $L$ were calculated as

$$K = \begin{bmatrix} 6.0497 & 0.7365 & 0.0611 \end{bmatrix}, \qquad L = \begin{bmatrix} 1 & 0 \\ 14.0934 & 0.0825 \\ 0.8845 & 0.0048 \end{bmatrix}. \tag{19}$$

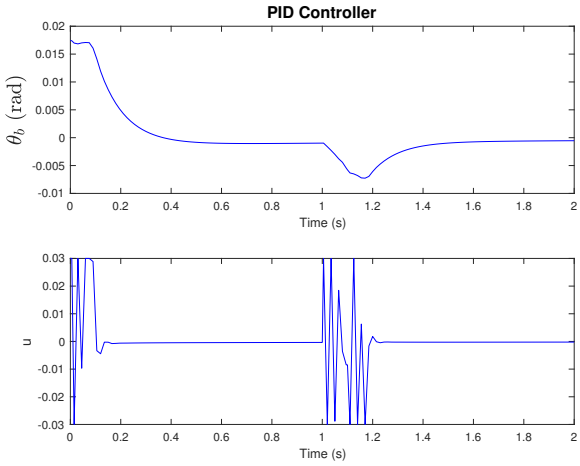


**Figure 7.** Simulation of the single axis setup with a disturbance, using the PID controller.

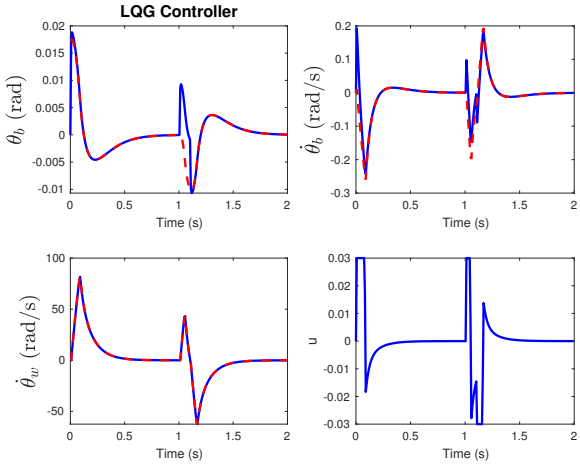


**Figure 8.** Simulation of the single axis setup with a disturbance, using the LQG controller. The blue line represents the estimated values and the red dashed line represents the values calculated from the state-space model.

While experimenting with different parameter settings for the simulation, we observed that some had a larger effect on the system than others. By changing the parameters in the PID simulation, with different factors, the parameters importance could be observed. The observation was based on how high, or low, of a factor a parameter could handle, before the system achieved instability. Sampling time, maximum torque and length to center of mass had the largest effect. This was followed by the mass of the frame and the radius of the wheel. In contrast, the system was still stable with a high friction coefficient and a low mass of the wheel. This qualitative observation is summarized in Table 4.

| Parameter | Factor | Importance |
|---|---|---|
| $\Delta t$ | 1.2 | Very high |
| $T_{\max}$ | $2.7^{-1}$ | High |
| $l$ | 2.8 | High |
| $m_b$ | 4.5 | Medium |
| $r_w$ | $4.5^{-1}$ | Medium |
| $m_w$ | $< 20^{-1}$ | Low |
| $C_w$ | 40 | Low |

**Table 4.** Parameters and their observed importance.

### 5.5 The physical setup

The discretized controller from (11) was implemented with a sample time of 10 ms with the original DC motors given to us. The motor was eventually swapped out to the new one[6]. After this, testing could begin again. This time the motor feedback from (14) was also implemented. Experimenting with PD-control and positive values of $K_C$ seemed promising but balancing still proved difficult. In a last ditch effort a new design for the pendulum body was made. The improved structure can be seen in Figure 2.

After some inspiration from[4] the controller now made use of integral action with a positive $K_I$ as well as reference adjustment according to (13). $K_C$ was now chosen to be negative and relatively small. The idea was to slow down the wheel over time when balancing. After some tuning stable balancing was finally achieved with the following control parameters:

$$K_P = 50, \quad K_I = 150, \quad K_D = 2.5, \quad K_C = -0.2, \\ I_{\max} = 127, \quad \Delta\theta = 0.004. \tag{20}$$

Note that the control signal ranges between ±255 (where 255 is simply the max speed of the motor) and the output signal is measured in degrees. The parameter $\Delta\theta$ is also in degrees. In the code implementation, $\Delta t$ was also baked into the constants to reduce the amount of calculations. In Figure 9 we can see roughly 15 seconds of balance using the control parameters above. The sensor readings shown are spaced 50 ms apart, but the controller still acted at the usual 10 ms intervals.

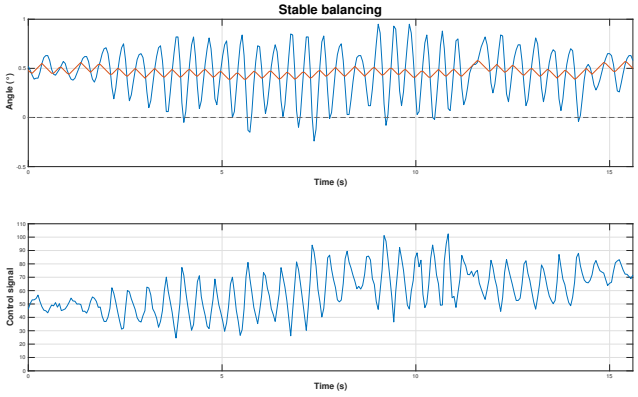


**Figure 9.** Top graph: the measured angle $\theta_b$ is in blue and $r$ is in orange. Bottom graph is the control signal.

There are some key takeaways from this graph. Firstly, The structure is not balancing around 0°. Instead, the reference is oscillating slightly around 0.46°. The standard deviation in error over this period was $\sigma_e = 0.24°$. Secondly, the control signal is also oscillating around a positive, non-zero speed.

Note that these 15 seconds were an excerpt from a longer period of balancing. In reality, the structure could balance for several minutes. The longest measured time balancing was just shy of 37 minutes. It could also handle small disturbances such as shaking the table and gently pushing the structure.

### 5.6 The cube

The physical model can be seen in Figure 10. The printed parts, structural screws and nuts had a total mass of 587 grams. The scale that was used had an upper limit of 1 kg, which the cube exceeded with the rest of the internal components assembled.

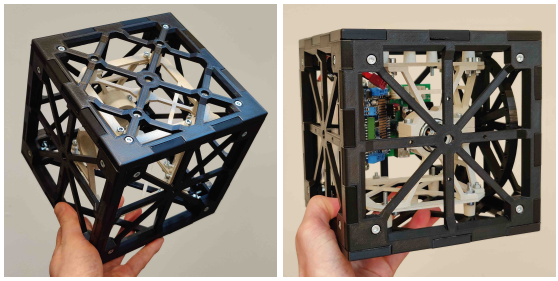

**Figure 10.** The printed cube with and without components

## 6. Discussion

Our supervisor said during a meeting that "There are two ways of solving a control problem. Either, one designs the system and actively chooses which components are to be used, or one is given a complete system and has to do a lot more work on control modelling and algorithms."

### 6.1 Limitations in the simulation

There were a number of limitations in the simulation that has to be addressed. First of all, the simulation uses the linearized model in (4). This does not completely represent the reality of the dynamics of the single axis setup. However, since the working range of the angle of the system is only a couple of degrees, it was reasonable to use the linearized model. Second of all, many of the parameters used in the model were approximated. The moments of inertia were all derived using standard formulas and could have been more accurate. Another example is the friction coefficient, which was taken from [5]. Although there are a number of limitations to the simulation, the purpose of it was not to get a working controller that could be directly applied to the real model, but instead to see how a PID controller compares to a LQG controller, and which parameters are important.

### 6.2 Interpreting the simulations

As seen in Figures 7 and 8, both controllers could stabilize the system with the given set of parameters. They could also handle an impulse disturbance of 0.5°. When comparing the

two controllers, one instantly notes the smoother response of the LQG controller. However, the LQG is highly dependent on a good model of the system and to what extent our model and parameters are sufficient, will not be answered in this paper, since a LQG controller was not implemented in the real system. Both controllers in the simulation used a sampling time of 15 ms. This was done in order to insure that the sampling time was not impossible to meet.

As mentioned in Table 4, the most significant parameters were the sampling time, the maximum torque that the motor can generate and the length to center of mass from the pivot point. This is reasonable since a longer length to the center of mass will increase the torque required cancel the gravitational force. The motor torque is also important to counter the torque generated by the body itself. It is also reasonable that the sampling time is significant, since even a small change in angle will have large effect on the necessary torque to counter the fall and the change in angle should be measured as fast as possible. Also, the mass of the frame and the radius of the wheel, seemed less important. This is likely due to the fact that the mass of the frame is already low, and the same applies to the length to center of mass. The same applies to the friction coefficient and the mass of the wheel, which both already had low values. It is important to note that this observation was only done qualitatively, since the simulation is based on approximations. The purpose was to observe the trends mentioned above rather than finding the exact parameters that can be handled. With this said, it would be recommended to initially lower the sampling time, increase the maximum torque and diminish the length to center of mass, before changing anything else.

Comparing the PID parameters in Table 3 with the real ones, one notices a large difference. This is because of a number of reasons. One being that the control signal is very different when comparing Figure 7 with Figure 9, since the simulation input signal is in ampere and the code for the real process is the velocity for the wheel. Also, the simulation inputs angles in radiance, while the real process inputs angles in degrees. Another reason why the PID parameters differ, is because the physical parameters differ between the simulation and the real process. This is true, since the real setup used is not the same as in Figure 4, which is the structure simulated. Also, most parameters were identified and can differ from the real physical parameters.

### 6.3 Tuning the PID controller on the single axis setup

The PID controller was tuned through trial and error with the help of some intuitive reasoning. Most of the early tuning was done with only PD-control. Due to the system's instability there can be no steady state error, leading us to believe no integral action should be used. But these ideas proved to be flawed.

Firstly, while it is true that we may want to ease up on the torque when the pendulum is already on its way up, that does not mean to lower the control signal. Remember that the control signal is (roughly) proportional to speed so a lower control signal actually means decelerating the wheel. If the wheel decelerates while the pendulum is rising the effect is a torque going the opposite way, sending the pendulum back down. So a too large $K_D$ may hurt the balancing.

Secondly, by introducing integral action we can get the desired behavior of increasing the speed, which maintains torque, even when the pendulum is rising. The introduction of integral action is likely also why the control signal oscillates around a positive base speed instead of 0. It is then important to have integral action that reacts quickly, but does not grow too large. This is why setting $I_{\max} < u_{\max}$ and adding reference adjustment eventually led to the best results. Another advantage of reference adjustment is that it allows the process to find its own equilibrium. A very interesting example of this can be found in [4] at 11:25. If there is a discrepancy between what the IMU registers as 0° and the physical equilibrium, it could be hard to achieve balance if always controlling to $r = 0$. This explains the results seen in Figure 9. The reference adjustment turned out to be crucial for this reason.

### 6.4 IMU signal

Surprisingly, when measuring the output angle from the IMU we could not detect any noise at all (Figure 5). Despite this the raw data readings from the gyrometer were very noisy (Figure 6). This is problematic if implementing state feedback in the microcontroller. A good observer would then be needed, consequently requiring a more accurate model to develop this observer with the help of simulations. An alternate way of estimating $\dot{\theta}_b$ is of course with a difference quotient, as in the discretized PID controller. We considered using the gyrometer readings as a more accurate estimate of the derivative even with PID control, but after discovering the noisiness we decided against it.

### 6.5 Weight reduction

In cases where more time is available, the final design can be developed through several iterations. For this project, time was sufficient for designing a single axis setup as a pilot study for the actual cube. While the thickness of the cube frame was heavily reduced, compared to the previous prototype, at points where it could be done, after printing it was concluded that the thickness could have been reduced even more in order to save weight. Although, the group is concerned if such a weight reduction would be enough for the motors to be strong enough to balance the cube. The components on their own also contributed to the total weight. With this in mind, even stronger motors would almost certainly be needed if the cube is to be driven by a large battery like in this case.

## References


[1] Adafruit. *Adafruit motor shield v2 — overview*. 2013. URL: https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/overview.

[2] Adafruit. *Adafruit motor shield v2 arduino library*. URL: https://adafruit.github.io/Adafruit_Motor_Shield_V2_Library/html/annotated.html.

[3] Arduino. *Uno r3 — arduino documentation*. URL: https://docs.arduino.cc/hardware/uno-rev3.

[4] B. E. Channel. *Lego, raspberry and python project - reaction wheel inverted pendulum*. 2022. URL: https://www.youtube.com/watch?v=WObG2LoSEwQ.

[5] M. Gajamohan, M. Merz, I. Thommen, and R. D'Andrea. *The cubli: a cube that can jump up and balance*. 2012. URL: https://ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-control/idsc-dam/Research_DAndrea/Cubli/Cubli_IROS2012.pdf.

[6] M. Motors. *Gear-motors: series l149*. URL: https://media.distrelec.com/Web/Downloads/80/06/05448006.pdf.

[7] ReM-RC. URL: https://github.com/remrc?tab=repositories.

[8] B. Sensortec. *Bno055: intelligent 9-axis absolute orientation sensor*. 2014. URL: https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf.