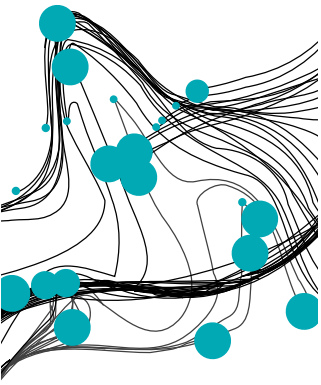
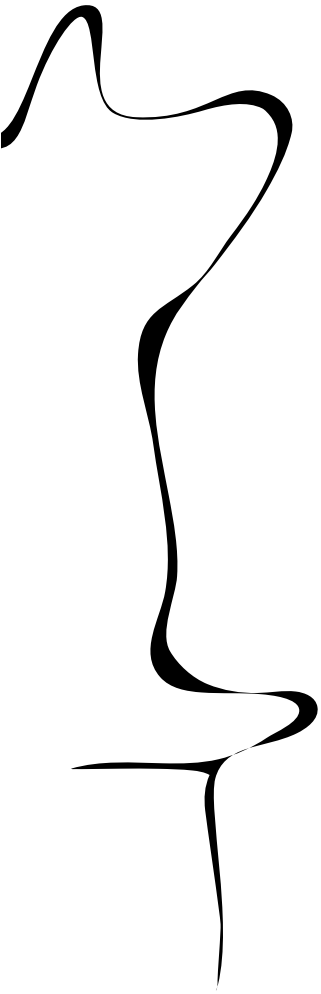


RAM ROBOTICS AND MECHATRONICS



ENERGY RECUPERATION BETWEEN DC MOTORS USING DYNAMICALLY RECONFIGURABLE SWITCH GRID

E.J.J. (Jelle) Wilbrink

MSC ASSIGNMENT

Committee:
dr. F.J. Siepel
dr. V. Groenhuis, MSc
dr. ir. A.B.J. Kokkeler

August, 2024

048wilbrink2024
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE
UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Acknowledgements

I am deeply thankful to dr. Vincent Groenhuis, MSc, as my daily supervisor. With his ample experience supervising students, he has guided me throughout the process of this thesis. His deep understanding of the system has brought new insights to the table, his practical skills proved valuable for making the prototype, and he aided the scientific quality by asking critical questions at the right times and giving elaborate feedback on my work.

I also want to express my gratitude to the technicians at RaM for their help with making the experimental setup, and to prof. dr. ir. S. Stramigioli, for sharing his ideas and input.

Lastly, I want to thank dr. F.J. Siepel and prof. dr. ir A.B.J. Kokkeler for being part of my MSc committee.

Summary

Energy saving in industrial robotics can directly contribute to reaching climate goals, as well as reducing operating costs. Several methods exist for this, such as regenerative braking, where braking energy from motors is recuperated and reused to accelerate motors again. This report proposes a novel regeneration method where motors are directly connected to each other to share and reuse their energy. The benefits and drawbacks of this method are investigated, and it is found to be favourable over conventional capacitor-based regeneration in terms of the systems size and weight, but the motor-to-motor regeneration system designed in this research costs more than capacitor-based regeneration would.

A setup is designed and implemented to evaluate the main advantage the proposed motor-to-motor regeneration might have: reduced power consumption. The design requires an interconnection of switches to dynamically connect the various power sources and consumers. Several interconnect topologies are compared and finally, a minimal crossbar topology is chosen for the design. The interconnect also generates a PWM signal to control the speed of the motors. The maximum frequency for this was found to be 10 Hz limited by the speed of the switches in the interconnect and by the I2C buses used to control the switches from the microcontroller that was also included in the system. After calibrating the voltmeters and motor constants for the experimental setup, the power consumption of the system was measured.

The experiments showed that motor-to-motor regeneration reduced the system's power consumption by up to 30 %, or even up to 33 % when combined with capacitor regeneration, whereas capacitor regeneration alone only yielded a reduction of 15 %. However, even with regeneration, the system was still significantly less efficient than a state-of-the-art motor controller, but it seems reasonable to believe that adding the regeneration methods discussed in this report would further allow for further power savings.

Looking at the results in more detail, motor-to-motor regeneration was found to be ineffective when the motors are in-phase and most effective when they are in anti-phase. Capacitor regeneration does not have this dependency on phase shift. Furthermore, regeneration brings a larger reduction when the motors are storing much energy and have high voltage. Besides, when the period of the waveform increases the power usage gets lower independent of regeneration used because keeping the motors running at constant speed requires less energy than accelerating or decelerating them. Finally, it is found that using motor-to-motor regeneration causes the motors to accelerate and brake slower.

Contents

1 Introduction

Huge amounts of energy are used each year for powering electric motors in robots and other machines. Industrial robotics alone might use as much as 0.8 % of the total electrical energy in the United States in 2025 [1]. Therefore, powering motors more efficiently can directly contribute to reaching climate goals. The European Union, for example, has committed to reducing the annual energy consumption by 32.5 % by 2030 [2]. Besides, power savings can also bring operating costs down.

To achieve a lower power consumption of electric motors, one can look at how they are operated. They repeatedly accelerate and decelerate during their operations. This behaviour can be used to power them more efficiently, for example by recuperating and storing braking energy such that it can be reused. This process is known as regenerative braking and it can be implemented by adding a capacitor to the system to store the energy [3]. This report proposes a novel regeneration method where braking energy from one motor can directly be reused by another motor by electrically connecting them: motor-to-motor regeneration (Figure 1.1).

The main goal is to evaluate the feasibility of this concept, considering the power draw, physical dimensions, mass, and cost. How does its power consumption compare to more traditional capacitor-based regeneration and to not using any regeneration? How does the power consumption depend on the properties of the target waveform? How does its power draw compare to the power draw of a state-of-the-art motor controller? How does motor-to-motor regeneration compare to capacitor-based regeneration in terms of cost, size, and mass? How well can the prototype presented in this report be scaled down?

Motor-to-motor regeneration requires that components can be dynamically connected or disconnected from each other. Therefore, an interconnection of switches is added to arrange the desired connections. The pulse width modulation (PWM) used to control the DC motors can also be implemented by the interconnect. What PWM frequency can be achieved with such an interconnect? What parameters need to be calibrated for the design?

The remainder of this report will first elaborate on the relevant theory in Chapter 2: it will place the work in its scientific context, provide some theory on interconnects and theoretically analyze the potential merits of motor-to-motor regeneration. Then the setup is designed in Chapter 3 and experiments are planned. The setup is then built and characterized in Chapter 4 and the experiments are performed and their results presented and interpreted. Finally, the questions posed here in the introduction will be answered in Chapter 5 and some recommendations for future work will be provided.

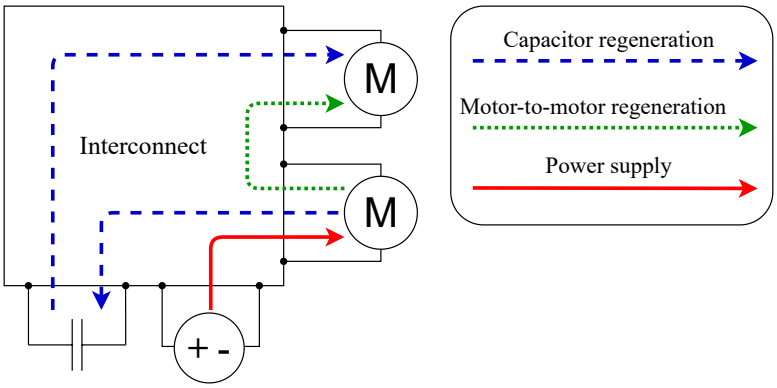


Figure 1.1: Energy flows when powering a motor from a power supply, via capacitor regeneration, or motor-to-motor regeneration.

2 Background

2.1 Related work

Numerous methods to improve the energy efficiency of industrial robotic systems have been proposed and studied in the literature. Carabin *et al.* [4] made an overview of the various methods, categorizing them as shown in Figure 2.1. The first distinction they make is whether the approach is hardware or software-based. In the software approaches, they distinguish between methods that optimize the trajectory of the robot and methods that optimize the scheduling operations. Amongst the hardware approaches, they distinguish methods that save energy by choosing a more appropriate robot for the task at hand, methods that replace parts within a single robot, such as using lighter parts, and methods that add hardware. This additional hardware can then be used to share the braking energy of one actuator to drive another actuator via a common network. This is typically implemented using a shared DC bus. Alternatively, hardware can be added to a robot to store energy release during braking of the actuators. These are further subdivided into mechanical kinetic-energy recovery systems (KERS), e.g. flywheels, electric KERS, e.g. supercapacitors, and hydraulic KERS, e.g. a hydro-pneumatic accumulator. However, if multiple actuators brake at the same time, the brake energy might exceed the installed energy-storage capacity, so a chopper resistor is needed to dissipate this excess energy as heat. It is also possible to combine methods from multiple of these categories.

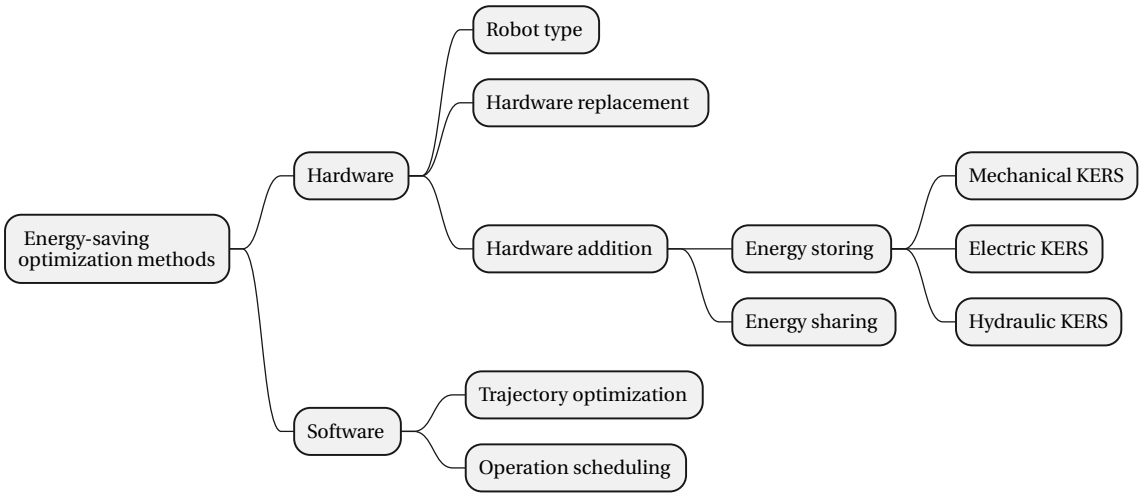
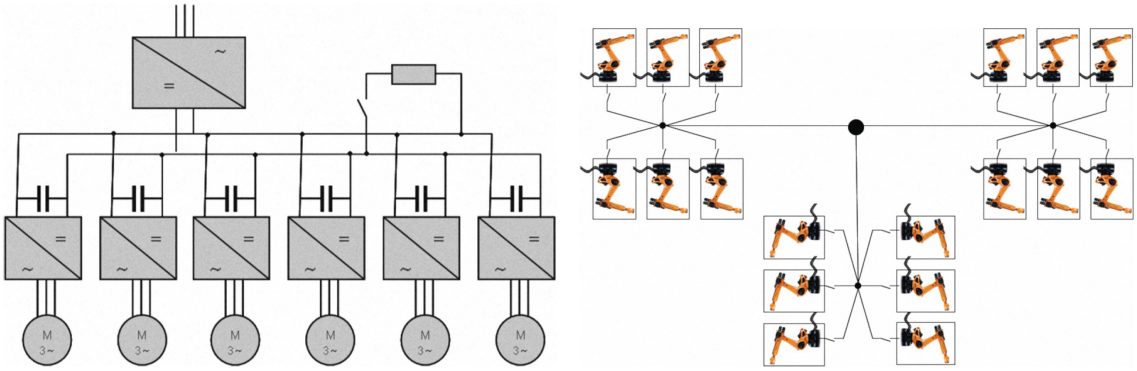


Figure 2.1: Optimization methods to save energy in robotics and automation systems as categorized by Carabin *et al.* [4].

The most common hardware approach is to add energy storage elements to the system such as (super)capacitors [3], [6] or flywheels [7], [8]. These enable regenerative braking, which has been used for many applications in industrial robotics such as robot arms, conveyors and elevators [9], and its use in electric vehicles has also been extensively studied [10], [11] and widely adopted. These energy storage solutions have also proven their use in different industries, such as renewable energy storage, where flywheels [12]–[15] and gravity batteries [16] have been developed.

This report, however, will focus on the seemingly less common approach: energy sharing. Meike *et al.* [5] evaluate a system where multiple motors are connected to a shared DC bus that is also equipped with capacitors to store the braking energy Figure 2.2a, and they also connect multiple robots to a shared DC bus, a solution they call *EnergyTeam* Figure 2.2b. Both solutions were found to be able to achieve energy savings ranging from 5 % to 20 % in realistic scenarios, but the investment to add capacitors might take a too long time to earn itself back, although



(a) Multiple motors on shared DC bus. [5]

(b) *EnergyTeam*: Multiple robots on shared DC bus. [5]

Figure 2.2: Configurations for energy sharing.

it should be noted that this conclusion was drawn before the advent of supercapacitors, and *EnergyTeam* requires significant changes to the existing hardware. Khalaf *et al.* [3] found similar energy savings ranging from 10 % to 20 % for a robot arm where some of the joints were powered by a shared supercapacitor. To maximize the energy savings, they also optimize the trajectory of the robot arm. Gritzner *et al.* [17] also propose a trajectory optimization method but for a shared DC bus between multiple robots.

2.2 Single motor control

The angular velocity and direction of a single DC motor can be controlled by using an H-bridge, which is a circuit consisting of four switches as shown in Figure 2.3. By enabling S_1 and S_4 at the same time, current can flow from the supply through the motor, causing it to rotate in one direction (Figure 2.3a), and similarly, enabling S_2 and S_3 will cause the motor to rotate in the opposite direction.

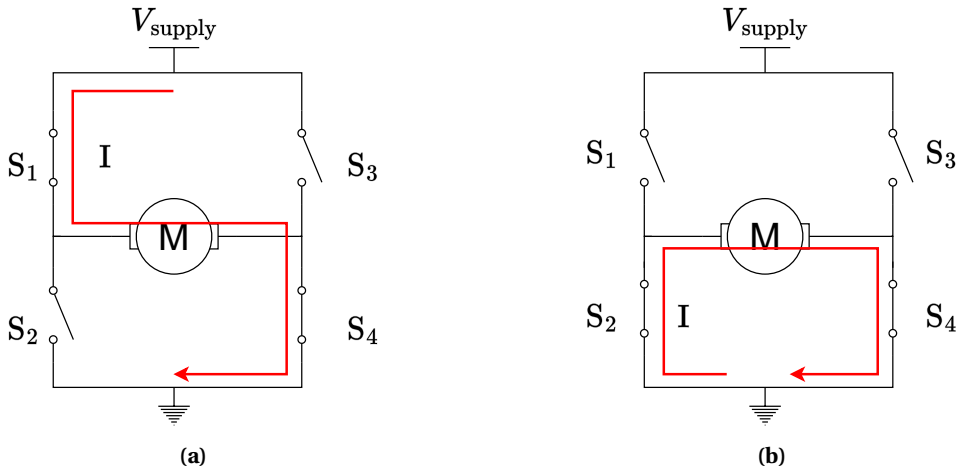


Figure 2.3: On-time (Figure 2.3a) and off-time (Figure 2.3b) states in a PWM cycle of a H-bridge driving a motor.

The angular velocity of the motor is proportional to its voltage, which can be controlled by alternating between connecting the motor to the voltage supply and connecting it to ground. Periodically doing this is called pulse width modulation (PWM). The PWM is characterized by its period T and its duty cycle, i.e. the percentage of the period that the motor is connected to ground. Alternatively, the PWM cycle can be described by the on-time τ_{on}, the time within a period that the motor is connected to the voltage supply, and the off-time $\tau_{off} = T - \tau_{on}$, as

illustrated in Figure 2.4. This method of controlling a DC motor is called sign-magnitude drive [18], [19].

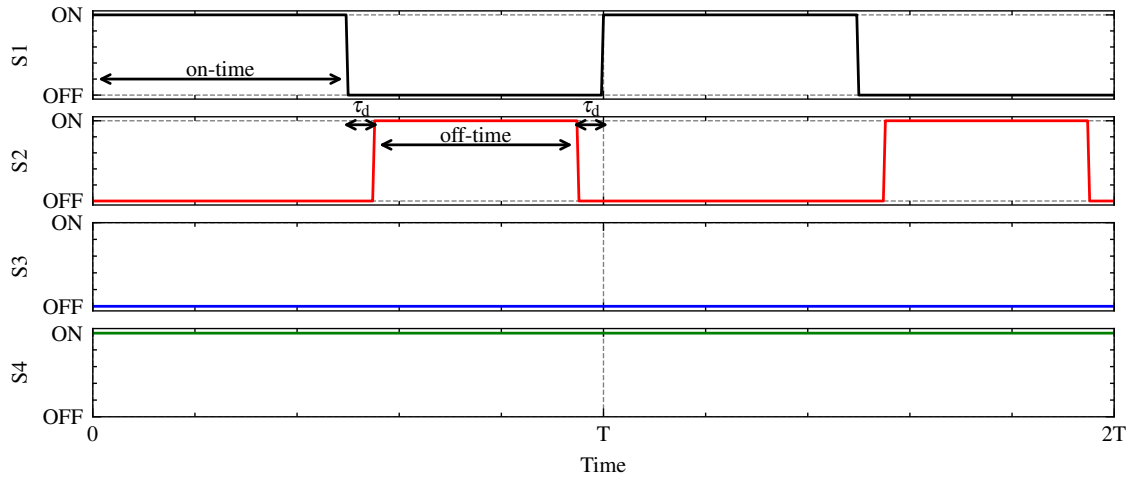


Figure 2.4: H-bridge switch states for PWM signal with a duty cycle of 50 %, period T, and dead time $\tau_d = 0.05$T.

Ideally, S_2 switches on at the same time as S_1 switching off, and vice versa. In reality, however, switches have a non-zero rise time and fall time, so they cannot instantly toggle on or off. Having S_1 and S_2 on at the same time would cause a short between the supply and ground, wasting energy and potentially damaging the circuit due to a large current flow. To prevent this, a dead time τ_d is added in between Figure 2.4.

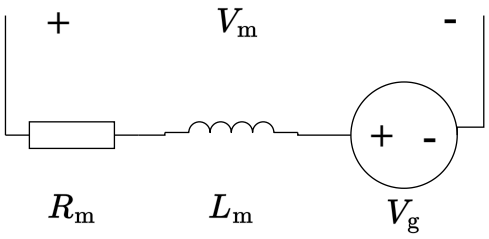


Figure 2.5: Model of a DC motor as a resistor, inductor and voltage source.

A DC motor behaves as a series circuit of a resistor R_m, inductor L_m and gyrator voltage V_g (Figure 2.5). During the dead time, the motor is connected to neither supply nor ground, leaving no path for the current to flow. However, the current flowing through L_m cannot change to 0 A instantaneously. Hence, the voltage will rise as high as it needs to keep this initial current flowing before gradually lowering the current as the energy stored in the inductor reduces. This can cause very high voltage peaks that could damage the circuit. Therefore, a diode typically is placed parallel to each switch to allow a path for the current through the motor during the dead time [18].

These diodes might not be required, however, if the inductor current can reach 0 A during the time that the switch is turning off without raising the voltage to dangerous levels. The circuit equivalent to a switch opening to disconnect the from the supply is shown in Figure 2.6, where the switch that is opening is modelled as a resistor whose resistance is rapidly increasing. This is an example of the well-known RL circuit [20]. To keep the voltage peak low, the inductor must be able to discharge when the equivalent resistance $R_{eq} = R_s + R_m$ is still low when the switch is just starting to open. The time the inductor needs to lose all of its energy is approximately $5\frac{L}{R_{eq}}$, so the diode can be left out if the following condition holds:

$$\text{fall time} \gg 5\frac{L}{R_\text{s} + R_\text{m}}. \tag{2.1}$$

However, because the switch's resistance rapidly increases, causing a larger voltage peak and faster discharge of the inductor, it is impractical to compute how fast the inductor will discharge, so this can best be measured instead.

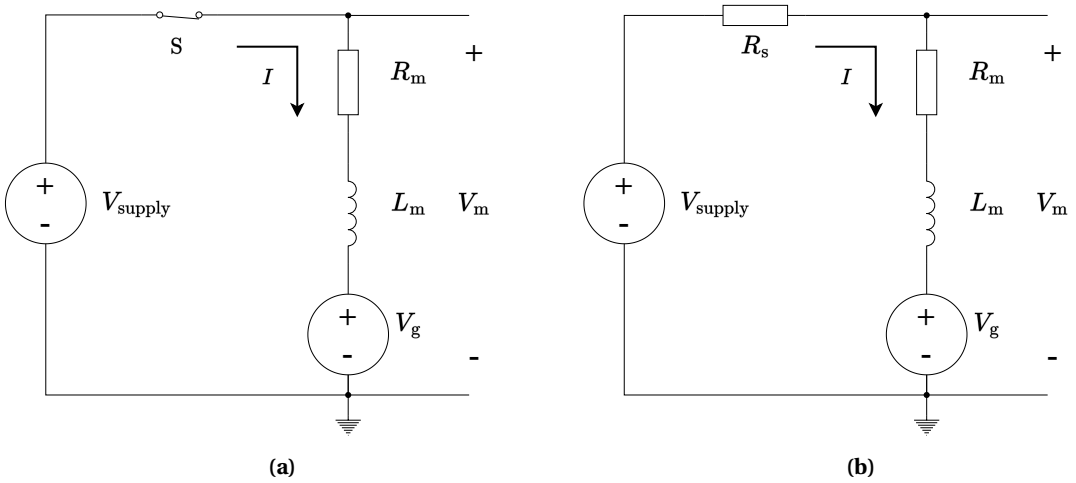


Figure 2.6: Current through a switch when it is closed and when it is closing.

If the inductor completely discharges during the fall time of the switch, a part of its energy is dissipated by R_eq, and a part is transferred to the mechanical side of the motor, slightly increasing its angular velocity and thus voltage V_g. To analyze whether this contributes to significant energy losses, the total energy added to the motor within one PWM period can be compared to the energy stored in the inductor.

Assuming that it takes many PWM cycles to accelerate the motor to full speed, the voltage V_g can be regarded as constant. After the inductor has charged up, which is assumed to happen within a fraction of the PWM period T, the current through the motor is determined by the internal resistance R_m, so the energy added to the motor in a single PWM period is

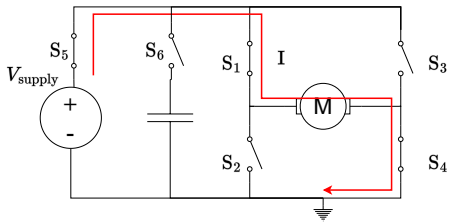
$$E_\text{PWM, period} = P \cdot \Delta t = \frac{\left(V_\text{supply} - V_\text{g}\right)^2}{R_\text{m}} T = I^2 \cdot R_\text{m} \cdot T. \tag{2.2}$$

The energy in the inductor in an inductor is

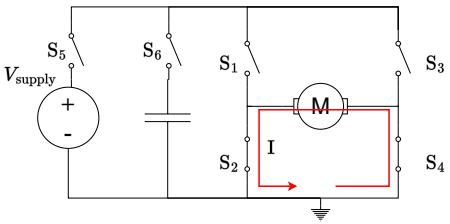
$$E_\text{L} = \frac{1}{2} L I_0^2, \tag{2.3}$$

where I_0 is the current through the inductor at the moment that the switch starts opening, which equals I in this case. Both Equation 2.2 and Equation 2.3 depend on I^2, so the ratio between them is constant.

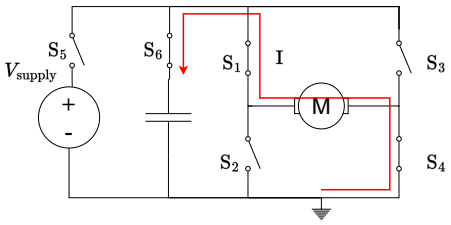
Apart from controlling the motor's direction and angular velocity, an H-bridge can also be used to recuperate energy from the motor. The main requirement for this is that the voltage at the motor is higher than that of the component storing the recuperated energy, such that current can flow from the motor to this component. If the H-bridge switches sufficiently quickly, the L_m can be used to boost the voltage sufficiently high (Figure 2.7b and Figure 2.7c) [6]. Otherwise, if slower switches are used, regeneration is still possible, but only if the receiving component has a lower voltage than the motor, for example, a capacitor or the input of a buck-boost converter (only Figure 2.7c).



(a) Powering the motor with the supply.



(b) Charging L_m by shorting V_g.



(c) Regenerative braking: storing energy in the capacitor.

Figure 2.7: Different H-bridge states for regenerative braking.

2.3 Energy recuperation between motors

Moving on from controlling a single motor, this report proposes a novel method for energy sharing: motor-to-motor regeneration. Instead of connecting all motors to the same shared DC bus all the time, each motor is dynamically connected to another motor, capacitor, or power supply as needed. To achieve this, the optimal energy source or drain for each motor is continuously re-evaluated, and the circuit is dynamically reconfigured by enabling and disabling switches.

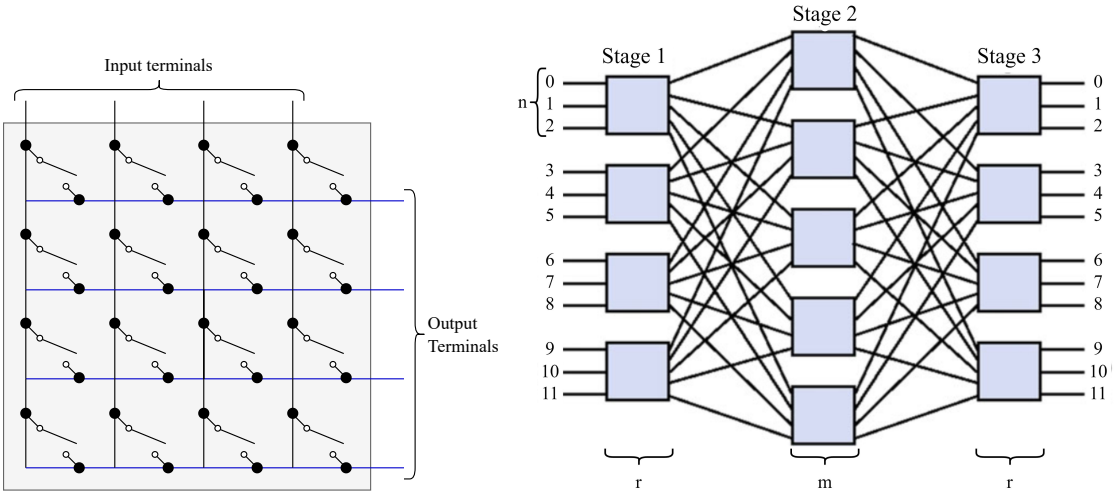
No literature demonstrating the use of such a reconfigurable switch grid in robotics has been found, but dynamic reconfiguration has been applied to battery management systems [21]. Numerous benefits were found concerning the reliability and lifetime of the batteries, but energy efficiency was not considered.

2.3.1 Interconnection theory

The connections between components in the circuit can be changed on-demand using switches. This is commonly referred to as a dynamic interconnection [22] and has been used in several fields over time. In computer networks and data centres, interconnections are often used to transfer data streams between devices, which requires low power and might see time-division multiplexing of data streams. However, the design in this report needs interconnections to connect electrical components, so time-division multiplexing is not possible. Field-programmable gate arrays (FPGAs) also use interconnect without time-division multiplexing [23], but these are also designed for low-power digital signals, whereas the interconnect that will be used in this report should be able to withstand much more current. This application is, therefore, more similar to the use of interconnections in telecommunications in the 1950s [24] or (radio frequency) test benches to connect test equipment to devices [25]. However, connecting electrical components here is still different from the literature, because the electrical components that need to be connected all have multiple terminals instead of one.

Interconnection networks can be categorized into static networks, where connections between the inputs and outputs are fixed links, and dynamic networks, where the connections can be established by switches or by using a shared bus [22]. Typical use cases for shared bus interconnects are computer systems or computer networks, where data is transferred [22]. However, a shared bus is not an option for the design in this report because it would electrically connect all components together. A static network could be used in combination with bi-directional

(de)multiplexers, but this would quickly lead to impractical amounts of wiring, leaving only switch-based interconnects.



(a) 4 × 4 crossbar switch.

(b) Non-blocking Clos network with (m,n,r)=(5,3,4). Adapted from [22].

Figure 2.8: Examples of crossbar and clos interconnect.

The simplest type of a switch-based interconnect is the crossbar (Figure 2.8a). Any input can be connected directly to any output by turning on the switch at the point where their lines cross. A crossbar interconnection with N inputs and M outputs requires $N \cdot M$ switches [24].

Clos networks can also connect any input to any output whilst using fewer switches [24]. A Clos network is made up of multiple stages consisting of switches. Every switch in a layer is connected to every switch in the previous stage and every switch in the next stage. For instance, a three-stage Clos network has a first stage of r_1 switches with n_1 inputs each, a middle stage of m switches and a third stage of r_2 switches with n_2 outputs each. The network is symmetric if $r = r_1 = r_2$ and $n = n_1 = n_2$, for example Figure 2.8b.

Contrary to crossbar networks, connections may not always be possible in a Clos network, which can be non-blocking, rearrangeable non-blocking, or blocking [22]. When a network is non-blocking, it is always possible to find a free route between any unconnected input and any unconnected output, whereas rearrangeable non-blocking networks might require rerouting some of the existing connections. On the other hand, blocking networks might make connecting an unused input to an unused output impossible due to an existing connection.

In the design in this research, multiple electrical components need to be connected in a flexible manner, but connections can be rearranged, so a rearrangable non-blocking or non-blocking network is required. This is the case for $m \geq n$ and $m \geq 2n-1$, respectively [22]. However, this design also requires connections from a single input to multiple outputs, i.e. multicast. Hwang [26] proved that multicasting from any group of up to q_1 input terminals to any group of up to q_2 output terminals is rearrangeable non-blocking if the following condition is met:

$$m \geq \max(\min(n_1 q_2, n_2 r_2), \min(n_2 q_1, n_1 r_1)). \tag{2.4}$$

Interconnect comparison

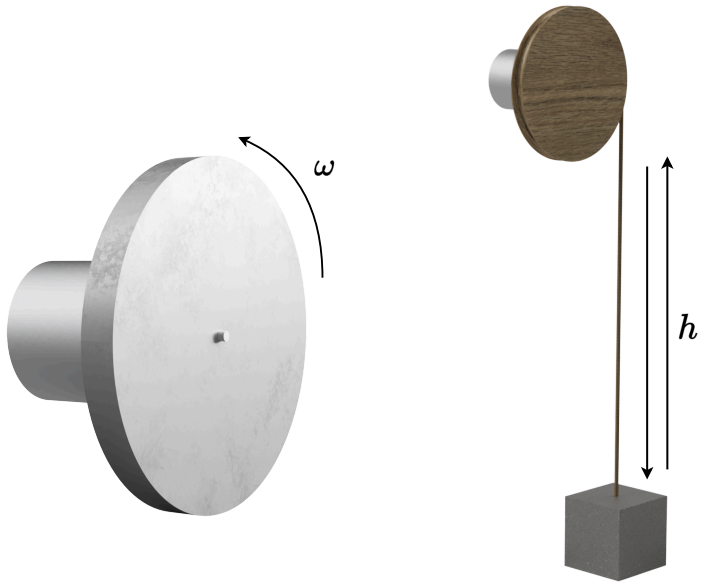
Clos networks rely on multiple-input and multiple-output (MIMO) switches, so whether they require fewer switches than crossbars depends on how the MIMO switches are implemented. They can be made from crossbars, or if they have more than four inputs or outputs, they can be recursively made from smaller Clos networks. This means that there are many possible com-

binations of m, n and r to construct an interconnection, so finding one that uses significantly fewer switches than an equivalent crossbar switch can take some effort. In general, Clos networks can only offer a worthwhile reduction in the number of switches when the number of inputs and outputs gets large, which is not the case for the design in this report. Furthermore, routing connections in a Clos interconnection is computationally more complex, and the algorithm is less straightforward than routing in a crossbar. The reduced number of switches becomes more significant when using interconnecting many components, but that is not the case in this research. Therefore, the design in this report will use a crossbar interconnect.

2.3.2 Energy efficiency

In order to measure the power draw of the proposed energy recuperation method, a setup is built. To keep the findings general, a flywheel and a mass can be used as these can store inertial and gravitational energy, respectively. One of them would also serve to evaluate the energy efficiency of the proposed method, but the gains could differ between the methods since the flywheel needs to decelerate from high speed to release much energy, whereas the mass can be lowered very slowly to release the energy. When estimating the efficiency improvement that this method could offer for a given application, one can then choose which component, either the flywheel or the mass, has the most likeness to the components application.

The energy recuperation between motors is only dependent on the mechanical energy stored by the motors, so the results might also hold in general independent of the type of motor used, but empirically testing this is outside the scope of this research.



(a) **(b)**

Figure 2.9: Example of flywheel attached to an electric motor (Figure 2.9a) and mass that can be lifted by an electric motor via a pulley (Figure 2.9b) with movements to store energy indicated.

To clarify that the mass and the flywheel are different in how they store energy and that the flywheel is similar to a capacitor, their energy equations are listed next. The energy stored by the mass is

$$E = mgh, \tag{2.5}$$

where m is the mass, g the gravitational constant and h the height. The energy for a flywheel is

$$E = \frac{1}{2} J\omega^2, \tag{2.6}$$

where J is the inertia of the flywheel and ω is its angular velocity. If the inertia of the flywheel can be approximated by a circle, where all mass is located at a constant distance from the center of rotation, this would result in

$$E = \frac{1}{2} mr^2\omega^2, \tag{2.7}$$

where m is the mass of the flywheel and r its radius.

Finally, the energy of a capacitor is given by

$$E = \frac{1}{2} CV^2, \tag{2.8}$$

where C is the capacitance and V the voltage.

Not all of the braking energy can be reused because the components dissipate some of the energy. Figure 2.10 shows the energy flows for regeneration using a capacitor or motor directly to store the energy. However, when connecting these components directly, energy recuperation is only possible when the voltage of the source component is higher than the voltage of the drain component. A DC-DC converter could be added in between to resolve this issue, but this would introduce additional losses, as shown in Figure 2.11.

The resulting expectations for the efficiency improvement of adding regeneration compared to always using the power supply are displayed in Table 2.1. Estimated values for the percentage of reusable energy are also estimated in this table, assuming that the energy efficiency of the motor and DC-DC converter is 80 % [27], and the capacitor's efficiency is 98 % [28]. When first powering up the system, all of the motor's energy will come from the power supply. Every accelerating/braking cycle from then on will reuse at most the percentage of energy listed in Table 2.1 and require the remainder of its energy from the power supply so that eventually the energy efficiency of the system converges to (1 − reusable energy). Table 2.1 suggests that using a DC-DC converter will perform significantly worse than directly connecting the components, but these equations do not take into account that regeneration is not always possible in the latter case.

(a) Capacitor regeneration. **(b)** Motor-to-motor regeneration.

Figure 2.10: Regeneration energy flow.

Table 2.1: Equations for regeneration efficiency and estimated values, where η_C, η_M and η_{DCDC} are the efficiencies of the capacitor, motor and DCDC converter, respectively.

Regeneration type	Regeneration efficiency	Reusable energy (%)
Capacitor	$\eta_C^2\eta_M^3$	49
Motor-to-motor	η_M^3	51
Capacitor with DC-DC	$\eta_C^2\eta_M^3\eta_{DCDC}^2$	31
Motor-to-motor with DC-DC	$\eta_M^3\eta_{DCDC}^2$	33

(a) Capacitor regeneration.

(b) Motor-to-motor regeneration.

Figure 2.11: Regeneration energy flow with DC-DC converter.

2.3.3 Cost

There are also other factors to consider next to energy efficiency. Adding capacitors or supercapacitors to the system to recuperate energy might yield similar improvements and has already been widely adopted as mentioned before. However, adding these components to the system takes up space, which might be constraint. Besides, it adds mass to the system, which increases the power consumption if the mass is added to a moving part of the system. Furthermore, the return on investment of adding sufficient capacitors for regenerative braking might be unattractive, as capacitors add significant costs and electricity is relatively cheap. Although it is hard to put an exact number on it, current prices for electrolytic capacitors seem to be around €10 per 0.1 F, or about €1 per 1 F for supercapacitors, but these have a limited voltage rating, so they would require the addition of more components, such as a DC-DC converter or more supercapacitors in series, which would increase the cost of supercapacitors.

Motor-to-motor regeneration does not appear to have any of these drawbacks. Because only existing parts of the system are used, no extra space or mass is added. Besides, multiple motors in the same system can typically store similar amounts of kinetic energy, so they are naturally dimensioned well to store the amount of braking energy released by their peers. Despite not adding components to the system, motor-to-motor regeneration might increase the cost because it uses many more switches than the conventional motor controller that it replaces. Depending on how well this can be optimized for mass manufacturing, this cost might be in the same order of magnitude as the conventional controller, or it might be more expensive.

However, motor-to-motor regeneration introduces a drawback of its own: it can only be effectively utilized if motors happen to be braking and accelerating at the same time, which will realistically not always be the case. Connecting a braking motor to a motor spinning at constant velocity would be much less effective because, at a constant speed, the motor requires much less energy than when it is accelerating, so the braking motor would decelerate significantly slower. Capacitor-based alternatives do not have this drawback because they can store energy over time.

3 Design

The reconfigurable grid that has been designed in this paper consists of several parts as shown in Figure 3.1. At its core, electrical components or circuits need to be connected to each other. The connections are made by the interconnect, which consists of a number of switches and is controlled by a microcontroller.

In this chapter, the interconnect is designed, followed by the required mechanical components, the flywheel and the weight. Then, electrical components are chosen, and a printed circuit board (PCB) is designed to join all the electrical components together. Finally, when the physical design is finished, firmware for the microcontroller is developed to control the whole system, and experiments are planned.

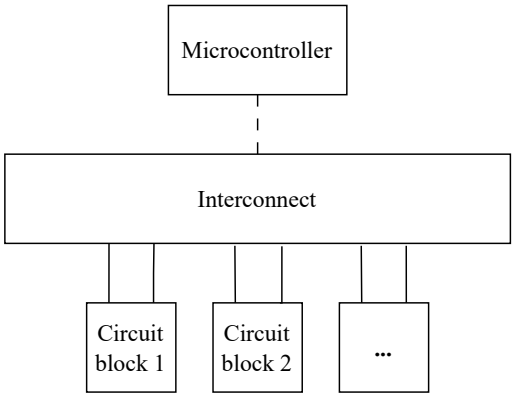


Figure 3.1: High-level system overview.

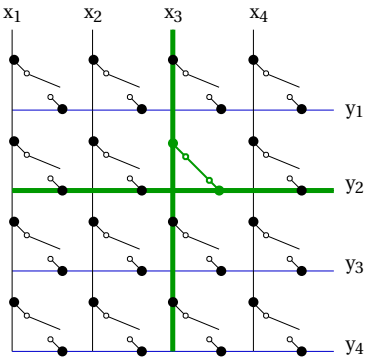
3.1 Interconnect design

Interconnection theory (Section 2.3.1) reasons about interconnects that use as few switches as possible whilst guaranteeing that the required connections between an input terminal and an output terminal can always be established. However, since the electrical connections are bidirectional, the terms input and output make little sense, so they will be referred to as x-axis and y-axis for the remainder of this report. As illustrated in Figure 3.2a, any terminal on the x-axis can always be directly connected to any terminal on the y-axis, but terminals on the same axis require a free terminal on the other axis for the connection to be feasible (Figure 3.2b). In Section 2.3.1, several interconnect topologies have been discussed, and an argument was made for using a crossbar interconnect.

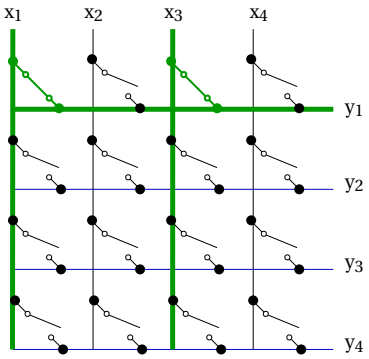
There are several options to tailor the crossbar interconnect for the task at hand. The footprint and cost of the interconnect are directly proportional to the number of switches, so it is beneficial to minimize the number of switches in the interconnect.

For maximum flexibility of the circuit, every terminal should be able to connect to every other terminal, so interconnection theory suggests that every electrical terminal should be connected to both the x- and y-axis of the crossbar. As each electrical component needs at least two terminals to form a closed circuit loop, N_c electrical components would require $(2N_c)^2$ switches.

Alternatively, a crossbar that corresponds more closely to an electrical circuit consisting of components and nodes can be used. The components can be connected to the nodes to form circuits. This can be directly mapped to a crossbar where the electrical components are on the x-axis and the nodes are on the y-axis (Figure 3.3a). An electrical circuit can have at most N_c nodes, occurring when every electrical terminal is connected to exactly one other terminal. This would result in $2N_c^2$ switches, a 50 % reduction.

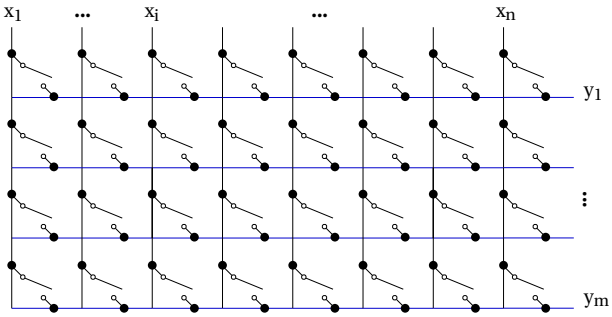


(a) Direct connection between components on opposite axes.

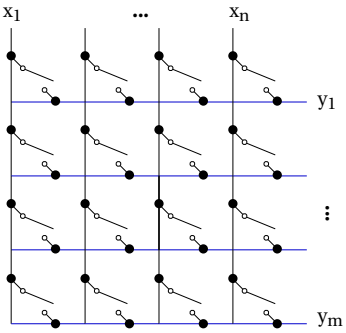


(b) Indirect connection between components on the x-axis, using a free line on the y-axis.

Figure 3.2: Example of direct and indirect connections between terminals in a crossbar. The terminals on the horizontal lines are referred to as the y-axis, and the terminals on the vertical lines as the x-axis.



(a) Crossbar with components on the x-axis and nodes on the y-axis, so n = $2N_c$ and m = N_c.



(b) Crossbar with components on both x-axis and y-axis, so n = m = $2N_c$.

Figure 3.3: Crossbars with fewer switches.

The number of switches used by the crossbar interconnection can be further reduced by connecting the electrical components to both the x- and y-axis of the interconnection, as shown in Figure 3.3b. However, connections between any two components are not guaranteed to be possible, as mentioned before and illustrated in Figure 3.2b. Whether this is an acceptable compromise depends on which connections might be prohibited, so this will be analyzed next.

To act as a proper motor controller, every motor must be controllable at any given time. This is ensured when a route can always be created between every motor and the voltage supply. To also allow for maximum energy recuperation, connections between motor and DC-DC converter, between motor and capacitor, or directly between motors also have to be possible.

The components can be placed strategically to minimize the number of desired connections that can be blocked. Motors are the only components that sometimes need to short themselves, so the terminals of one component are placed on the same axis. Besides, the motors are placed on one axis of the crossbar and the components that connect to the motors on the other axis. This way, connections between two motors or when a motor shorts itself could be blocked. Two terminals can be added to the opposite axis for each pair of motors to ensure that these connections are always possible.

Even after adding extra lines, the interconnect in Figure 3.3b still has fewer switches than the alternative in figure Figure 3.3a, so this crossbar is best in terms of minimizing the number of switches.

However, minimizing the number of switches is not the only factor to consider. The different interconnects may also result in different choke points for peak currents. The design should also have realistic peak currents at various places: peak supply current $I_{\text{supply, max}}$, peak current $I_{\text{l,max}}$ through a line, switch peak current $I_{\text{s, max}}$ and the peak current $I_{\text{controller,max}}$ through the controller to drive the switches. $I_{\text{supply, max}}$, $I_{\text{l,max}}$ and $I_{\text{s, max}}$ depend on the number of motors N_{m} and the maximum motor current $I_{\text{m,max}}$, and $I_{\text{controller,max}}$ depends on maximum number of switches N_{en} and the current required to turn a switch on $I_{\text{s,on}}$, assuming that the off-state current $I_{\text{s,off}}$ is negligible.

The maximum current drawn from the voltage supply occurs when all motors are simultaneously powered by the same source and draw the maximum current. The line connected to this supply must then be able to carry that current, and in both the folded and minimal crossbar, all this current flows through a single switch in the worst case because the current from the supply must make a U-turn to reach the motors, i.e. all connections are indirect as shown in Figure 3.2b. The minimal crossbar can get a lower $I_{\text{s, max}} = I_{\text{m,max}}$ if the power supply and the motors are connected on opposite axes of the crossbar because the U-turn can then be avoided.

To determine a general expression for $I_{\text{controller,max}}$, consider that any component might want to connect to another component. In the worst case, these are all indirect connections. Then the number of switches required is $2N_{\text{c}}$, where N_{c} is the number of electrical components, yielding the expression in Table 3.1. Note that when only using the crossbar for motor control, the maximum number of switches enabled at any time is lower, because during normal operation, an H-bridge has at most two switches enabled at any given time, yielding a maximum of $2N_{\text{m}}$ switches enabled simultaneously.

In the worst case, the maximum currents are the same for both the folded and minimal crossbar interconnects (Table 3.1), so this does not provide a reason to choose one interconnect over the other.

Table 3.1: Parameter sweeps to be performed.

Property	Maximum current
$I_{\text{supply, max}}$	$N_{\text{m}} I_{\text{m,max}}$
$I_{\text{l,max}}$	$N_{\text{m}} I_{\text{m,max}}$
$I_{\text{s, max}}$	$N_{\text{m}} I_{\text{m,max}}$
$I_{\text{controller,max}}$	$2N_{\text{c}} I_{\text{s,on}}$

3.1.1 Interconnect dimensioning

Finally, the dimensions of the crossbar need to be chosen. The minimal viable circuit to be able to test motor-to-motor energy recuperation consists of two motors and a single power supply. Besides, a DC-DC converter might add value to this circuit, and a capacitor is required to replicate the current state-of-the-art regeneration methods. In order to have some flexibility for the experiments to expand the circuit in the future, some additional terminals are added to the crossbar, at least enough to add a second DC-DC converter and two more motors. This brings the total to 10 terminals for both axes of the crossbar.

Besides, some terminals are added for voltage meter probes, allowing the circuit to measure the voltage at its various nodes. This yields an overall crossbar design of 10×12 terminals.

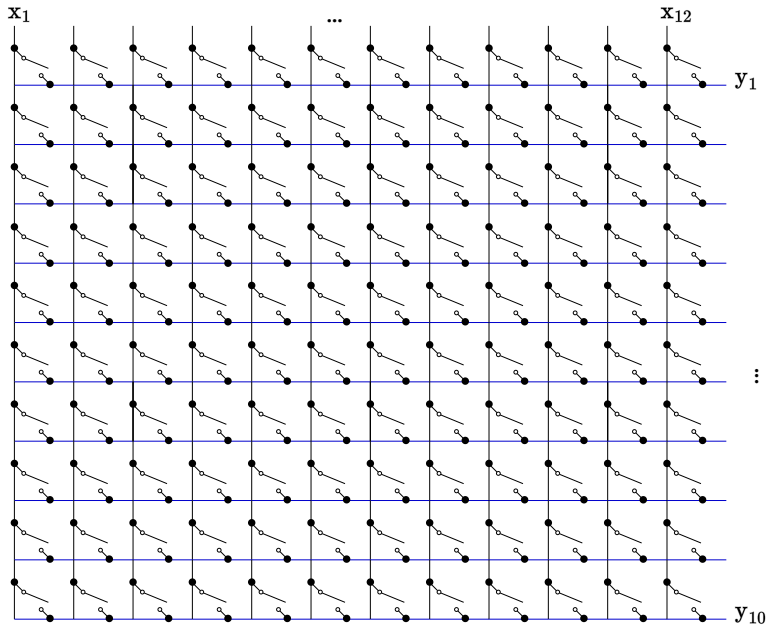


Figure 3.4: Final crossbar interconnect of 10 × 12 terminals.

3.2 Mechanical design

3.2.1 Energy dimensioning

Before designing or selecting any components, the requirements need to be established. The goal is to share energy between multiple motors, so the energy in the motors should be in the same order of magnitude.

The main consideration for dimensioning the system is that the expected efficiency improvement must be significantly larger than any energy losses due to non-ideal side effects, such as resistance or friction, in order to be able to measure any efficiency improvement. Besides, the setup should fit on a desk for practical reasons.

Since $E = mgh$ (Equation 2.5), lifting 1 kg for 1 m off the floor costs 9.81 J of energy. Lifting a 1 kg mass seems realistic, so the design aims for 10 J mechanical energy weight and flywheel. Parts are designed for this in the remainder of this section.

3.2.2 Flywheel design

To keep the flywheel tabletop sized, a radius of 5 cm is chosen. Assuming the flywheel has a top speed of 2×10^3 rotations per minute (rpm), and the flywheel can be approximated as a circle, Equation 2.7 yields a mass of 0.7×10^2 g to achieve an energy in the order of magnitude of 10 J. The flywheel in Figure 3.5 has an adjustable mass around this value. The mass can be adjusted by changing the number of M10 bolts and nuts mounted through the holes in the flywheel. The masses of the different parts of the flywheel are shown in Table 3.2. In the end, four bolts and nuts have been added, making the total flywheel mass 167.9 g and placing the combined centre of mass at a radius of 4.1 cm, thus yielding an inertia of 2.9×10^{-4} kg m^2.

The flywheel can be mounted to the axle of a motor using four M2 bolts. But the motor also needs to be kept in place. A mechanical breadboard with M6 holes spaced 2.5 cm [29] is available to mount the motor on, so the part in Figure 3.6a is designed. To lock the motor more firmly into place, the part in Figure 3.6b is added around to this.

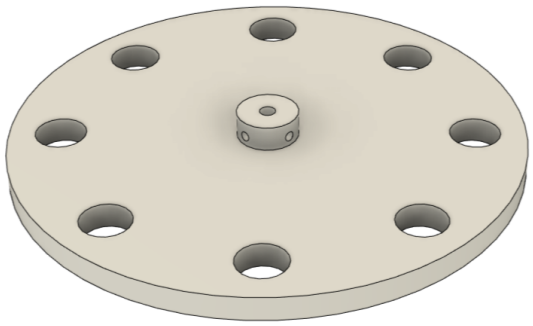
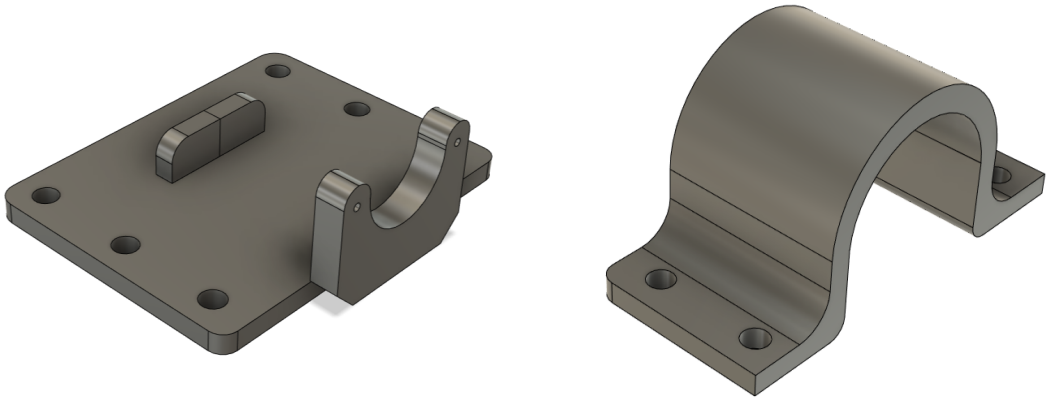


Figure 3.5: 3D model of flywheel.

Table 3.2: Flywheel mass contributions.

Component	Mass (g)
Nut	10.3
Bolt	26.3
Plastic part	21.5

The flywheel can spin at thousands of revolutions per minute, posing a safety hazard if a part detaches. Therefore, a 3D-printed dome is added around the flywheel and motor, as shown in Figure 3.7.



(a) Base to mount motor to the mechanical breadboard. **(b)** Top cover to fasten motor.

Figure 3.6: 3D-printed parts to keep the flywheel motor firmly in place.

3.2.3 Weight design

The weight-lifting setup needs a motor to lift the weight. Besides, an encoder is used to measure the angle of the motor, which can be converted to the height of the mass. To mount the motor, a similar part is used for the flywheel motor, as shown in Figure 3.8a. A part to mount the encoder on is added to this in Figure 3.8b. And again similar to the flywheel motor, the bracket in Figure 3.8c is added around the motor to lock it more firmly into place.

The motor and encoder also need to be mechanically connected. For this, the axle in Figure 3.9 is designed. The motor axle slides into this axle, which in turn slides into the encoder. Because

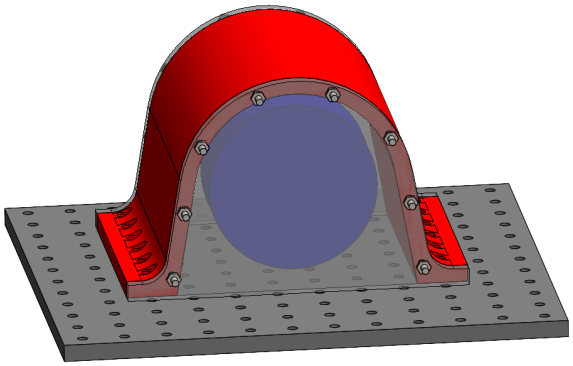


Figure 3.7: 3D model of the protective dome around the flywheel.

the tolerances are too tight for the available 3D printers, this axle is machined from stainless steel. The axle also features a pulley, around which a rope can be wound to lift the weight.

3.3 Electrical Component selection

The required electrical components are known from the interconnect and mechanical design. They are selected in this section.

3.3.1 Power supply

24 V is widely used in the robotics industry. Choosing this voltage has the advantage of a wide range of motors specifically designed for it. The XP Power VEH150 24 V supply is selected [30]. It can deliver up to 6 A, which is plenty.

3.3.2 Motors

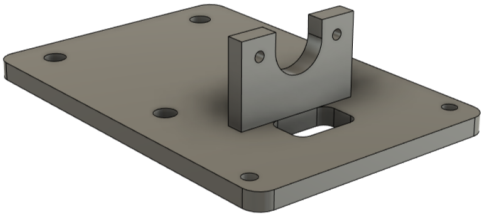
A flywheel's energy is quadratically related to its angular velocity, so a motor that can spin it fast is required to store a significant amount of energy. The motor should also be able to spin smoothly and not have too much internal resistance because that lowers the overall energy efficiency. With a no-load speed of 4110 RPM and internal resistance of 40 Ω, the Maxon F2140.937-22.112.050 [31] (Figure 3.10a) matches these requirements and is readily available, so it is used to drive the flywheel. Other relevant specifications are its listed speed constant of 173 RPM/V, terminal inductance L_m of 5.02 mH, and maximum efficiency of 81 % [31].

Datasheet [31] provides sufficient characteristics about the motor to be able to analyze the losses caused by the L_m. When the motor is at a standstill, $V_g = 0\,\mathrm{V}$ and $I = 0.6\,\mathrm{A}$. Substituting these values in Equation 2.2 and Equation 2.3 shows that the energy losses due to the L_m are $\frac{E_\mathrm{L}}{E_\mathrm{PWM,period}} \cdot 100\,\% = 0.1\,\%$, so these losses are negligible.

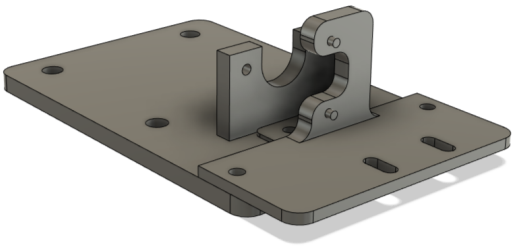
The weight-lifting application has quite different requirements for the motor, the main one being that it should be able to lift 1 kg, so according to Newton's second law, the motor needs to have a stall torque of at least 9.8 N cm. The torque requirements could also have been set lower by adding a gearbox between the motor and the mass, but this would add friction to the system and reduce its efficiency more than the lighter motor would compensate for. The Mellor RS555 [32], shown in Figure 3.10b, meets the specs, so this motor is selected to lift the mass. It has a stall torque of $10.0\,\mathrm{N\,cm^{-1}}$, a no load speed of 3500 RPM, and a no load current of 0.15 A [32].

3.3.3 Switches

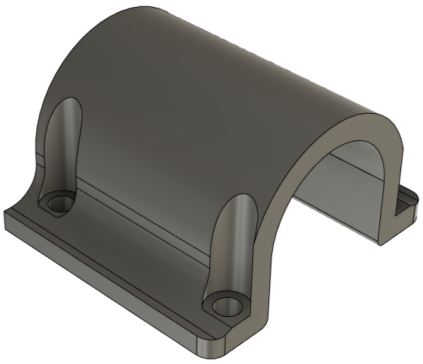
The Maxon F2140 motor lists a maximum starting current of 0.6 A. For motor-to-motor regeneration to be possible, the circuit must be able to support at least two motors at the same time.



(a) Base to mount motor to the mechanical breadboard.



(b) Base to mount motor to mechanical breadboard including a mount for the encoder.



(c) Top cover to fasten weight lifting motor.

Figure 3.8: 3D-printed parts to mount weight lifting motor and encoder.

Besides, some overhead should be kept to have some future flexibility for using the circuit, for example, adding a third motor. Therefore, the requirement for the switches is that they can withstand 2 A. Besides, the switches are used to generate a PWM signal. Since this design is a proof of concept, they do not need to be very fast, so at least 100 Hz switching is set as a requirement. If the switching would be much slower than this, the motors would significantly slow down during PWM off-time. On the other hand, faster switching generally generates a smoother signal on the motor at the cost of requiring a more capable microcontroller. The third requirement of the switches is that they offer a bidirectional signal path, i.e. currents in both directions can be switched.

A summary of switch types and the requirements that they meet is presented in Table 3.3, showing that solid-state relays are the only type of switch that meets all the requirements. Next, more elaborate reasoning for choosing a switch will be given, as well as the choice for a specific model.

Table 3.3: Types of switches and requirements they meet.

Device	$I_{max} \geq 2\,A$	$f_{max} \geq 100\,Hz$	Bidirectional signal
Crossbar IC	No	Yes	Yes
(De)multiplexer IC	No	Yes	Yes
MOSFET	Yes	Yes	No
Mechanical relay	Yes	No	Yes
Solid state relay	Yes	Yes	Yes

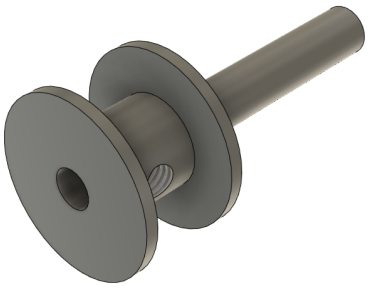


Figure 3.9: Axle between motor and encoder with built-in pulley.

(a) Maxon F2140 [31].

(b) Mellor RS555 [32].

Figure 3.10: DC motors selected for flywheel (3.10a) and weight (3.10b)

Multiple options were investigated for the switches. To keep the footprint of the circuit minimal, ideally, integrated circuits (ICs) that combine multiple switches are used. There are off-the-shelf crossbar ICs available. These are designed for digital signals, such as video, so they can switch very fast, but are not rated for the high currents required for the analogue circuit in this design. Multiplexer ICs have the same drawbacks.

Single switches have also been considered, starting with MOSFETs, but these need to be biased and thus cannot support potential drops in both directions, i.e. not acting as a true switch. This is a requirement for the circuit, so MOSFETs are not suitable. Relays, on the other hand, can be used bidirectionally. These can be categorized as mechanical relays and solid-state relays (Figure 3.11). Solid-state relays are more compact, have a longer lifetime, and make less switching noise than mechanical relays [33]. Typically, solid-state relays are also faster, but signal relays can have similar speeds for the same 2 A current rating. They are also in the same price range. Eventually, the TLP241A [34] solid-state relay is selected, which has a typical turn-on time of 2.8 ms, turn-off time of 0.3 ms, and a maximum continuous current rating of 2 A, meeting the requirements. A current between 5 mA and 19.5 mA is recommended to drive the LED, which typically has a forward voltage of 1.27 V [34].

A downside of these switches is that they are hard to scale down to reduce the footprint of the circuit. Compared to MOSFETs, these switches have a relatively high internal resistance of around 100 mΩ, resulting in more thermal dissipation, so they require a larger area for sufficient cooling. Besides, the internal structure consists of multiple transistors and an LED, constraining how small the switch can be made. Another downside of this switch is that a single one costs about €1, so the motor-to-motor regeneration added in this prototype has no financial benefit compared to adding capacitors that can store similar amounts of energy as a flywheel or mass.

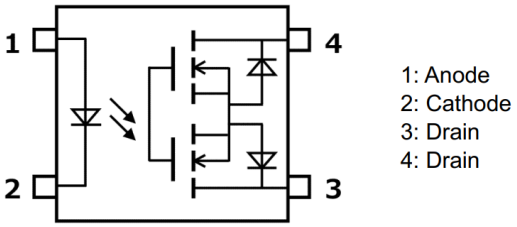


Figure 3.11: Circuit inside a solid-state relay [34]. Applying a positive voltage between terminals 1 and 2 creates a low-resistance path for current between terminals 3 and 4, thus turning the switch on.

As mentioned in Section 2.2 a diode could be needed in parallel to prevent large voltage peaks caused by the motor's inductance. However, the turn-off time of this switch is slow enough to satisfy the condition in Equation 2.1, so no diodes are needed.

3.3.4 Microcontroller

A microcontroller is needed to control the switches. The microcontroller should be able to control all 120 switches in the interconnect. So, it needs at least as many digital output pins, or it should be possible to add more pins to it. Besides, it should be able to supply enough current to drive the switches in all conceivable scenarios. From Table 3.1 gives the maximum current needed as $2N_c I_{s,on}$. Given that the crossbar consists of 10×12 switches, i.e. $N_c = 11$ and assuming that a switch draws about 10 mA in on-state, the microcontroller must be able to provide at least 120 mA to the switches. It also needs to power some other components, so the minimum requirement is set at 150 mA.

Further considerations for selecting the microcontroller are ease of use to keep the development time minimal. This can be influenced by the compatible programming languages and the availability of high-quality documentation and libraries. Finally, there are practical limitations that should be considered, such as the cost and whether they are in stock.

Microcontroller vs FGPA

An FPGA could be an alternative to a microcontroller in the system. They can generate very fast PWM signals and can have more GPIO pins than a microcontroller. However, no models with the required number of pins were found. Besides, FPGAs come at significantly higher cost and are significantly more time-consuming to program than microcontrollers. Therefore, this design uses a microcontroller.

Microcontroller comparison

The PWM signal needed to drive the switches is <1 kHz, so having a higher clock speed is not critical. Hence, simple and user-friendly hobbyist boards such as Arduino, ESP32, or Raspberry Pi Pico can be used. For Arduino the Due [35] has been considered as it is much faster and has more general-purpose input/output (GPIO) pins than the Uno [36]; for ESP32 the WROOM devkit [37] has been considered, and for Raspberry Pi the regular Pico [38] has been considered since wireless communications are not needed.

None of the microcontrollers have the required number of digital IO pins, so external expansion is used. Therefore, the number of communication buses has been considered instead of the number of pins. Besides, implementing the application onto two separate cores could improve the jitter and increase the overall speed, so this possibility is considered beneficial. However, using this feature can be quite complex, so ease of use is also considered here.

The resulting comparison can be found in Table 3.4. The values from the comparison are mapped to the numbers -2 for $--$ to 2 for $++$. The Raspberry Pi Pico gets the highest score,

Table 3.4: Microcontroller comparison. Except for the documentation quality, all scores are based on numbers, but they are denoted as $\{--, -, +, ++\}$ to express that they have different subjective importance to the final decision.

	Arduino Due	ESP32 WROOM devkit	Raspberry Pi Pico
Clock speed	+	++	++
ADC resolution (bits)	++ (12)	++ (12)	++ (12)
ADC pins	++ (12)	++ (15)	+ (3)
I2C busses	- (1)	+ (2)	+ (2)
SPI busses	- (1)	+ (2)	+ (2)
Price	-	+	+
Dual-core	-	+	++
Documentation	++	+	++
Total score	3	11	12

so it is best suited for this system. It has a higher resolution ADC and more communication buses than the Arduino, and better documentation and ease of use than the ESP32. The one point where the Raspberry Pi Pico falls behind is the number of ADC pins, but because they are connected to switches in the interconnect, three ADCs are sufficient.

The Raspberry Pi Pico can be programmed using either C++ or MicroPython. MicroPython has better ease of use, but it is slower than C++. The system has hard real-time constraints, so the firmware is written in C++ because it runs faster. To improve the ease of use of C++, the Arduino framework and the PlatformIO IDE [39] are used.

3.3.5 GPIO expander

Singh [40] describes four possible approaches to get more GPIOs: matrix scanning, shift registers, dedicated GPIO expander ICs, and more microcontrollers. Matrix scanning and shift registers both use time multiplexing to connect the available GPIOs on the microcontroller. This means that the switches would be updated only every once in a while. The same holds for adding a dedicated GPIO expander IC, but this IC makes it possible to update the GPIOs in any order or skip a pin from receiving an update if its value stays the same. For matrix scanning and shift registers, unchanged pins cannot be skipped and the order of addressing them is (more) fixed, thus reducing the maximum achievable PWM frequency.

Adding more GPIO can achieve faster GPIOs speeds because this reduces the issue of limited communication speed between the microcontroller and external GPIOs. However, adding microcontrollers is much more expensive than the alternatives and it makes the system significantly more complex as well. Therefore, the system uses GPIO expander ICs. The MCP23017 [41] is selected because it is the most widely used and well-documented IC. There are multiple libraries available online that implement a simple interface to this IC.

One MCP23017 adds 16 GPIO pins to the microcontroller, whilst only using two for I2C communications. In total 10 of these ICs have been added, one for each row for the crossbar interconnect.

3.3.6 Capacitor

As mentioned before in Chapter 2, a capacitor has been added to store the braking energy from the motors. Ideally, the capacitor can store all energy, so 10 J per motor, and it should be almost empty when it starts to get charged and almost full, i.e. at the supply, when the motor has halted. A lower voltage swing could be chosen, but that would require more current to transfer all

energy in the same time frame, which would quickly lead to currents that the switches cannot handle.

The required size of the capacitor can be calculated by rewriting its energy equation into

$$C = \frac{2E}{\Delta V^2}, \tag{3.1}$$

where E is the energy stored in the capacitor in J and ΔV is the voltage difference over the capacitor before and after (dis)charging it. Substituting $E = 10$ J and $\Delta V = 24$ V yields a capacitance of 35 mF per motor, so when designing for two motors, a capacitor of 70 mF is a nice estimate for a capacitance value.

It proved challenging to find a suitable capacitor because there are not many regular capacitors with such high capacitance, and the available ones are not rated for 24 V or higher voltages. One could also consider putting many supercapacitors, i.e. capacitors with much higher capacitance and lower voltage ratings, in series, but this is not practical. Eventually, a 22 mF capacitor is selected because this increases voltage swing compared to the calculated 70 mF.

3.3.7 Buck-boost converter

The final component that needs to be selected is a buck-boost converter, a type of DC-DC converter that can both step-up and step-down voltage. This should ideally have a range from 0 V to 24 V, but in reality, it has a minimum voltage, so it should be selected such that the minimum input voltage is as low as possible. One could choose a buck-boost IC and add some passive components around it, but to reduce the complexity, this system has used an off-the-shelf buck-boost converter PCB [42].

3.4 PCB design

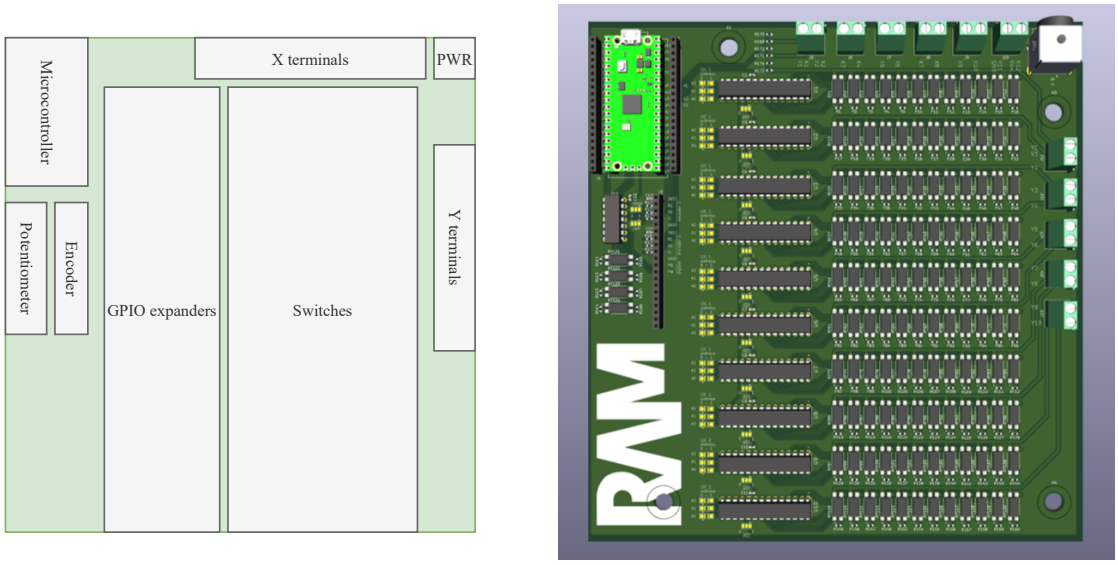
The electrical circuit requires 120 switches plus several other components, so a PCB is designed for this. The advantages of using a PCB are that the circuit footprint can be much smaller, much less tedious to solder, and that the quality is more constant and repeatable.

The PCB displayed in Figure 3.12 is designed using KiCad EDA [43]. Its dimensions are 19 cm by 20 cm. External components, such as motors can be connected to the screw terminals on the right and top edges of the PCB. The horizontal lines are labeled Y1 to Y10 and the vertical lines X1 to X12. Besides, a dedicated connector for the 24 V power supply has been added in parallel with the screw terminals on X11 and X12, and voltage dividers that are connected to the ADCs of the microcontroller have been added on X1, X2 and Y1.

The switches are laid out in a grid pattern, and a pin expander IC is added to each row. These ICs are controlled by the microcontroller via I2C. They only have eight unique addresses, so two I2C buses are used for them. To prevent wrongly wiring the addresses in the PCB design, the address pins are connected to solder jumpers, such that the addresses can be changed after production. Other configuration pins, such as reset pins, are also connected to jumpers in order to prevent mistakes with wiring them during the PCB design.

Pull-up or pull-down resistors are required by the encoder and the I2C bus. For the I2C bus, the value of these is a trade-off between speed and power budget. The maximum and minimum values have been calculated, and a pull-up value of 1.8 kΩ is chosen [44].

The Raspberry Pi Pico can be mounted as a daughter board onto the PCB and additional female headers are added to keep its GPIOs easily accessible, adding possibilities for design changes without replacing the PCB and for debugging. Furthermore, a header is added for two quadrature encoders, and a digital potentiometer is added to control the buck-boost converter. This circuit is a bit more elaborate than the other parts of the PCB, so it will be elaborated in its own subsection.



(a) PCB layout overview.

(b) 3D render of PCB.

Figure 3.12: PCB design top-down view.

3.4.1 Digital potentiometer

For the digital potentiometer, the MCP4261 is selected [45]. This digital potentiometer has a range from 0 kΩ to 5 kΩ and can be controlled via SPI. The maximum potential difference between its pins is limited to the voltage it is powered with, in this case, the 3.3 V supplied by the microcontroller. However, the voltage ranges from 0 V to 30 V to control the buck-boost converter. Other digital potentiometer ICs have the same issue, so additional discrete resistors are added in series with the MCP4261 to reduce the voltage drop over the MCP4261 and to increase the resistance range because the buck-boost converter also needs a larger resistive range.

Figure 3.13 shows the schematic for this potentiometer design, such that the resistance between terminals A and B can be varied from 0 kΩ to 32 kΩ. To sufficiently limit the voltage drop over the MCP4261, it shall not be used for more than 2 kΩ resistance. Lastly, the same switches as for the crossbar, i.e. TLP241A, can be used here.

Voltmeters

The ADCs on the microcontroller are used to measure the voltage of the motors and other components. However, the Raspberry Pi Pico can only measure up to 3 V. This is lower than the 24 V of the power supply, so a voltage divider with resistors of 10 kΩ and 150 kΩ is placed in between, changing the measurable range to 0 V to 48 V providing some headroom if voltage spikes occur. The resolution of the ADCs is 12 bits, so steps of $\frac{48}{2^{12}} = 12\,\text{mV}$ can be measured with this voltmeter. This corresponds to steps of $0.012 \cdot 173 = 2\,\text{RPM}$ when measuring the speed of the flywheel, although it is not expected that such accuracy can be achieved due to noise in the measurements.

3.4.2 Trace width

During the PCB design, the trace width is also considered to ensure that the traces on the interconnect can safely carry 2 A without causing the temperature to rise more than 10 K to prevent overheating the PCB. The built-in calculator tools in KiCad show that the minimum trace width for outer layers of the PCB is 0.8 mm, so to be safe 0.9 mm trace width is chosen. The traces that are only connected to the microcontroller do not need to handle such high currents, so they are kept at the default 0.15 mm.

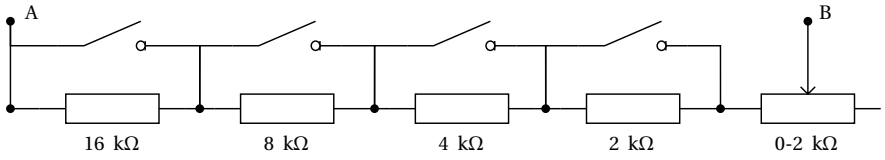


Figure 3.13: Digital potentiometer schematic with variable resistance from 0 kΩ to 32 kΩ between terminals A and B.

3.5 Firmware design

3.5.1 Controller design

The microcontroller is responsible for enabling/disabling the right switches at the right time. By doing so, it can electrically connect different components to form the desired circuit. Figure 3.14b shows how this can be implemented to form an energy-aware motor controller. The controller takes a percentage of the supply voltage as input for the motor as input, just like a regular motor controller (Figure 3.14a).

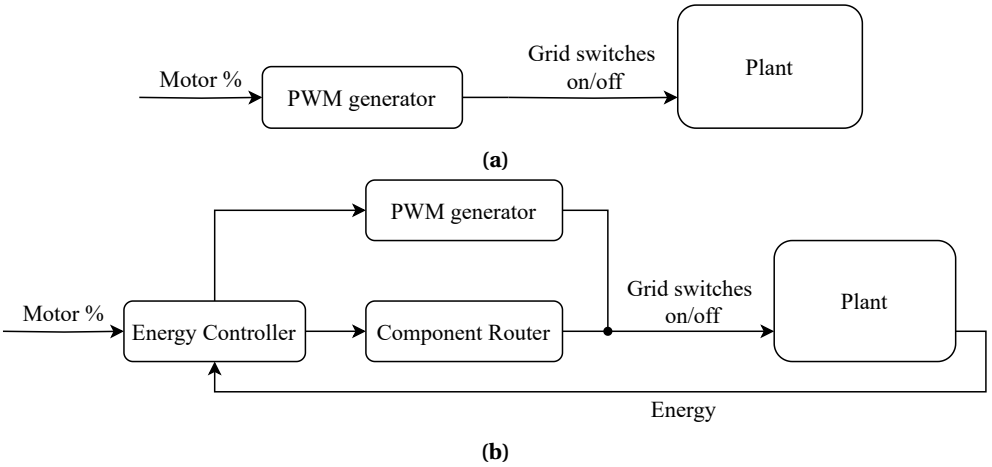


Figure 3.14: High level control loop for regular motor controller (Figure 3.14a) and motor controller for this research (Figure 3.14b).

The Energy Controller code looks at the desired duty cycle and decides on which components to connect to each other. The desired connections are then passed to the Component Router, which computes which switches to turn on to realize the desired connections. Besides, the motors should receive a PWM signal, so this is superimposed by the PWM generator on the switches that are turned on by the Component Router.

Energy Controller

The Energy Controller evaluates the energy needs of each motor multiple times per second. This information is then used to decide which component to connect to the motor based on the following priority rules:

1. Use another motor's energy
2. Use a capacitor energy
3. Use supply energy

Since electrical connections are bilateral, these rules apply both for accelerating and decelerating motors.

The priority rules are made under the assumption that it is possible and acceptable to change a motor's power source at any moment. For example, within a single acceleration, a motor might

first be connected to another motor, then to a capacitor and finally to the voltage supply. This might cause non-linearity in the motor's motion profile.

Intuitively, the priority rules seem optimal because they ensure that any braking energy is always used as soon as it becomes available. However, analyzing or proving optimality is outside the scope of this report, which aims to design a working concept.

Circuit Router

The Component Router needs to find a set of switches in the interconnect to enable, in order to realize the component connections as dictated by the Energy Controller. The interconnect is a crossbar with a horizontal screw terminal axis X and a vertical screw terminal axis Y. Assuming that all terminals of any given component are always connected to the same axis, the following rules can be used to find a feasible route:

1. If the terminals are on opposite axes, connect them directly.
2. If the terminals are on the same axis, each terminal requires one line on the opposite axis.
   (a) Use free lines without any component connected.
   (b) If there are not enough free lines, prioritize function over efficiency.
   In the case of motors: fall back by connecting them directly to supply.

To make the Component Router more application agnostic for rule 2b, the fallback connections can be decided by the Energy Controller, because it has full knowledge of the specific circuit.

Another potential drawback of the routing rules is that all connections are always recalculated, so even if two components stay connected to each other, they might be connected through different switches over multiple route computations. In general, this means that connections between the components would temporarily be broken, which might cause undesired side effects, for example, because currents through inductive components cannot instantly stop.

3.5.2 Firmware architecture

The firmware architecture implementing the control loop is designed to be modular and flexible, such that the code can be easily extended or modified for different use cases. For example, the firmware should continue to work when connecting any supported component to any row or column of the switch grid, or even when using another switch grid with a different number of rows and columns. Besides, a given implementation of a class can be overridden to modify its behaviour, for instance allowing to test different energy controller algorithms or to add additional component types to the grid. The main drawback of this degree of modularity is that it adds significant complexity and thus time to the development of the firmware.

Hardware definitions

Another advantage of the proposed firmware architecture is that the code contains a minimal amount of hardcoded information. This is all added separately to the code, making it easy to change the hardware configuration without deep knowledge of the code base.

All constants related to physical components, such as motor constant, are hardcoded as global variables. The switch grid and components themselves are defined in `grid_configs.h`. The I2C address for each switch is always the same for a given PCB, so these are all defined in a class, for example, `class PcbV1`. Child classes are then inherited from this class to define which component is connected to which screw terminal on the PCB.

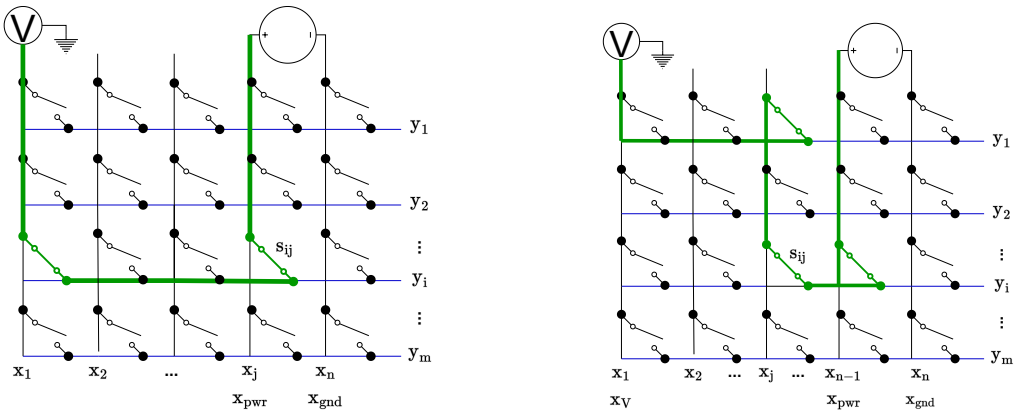
Self-test

The child class defining the specific hardware configuration may also contain a self-test. Switches can be automatically tested by the firmware if there is a path possible between the

supply voltage and a voltmeter: if the voltmeter reads the supply voltage the switch is working properly, and otherwise it is broken.

For the PCB designed in Section 3.4, self-testing is implemented for any switch s_{ij} which is located at the intersection between the lines y_i and x_j, except when $x_j = x_{gnd}$, where x_{gnd} is the line connected to supply ground.

The paths created between the voltage supply and the voltmeter are illustrated in Figure 3.15. If $x_j = x_{pwr}$, where x_{pwr} is the line connected to supply voltage, the following switches are enabled to create a path to the voltmeter connected to line x_V: $\{s_{i,V}, s_{i,pwr}\}$. In the more general case, the following switches are enabled: $\{s_{ij}, s_{1,V}, s_{1,j}, s_{i,pwr}\}$.



(a) Self-test for switch s_{ij}, where $x_j = x_{pwr}$.

(b) Regular self-test for switch s_{ij}.

Figure 3.15: The switches at the green intersections are enabled to perform a self-test on switch s_{ij}.

Implementation

The control loop in Figure 3.14 is further detailed in Figure 3.16. This figure shows the different classes that make up the firmware, as well as how the different subroutines are triggered, and how the firmware can be split up over different cores of the Raspberry Pi Pico.

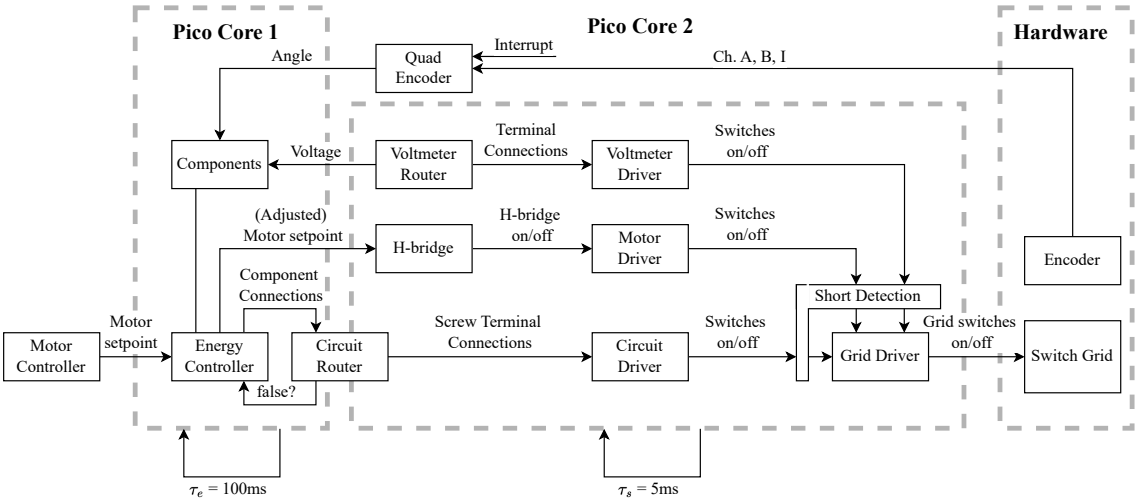


Figure 3.16: Firmware architecture, indicating all main classes as blocks with the information flow between them on the arrows, as well as the timing and hardware allocation.

Figure 3.16 has several changes compared to the high-level control loop in Figure 3.14.

Firstly, Components are added. This is a collection of classes that acts as a central place to store all relevant information about the electrical components connected to the screw terminals of

the Switch Grid. Each supported electrical component is a child of `class Component`, storing at least to which screw terminals the component is connected. The different components add their specific needs to this, such as an H-bridge for flywheel motors and an encoder for mass lifting motors. The supported component classes are shown in Figure 3.17.

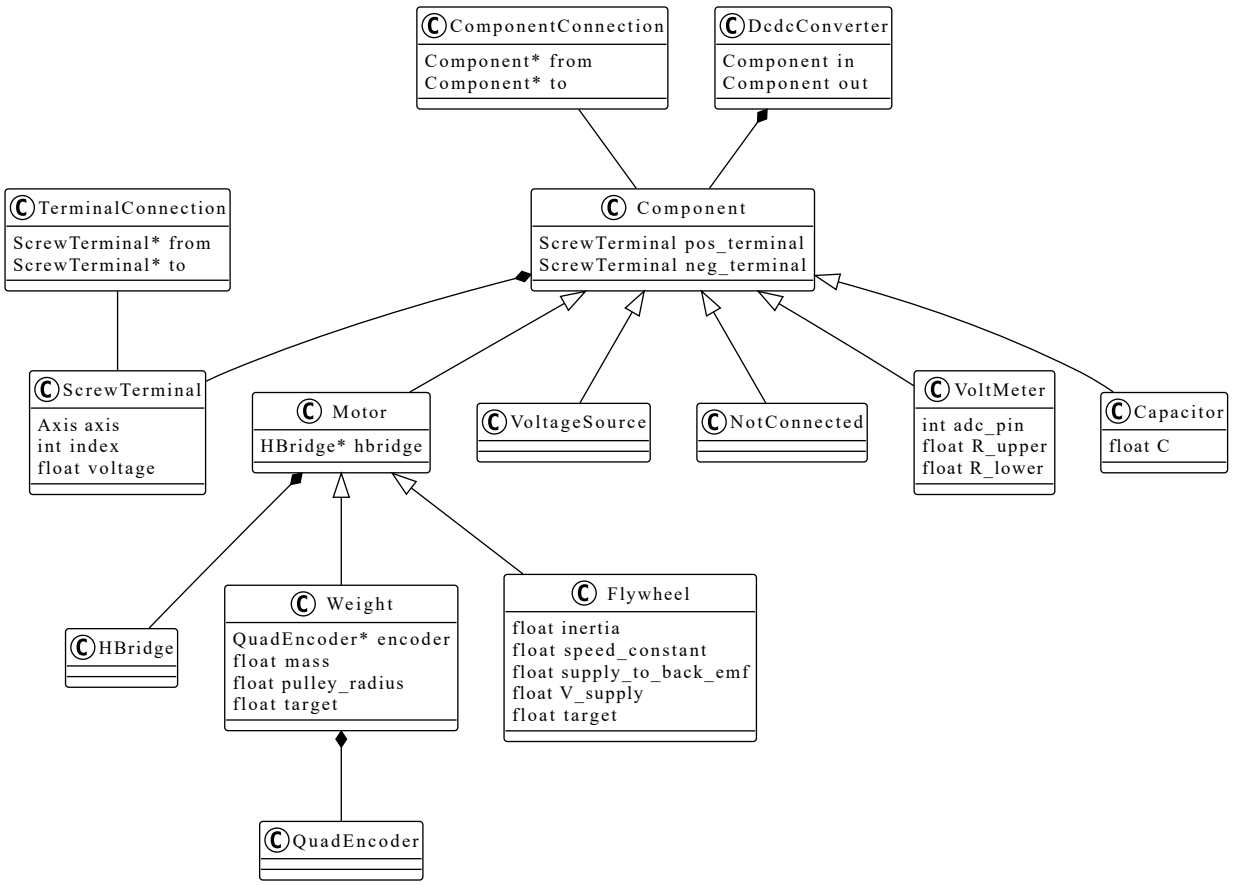


Figure 3.17: Simplified UML class diagram of the supported components. These classes are used to store the electromechanical properties of the system.

Secondly, the Component Router from Figure 3.14 has been subdivided into several router and driver classes in Figure 3.16. The circuit router is used to make a route for an electrical connection between components. This does not suffice for motors because they need a PWM signal, so a H-bridge class is added. Voltmeters also need some additional logic to decide when to measure which component, so they also get a dedicated router class. The router classes only indicate which screw terminals (horizontal/vertical lines in the grid) they want to connect to each other. The driver classes then write this information to a matrix where each switch is represented as a boolean, and finally, the Grid Driver class looks up the physical address of the pin corresponding to each switch and writes the desired value to it.

Furthermore, Figure 3.16 adds classes to measure the energy in each component. The energy of capacitors can be computed from their voltage (Equation 2.8). The energy of the flywheels is correlated with its angular velocity ω (Equation 2.7), which can also be computed from its open circuit voltage $V_{\mathrm{m,open}}$ via

$$\omega = k_{\mathrm{v}} V_{\mathrm{m,open}}, \tag{3.2}$$

where k_{v} is the motor's velocity constant. Therefore, voltmeters have been added to the system.

The gravitational energy stored in a mass lifted in the air is related to the height it has been lifted (Equation 2.5). To measure this, a quadrature encoder is added. The flywheels are con-

trolled using PWM, so their voltage is only representative of their angular velocity when it is not connected to the voltage supply or shorted. Therefore, the flywheel's voltage must be measured during the dead time τ_d, which is a period in the PWM cycle when the motor is connected to neither supply nor ground to prevent a short circuit between these two. To keep the scheduling implementation straightforward, the voltmeters must be able to measure every motor within a single dead time, constraining minimum to

$$\tau_\mathrm{d} \geq \frac{N_\mathrm{m}}{N_\mathrm{V}} \tau_{\text{pulse, min}}, \tag{3.3}$$

where N_m is the number of motors to be measured, N_V is the number of voltmeters, and $\tau_{\text{pulse, min}}$ is the minimum pulse duration that the switch can generate.

Lastly, the Short Detection class acts as a software safety layer that checks that no illegal shorts are created. For some component types, such as motors, a short between their terminals is sometimes desired, but for others, such as a voltage source or capacitor, a short would lead to unrestricted current flow, which could physically damage the switch interconnect. Starting from the positive terminal of a component, the Short Detection class recursively checks the screw terminal to which an enabled switch makes a connection. If this connection turns out to be connected to the negative terminal of the same component, a short is detected, and this switch configuration will not be sent to the physical switches on the PCB.

Motor control

The controller gets the desired angular velocity for the flywheels and the desired height as input. It then needs to convert these values into a PWM duty cycle to reach the target velocity or height. The duty cycle of the flywheel D_flywheel is controlled using a simple feed-forward controller based on

$$D_\text{flywheel} = \frac{\omega_\text{target}}{k_\mathrm{v} k_\mathrm{g} V_\text{supply}}, \tag{3.4}$$

where ω_target is the target angular velocity, k_v is the motor's speed constant in RPM/V, and k_g is the conversion factor from the supply voltage V_supply to the motor's generator voltage V_g. This assumes that the constants are correct, but the measurement of the angular velocity depends on the same assumption (as will be shown later in Equation 3.2), so applying a feedback controller based on this measurement would not reduce the controller's error.

The motor lifting the weight gets a simple proportional controller with a dead band. Given the error e = target height − measured height, its duty cycle D_weight is computed using

$$D_\text{weight} = \begin{cases} D_\text{up,min} & \text{if } |e| \leq d\,, \\ \min\left(2e + D_\text{up,min}, 1.0\right) & \text{otherwise}\,, \end{cases} \tag{3.5}$$

where d is the deadband in cm, and $D_\text{up,min}$ is the minimum duty cycle for the weight to move upwards.

Timing

As shown in Figure 3.16, the controller is split into two loops: a slow loop with period τ_e, where the motor setpoints are read and the energy controller decides which components to connect to each other, and a fast loop with period τ_s where the switches are controlled, and the PWM is generated. $\tau_\mathrm{s} = 5.00\,\text{ms}$ is chosen because the selected switches would not switch much faster than this. The resolution of the PWM signal is $\frac{\tau_\mathrm{e}}{\tau_\mathrm{s}}$, so $\tau_\mathrm{e} = 100\,\text{ms}$, i.e. 10 Hz, is chosen to achieve a PWM resolution with 5 % steps.

The maximum speed of the control loop is not only dependent on the rise and fall time of the switches, but the microcontroller and the GPIO expanders also have to keep up. The most straightforward communication to the GPIO expanders is to explicitly update every pin every

cycle. However, the experiment in Section 4.3.2 shows that the execution time is too long. The I2C communications to the GPIO expanders are the bottleneck, so these communications are optimized. The clock speed of the I2C bus is increased from the default 100 kHz to 400 kHz, and I2C packets are only sent to a GPIO pin changes value, and the microcontroller buffers all GPIO changes for a single GPIO expander and sends them as a single packet.

Both control loops are completely separate, so they can both run on a different core of the Raspberry Pi Pico, as indicated in Figure 3.16. This would improve the jitter and increase the overall processing power that the controller can use. However, it increases the complexity of the firmware and introduces concurrency concerns that would be hard to debug. Therefore, the controller has not been implemented on multiple cores because no jitter or computing speed problems have been encountered.

When running the control loops on separate cores, concurrency issues might arise. For variables, making them `volatile` fixes this, as long as it is acceptable that some variables might be slightly outdated. However, when the Energy Router asks the Circuit Router to generate a new list of Screw Terminal Connections, this list must be updated atomically because otherwise, faulty circuits, possibly with shorts, could be created. This can be achieved by first generating the new connection list in a temporary variable and then briefly disabling interrupts when writing it to the actual variable:

```
noInterrupts();
connections_.swap(new_connections);
interrupts();
```

3.6 Experiments

To find an answer to the research questions posed in Chapter 1, several experiments are planned in this section. First, some sensors are added to measure the power that the designed system uses. Then, experiments are described to characterize the switches and calibrate the motor constant of the flywheels and the voltmeters. Next, a validation experiment is described to measure how well the controller can track a target signal, and finally, experiments are described to measure the power consumption of the system for various control algorithms and target signals.

3.6.1 Experimental setup

In order to evaluate the energy efficiency of the proposed design, its power draw needs to be measured over time.

One can recognize two separate power flows in the design: a digital low-voltage flow for the microcontroller and devices it controls and an analogue high-voltage flow consisting of the circuit components, such as motors, because a real-world system would probably already include a microcontroller, so only the high voltage motor driver part would have to be added. Therefore, measuring only the power usage of this part of the circuit is most interesting.

The microcontroller and analogue circuit are powered by different sources, so they can also be measured separately. The power, current, and voltage can be measured over time. The current can then be measured using an R&S RTB2004 Oscilloscope [46] as the voltage drop over a shunt resistor with resistance $R_{\text{shunt}} = 0.5\,\Omega$. The shunt resistor is added on a separate perf board PCB, for which the layout is shown in Figure 3.18. This PCB also includes a 2A fuse to protect the switches from too large currents from the power supply if they were to happen.

The power P can then be computed as follows:

$$P = V \cdot I = (V_{\text{supply}} - V_{\text{shunt}}) \frac{V_{\text{shunt}}}{R_{\text{shunt}}}, \tag{3.6}$$

where V_{supply} is the supply voltage, and V_{shunt} is the voltage over the shunt resistor.

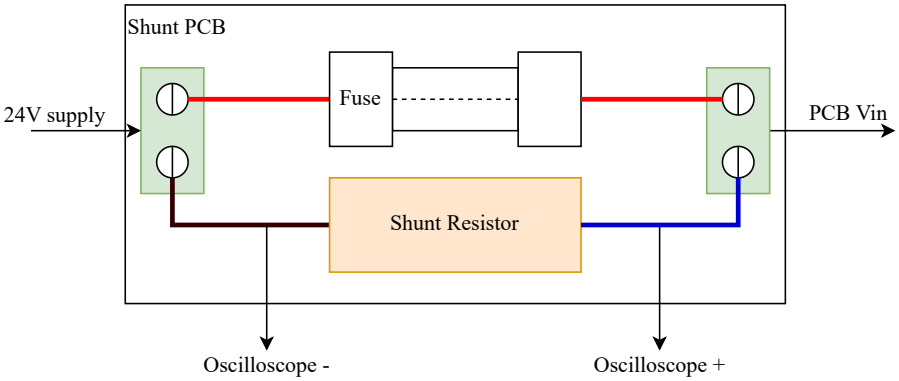


Figure 3.18: Shunt resistor PCB layout, consisting of a shunt resistor, fuse and two screw terminals.

Next to measuring the power consumption, the microcontroller measures the voltage of components and the height of the weight. It also makes decisions on which components to connect to each other. Logging this information is useful because it can help explain the power measurements, and it can be used to validate the system and debug it if necessary. Therefore, a logging class is added to print a line formatted as JSON [47] to the serial console every period τ_e.

In a real-world application, the velocities of the target velocities for the flywheels and the target height for the weights would be set by an external controller for the application, but for the experiments, the targets can be hardcoded into the firmware as a schedule. Such a schedule is a list of timestamps and targets for every flywheel and weight corresponding to the timestamps. Besides, functions have been implemented to automatically generate these schedules for some specific waveforms and concatenate them into longer schedules.

During the experiments, the components were connected to the screw terminal on the PCB as shown in Figure 3.19. If a component is not needed for a given experiment, it has been disabled in the firmware, so it can safely be left connected to the grid, although it can also physically be disconnected.

3.6.2 Experiment scenarios

Characterization and calibration experiments

Before the system designed in this report can be used as a motor controller, some of its physical properties need to be measured so that they are known to the firmware. First, the switches have been characterized. Secondly, the motor constant is determined, and lastly, the voltmeters are calibrated. The HP 54603B oscilloscope [48], Agilent 34401A digital multimeter [49], and Testo 470 tachometer [50] are used for these measurements.

Before the design has been completed, the rise time of a switch is already measured using the circuit in Figure 3.20 for a 1 kΩ load resistor R_L and values of resistor R_D ranging from 330 Ω to 6.8 kΩ. The current through R_D is also measured. This measurement is performed to confirm the rise time in the datasheet since its ability to generate PWM is crucial to the design. Besides, this experiment investigates if the switches could be driven with less current, allowing more switches to be enabled simultaneously within the current limitations of the microcontroller. V_{GPIO} was driven by a GPIO pin of the Raspberry Pi Pico, and both V_{GPIO} and V_{supply} were captured by the oscilloscope to measure the rise time.

Later, when the design is completed, the duty cycle of the generated PWM signal was measured both at the GPIO pin and at the load side of the switch. For this experiment, the value of R_D that was chosen in the previous experiment was used, and R_L was 1 kΩ again.

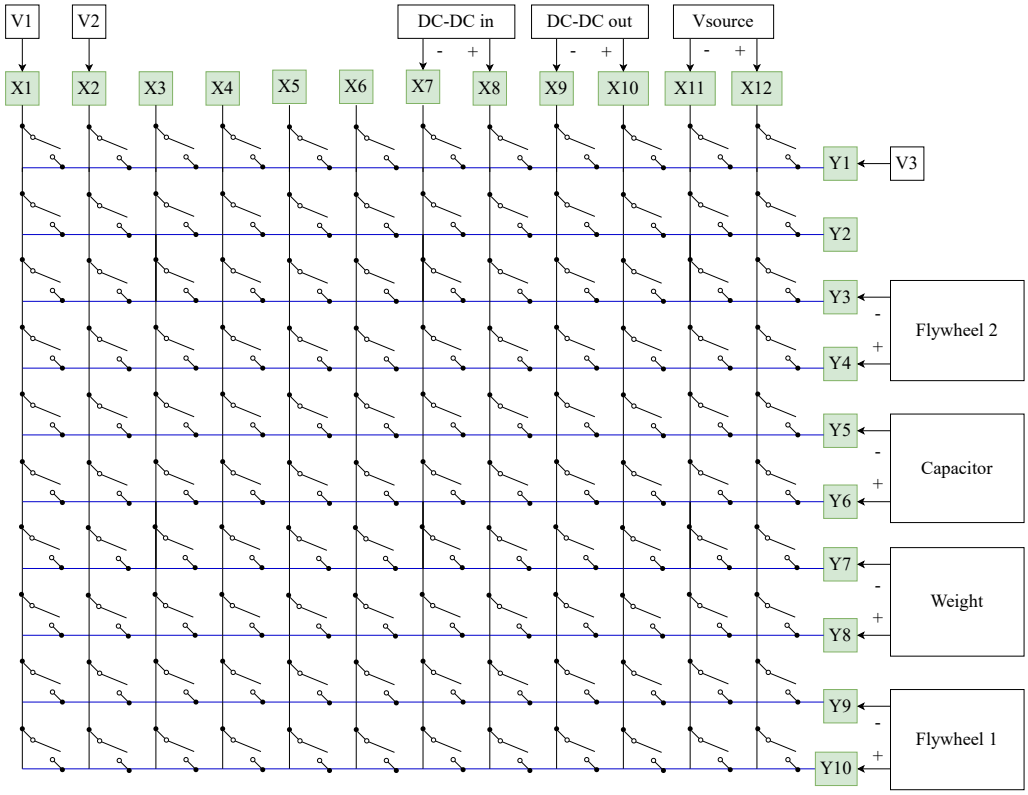


Figure 3.19: Components connected to the screw terminals on the PCB for measurements.

In order to determine the motor's speed constant k_v from Equation 3.2, a fixed voltage V_{supply} is supplied to the motor. Its angular velocity can then be measured using a tachometer to determine the relationship between supply voltage and angular velocity. However, some of the voltage from the supply drops over the series resistance in the motor, so it is not converted into mechanical energy in the flywheel. Therefore, the relation between V_{supply} and the open circuit motor voltage $V_{m,open}$ also needs to be measured to find the speed constant.

The voltmeters are calibrated by measuring the resistors in the voltage dividers. These values are added in the firmware, and then the voltmeters are read out for known V_{supply}. If necessary, a voltage offset was introduced in the firmware.

Validation measurements

To validate that the controller can track a target signal and thus perform the primary function of a motor controller, a sine wave target signal with a period of 40 s and velocities from 500 RPM to 3000 RPM was measured. The period is not chosen faster because the wave would be too fast for the controller to follow the target signal for lower periods properly.

Assuming a setup with two flywheels, each measurement has been performed for each of the following different Energy Controller types:

- No regen: use the power supply only.
- C regen: use the power supply and regeneration with a capacitor.
- M regen: use the power supply and regeneration with a motor.
- MC regen: use the power supply and regeneration with motor and capacitor.

To also investigate the energy efficiency of the design when lifting a mass, the same experiments can be performed for a setup with one weight and one flywheel.

The power consumption of the different controllers has been measured, and to quantify the quality of the sine wave and compare the different controllers, the Root Mean Square Error

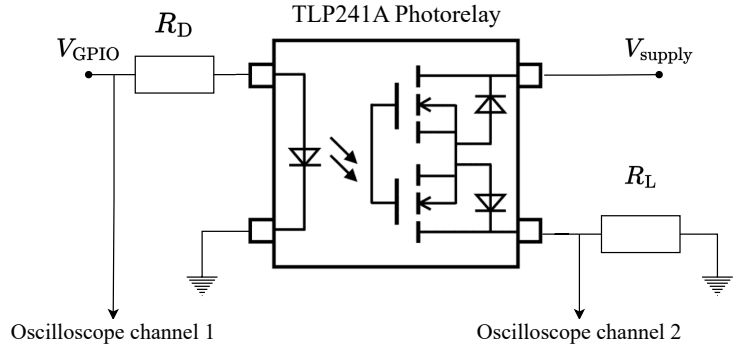


Figure 3.20: Circuit to measure rise time of a switch.

(RMSE) is calculated using

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\omega_{\text{measured}} - \omega_{\text{target}}\right)^2}, \tag{3.7}$$

where N is the total number of samples measured, ω_{measured} is the measured angular velocity and ω_{target} is the target angular velocity. The delay is also calculated. It is defined as the time shift between the target and measured velocity that yields the maximum correlation between these two signals.

Efficiency measurements

The average power delivered to the system by the power supply has been measured to quantify any power savings that motor-to-motor regeneration might yield, thus providing a measure of the system's efficiency. This has been measured for the same four controllers as the validation experiment above. The properties of the target signal have been varied. It is a square wave, shown in Figure 3.21, because this waveform is periodic and applies the maximum amount of braking.

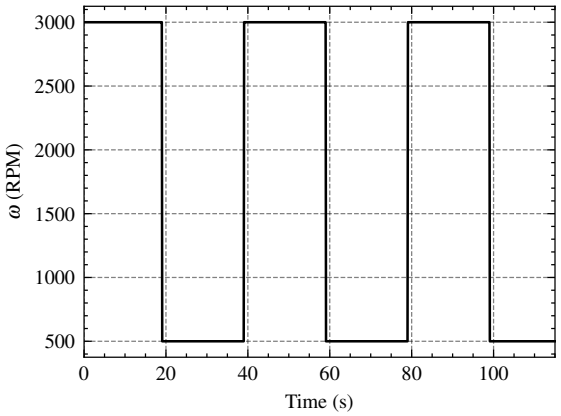


Figure 3.21: Default target square wave with a period of 40 s and velocities from 500 RPM to 3000 RPM.

The default period for the square wave is chosen to be 40.0 s. To maximize the potential motor-to-motor regeneration, the square waves for consecutive motors are in anti-phase. For the flywheels, the lower amplitude of the square wave is set at 500 RPM and the higher amplitudes at 3000 RPM. The lower side is not chosen at 0 RPM to reduce the effect of a large starting current, which could lead to less consistent measurements and stress the switches. The weights are lifted from 0.0 m, allowing to rest the weight on the floor, to 0.6 m, limited by the height of the desk on which the motor is mounted.

To get a better sense of the average power consumption, the square wave has been repeated 8 times. This number is limited by how long the oscilloscope can measure at once.

The various properties of the square wave have been varied as described in Table 3.5 to investigate their influence on the energy efficiency of the proposed design.

Table 3.5: Parameter sweeps to be performed.

Parameter	Start	End	Step
Flywheel high amplitude (RPM)	1000	3000	500
Weight high amplitude (m)	0.10	0.60	0.10
Period (s)	10	60	10
Phase shift (°)	0	180	45

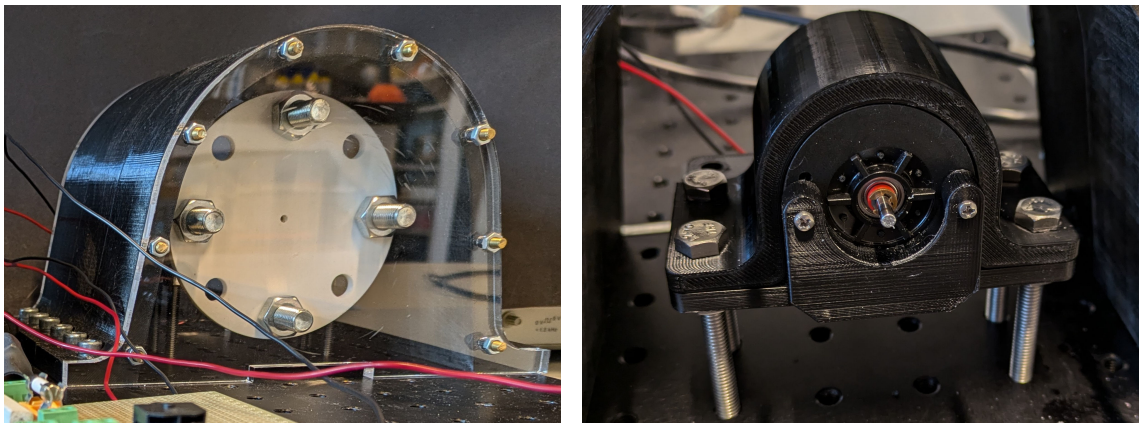
Next to these experiments, the Adafruit Motorshield v2.3 for Arduino [51] has been measured as a point of comparison to a state-of-the-art motor controller. The Motorshield supports a maximum supply voltage of 12 V, so the comparison has been made at this voltage and for the default square wave. It also runs at a much higher PWM frequency of 1600 Hz. To investigate if this difference in PWM frequency affects the comparison, the motor controller designed in this research has also been measured when powered by a 12 V adjustable lab voltage supply [52], with the PWM set at 100 % duty cycle, effectively eliminating the switching between power and ground, but keeping the dead time to be able to measure the angular velocity of the flywheels.

4 Results and Discussion

The design from the previous chapter has been realized and will be shown. Then, the measurement setup is characterized and validated. Next, the system is characterized and validation experiments are performed. Finally, the system's power draw is measured and discussed.

4.1 Setup

The flywheels have been successfully realized. One of them is shown in Figure 4.1.



(a) **(b)**

Figure 4.1: Flywheel with protective dome (4.1a) and closeup of the motor assembly (4.1b).

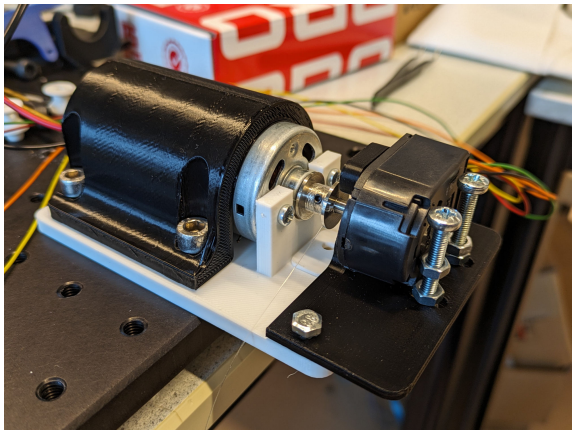


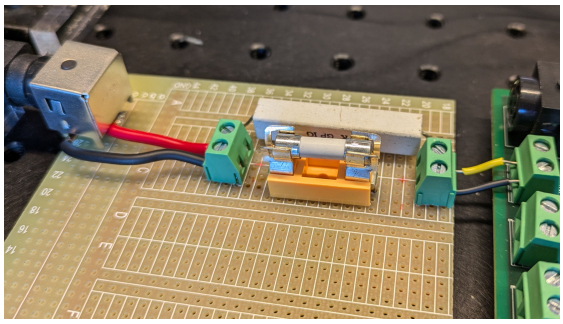
Figure 4.2: Weight lifting motor and encoder and 3D-printed mounting parts.

The weight-lifting motor is shown in Figure 4.2. After assembling, it was first tested with a 0.25 kg mass. Looking at the control signals from the microcontroller, the controller could bring the mass to a predetermined height and keep it around that point, but it oscillated wildly. This caused the 0.26 mm thick nylon rope used to lift the mass to snap consistently within one minute. Using a stronger Berkley X9 braid rope with a diameter of 0.43 mm [53] resolved the snapping but not the wild oscillations, causing a peak current exceeding the rated 2 A, and consequently breaking the circuit. The oscillations are probably because the PWM signal is only 10 Hz, giving the mass some time to fall during the off-time of the PWM cycle. The weight-lifting did not consistently work, so it has not been used for any of the experiments.

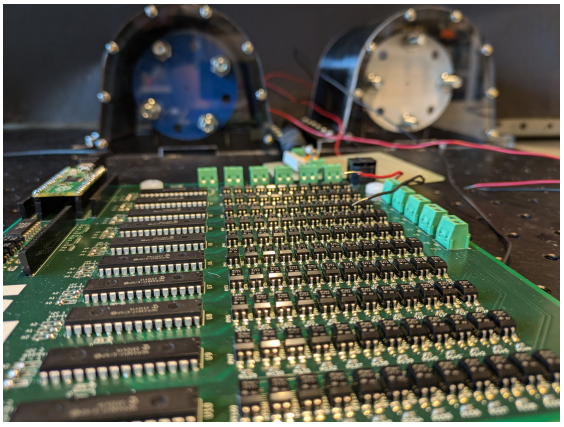
Similarly, regeneration using the buck-boost converter has been tried. For the test, a discrete resistor feedback resistor has been placed in the converter, configuring it to permanently boost

the output voltage to 24 V. When connecting it to a high-voltage decelerating flywheel and a low-voltage accelerating flywheel, the latter would barely accelerate. The same was observed when connecting the DC-DC converter between the capacitor and a flywheel. On the contrary, when directly connecting the flywheels, or a flywheel and a capacitor, without the DC-DC converter in between, the low-voltage flywheel accelerates much more. Hence, the DC-DC converter is not used in further measurements, nor is the digital potentiometer designed in Section 3.4.1 used to control the DC-DC converter. Reasons for this poor performance of the DC-DC converter could be that it is used near its limits, i.e. with high voltage gain and large input current, and that the converter is not powered during the dead time of the PWM, so 10 ms during every 100 ms, which could allow the energy storage elements in the converter to discharge, thus wasting energy.

To finalize the setup, the PCBs have also successfully been realized, as shown in Figure 4.3.



(a) Realization of shunt resistor PCB from Figure 3.18.



(b) Realization of main PCB from Figure 3.12 with flywheels in the background.

Figure 4.3: Realized PCBs

4.2 Measurement setup properties

The accuracy of the measurement setup is determined in this section. The resistance of the shunt resistor is measured to be 0.450 Ω using the HM8118 LCR meter [54]. The RTB2004 oscilloscope [46] has a resolution of 10 bits, so if the range is set to 1 V, the current resolution that can be derived from measuring the voltage over the shunt resistor is 2 mA. Using an Agilent 34401A digital multimeter [49], the voltage from the supply is measured to be 24.088 V without load, during peak load it was measured to drop no lower than 24.040 V. So the resolution for power is $0.04\,\text{V} \cdot 2\,\text{mA} = 48\,\text{mW}$.

4.3 Characterization and calibration results

4.3.1 Switch characterization

The rise time and current draw of the switch are measured as described in Section 3.6, yielding the results in Table 4.1. Figure 4.4 shows that the fall time is much shorter than the rise time, so the rise time is the limiting factor for the maximum switching frequency. The measured values correspond to those in the datasheet, and lower values of R_D result in higher currents and lower rise times, as expected [34]. The microcontroller can only supply a limited current, so when enabling a switch draws more current, fewer switches can be enabled at a time. In other words, the resistor value is a trade-off between the rise time and the maximum number of switches that can be enabled simultaneously. Therefore, the maximum resistor value that meets the required rise time requirement must be chosen. To generate a PWM signal with 5 ms

steps, the switch must be able to turn on and off within this time step. The maximum resistance for which this is the case is 220 Ω, so this value is chosen.

Table 4.1: Current draw and load side rise time to 15 V for different resistor values when driven at 3.3 V by Raspberry Pi Pico.

R_D (Ω)	Current (mA)	Rise time (ms)
2.2×10^2	7.33	3.7
3.3×10^2	5.30	5.4
6.8×10^2	2.81	10.0
1.0×10^3	1.95	14.8

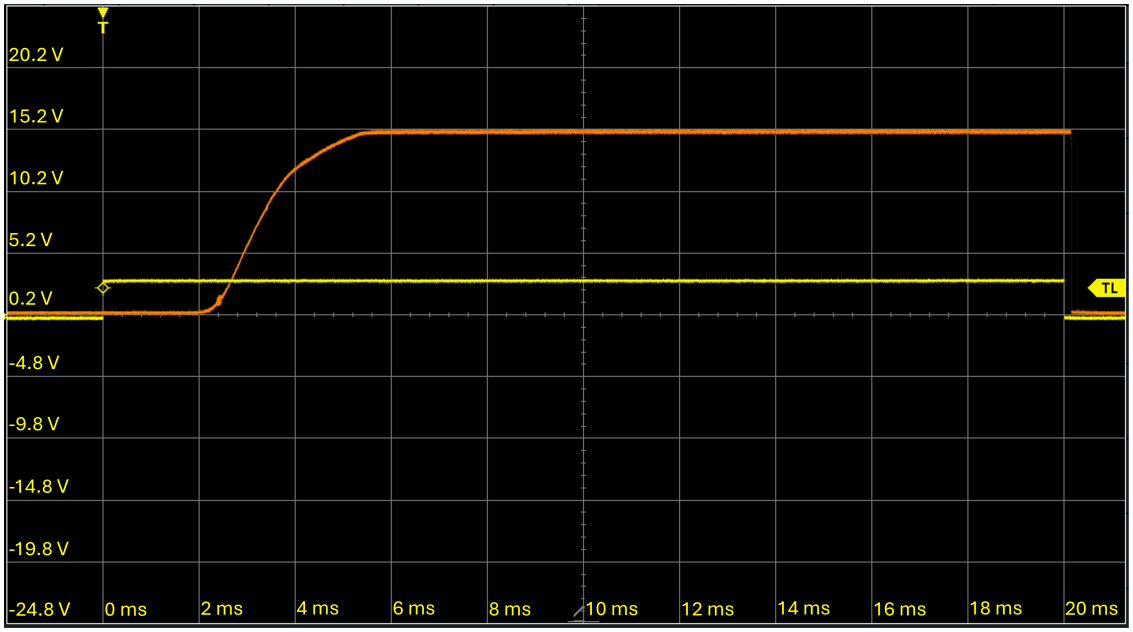


Figure 4.4: Oscilloscope screenshot showing the input pulse (yellow) and response (orange) for TLP241A switch [34] with 330 Ω resistor connected to the Raspberry Pi Pico [38] showing rise time and fall time.

4.3.2 PWM validation

As mentioned in Chapter 3, the system should generate a 10 Hz PWM signal, with 5 ms steps. However, the implementation of the control loop is measured to be 55 ms when only generating a simple PWM signal for a single motor. It would be even slower in a more demanding scenario and with an additional motor. Therefore, optimizations have been implemented in Section 3.5.2, yielding code execution times of around 1 ms, allowing the control loop to run at the desired 5 ms.

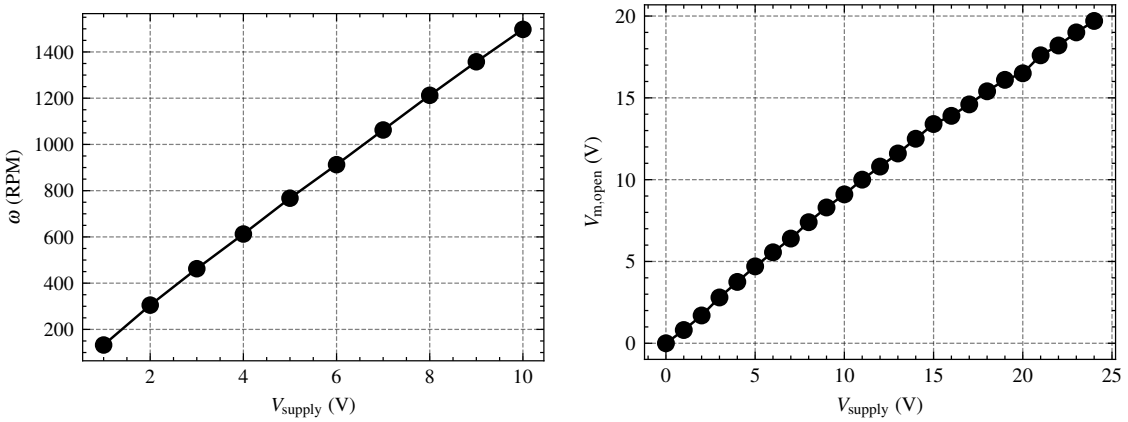
To validate this, PWM signals with target duty cycles ranging from 0 % to 100 % are generated and measured at the GPIO pin and at the load side of the switch. Table 4.2 shows that a PWM signal can successfully be generated with an accuracy of 1 % at the GPIO pin. The duty cycle seen by the load is consistently 2 % to 3 %, i.e. 2 ms to 3 ms, below the target value. This is due to the rise time of the switch, which is in the same order of magnitude as the PWM resolution of 5 ms. It is also noteworthy that the maximum duty cycle is 90 %. This is due to the dead time, which is 10 ms, i.e. 10 % of the PWM period.

Table 4.2: PWM cycles measured on and load side of a switch and the controlling GPIO pin.

Target duty cycle (%)	GPIO duty cycle (%)	Load duty cycle (%)
0.0	0.0	0.0
10	10	7.0
20	21	17
30	30	27
40	40	37
50	50	47
60	60	57
70	69	67
80	81	77
90	91	88
100	91	88

4.3.3 Motor constant calibration

To determine the speed constant k_v of the flywheel motors (Equation 3.2), their angular velocity is measured for various supply voltages as described in Section 3.6. The resulting graphs are shown in Figure 4.5a. The average slope of this line, i.e. the conversion factor from supply voltage to flywheel velocity, is 150 RPM/V. Next, the open circuit voltage of the motor for various supply voltages in Figure 4.5b. The relation $V_{m,open} = 0.85 V_{supply}$ is found. From this, k_v is found to be 176 RPM/V. This value is found for both flywheels, so the Figure 4.5 only shows the graphs for one of them.



(a) Angular velocity for varying supply voltages V_{supply}.

(b) Open motor voltage $V_{m,open}$ for varying supply voltage V_{supply}.

Figure 4.5: Measured curves to calibrate the flywheels' motor constant.

4.3.4 Voltmeter calibration

The voltmeters consist of a voltage divider connected to the ADC of the Raspberry Pi Pico, so to improve their accuracy, they are calibrated. The measured values of the resistors in the voltage dividers are shown in Table 4.3. The values are programmed in the firmware, and then the voltage read by the ADC is measured for various supply voltages, yielding Figure 4.6. The voltmeters consistently measure the voltage too high. The median error is 0.7 V, so this is added to the firmware as a DC correction term. This is expected because when the internal ADC reference of the Raspberry Pi Pico is used, which is known to result in a drop of about 30 mV [38].

This corresponds to a much larger voltage error on the high side of the voltage divider, causing the voltmeters to measure too high consistently.

Table 4.3: Resistors in the voltage dividers for voltmeters V_1, V_2 and V_3.

Resistor	Resistance (kΩ)
$R_{V1,upper}$	149.3
$R_{V1,lower}$	10.01
$R_{V2,upper}$	149.1
$R_{V2,lower}$	9.96
$R_{V3,upper}$	150.3
$R_{V3,upper}$	10.0

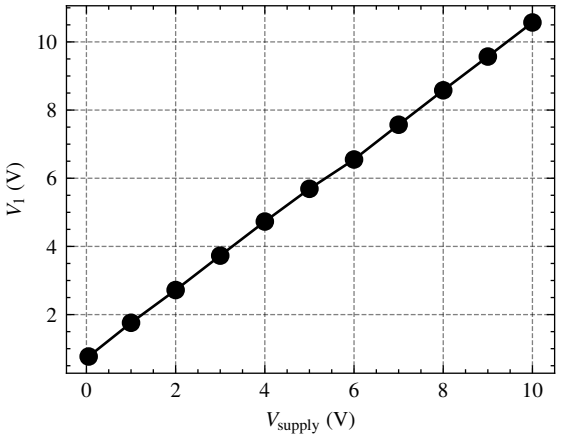


Figure 4.6: Calibration curve of voltmeter 1 voltage V_1 for different supply voltages V_{supply}. The other voltmeters yield a similar curve, so these are not shown.

The voltmeters need to be connected to the desired component by switches in the interconnect. For example, the open circuit voltage of the motors needs to be measured during the dead time, as explained before. This behaviour is validated in Figure 4.7.

4.4 System validation results

To validate that the designed system can act as a motor driver, a sinusoidal target signal is provided to the system with all four different controllers. Figure 4.8 shows the sine wave for the MC regeneration controller, and all controllers are shown in Appendix A.1. The waveform in Figure 4.8b clearly shows the resemblance of the sine wave, but it is not ideal. For example, the flywheel slows down too much to around 300 RPM, which is significantly lower than the target of 500 RPM, because the motor continuously shorts the motor when braking instead of using the PWM to power the motor a part of the time. This makes the deceleration faster, but the Energy Controller only stops the braking after the speed has already dropped below the target by up to $\tau_e = 100$ ms, causing the angular velocity to undershoot the target. This effect can be observed more clearly later in Figure 4.11b. Another example where the controller does not track the target sine wave well is when it switches between power sources. Then the acceleration is not smooth, and it lags behind the target, for example, around 1300 RPM.

The power consumption, RMSE and delay measured for the different controllers are shown in Figure 4.9. As expected, C regeneration consumes less power than only using the power supply, M regeneration uses less power, and MC regeneration consumes the least power. The bar charts of the delay and the RMSE are similar because a larger delay yields a larger error. This also suggests that the delay is the main factor in the RMSE, an interpretation reinforced by

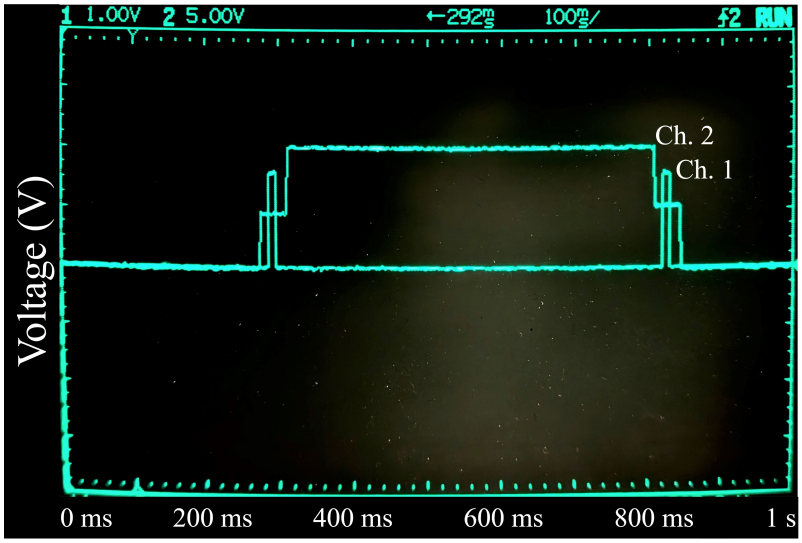


Figure 4.7: Annotated screenshot of an oscilloscope displaying the voltmeters (Ch. 1) pulsing to measure during the dead time of the PWM signal on the motors (Ch. 2). The PWM frequency in this image is intentionally slower than the final frequency of 10 Hz to aid debugging.

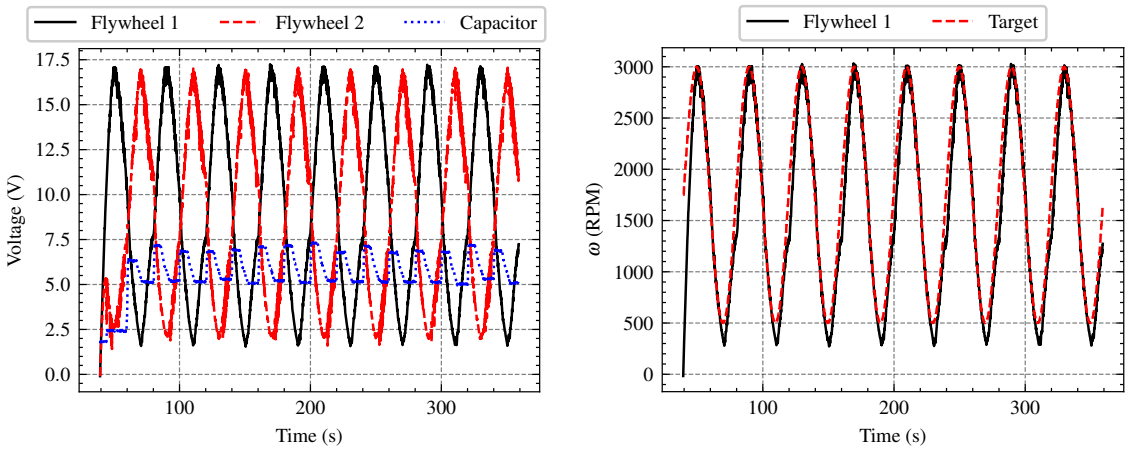
the observation that the output in Figure 4.8 resembles the target sine wave. The charts show that the quality of the sine waves is similar for all three controllers, although the C regeneration seems a bit worse. It is not immediately obvious why this is the case for C regen, but not for MC regen. A possible cause is that the PWM control signal assumes that the voltage supplied to the motor is always 24 V, but in reality, the capacitor is quickly charging or discharging, so the amount of acceleration or deceleration provided to the motor is lower than the power supply would provide and it is not constant. The same is true for the MC regen case, but here the motor provides additional energy capacity, so the acceleration is not degrading as fast.

4.5 Efficiency

As elaborated in Section 3.6.2, the power consumption of this design has been measured by providing the system with square wave targets with varying amplitudes, periods and phase shifts for different controller algorithms. As Figure 4.10 illustrates, these parameter sweeps are all concatenated in a single measurement for each controller, limiting the overhead time required for manually starting and shutting down a measurement. This long measurement is then sliced into the individual measurements, for example, the one in Figure 4.11.

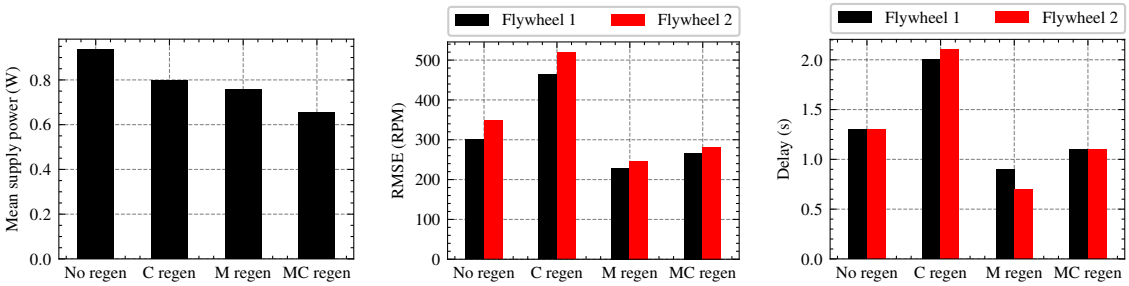
Figure 4.11 can also be used to illustrate the measurement taking place. It shows the ideal square wave (target in Figure 4.11b) as well as the angular velocity measured by the microcontroller. The voltage and velocity of the flywheel are the same curve but scaled by the motor constant, so the voltage curve also displays that the flywheels are in anti-phase. Besides, one can see that the capacitor charges when one of the flywheels brakes and later discharges again when both require more energy.

The average power consumption for the various square wave and control algorithms is displayed in Figure 4.12. Overall, the MC regeneration controller reduces the power consumption most with up to 33 % compared to only using the power supply. Second most efficient is the M regeneration controller with power reduction up to 30 %, and finally the C regeneration controller with up to 15 %.



(a) **(b)**

Figure 4.8: Example of voltages in the system (4.8a) and flywheel velocity (4.8b) when tracking a sinusoidal target signal (4.8b). The velocity is not directly measured but calculated by scaling the voltage measured by the Raspberry Pi Pico.



(a) **(b)** **(c)**

Figure 4.9: Comparison of performance metrics for a sinusoidal target signal on controllers that use no regeneration (No regen), capacitor regeneration (C regen), motor-to-motor regeneration (M regen) and both capacitor and motor-to-motor regeneration (MC regen). The compared metrics are the mean power drawn from the supply (4.9a), the RMSE relative to the target (4.9b), and the delay relative to the target (4.9c). Lower values are better for all of these three metrics.

4.5.1 Amplitude dependency

Looking at the amplitude sweep in Figure 4.12, lower amplitudes generally seem to use less power, which is expected since the flywheels simply spin slower and store less energy. Until 2000 RPM, there is a steep increase in power consumption, but the power consumption stays more or less stable for higher speeds. This is probably due to the controller balancing around the target amplitude. The PWM signal is only at 10 Hz, so the generated signal is not very smooth, and during the off-state part of the PWM cycle, the motor is shorting to itself, so it is actively braking, and it keeps requiring more energy to stay at the same angular velocity. The observation that the power draw is stable for high amplitudes could might also be caused by the fact that the maximum effective PWM duty cycle is 88 %. However, the maximum amplitude required for this measurement is 3000 RPM, which corresponds to a duty cycle below 88 %, so the limited duty cycle is not the issue here.

After this general discussion, the power consumption of the various controllers will be compared. Using only the power supply uses the most energy, next the controller that also employs capacitor regeneration, and the controllers that use motor regeneration use the least energy.

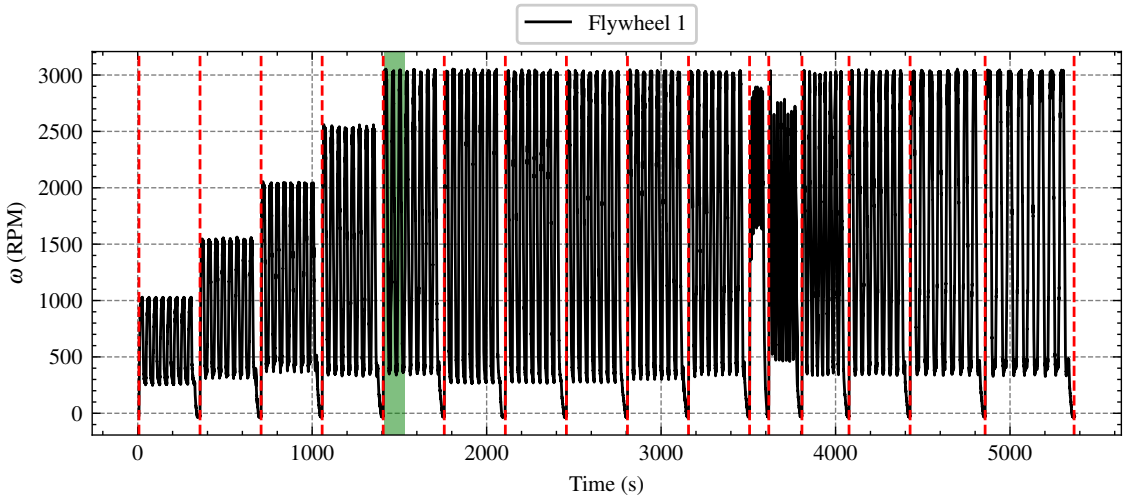


Figure 4.10: The amplitude, phase shift and period sweeps are concatenated into one measurement and then split at the dotted lines when processing. The highlighted green period is shown in more detail in Figure 4.11.

Amongst the controllers that use motor generation, the one using both capacitor and motor regeneration is even more efficient.

The controllers using motor regeneration also have a much more constant power usage over the entire amplitude range than the others. Besides, at low amplitudes, the regeneration controllers are not performing much better than the controller that only uses the supply, because the motors only have relatively little braking energy available. In fact, when the potential difference between the two components is too small, the acceleration/deceleration would be so small and so slow that connecting the components would not be beneficial.

4.5.2 Period dependency

The general trend for the phase sweeps in Figure 4.12 is that the No regen en C regen control algorithms use less power for longer periods, whereas the M regen and MC regen use very little power at small periods and increase with frequency. The first observation can be simply explained by the fact that accelerating a mass costs more energy than keeping it moving, and for longer periods, the flywheels are accelerating and decelerating a smaller portion of the time as they are longer at constant speeds. The observation that the M and MC regeneration use much less energy than the others at low periods becomes apparent when looking at Figure 4.13. The flywheels do not reach their target velocities there, and they accelerate and decelerate much less. This might be due to the longer RC time caused by the larger energy capacity of the flywheel compared to the capacitor and the larger resistance caused by placing two flywheels in series. Also, the lower potential difference between the motors compared to when powering from the supply causes slower acceleration. Although this is an interesting observation, it means that the data points below 40 s do not provide a fair comparison between the curves because the M and MC regeneration controllers do not reach the maximum target velocity.

So, comparing the different controllers for periods of 40 s and higher, the same order can be seen as previously for the amplitude: MC is most power efficient, then M, C and finally, No regeneration is least power efficient. No regen and C regen use less power for higher periods, as explained before. On the contrary, the power consumption of M and MC regeneration increases for higher periods because the supply is used for longer when the target is 500 RPM when the square wave period gets longer. This behaviour is also expected for the capacitor regeneration case, but analyzing the voltages during this measurement (Figure A.5) shows that the capacitor

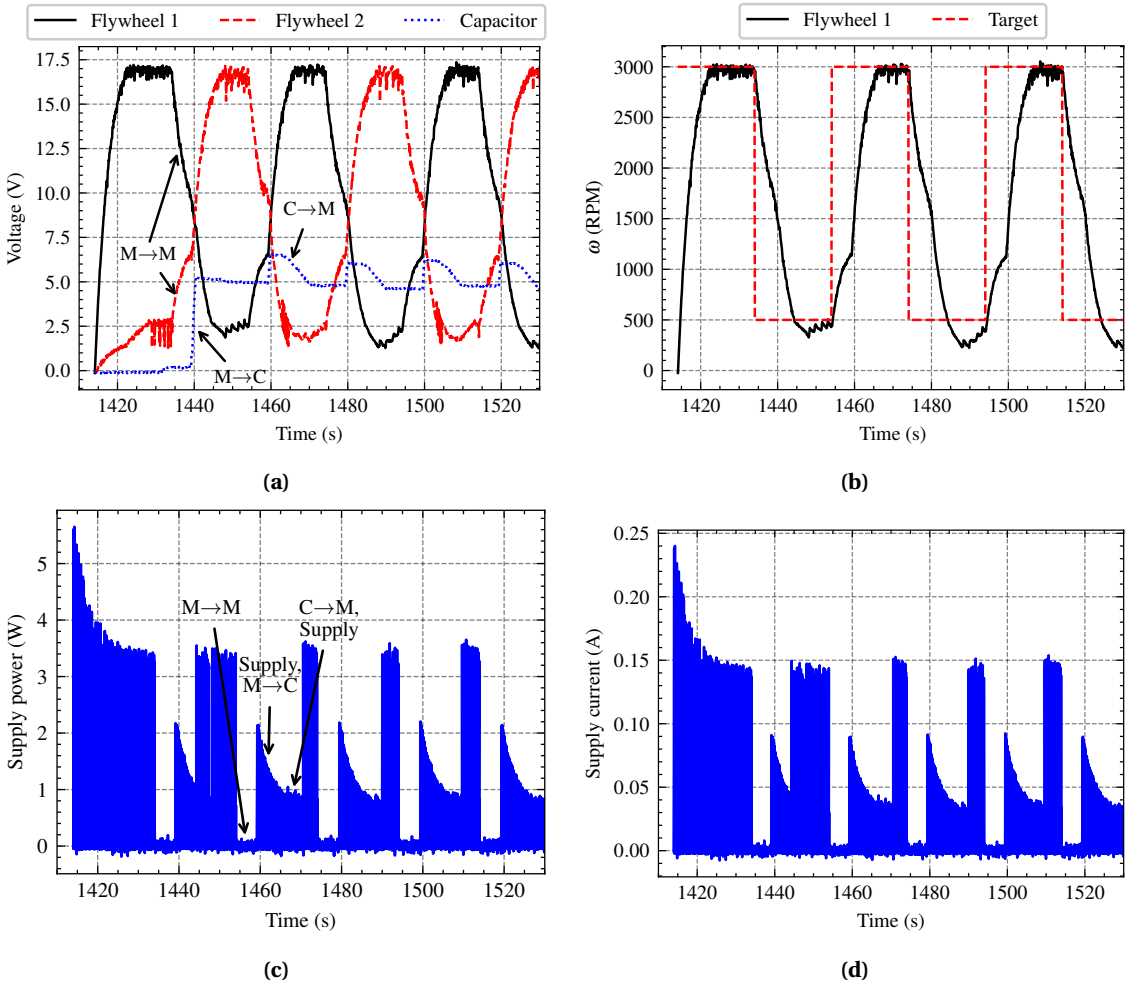


Figure 4.11: Partial slice from the total measurement in Figure 4.10: voltages in the system (4.11a) and flywheel velocity (4.11b) for a square wave target signal (4.11b), as well as the power and current delivered by the supply (Figure 4.11c and Figure 4.11d). Some instances of motor-to-motor regeneration, capacitor regeneration and the capacitor powering the motor are marked.

regeneration takes the entire time, so the voltage supply is not used when the target is 500 RPM, and that the capacitor regeneration causes a large undershoot of the target. This indicates that the voltage difference required before the capacitor gets connected as an energy source to a flywheel should be set higher.

4.5.3 Phase shift dependency

When varying the phase shift between the two flywheels, the power draw of the controller that only uses the supply stays constant as expected. The capacitor regeneration controller also consumes roughly the same power independent of phase shift, although its power consumption is a bit lower for a phase shift 180° than for the other phase shifts. It is more efficient than the controller that only uses the supply, as expected.

For a phase shift of 0°, the controller that only uses M regeneration uses as much power as the one that only uses the supply, and the MC regeneration controller uses as much as the C regeneration controller. This is because both flywheels are in phase here, so they always accelerate and decelerate at the same time, leaving no opportunity for sharing any energy between the motors. Therefore, M regeneration does not provide any efficiency gains when the flywheels are in phase. The C regeneration already boosts efficiency here, because the capacitor can

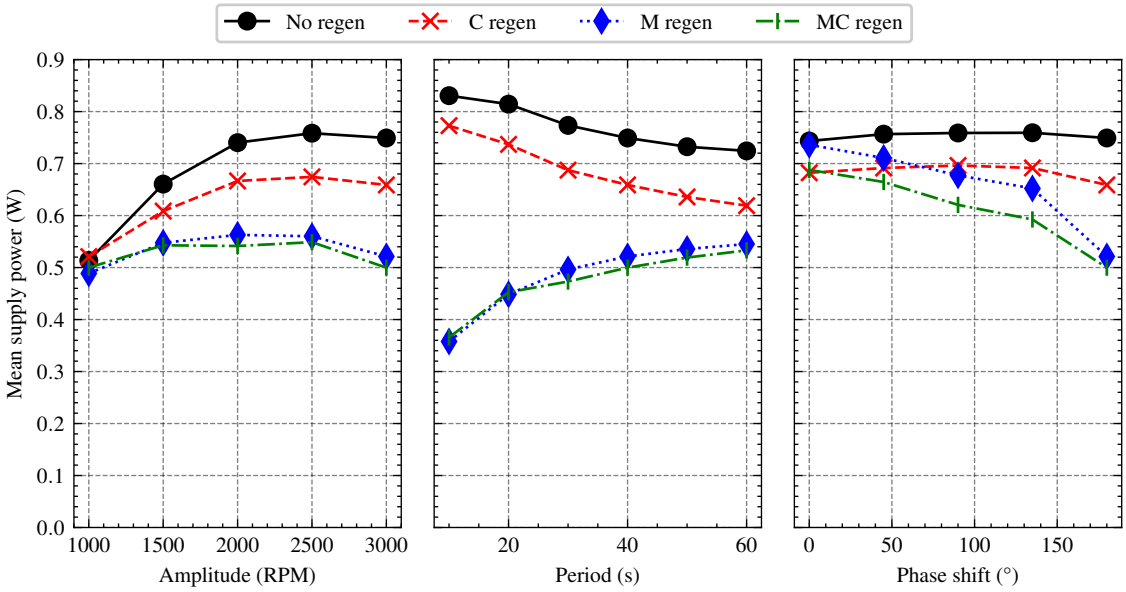


Figure 4.12: Mean power drawn from supply for varying values of amplitude (left), period (middle) and phase shift (right). Multiple controllers are compared, without any regeneration, with only capacitor regeneration, only motor regeneration, and both motor and capacitor regeneration.

store the energy released during braking for some time and release it later when the flywheels are accelerating again.

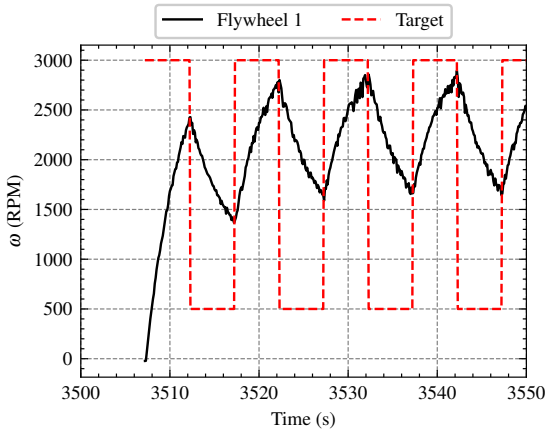
At larger phase shifts, the M and MC regeneration controllers gradually consume less power, showing that motor-to-motor regeneration positively impacts the system's efficiency as soon as the motors are not exactly in phase. Finally, there is a sharper drop in power consumption when the motors are exactly in anti-phase.

One might also expect the capacitor regeneration to yield the same efficiency improvement as the flywheel regeneration, but that is not the case here because the capacitor can simply store less energy.

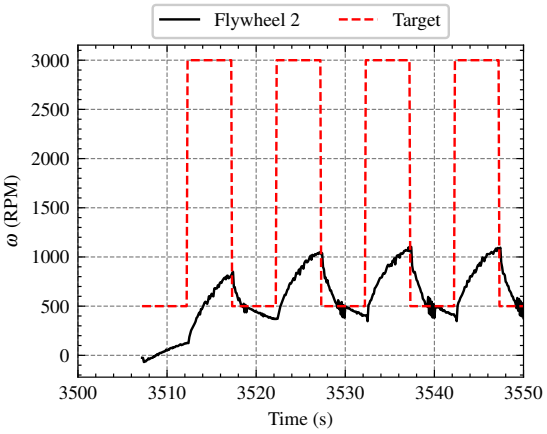
4.6 Comparison to state-of-the-art

The power consumption of the Adafruit Motorshield v2.3 for Arduino [51], hereafter referred to as Motorshield, is also measured as a reference of a state-of-the-art motor controller as described in Section 3.6.2. The results of this comparison are shown in Figure 4.14.

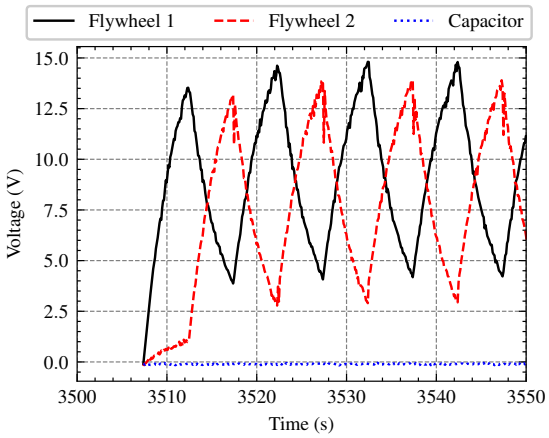
When powered at 24 V, the motor controller designed in this research uses about 83 % more power than the Motorshield. However, when the motor controller designed in this research is powered at 12 V and 100 % duty cycle, it uses a bit less power than the Motorshield. This suggests that the differences are primarily due to the Motorshield running at a much higher PWM frequency of 1600 Hz compared to the 10 Hz of the controller presented in this research. This allows the Motorshield to generate a much smoother control signal where the motor is short braking for much shorter continuous durations, so the flywheel does not have time to slow down significantly. So, if this design can be optimized to run at a similar frequency, it might achieve similar efficiency, and then the efficiency gains from the regeneration would still be added on top of that.



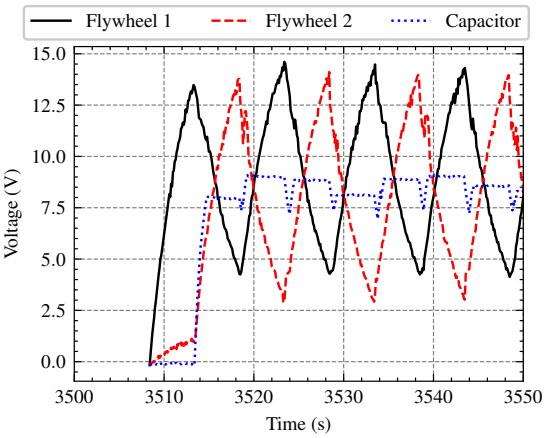
(a) Flywheel 1 velocity and its target for M regeneration.



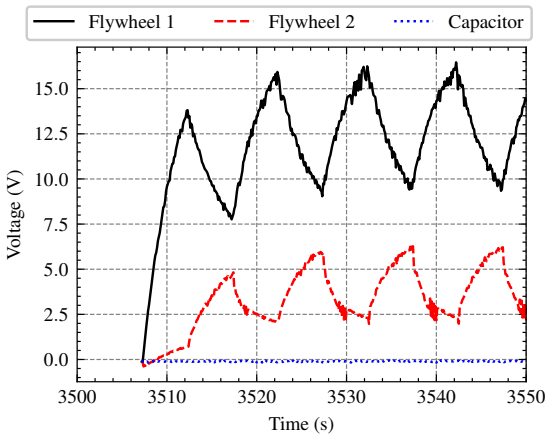
(b) Flywheel 2 velocity and its target for M regeneration.



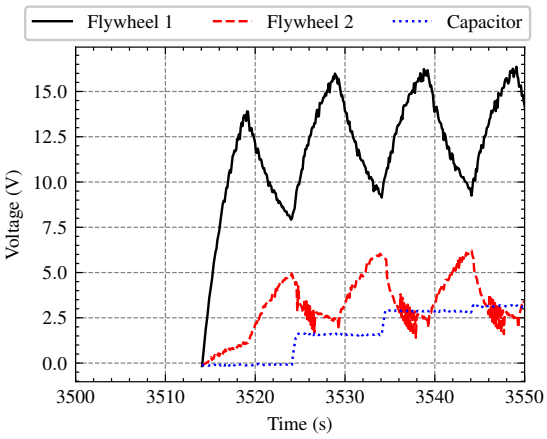
(c) No regeneration



(d) C regeneration.



(e) M regeneration.



(f) MC regeneration

Figure 4.13: Partial slices from Figure 4.10 for a square wave with period 10 s, showing that the target velocities are not reached for this period. No regeneration and C regeneration yield similar plots. M regeneration and MC regeneration are similar as well.

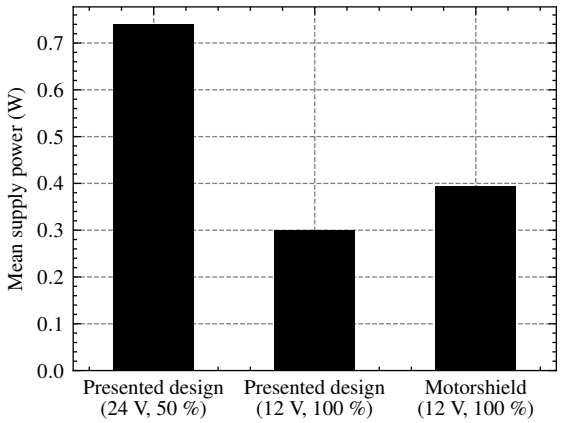


Figure 4.14: Comparison of average power consumption of the motor controller controlled presented in this research, without any regeneration and Adafruit Motorshield v2.3 [51]. The design presented in this research is measured once with a 24 V supply and a 50 % duty cycle and once with a 12 V supply and a 100 % duty cycle.

5 Conclusions and Recommendations

5.1 Conclusions

In this report, a controller for multi-motor systems has been designed using a reconfigurable switch grid, enabling a novel regeneration method, motor-to-motor regeneration, as well as more traditional capacitor-based regeneration. As this is a proof of concept no particular application is used, but instead generic flywheels to store inertial energy and a weight to store gravitational energy are used. However, the weight lifting could not be used during the measurements because the peak current draw exceeded the maximum current of the interconnect.

First, several circuit topologies have been studied. After choosing for a crossbar interconnect with components on both axes, components for the system have been selected, and some parts have been 3D-printed. A PCB has been designed to compactly implement the circuit, and firmware has been designed and implemented to control the system.

The maximum achievable PWM frequency for this design was found to be around 10 Hz. This was mainly limited by the rise time of the switches, but even if faster switches were found, the maximum frequency would quickly be limited again by the speed of the I2C bus controlling the pin expander ICs. Besides, the rise time and corresponding trigger current of the switches have been characterised, and the generated PWM signal has been validated. Furthermore, the motor constant and voltmeters were calibrated with the use of a digital multimeter and tachometer, respectively.

The system is capable of controlling the speeds of the flywheels given a sinusoidal target signal. Multiple controller variations have been implemented to evaluate the efficiency of the system. One controller only uses the power supply as in traditional motor controllers, another one uses a capacitor for energy recuperation, and finally, there are two controllers applying motor-to-motor regeneration, one only using it and one combining it with capacitor regeneration. The power consumption of these controllers has been measured for varying amplitudes, periods and phase shifts of a square wave target signal. When the flywheels are in anti-phase, using motor-to-motor regeneration has been found to reduce the power usage by up to 30 %, whereas capacitor-based regeneration only yields an improvement of up to 15 %. This is thought to be because the capacitor can store less energy than the flywheels, but this would have to be confirmed by additional experiments. Combining both regeneration methods has yielded a slight further improvement of the power draw to at most 33 %. The measured improvements are significantly lower than the expectations of around 50 % because the braking energy can only be recuperated when the voltage of the braking motor is higher than the voltage of the accelerating motor or the capacitor's voltage.

Next, comparing the system's performance without regeneration to a state-of-the-art motor controller, the Adafruit Motorshield for Arduino, shows that the design is this report is not quite optimal yet, as the Motorshield is about 83 % more efficient. This is thought to be due to the Motorshield running at a much higher PWM frequency than the presented design. However, the Motorshield does not implement any form of regeneration, suggesting that if the motor controller in this report would run at the same higher PWM frequency, it might be about as power efficient as the Motorshield and adding capacitor and motor-to-motor regeneration would further reduce the power consumption.

When looking at the effects of the different waveform properties, a higher amplitude generally increases the power usage, but it flattens for speeds close to the maximum. Regeneration also has the largest effect for higher amplitudes because much more braking energy is available. For larger periods, the power usage generally decreases because keeping a constant speed becomes a larger portion of the measurement. However, it was also found that the controllers using

motor-to-motor regeneration have slower acceleration and deceleration, causing them to miss the target amplitudes for short periods. Finally, providing target signals with different phase shifts shows that motor-to-motor regeneration does not improve energy efficiency when the flywheels are in phase, but as soon as there is any phase shift, improved efficiency is observed, with the most efficient scenario being that the flywheels are in anti-phase. On the contrary, the capacitor regeneration yields an improved efficiency irrespective of phase shift because of its capability to store energy for some time.

Finally, motor-to-motor regeneration uses the existing components of the system, whereas capacitor-based regeneration requires capacitors to be added to the system, making it larger and heavier. However, the switches used in the current prototype cost more than adding capacitors with similar energy storage as a flywheel, and the switches were found hard to scale down. A potential solution to both of these downsides is recommended in Section 5.2, but for the current design, it depends on the application requirements whether motor-to-motor regeneration is more attractive than adding capacitors.

5.2 Recommendations

Some recommendations for future research can be made. First, suggestions are made to achieve a design with faster PWM. Next, some improvements to the current design are recommended, and finally, some ideas for additional experiments with the current design are listed.

5.2.1 Increasing PWM frequency

Multiple issues were encountered with the current design due to the low PWM frequency, so implementing faster PWM potentially yields a great performance boost. Using pulse density modulation (PDM) instead of PWM increases the switching frequency because it switches multiple times per period [55]. This can be implemented without any hardware modifications. Another method to increase the PWM frequency with the current PCB is to attach an external PWM generator to the screw terminals, such as the one on the Motorshield. The switches on the PCB would still make the connections between the right components, but they would no longer need to generate the PWM signal. Besides, using faster switches for the voltmeters would also yield a performance improvement, because currently, the dead time in the PWM cycle has to be extended to ensure that the voltmeters can always measure the voltages of each motor that is in the dead time state.

In Section 3.3.3 the choice was made to use solid-state relays instead of MOSFETs because the latter cannot be activated bidirectionally for both negative and positive voltage drops between drain and source. However, gate drivers can be placed between the microcontroller and the MOSFETs to overcome this issue[56]. This makes MOSFETs preferred over solid-state relays because they can switch much faster, allowing for faster and more power-efficient PWM. Besides, they are simpler devices and have a lower internal resistance, so they dissipate less power, making it possible to scale them smaller than the solid-state relays. However, the faster PWM would require diodes to be placed in parallel to the switches, as explained in Section 2.2. These could be connected as discrete components to the screw terminals, allowing them to be connected in the correct orientation via the switches. On the other hand, the faster PWM makes it possible to use the H-bridge in combination with the motor's inductance to boost the voltage, allowing regeneration without changing the orientation of the diode, so it could then be permanently connected, reducing the number of switches required. However, this makes the regeneration more complex to control [18]. When considering this approach, off-the-shelf H-bridge ICs can be used. These contain multiple MOSFETs and might have an integrated gate driver, reducing the number of separate ICs required for the design and, thus, the footprint. This makes the circuit more specialized to motor control, limiting its use for different applications.

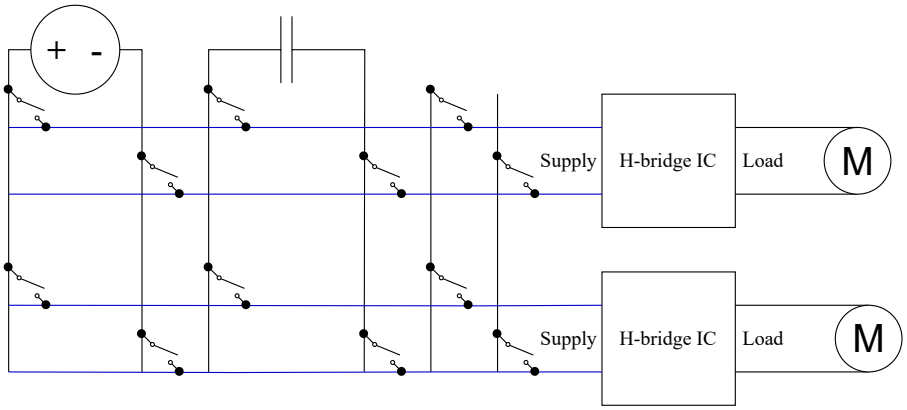


Figure 5.1: Suggestions of an improved design using H-bridge ICs allowing for faster PWM and smaller footprint. The discrete switches could be either solid-state relays or MOSFETs in combination with gate drivers. The four switches on the right that are only connected to the H-bridge can be used to connect the H-bridge ICs together, allowing for motor-to-motor regeneration.

In the current design, the effect of adding faster switches to increase the PWM frequency would be limited by the fact that the PWM signals are generated by the microcontroller and then sent to the pin expander ICs via the I2C bus. PWM generator ICs could be used instead of pin expanders. For example, the PCA9685 used on the Motorshield can generate 16 PWM signals between 24 Hz and 1526 Hz [57].

5.2.2 Design improvements

In hindsight, some improvements can be made to the PCB. Since the supply can deliver more current than the switches are rated for, a fail-safe should be added, for example, a fuse or a current-limiting diode. Adding a physical toggle button on the power line would also make the system easier to use because now the entire power supply must be unplugged from the wall socket to turn the circuit off. Furthermore, an external voltage reference for the Raspberry Pi Pico could be added because this improves the ADC performance [38]. The power supply ground should also be connected to the AGND pin of the Raspberry Pi Pico instead of DGND because this could reduce noise.

The controller can also be changed to use motor-to-motor regeneration when one motor is decelerating and the other is spinning at a constant speed instead of only when the motor is accelerating. The accuracy of the controller can also be improved, for example, by calibrating the PWM.

5.2.3 Further experiments

Although it can be improved, the motor controller designed in this research can still be used for further experiments.

For starters, there are some alternative configurations imaginable that this research has not evaluated. For example, a buck-boost converter configured to always boost the voltage to 24 V has been briefly tried in this research before being discarded for using too much power. However, the buck-boost converter could also be used in other ways. For example, it can be controlled using the digital potentiometer that is already included in the PCB to have a variable output. The buck-boost converter can also be used to take over the system's PWM generator function. Alternatively, it might be beneficial to replace the buck-boost converter with a more efficient boost converter or to create a custom buck-boost converter by hooking up multiple capacitors to the circuit and connecting them in parallel when a low voltage is needed and in series to create a higher voltage.

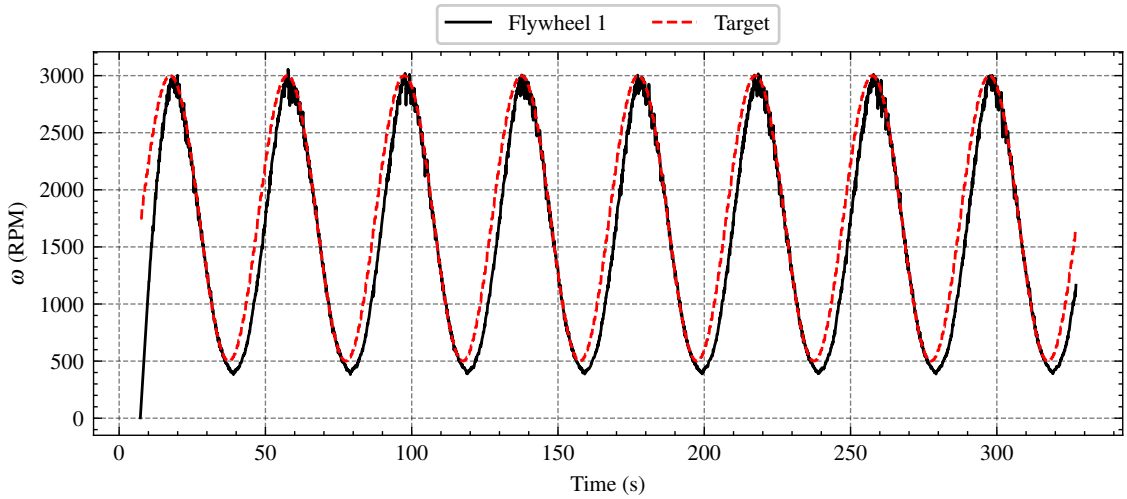
Secondly, this research did not manage to complete measurements for lifting a weight. The mass wildly oscillated around the target height, causing too large peak currents, so the voltage applied to this motor must either be smoother or have a lower amplitude for the weight lifting to be successful. Adding capacitors in parallel with the motors would smoothen the signal, but charging them up would cost additional energy. This solution can be evaluated in future work.

Finally, the grid of switches that this research uses to control motors can also be used for other applications that would benefit from dynamically connecting or disconnecting components in the circuit. For example, a DC-DC converter by connecting multiple capacitors to the screw terminals. The microcontroller can then decide to put them in parallel for a low voltage or to put two or more capacitors in series to achieve a higher voltage. Alternatively, an inductor can be connected to the interconnect to form a DC-DC converter. This can be done in combination with capacitors. An inductor can also be placed in series with a DC motor to boost the voltage higher when regeneratively braking.

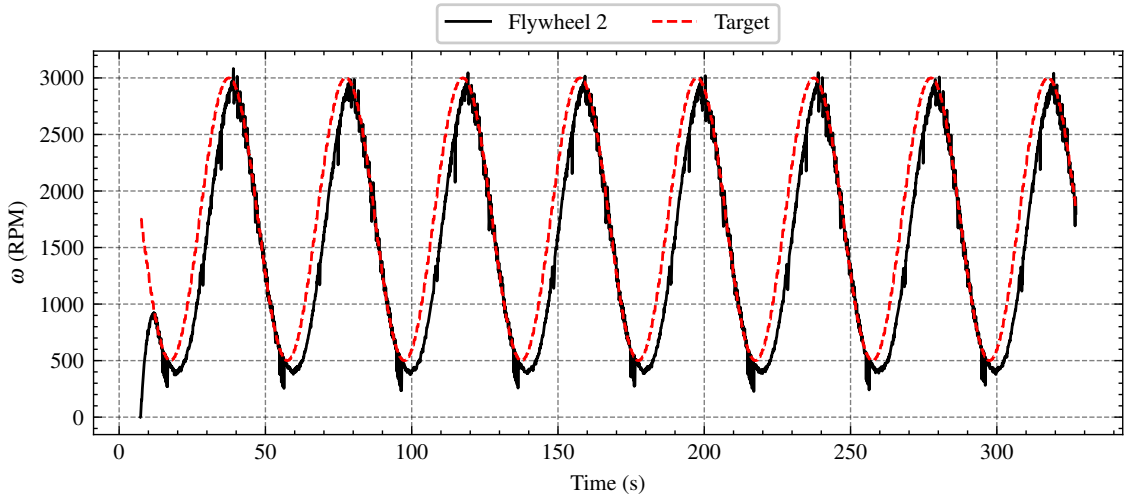
A Additional results

A.1 Sine over time

In Figure 4.9 the performance of the controller is summarized as bar charts. This appendix section elaborates on these figures by displaying the angular velocity and target signals over time, to substantiate the interpretation of the results in Figure 4.9.

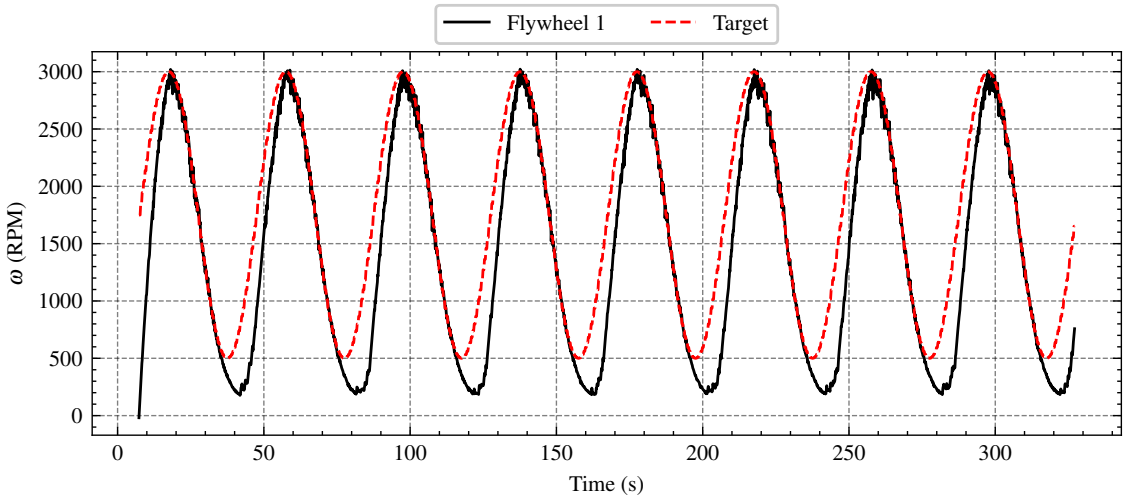


(a)

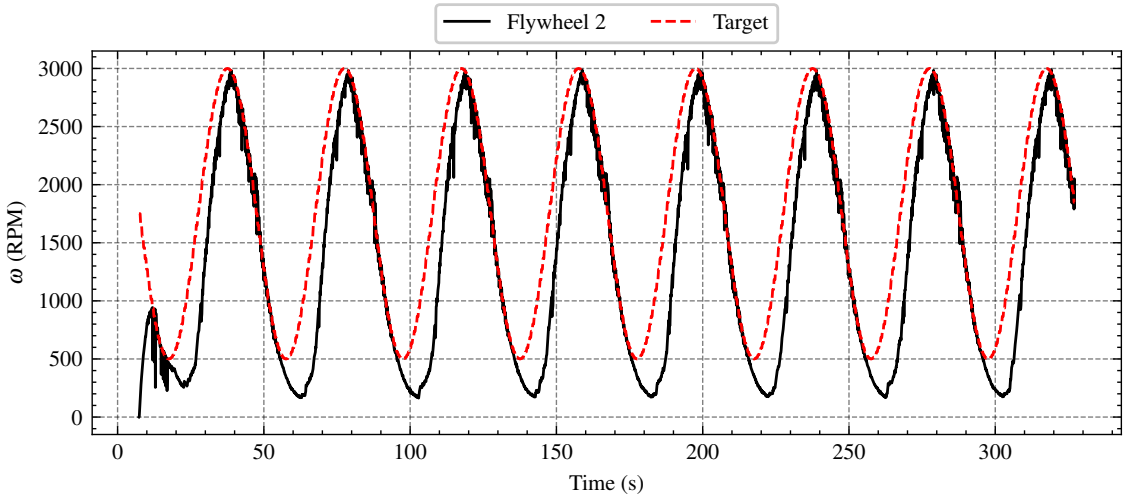


(b)

Figure A.1: Flywheel velocity for no regeneration controller when tracking a sinusoidal target signal.

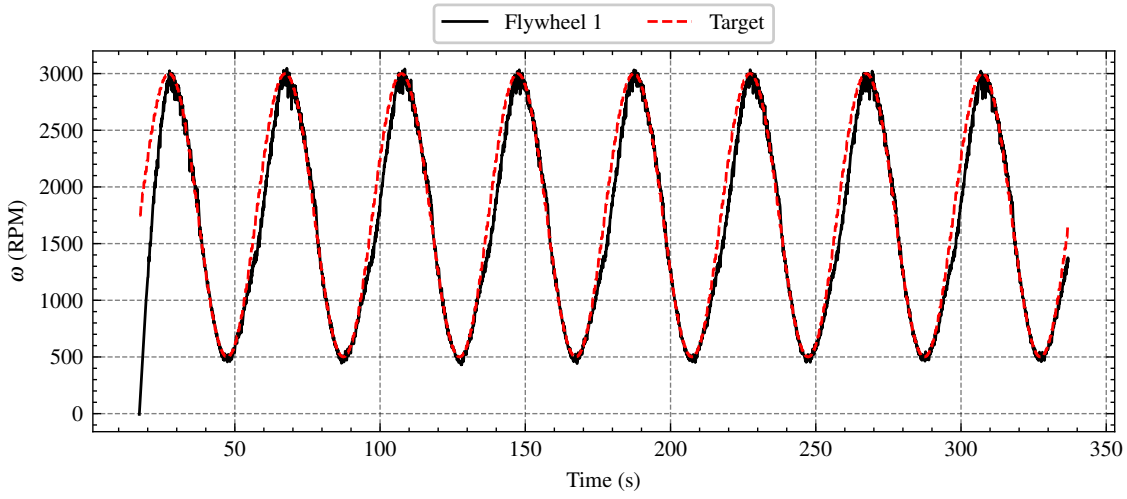


(a)

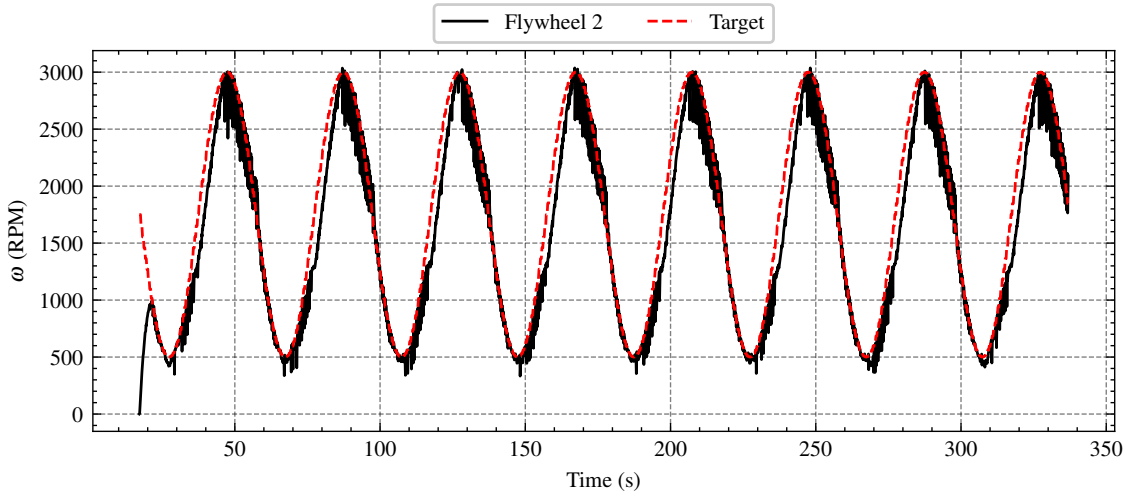


(b)

Figure A.2: Flywheel velocity for C regeneration controller when tracking a sinusoidal target signal.

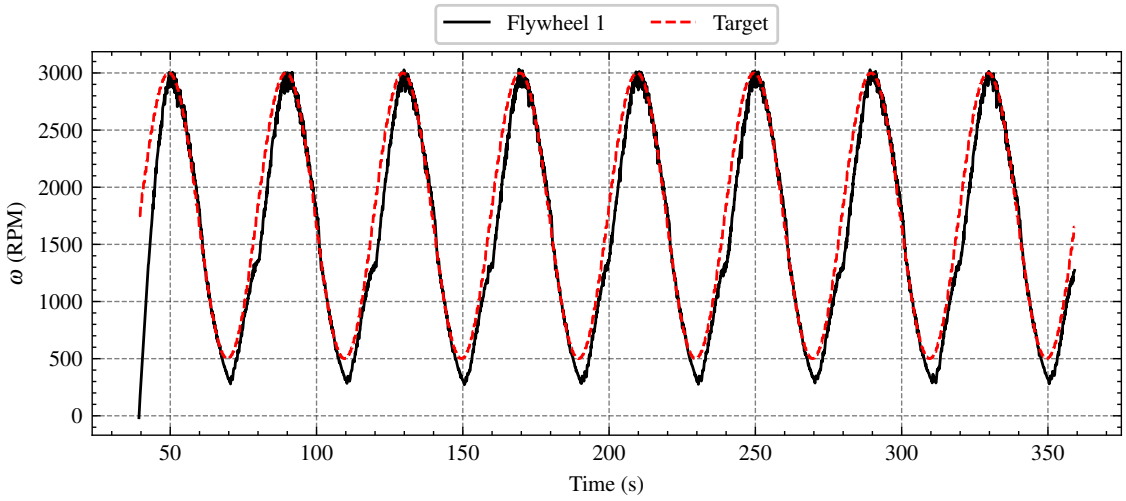


(a)

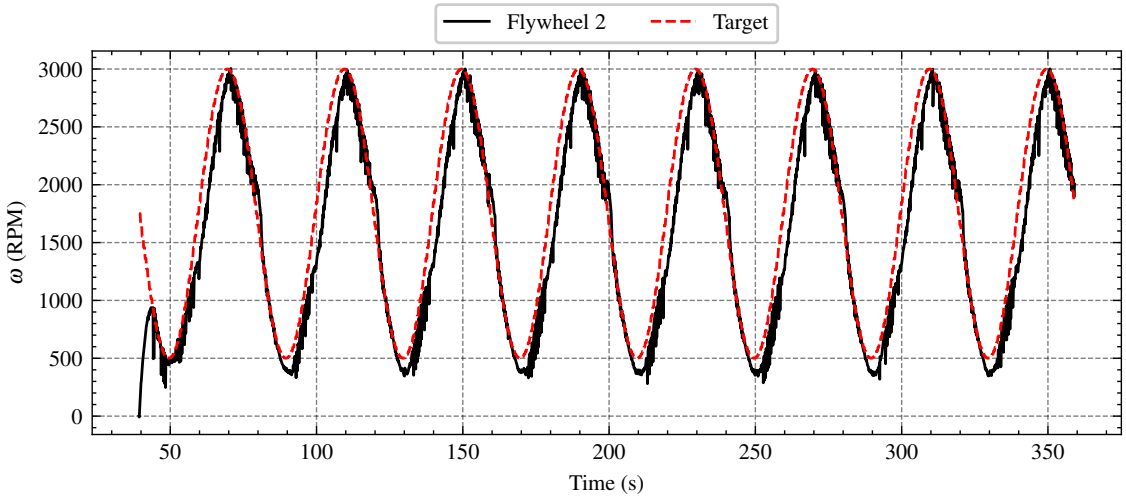


(b)

Figure A.3: Flywheel velocity for M regeneration controller when tracking a sinusoidal target signal.



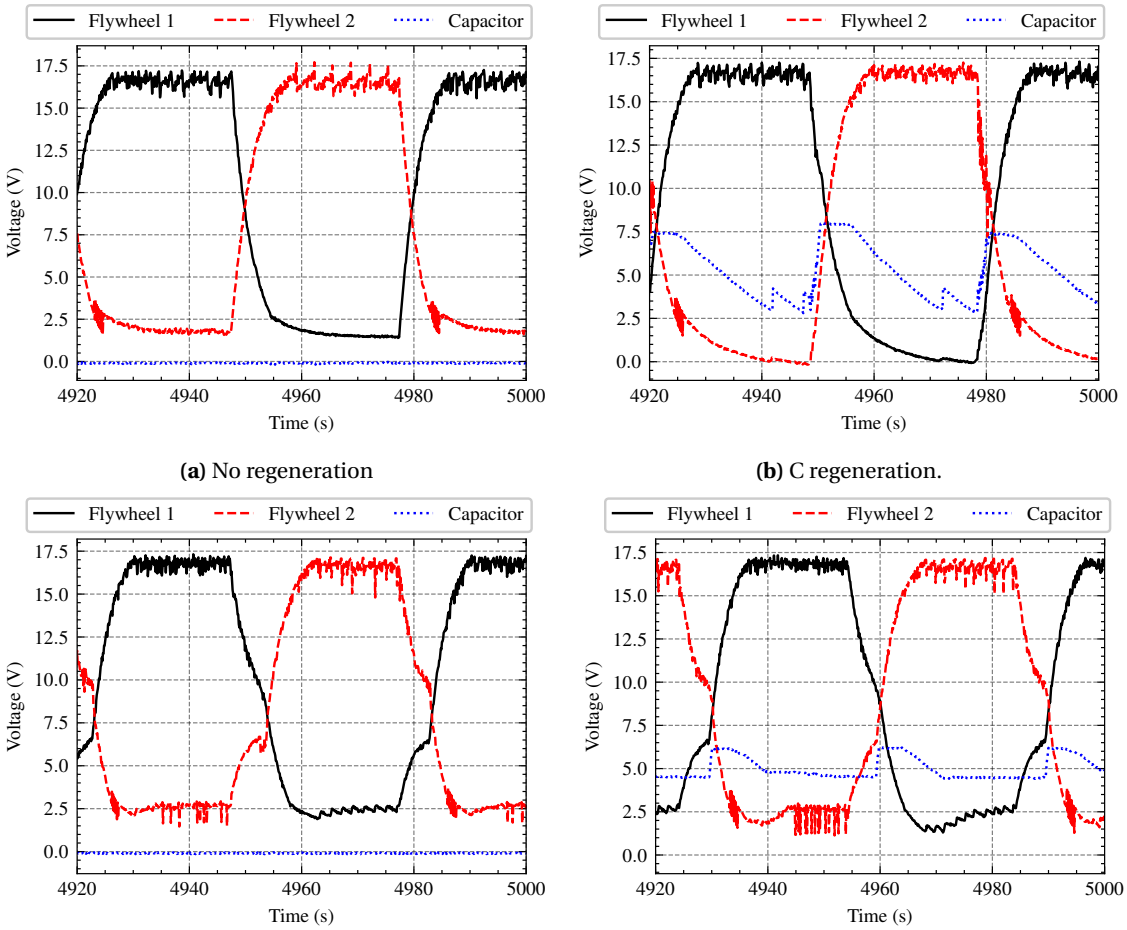
(a)



(b)

Figure A.4: Flywheel velocity for MC regeneration controller when tracking a sinusoidal target signal.

A.2 Voltages for period of 60 s



(a) No regeneration

(b) C regeneration.

(c) M regeneration.

(d) MC regeneration

Figure A.5: Partial slices from Figure 4.10 for a square wave with period 60 s, showing that all controllers that apply regeneration undershoot the target voltage and that this effect is worse for capacitor regeneration than for motor-to-motor regeneration.

Bibliography

[1] N. Barnett, D. Costenaro and I. Rohmund. 'Direct and indirect impacts of robots on future electricity load'. (2017), [Online]. Available: https://www.aceee.org/files/proceedings/2017/data/64395-aceee-1.3687710/t001-1.3687829/f001-1.3687830/a002-1.3687868/0036-0053-000029-1.3687903.html (visited on 22/07/2024).

[2] A. Sękala, T. Blaszczyk, K. Foit and G. Kost, 'Selected issues, methods, and trends in the energy consumption of industrial robots', *Energies*, vol. 17, no. 3, p. 641, Jan. 2024, Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1996-1073. DOI: 10.3390/en17030641. [Online]. Available: https://www.mdpi.com/1996-1073/17/3/641 (visited on 22/07/2024).

[3] P. Khalaf and H. Richter, 'Trajectory optimization of robots with regenerative drive systems: Numerical and experimental results', *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 501–516, Apr. 2020, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2019.2923920. [Online]. Available: https://ieeexplore.ieee.org/document/8760550 (visited on 17/01/2024).

[4] G. Carabin, E. Wehrle and R. Vidoni, 'A review on energy-saving optimization methods for robotic and automatic systems', *Robotics*, vol. 6, no. 4, p. 39, Dec. 2017, Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2218-6581. DOI: 10.3390/robotics6040039. [Online]. Available: https://www.mdpi.com/2218-6581/6/4/39 (visited on 17/01/2024).

[5] D. Meike and L. Ribickis, 'Recuperated energy savings potential and approaches in industrial robotics', in *2011 IEEE International Conference on Automation Science and Engineering*, ISSN: 2161-8089, Aug. 2011, pp. 299–303. DOI: 10.1109/CASE.2011.6042435. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6042435 (visited on 18/01/2024).

[6] A. Adib and R. Dhaouadi, 'Modeling and analysis of a regenerative braking system with a battery-supercapacitor energy storage', in *2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, Apr. 2017, pp. 1–6. DOI: 10.1109/ICMSAO.2017.7934897. [Online]. Available: https://ieeexplore.ieee.org/document/7934897/citations#citations (visited on 29/01/2024).

[7] T. R. Hsu, 'On a flywheel-based regenerative braking system for regenerative energy recovery', Nov. 2013. [Online]. Available: https://arxiv.org/abs/1311.6012v1.

[8] O. V. Egorova and N. N. Barbashov, 'Use of flywheel energy storage in mobile robots', in *Proceedings of the 2020 USCToMM Symposium on Mechanical Systems and Robotics*, P. Larochelle and J. M. McCarthy, Eds., ser. Mechanisms and Machine Science, Cham: Springer International Publishing, 2020, pp. 116–125, ISBN: 978-3-030-43929-3. DOI: 10.1007/978-3-030-43929-3_11.

[9] L. Eitel. 'Regenerative braking on motor-driven axes', DigiKey. (2020), [Online]. Available: https://www.digikey.nl/en/articles/regenerative-braking-on-motor-driven-axes (visited on 16/01/2024).

[10] M. Yoong, Y. Gan, G. Gan *et al.*, 'Studies of regenerative braking in electric vehicle', in *2010 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology*, Nov. 2010, pp. 40–45. DOI: 10.1109/STUDENT.2010.5686984. [Online].

Available: https://ieeexplore.ieee.org/abstract/document/5686984 (visited on 29/01/2024).

[11] Y. Xiao, M. Nemec, L. Borle, V. Sreeram and H. Iu, 'Regenerative braking of series-wound brushed DC electric motors for electric vehicles', in *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, ISSN: 2158-2297, Jul. 2012, pp. 1657–1662. DOI: 10.1109/ICIEA.2012.6360991. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6360991 (visited on 29/01/2024).

[12] S. M. Mousavi G, F. Faraji, A. Majazi and K. Al-Haddad, 'A comprehensive review of flywheel energy storage system technology', *Renewable and Sustainable Energy Reviews*, vol. 67, pp. 477–490, Jan. 2017. DOI: 10.1016/J.RSER.2016.09.060.

[13] R. Takarli, A. Amini, M. Khajueezadeh *et al.*, 'A comprehensive review on flywheel energy storage systems: Survey on electrical machines, power electronics converters, and control systems', *IEEE Access*, vol. 11, pp. 81 224–81 255, 2023, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3301148. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10203021 (visited on 17/01/2024).

[14] H. Ibrahim, A. Ilinca and J. Perron, 'Energy storage systems—characteristics and comparisons', *Renewable and Sustainable Energy Reviews*, vol. 12, no. 5, pp. 1221–1250, 1st Jun. 2008, ISSN: 1364-0321. DOI: 10.1016/j.rser.2007.01.023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032107000238 (visited on 17/01/2024).

[15] S. Choudhury, 'Flywheel energy storage systems: A critical review on technologies, applications, and future prospects', *International Transactions on Electrical Energy Systems*, vol. 31, no. 9, e13024, 2021, ISSN: 2050-7038. DOI: 10.1002/2050-7038.13024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/2050-7038.13024 (visited on 17/01/2024).

[16] T. green age. 'What are gravity batteries? - TheGreenAge'. (2020), [Online]. Available: https://www.thegreenage.co.uk/what-are-gravity-batteries/ (visited on 16/01/2024).

[17] D. Gritzner, E. Knöchelmann, J. Greenyer, K. Eggers, S. Tappe and T. Ortmaier, 'Specifying and synthesizing energy-efficient production system controllers that exploit braking energy recuperation', in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, ISSN: 2161-8089, Aug. 2018, pp. 408–413. DOI: 10.1109/COASE.2018.8560544. [Online]. Available: https://ieeexplore.ieee.org/document/8560544 (visited on 18/01/2024).

[18] A. Tantos. 'H-bridge secrets | modular circuits'. (20th Dec. 2011), [Online]. Available: https://www.modularcircuits.com/blog/articles/h-bridge-secrets/ (visited on 06/08/2024).

[19] Texas Instruments, *LMD18200 3a, 55v h-bridge datasheet*, 1999.

[20] S. J. Ling, J. Sanny and B. Moebs. '14.5: RL circuits', Physics LibreTexts. (1st Nov. 2016), [Online]. Available: https://phys.libretexts.org/Bookshelves/University_Physics/University_Physics_(OpenStax)/University_Physics_II_-_Thermodynamics_Electricity_and_Magnetism_(OpenStax)/14%3A_Inductance/14.05%3A_RL_Circuits (visited on 09/08/2024).

[21] W. Han, T. Wik, A. Kersten, G. Dong and C. Zou, 'Next-generation battery management systems: Dynamic reconfiguration', *IEEE Industrial Electronics Magazine*, vol. 14, no. 4, pp. 20–31, Dec. 2020, Conference Name: IEEE Industrial Electronics Magazine, ISSN: 1941-0115. DOI: 10.1109/MIE.2020.3002486. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9300283 (visited on 30/01/2024).

[22] M. Jahanshahi and F. Bistouni, *Crossbar-Based Interconnection Networks* (Computer Communications and Networks). Cham: Springer International Publishing, 2018, ISBN: 978-3-319-78473-1. [Online]. Available: http://link.springer.com/10.1007/978-3-319-78473-1 (visited on 09/02/2024).

[23] A. Boutros and V. Betz, 'FPGA architecture: Principles and progression', *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 4–29, 2021, Conference Name: IEEE Circuits and Systems Magazine, ISSN: 1558-0830. DOI: 10.1109/MCAS.2021.3071607. [Online]. Available: https://ieeexplore.ieee.org/document/9439568 (visited on 23/07/2024).

[24] J. Duato, S. Yalamanchili and L. Ni, *Interconnection Networks*. Morgan Kaufmann, 2003, 626 pp., Google-Books-ID: ZLoAAOMJ3egC, ISBN: 978-1-55860-852-8.

[25] Keithley, *Switching Handbook - A Guide to Signal Switching in Automated Test Systems*, 6th ed. 2008. [Online]. Available: https://www.tek.com/en/documents/product-article/switching-handbook (visited on 02/02/2024).

[26] F. K. Hwang, 'Rearrangeability of multi-connection three-stage clos networks', *Networks*, vol. 2, no. 4, pp. 301–306, 1972, ISSN: 1097-0037. DOI: 10.1002/net.3230020403. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230020403 (visited on 09/02/2024).

[27] Texas Instruments, *LM2577 simple switcher step-up voltage regulator*, 1999.

[28] KEMET Corporation, *Introduction to capacitor technologies*, 2013. [Online]. Available: https://www.mouser.com/pdfDocs/eb1001_what_is_a_capacitor.pdf.

[29] Newport Corporation. 'M-SA2-22 optical breadboard plate'. (), [Online]. Available: https://www.newport.com/p/M-SA2-22 (visited on 28/03/2024).

[30] XP Power. 'VEH150 series power product range', XP Power. (), [Online]. Available: https://www.xppower.com/product/VEH150-Series (visited on 29/04/2024).

[31] Maxon, 'Maxon DC motor f 2140', 2006. [Online]. Available: https://il.farnell.com/maxon-motor/2140-934-61-112-050/motor-12vdc-4090rpm/dp/1761260.

[32] Mellor Electrics Ltd. 'RS555 | mellor electric brushed DC motor, 24 v dc, 10 ncm, 3500 rpm | RS'. (), [Online]. Available: https://nl.rs-online.com/web/p/dc-motors/2483671 (visited on 18/03/2024).

[33] Omron. 'Overview of solid-state relays | OMRON industrial automation'. (), [Online]. Available: https://www.ia.omron.com/support/guide/18/introduction.html (visited on 16/07/2024).

[34] Toshiba, *Photocouplers photorelay TLP241a,TLP241af*, 2023.

[35] Arduino. 'Arduino due', Arduino Official Store. (), [Online]. Available: https://store.arduino.cc/products/arduino-due (visited on 07/08/2024).

[36] Arduino. 'Arduino uno rev3', Arduino Official Store. (), [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3 (visited on 07/08/2024).

[37] Espressif Systems. 'ESP32-DevKitC v4 getting started guide'. (), [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/hw-reference/esp32/get-started-devkitc.html (visited on 07/08/2024).

[38] Raspberry Pi Ltd. 'Raspberry pi pico datasheet'. (2021), [Online]. Available: https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html (visited on 23/07/2024).

[39] PlatformIO Labs. 'PlatformIO IDE', PlatformIO. (), [Online]. Available: https://platformio.org (visited on 07/08/2024).

[40] I. Singh. 'General purpose i/o: How to get more', Hackaday. (14th Jun. 2018), [Online]. Available: https://hackaday.com/2018/06/14/general-purpose-i-o-how-to-get-more/ (visited on 13/03/2024).

[41] Microchip Technology Inc., *I/o expander MCP23017-e/SP*, 2007.

[42] *DC-DC converter DEBO DCDC UP 3*. [Online]. Available: https://www.reichelt.nl/nl/nl/ontwikkelaarspanelen-spanningsomvormer-dc-dc-lm2577-lm2596s-debo-dcdc-up-3-p282582.html.

[43] Free Software Foundation Inc. 'KiCad EDA'. (2024), [Online]. Available: https://www.kicad.org/ (visited on 17/07/2024).

[44] Texas Instruments, *I2c bus pullup resistor calculation*, 2015. [Online]. Available: https://www.ti.com/lit/an/slva689/slva689.pdf.

[45] Microchip Technology Inc., *Digital potentiometer 5k, MCP4261-502e/p*, 2008.

[46] Rohde and Schwarz, *R&s RTB2000 OSCILLOSCOPE datasheet*, 2023.

[47] T. Bray, 'The JavaScript object notation (JSON) data interchange format', Internet Engineering Task Force, Request for Comments RFC 8259, Dec. 2017, Num Pages: 16. DOI: 10.17487/RFC8259. [Online]. Available: https://datatracker.ietf.org/doc/rfc8259 (visited on 23/07/2024).

[48] Keysight. '54603b 2-channel 60 MHz oscilloscope', Keysight. Section: Article Section. (), [Online]. Available: https://www.keysight.com/gb/en/product/54603B/2channel-60-mhz-oscilloscope.html (visited on 29/07/2024).

[49] Keysight. '34401a digital multimeter, 6½ digit', Keysight. Section: Article Section. (), [Online]. Available: https://www.keysight.com/gb/en/product/34401A/digital-multimeter-6-digit.html (visited on 25/07/2024).

[50] Testo. 'Testo 470 tachometer', Testo. (), [Online]. Available: https://www.testo.com/en-US/testo-470/p/0563-0470 (visited on 27/07/2024).

[51] Adafruit. 'Adafruit motor shield v2', Adafruit. (2011), [Online]. Available: https://www.adafruit.com/product/1438 (visited on 26/01/2024).

[52] Delta Elektronika, *Voedingsapparaat d 015_1.5*.

[53] Berkley. 'X9 braid filler spool', Berkley Fishing. (), [Online]. Available: https://www.berkley-fishing.com/products/x9-braid-filler-spool (visited on 20/08/2024).

[54] Rohde and Schwarz. 'R&s HM8118 LCR bridge/meter'. (), [Online]. Available: https://www.rohde-schwarz.com/products/test-and-measurement/rs-essentials-meters-and-analyzers/rs-hm8118-lcr-bridge-meter_63493-44101.html (visited on 25/07/2024).

[55] Adafruit. 'Adafruit PDM microphone breakout'. (), [Online]. Available: https://learn.adafruit.com/adafruit-pdm-microphone-breakout/overview (visited on 12/08/2024).

[56] B. M. Palaniappan Sampath. 'Introduction to gate drivers for power electronics', The Talema Group. (22nd Oct. 2018), [Online]. Available: https://talema.com/gate-driver-introduction/ (visited on 11/08/2024).

[57] NXP. 'PCA9685'. (2015), [Online]. Available: https://www.nxp.com/products/power-management/lighting-driver-and-controller-ics/led-controllers/16-channel-12-bit-pwm-fm-plus-ic-bus-led-controller:PCA9685 (visited on 11/08/2024).