Electric Discharge Machining

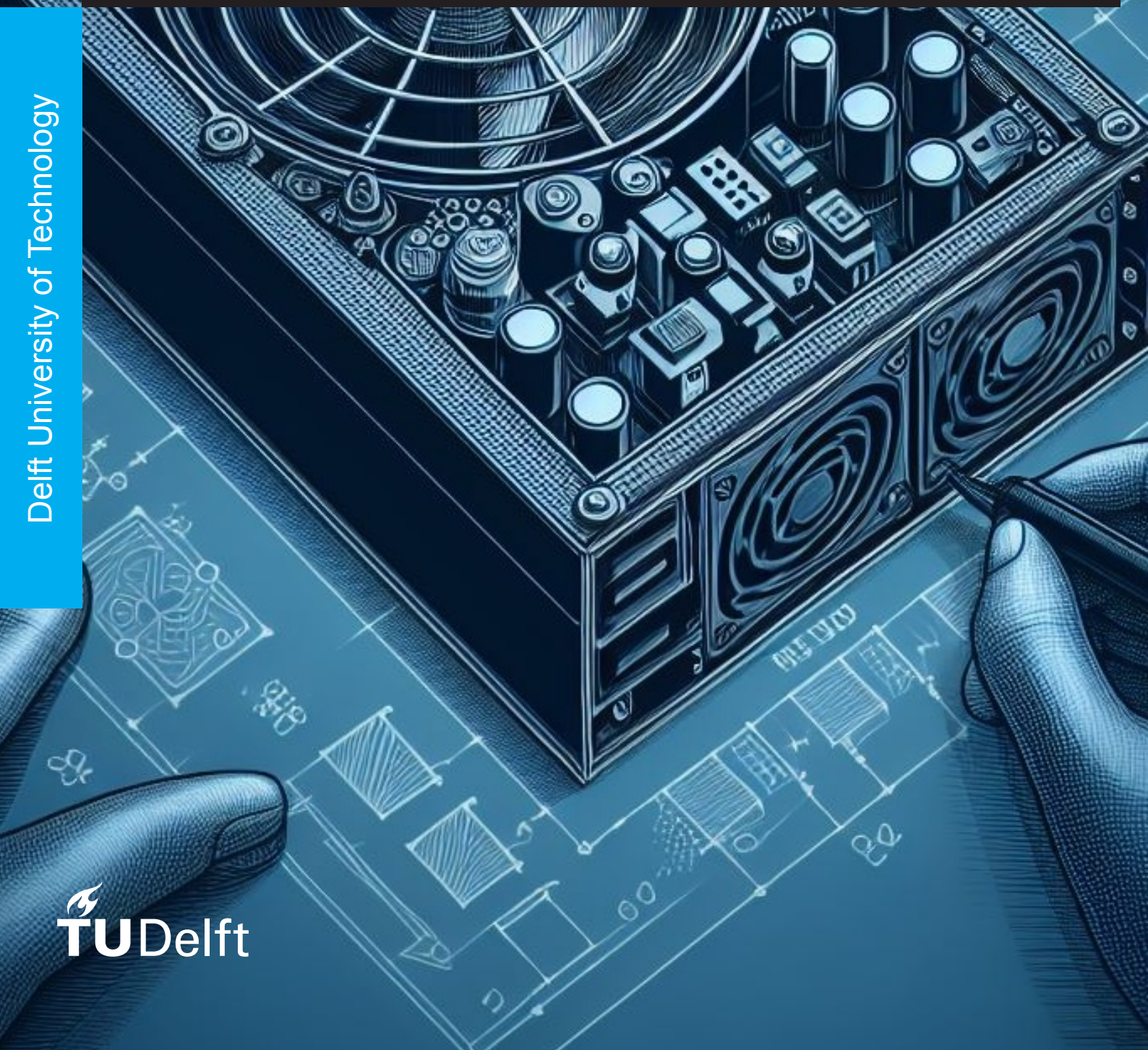
Design of a Power Supply

Bachelor graduation project electrical engineering

Tim van den Akker
Bart Pieper
Joey Cheung

July 6, 2024

Delft University of Technology



TU Delft

This page is intentionally left blank.

Electric Discharge Machining

Design of a Power Supply

Bachelor graduation project electrical engineering

by

Tim van den Akker
Bart Pieper
Joey Cheung

Date:	July 6, 2024
Supervisor:	Mohamad G. Niasar
Advisor:	Hani Vahedi
Thesis committee:	Ioan E. Lager, Mohamad G. Niasar, Hani Vahedi
Thesis defense:	June 25, 2024 13:30
Faculty:	TU Delft Faculty of Electrical Engineering, Mathematics & Computer Science

Abstract

The goal of this project is to create a power supply for an electrical discharge machine (EDM). This machine can drill holes and create shapes in metal objects by evaporating material using discharges from the electrode to the workpiece. The electrode should be supplied with a high-voltage pulse wave in order to initate sparks.

The power supply produces this waveform by turning a DC source on and off using a transistor as a switch. A current-limiting resistor that is placed between the source of the MOSFET and the electrode prevents short circuits during discharges.

The circuit was first created with a DC voltage of 15V. A gate driver circuit to drive this MOSFET was designed to reduce power losses and to make the frequency and duty cycle adjustable. Then the design was improved by adding a diode to remove oscillations caused by the parasitic inductance of the power resistor. Afterwards, the circuit was built on a custom PCB to reduce parasitics. A filtering capacitor was added to remove high-frequency noise, after which a stable 15V pulse signal was obtained.

A microcontroller is employed to regulate the pulse duration, detect spark occurrences, and relay this information to a speed regulator. This establishes a closed-loop system aimed at optimizing the overall system operation. To achieve this, circuitry is employed to convert the power supply output into a digital format, enabling the utilization of the microcontroller's specialized hardware for rapid sampling.

Preface and acknowledgements

Over the span of two months, we, three bachelor students majoring in electrical engineering, delved into the extensive but very interesting realm of power electronics to construct a power supply. This design process has proven to be a truly enjoyable project and, more importantly, a highly educational journey. Starting from scratch to developing a theoretical design, to finally having a physical prototype in our hands.

This project would not have progressed so smoothly without the support of our supervisor, Mohamad Ghaffarian Niasar. Additionally, we extend our sincere appreciation to Hani Vahedi for his assistance with power electronics designs, and to Dennis van der Born for sharing his knowledge of high-voltage applications. Of course, we also want to express our gratitude to the ESP technicians, Imke Splinter and Wim Termorshuizen, for providing support and a safe testing environment for high voltages. Also to the technicians from the Tellgenhal, Ton Slats and Martin Schumacher we want to give thanks, just as the Bachelor Graduation Project coordinator Ioan Lager for providing clear information about the project.

Finally, we are very thankful for our families having supported us in the previous three years of the bachelor, and of course, also for the other moments in our life. Without them, we could not have been where we are at the moment.

Tim van den Akker, Bart Pieper and Joey Cheung
June, 2024

Contents

1 Introduction

Electrical discharge machining (EDM), also known as spark machining, is an unconventional method of machining that uses thermal energy to process electrically conductive materials. The fundamental principle involves generating tiny electric discharges, eroding away at the conductive material regardless of its hardness. This erosion occurs due to the creation of thermal energy, which forms a channel of plasma (state of matter consisting of electrons and ions and therefore conducting) with temperatures reaching around 10,000 degrees Celsius [1, 2]. A visual representation of this process is illustrated in Figure 1.1. By precisely controlling these sparks at a high frequency, EDM serves as a viable alternative to traditional milling and drilling techniques, and it is often used in the manufacturing of molds and dies, electronic devices, automotive parts and surgical instruments for its preciseness.

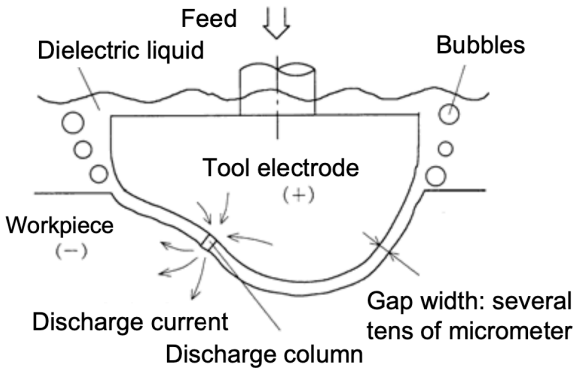


Figure 1.1: *Basic concept of electrical discharge machining [3]*

1.1. Working principle: creation of one spark

In order to understand the made choices and encountered challenges in the process of designing the power supply, it is of great help to have a deeper understanding of the sparks.
When a sufficiently large voltage is applied between the electrode and workpiece, it can break the fluid down electrically in charged fragments (ions). Due to this breakdown, the fluid locally loses its insulating properties, and as a consequence allowing current to flow. Since the fluid is constantly refreshed by means of a water pump, the conducting path between the electrode and workpiece is removed again. In practice, this whole process is observed as a spark, which lasts within fractions of a second.
In Figure 1.2, the different steps of the process are shown, including the corresponding gap voltage and current.

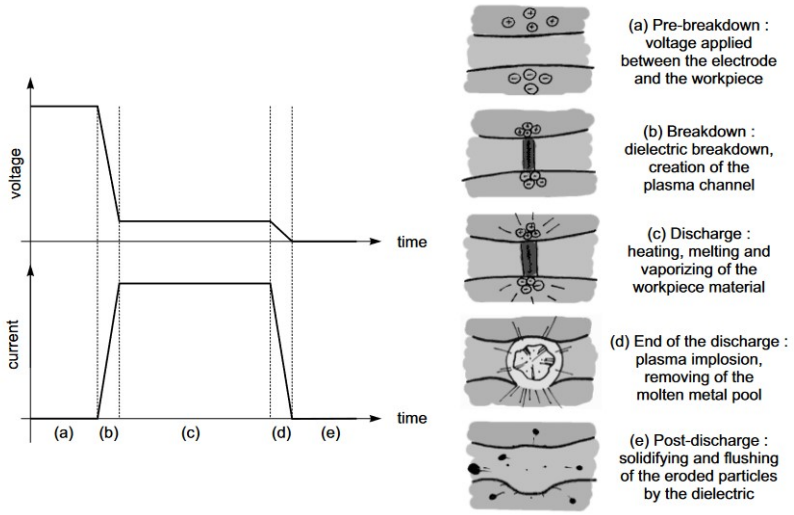


Figure 1.2: *The steps of the spark creation process in EDM [4]*

The basic principle behind EDM is that electrical energy gets converted into thermal energy (heat), which erodes some of the electrode and workpiece away. The exact physics behind this process is not fully understood yet. However, there is much evidence that substantiates the thermoelectric model [5].

The different stages of the process are as follows:

- The first stage is the pre-breakdown phase, where a fixed voltage, usually around 200V, is held across the electrode and workpiece. The reason why the breakdown doesn't happen immediately is because some statistical time lag, also called the "ignition delay time", is needed physically so that the intense electric field can dissociate the molecules and ionize the atoms in the fluid. Additionally, in the case of tap water as liquid, electrolysis takes place where the gases hydrogen and oxygen are formed. The duration of this period is of stochastic nature and thus varies for each discharge given one set of pulse conditions.
- In the next stage, the actual breakdown occurs. The insulating properties of the fluid have decreased due to the presence of ions and electrons, resulting in a plasma channel. As a result, a current rises quickly, and the voltage drops until the so called breakdown voltage, which is usually about 20V.
- In the discharge stage, the current has reached its maximum and remains constant, assuring the continuous bombardment of ions and electrons to the metals. This is where the plasma channel becomes larger and the temperature has become so high that parts of the two metals are vaporized, resulting in a metal pool at the surface of the metals. This is the first part of the material removal mechanism. Additionally, the voltage remains at the breakdown level.
- After this is the end-of-discharge stage where the voltage source is turned off and so the current decreases. The plasma channel implodes due to the pressure of the fluid, and this implosion results in a small crater in the workpiece. The implosion is the second and also dominant part of the material removal mechanism.
- In the post-discharge stage where the gap voltage and current are zero, all the particles have solidified and many are flushed away so that an insulating path is restored between the electrode

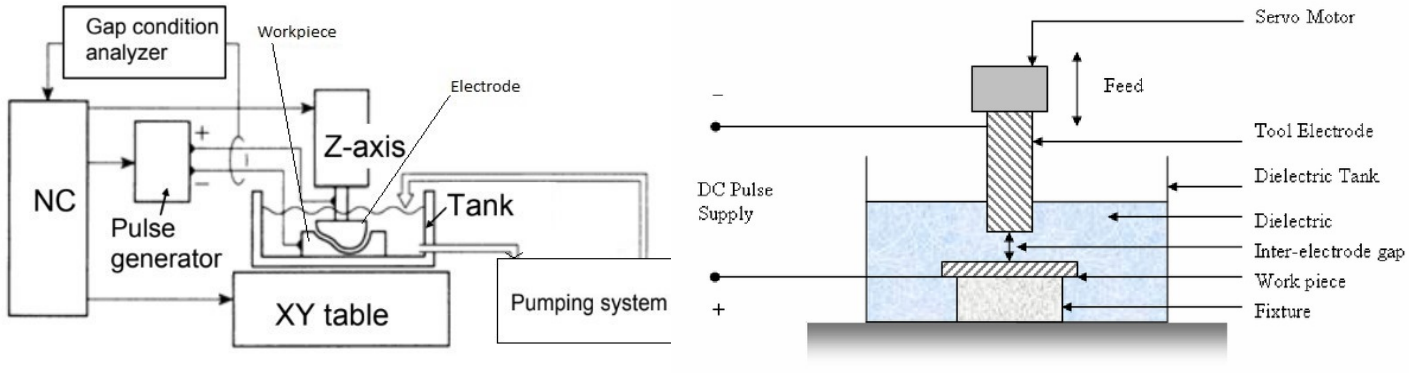
and workpiece for the next discharge cycle to happen. However, some particles may not be flushed away and will form a so called recast layer on the electrode and workpiece. The gases hydrogen and methane, which are generated by the dissociation are observed as bubbles [4].

1.2. Project objective

In order to gain insight into the working principles of EDMs, a Bachelor Graduation Project for Electrical Engineering students has been proposed. For this project, a group of six students was composed and divided into two subgroups. The first subgroup focuses on the intrinsics of the electrode, dielectric fluid and kinematics part of the system, detailed in [6], while the second subgroup explores the design of the power supply for an EDM. In this report, the design choices and achieved results concerning the power supply will be presented that enables the other subgroup to achieve a successful development of a complete EDM system.

1.3. Setup

As explained before, in EDM an electrode and metal workpiece are put closely to each other. A potential difference is put across the two metals and they are immersed into a dielectric fluid. The type of EDM is opted for in this project is called a sinker-EDM where a thin metal rod is used as electrode. In the other widely-used type called wire EDM, a thin wire is used as electrode.
Below, in Figure 6.13, the setup of the whole EDM system in the project is shown. Apart from the pulse generator, there is the a feedback system that manages the width of the gap by moving the electrode nearer or farther away from the workpiece. The flushing is done by means of a pumping system. NC stands for numerical control where the 3D printer and closed-loop control elements are included. The power supply group is responsible for the pulse generator, gap condition analyzer (sensor) and part of the closed-loop control. These subsystems will be highlighted more in the project decomposition (see Section 1.6). Note that this figure is only used to illustrate the whole system, and that the electrode is not representative in our project. Figure 1.3b does illustrate how the electrode and workpiece are set up.



(a) *The setup of our die-sinker EDM system [3]*

(b) *Close-up of the electrode and workpiece [7]*

Figure 1.3: *The project's setup*

1.4. Performance metrics

In EDM, there are three performance metrics, which are considered as the most important: machining speed, precision and electrode durability. These are reflected in the following technical terms:

- Material Removal Rate (MMR)
- Surface quality
- Tool Wear Ratio (TWR)

These parameters are controlled by the pulse conditions: duration, frequency and discharge peak current, and they are all related to each other. I.e., it is impossible to optimize all three of these parameters simultaneously. There are three possible scenarios:

1. Pulses with relatively small duration and high peak current result in a moderate MMR and good surface quality. However, the TWR will be higher due to the fact that there is not much time to form a thick recast layer on the electrode. This setting is used in wire EDM because electrode wear is less of an issue there.
2. Pulses with relatively large duration and large peak current result in a higher MMR because of the large peak current that creates large craters. In addition, TWR is now small because more time is allowed to form a thicker recast layer on the electrode. However, due to the large craters, the surface quality is poor. This setting is used in rough machining.
3. Pulses with relatively large duration but small peak current result in a better surface quality and lower TWR at the cost of a lower MMR. This setting is used for finishing

The scenarios are depicted in Figure 1.4. Note that in all cases, increasing the pulse frequency results in a higher MMR.

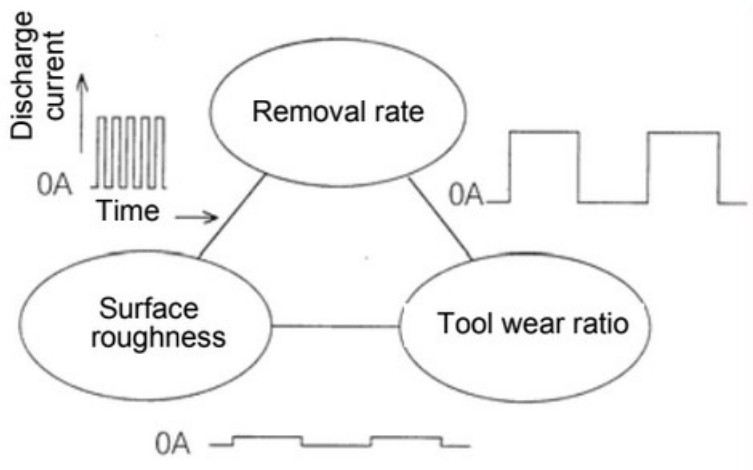


Figure 1.4: *Relation between discharge current pulses and machining characteristics [3]*

Depending on the EDM application, an appropriate combination of these metrics can be obtained so that the machining performance is optimized. This is one of the goals the other subgroup is aiming for [6].

1.5. Thesis outline

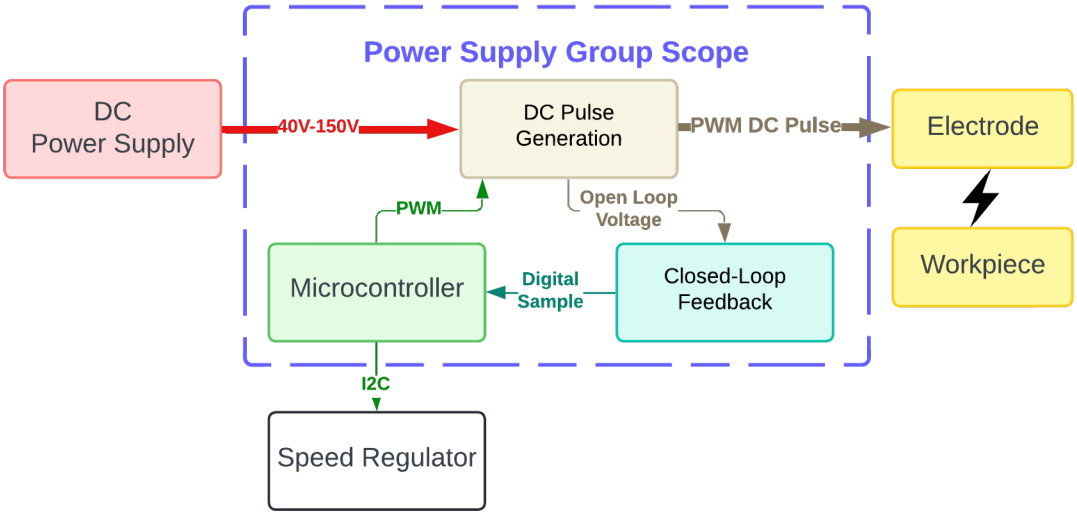


Figure 1.5: *Compact overview of the project scope for the power supply group*

The Power Supply Scope can be subdivided into multiple subsytems as visible in Figure 1.5. The outline of the thesis is as follows. In Chapter 2, the Programme of Requirements are presented. Then, in Chapter 3, the selected power supply topology and DC Pulse Generation will be discussed. This is followed by Chapter 4 where the aspects of the microcontroller are explained. In Chapter 5, the closed-loop feedback system will be elaborated on. Then, in Chapter 6, the experimental results at high voltage will be presented, along with the results obtained after integration with the other subgroup. Finally, In Chapter 7, a summary and conclusion of the results will be given, along with a discussion of the project and suggestions for future work.

In the appendix, information regarding the PCB can be found as well as the code used to program the Raspberry Pi Pico.

1.6. Project decomposition

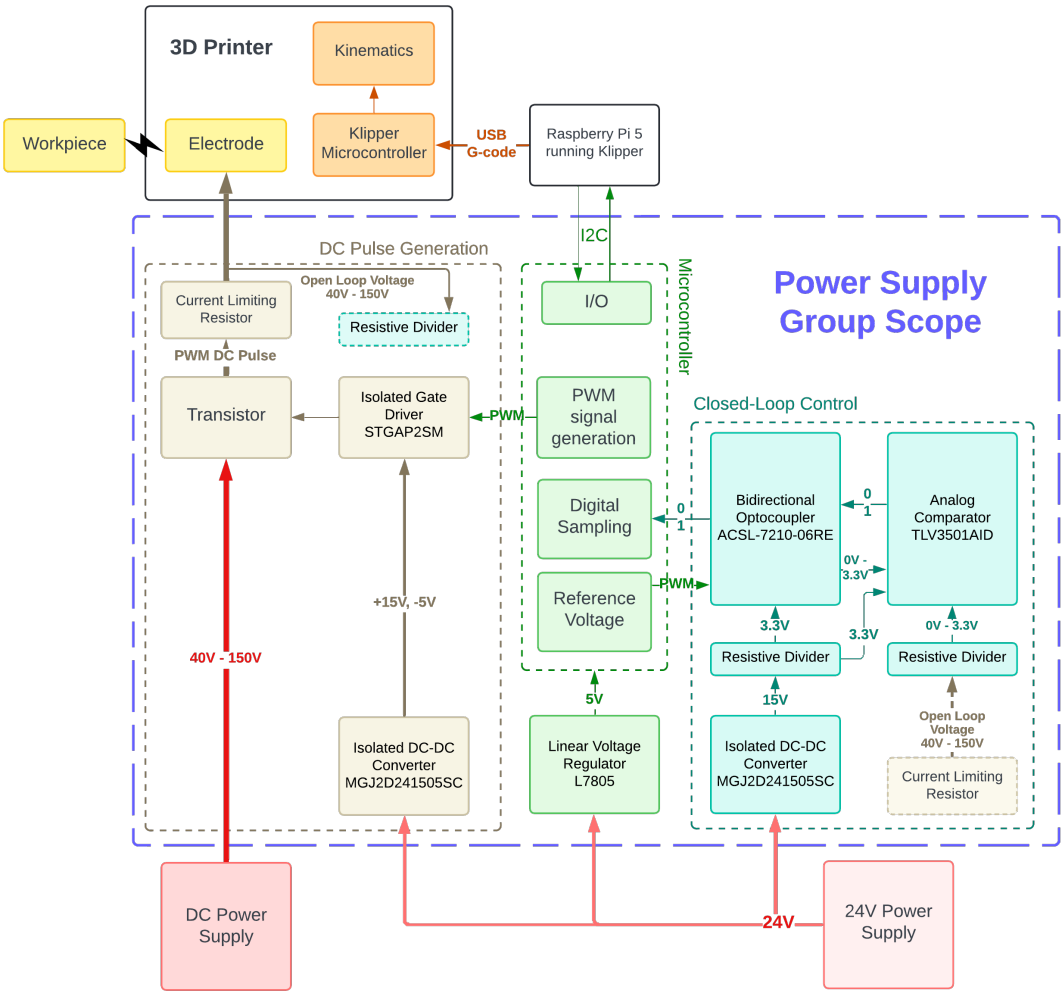


Figure 1.6: *Detailed diagram of the full project*

The full scope of the power supply design consists of multiple subsystems, which can be seen in Figure 1.6.

DC pulse generation This subsystem's goal is to generate and supply to the electrode DC pulses. It is powered by an external DC power supply.

Microcontroller As the previous subsystem is transistor-based, it needs to receive a PWM signal, which is generated by the microcontroller subsystem. It will also control the speed regulator of the electrode so that an approximately constant gap width can be maintained where communication is done by means of the I2C protocol. In order to do so, the voltage across the current-limiting resistor is sensed and given as input to the microcontroller.

Closed-loop feedback This subsystem focuses on the hardware related to digitizing the voltage across the current-limiting resistor for the microcontroller.

2
Programme of requirements

2.1. Power supply requirements

As previously mentioned, the primary objective of this subgroup is to develop a power supply. Achieving this necessitates thorough research into the specific functionalities required of the power supply. These requirements serve as the cornerstone for the design process, guiding the development of a power supply capable of meeting the desired objectives.

2.1.1. Range of parameters

In order to devise an appropriate power supply capable of effectively driving an EDM, it is imperative to establish some rough estimates. These estimates provide insights into the required voltage, frequency, and current ranges from the power source. This assessment is based on collecting data from various sources such as research papers [8–13], as summarized in Table 2.1. Given that the power supply is intended for die-sinking EDM, also known as ”macro EDM,” it is important to note that higher currents and lower frequencies are typically required compared to “micro EDM” applications [14].

Table 2.1: *Range of parameters found in literature review*

voltage (no load)	40 - 200 V
max current	1 - 14 A
duty cycle	0.1 - 0.8
t_{on}	5 - 50 µs
t_{off}	10 - 100 µs
frequency	2 - 100kHz

2.1.2. Decided requirements tiers

To anticipate potential project setbacks, a stratified approach is adopted with four distinct ”levels,” wherein the requirements of each level must be fulfilled before progressing to the subsequent one. The initial tier, denoted as ”fixed output,” prioritizes the establishment of a functional power supply with predetermined power specifications. At this stage, the focus is on ensuring the power supply operates with essential safety features, albeit possibly not at optimal electrical parameters. The requirements for this tier are detailed in Table 2.2, with fixed values chosen from a reasonable range outlined in Subsection 2.1.1.

Validation of these requirements incorporates methodologies such as analysis, inspection, demonstration, or testing, adhering to the NASA product verification model [15].

Table 2.2: *Fixed output requirements*

Fixed output requirements			Validation			
Req. ID	**Req. statement**	**Value**	**Ana.**	**Insp**	**Demo**	**Test**
EDM_PS_F_1	The power supply shall have a on/off switch	-	✗	✗	✓	✗
EDM_PS_F_2	The power supply shall have fixed output voltage	100V	✓	✗	✓	✗
EDM_PS_F_3	The power supply shall have limited output current	5A	✓	✗	✓	✗
EDM_PS_F_4	The power supply shall have fixed frequency	10kHz	✓	✗	✓	✗
EDM_PS_F_5	The power supply shall have fixed pulse width	20µs	✓	✗	✓	✗
EDM_PS_F_6	The power supply shall have short circuit detection	-	✓	✗	✗	✓
EDM_PS_F_7	The power supply shall have emergency shutoff switch	-	✓	✗	✓	✗
EDM_PS_ST_1	The power supply shall have proper grounding	-	✓	✓	✗	✗
EDM_PS_ST_2	The power supply shall be build as a PCB	-	✗	✓	✗	✗

Once the requirements for the fundamental fixed output power supply are fulfilled, the subsequent phase involves the ability to adjust electrical parameters. This capability holds significant importance in optimizing performance, considering the varying behavior of materials and electrodes at different parameter settings for different materials. Facilitating the flexibility for electrode and dielectric adjustment within this project necessitates the ability to swiftly and easily modify these parameters. The preferred approach involves employing transistors to regulate the primary electrical parameters for the power supply, as elaborated in Subsection 2.1.1. These transistors, along with other systems, can be controlled using a microcontroller [16]. Additionally, given the heightened technical complexity of this power supply version, documentation outlining the operation and reprogramming of the microcontroller to adjust parameters is indispensable. The complete requirements for this iteration of the power supply are outlined in Table 2.3.

Table 2.3: *Microcontroller controlled requirements*

Microcontroller controlled requirements			Validation			
Req. ID	**Req. statement**	**Value**	**Ana.**	**Insp**	**Demo**	**Test**
EDM_PS_F_8	The power supply shall have adjustable output voltage	40 - 150V	✓	✗	✗	✓
EDM_PS_F_9	The power supply shall have adjustable output current	0A - 15A	✓	✗	✗	✓
EDM_PS_F_10	The power supply shall have adjustable pulse frequency	1 kHz - 50kHz	✓	✗	✗	✓
EDM_PS_F_11	The power supply shall have adjustable pulse width	2 µs - 250 µs	✓	✗	✗	✓
EDM_PS_P_1	The power supply shall have stable good pulse stability	-	✓	✗	✗	✓
EDM_PS_ST_3	The power supply shall have a programmable microcontroller	-	✗	✓	✗	✗
EDM_PS_SPEC_1	The power supply shall have documentation for its operation	-	✗	✓	✗	✗

Earlier iterations of the power supply simply deliver pulses without considering the material being cut. By analyzing the behavior of the current spike, it is possible to determine the distance between the electrode and the material being cut. This version of the power supply communicates this crucial information to the servo, enabling it to adjust the electrode precisely to maintain the optimal gap width between the electrode and the workpiece. Such precision can notably enhance the material removal rate [8].

Table 2.4: *External Feedback Control Requirements*

Feedback control requirements			Validation			
Req. ID	**Req. statement**	**Value**	**Ana.**	**Insp**	**Demo**	**Test**
EDM_PS_ST_4	The power supply shall have a feedback loop with movement team	-	✓	✗	✓	✗
EDM_PS_P_2	The power supply shall have monitoring and logging of voltage, current and frequency	-	✓	✗	✓	✗

In this iteration, the power supply is designed to accommodate G-code, ensuring full compatibility with systems like a retrofitted 3D printer. This integration facilitates fully automated toolpaths for the EDM process. Electric parameters can be programmed via code, and the power supply can communicate essential information with the main control board. However, it is worth noting that this version of the power supply is considered an additional feature, to be addressed only if previous iterations encounter no issues. It is primarily seen as an optional enhancement. Requirements for this feature can be found in Table 2.5.

Table 2.5: *G-Code control requirements*

G-Code control requirements			Validation			
Req. ID	**Req. statement**	**Value**	**Ana.**	**Insp**	**Demo**	**Test**
EDM_PS_ST_5	The power supply shall have G-code Compatibility	-	✓	✗	✗	✓

3
Circuit topology

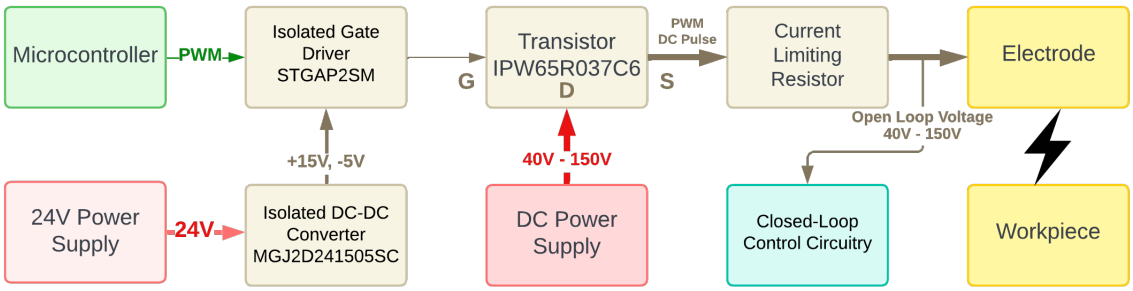


Figure 3.1: *Schematic overview of DC pulse generation*

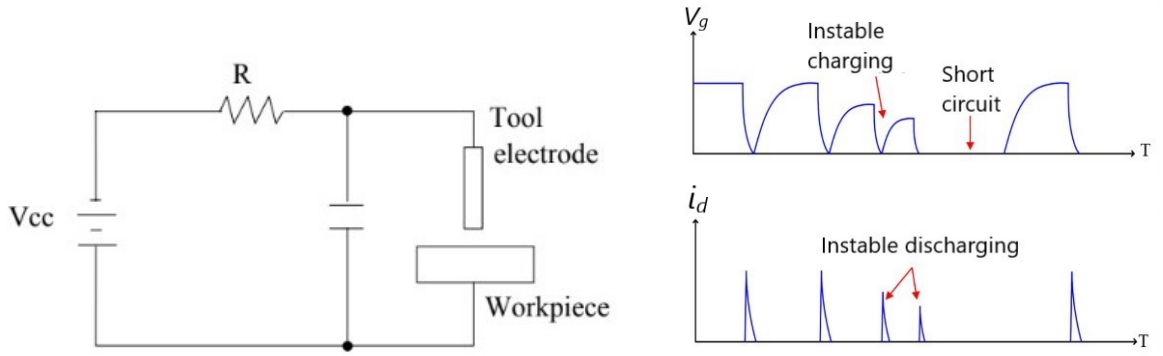
This chapter discusses the choice and analysis for the DC pulse generation, with a schematic view provided in Figure 3.1. First, the choice of the power supply type will be justified, followed by the detailed explanation of each component. Finally, the obtained results and the analysis will be presented.

3.1. Different topologies

Since the first developed EDM in the mid 1900's, many different power supply configurations have been developed. However, there are two configurations, which are most widely used and known. In the first configuration, an RC-type power supply delivers the pulses, while the second one is transistor-based where very fast switches are used such as power MOSFETs. In this section, the focus will be put on the main differences between these two types of power supply configurations. Note that the terms "power supply" and "pulse generators" have the same meaning and will be used interchangeably.

3.1.1. RC-based

The first configuration is based on an RC circuit, and was also used in the first EDM in the 1950s. Even though this type of power supply is nowadays also used in combination with a solid-state switch for some added machining flexibility, the pulse generation is still the same as in the case without a switch. The schematic can be seen in Figure 3.2a. Note that the polarity of the voltage source is reversed, such that the tool electrode is at a lower potential than the workpiece. This leads to lower a TWR [17]. The working principle is fairly straight-forward: the capacitor gets charged through the resistor during a certain period, until the capacitor voltage becomes large enough to initiate a dielectric breakdown of the fluid. The resistor serves to limit the current. When a discharge happens, the gap between the electrode and the workpiece temporarily becomes a short circuit. The resistor ensures that the DC power supply is not short-circuited when this happens. The energy stored in the capacitor is released rapidly across the gap where a short current pulse will be transferred from the tool electrode to the workpiece. A typical waveform of the capacitor voltage (also known as gap voltage or open loop voltage) v_g and the discharge current i_d can be seen in Figure 3.2b. As one may notice, Figure 3.2b does not really resemble the waveforms in Figure 1.2. This is true since the waveforms in the latter apply for a switch-based power supply as will be discussed next. However, the physics behind the process remain the same.



(a) *RC-type pulse generator circuit [8]*

(b) *Typical gap voltage and discharge current waveforms with RC-type configuration [18]*

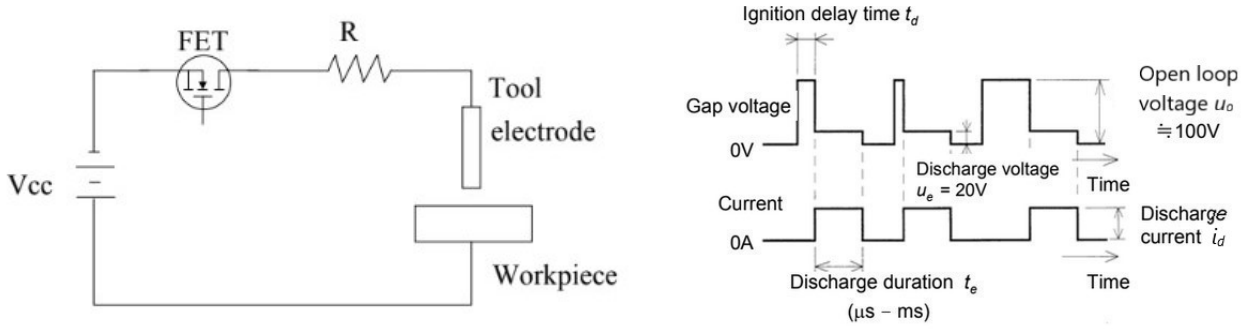
Figure 3.2: *Profile of an RC-type pulse generator*

The performance of the EDM can be adjusted by changing the values of R and C. A higher value for C means more energy per spark, which implies a higher MMR.

However, this leads to a poorer surface finish and a higher TWR. Another disadvantage of the RC-type pulse generator is the low pulse frequency it can achieve due to the relatively long charging time of the capacitor. Also, a uniform surface finish is often difficult to achieve because the discharge energy stored on capacitor varies as can be seen in Figure 3.2b. This is because if a short circuit occurs between the electrode and workpiece, the capacitor will already be discharged, resulting in sparks with different energy each time [8, 18].

3.1.2. Transistor-based

The second configuration is transistor-based, usually using power MOSFETs and is mostly used nowadays because of its higher flexibility, although not superior in some cases where the RC pulse generator is preferred. Many topologies exist but the most basic schematic can be seen in Figure 3.3a. Sometimes multiple switches can be used in parallel to decrease the current going through each one, allowing for the usage of cheaper switches.



(a) *Transistor-type pulse generator circuit (using a FET transistor) [8]*

(b) *Typical gap voltage and discharge current waveforms with transistor-type configuration (note that the discharge duration need not be equal each cycle) [3]*

Figure 3.3: *Profile of a transistor-based pulse generator*

This working principle is as follows. The switch will be turned on and off in a fast and periodic manner.

If it is turned on, a small delay (the ignition delay time of stochastic nature t_d) is needed for the dielectric fluid to become ionized, which happens at the open loop voltage v_o (usually about 100V). After this, the discharge current i_d is transferred, which happens at the discharge voltage, v_e (usually about 20V). Then the switched is turned off so that flushing can take place, allowing the process to start all over again. A typical waveform of the gap voltage and discharge current can be seen in Figure 3.3b.

With this configuration, a more flexible adjustment is possible of the pulse duration (on-time) and discharge current as the switching behaviour can be easily influenced by adjusting the PWM signal applied to the gate (in case of a MOSFET). It is often better at achieving a higher pulse frequency because no capacitor needs to be charged. As a consequence, this leads to a higher MMR. The TWR and surface finishing are not necessarily better. However, the TWR can be improved by optimizing the machining parameters such as pulse duration, frequency and discharge peak current. On the other hand, the surface finishing needs not necessarily to be better than RC-type power supplies. This is due the fact that the discharge duration varies with the ignition delay time, which is of stochastic nature. Due to this varying delay time, which sometimes takes so long, electrical breakdown may be not achieved, wasting an opportunity for a discharge to occur. This leads to a lower discharge frequency as well. However, this well-known problem can be resolved by using a so-called isopulse generator, which is an improved version of the traditional transistor-based pulse generator [3, 19, 20]. As one might already expect, the transistor-based configuration is a bit more complex and requires more design considerations. One can think of controlling the transistor (using a PWM signal), and the side effects of fast switching like voltage spikes and ringing.

3.1.3. Chosen topology

As mentioned before, the RC-type pulse generator is not that flexible as parameters such as discharge duration cannot be controlled easily. In contrast, the transistor-type pulse generator can, while also possibly achieving a higher discharge frequency. Mainly because of the increased versatility, the aim is to build a transistor-type pulse generator, despite the added complexity. If a better surface finishing is desired like in micro-EDM, an isopulse generator can be the goal. However, this is very hard for those who are new to EDM, which is why only the conventional transistor-type pulse generator is now opted for.

3.2. Basic circuit

The most basic version of the circuit in shown in Figure 3.4.

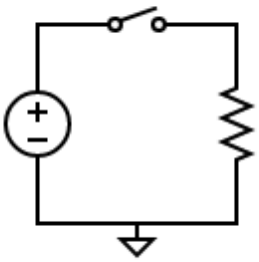


Figure 3.4: *Basic circuit*

The circuit consists of a DC source, a current-limiting resistor and a switch, which is needed to turn the DC supply on and off. The circuit has a high-side switch. The main reason is that the workpiece, in this case the 3D printer, should be grounded to ensure safe operation. As discussed above, this topology

generates the high voltage PWM signal by turning the DC supply on and off. In the following sections all these components will be discussed separately.

3.2.1. Switch

The most important part of the circuit is the switch. There are two main types of switches: BJTs (bipolar junction transistors) and MOSFETs (metal oxide field effect transistors). Out of these two the MOSFET was picked, mainly because it switches a lot faster than a BJT. This is very important since the switch dissipates the most power during switching. By minimizing the switching time, the switching losses will be low. The MOSFET needs to meet some specifications. The main specifications are listed in Table 3.1.

Table 3.1: *Required specifications of the switch*

Quantity Specification	Specification
Maximum drain-source voltage	>200 V
Maximum drain-source current	>20 A
Drain-source resistance	Minimal
Switching time/gate charge	Minimal

The reason why the drain-source resistance (also known as the on-resistance) and switching time should be minimal is to decrease the power losses associated to the MOSFET. There are two kinds of losses: conduction and switching losses. The first is related to the power dissipated by the MOSFET once it is already on. During the on-state, the MOSFET conducts, and its small on-resistance will dissipate power. In addition, the intrinsic body diode of the MOSFET may also conduct current if it is forward-biased, resulting in an additional conduction loss.

The second kind of loss are switching losses that occur when the MOSFET transits from its on to off-state and vice versa. The two losses are given by the following equations respectively:

$$P_{cond} = I_{D,RMS}^2 \cdot R_{DS(on)} \tag{3.1}$$

and

$$P_{on} = E_{on} f_{sw} = \left(\frac{I_{D(on)} V_{DD} t_{on}}{2} \right) f_{\text{sw}} \tag{3.2}$$

where P_{on} represents the case where the MOSFET goes from the off to the on-state, and t_{on} the amount of time this transition takes. P_{off} is calculated by replacing E_{on} by E_{off}, and t_{on} by t_{off} [21]. However, since creating an energy-efficient power supply is not one of the requirements, this is not that big of a concern for the design. It is good to minimize it where possible though, which is why the specifications for the drain-source resistance and switching time are set to be “minimal”.

The MOSFET which was chosen is the IPW65R037C6 from Infineon Technologies, an NMOS transistor (Figure 3.5). Normally PMOS transistors are preferred for high-side switching. However, since a PMOS has a higher drain-source resistance and slower switching time, the losses would be a lot larger. Therefore, an NMOS is used. The high-side problems will be solved by the gate driver circuit as discussed in the next section. The IPW65R037C6 meets the specifications in Table 3.1. Its specifications are listed in Table 3.2.

Table 3.2: *Specifications of the IPW65R037C6 [22]*

Quantity	Specification
Maximum drain-source voltage	700V
Maximum drain-source current	297A
Drain-source resistance	33 mΩ
Turn-on time	54 ns
Turn-off time	147 ns
Gate charge	330 nC

Figure 3.5: *IPW65R037C6*

The IPW65R037C6 obviously meets the required drain-source specifications. A rough estimate of the power dissipation would be handy in order to decide whether a heatsink is needed. For the PWM signal, $I_{D,RMS} = I_{D(on)}\sqrt{D}$ where D is the duty cycle [21]. As will become clear in Subsection 3.2.3, the used duty cycle will be 10%, current 10A and voltage 100V. So

$$P_{cond} = I_{D,RMS}^2 \cdot R_{DS(on)} = (I_{D(on)}\sqrt{D})^2 \cdot R_{DS(on)} = (10 \cdot \sqrt{0.1})^2 \cdot 0.037 = 0.37W \tag{3.3}$$

and with a maximum switching frequency of 50 kHz, on-time of 54 ns and off-time of 147 ns

$$P_{on} = E_{on} f_{sw} = \left(\frac{I_{D(on)} V_{DD} t_{on}}{2}\right) f_{\text{sw, max}} = \left(\frac{10 \cdot 100 \cdot 54 \cdot 10^{-9}}{2}\right) \cdot 50 \cdot 10^3 = 1.35W \tag{3.4}$$

$$P_{off} = E_{off} f_{sw} = \left(\frac{I_{D(on)} V_{DD} t_{off}}{2}\right) f_{\text{sw, max}} = \left(\frac{10 \cdot 100 \cdot 147 \cdot 10^{-9}}{2}\right) \cdot 50 \cdot 10^3 = 3.675W \tag{3.5}$$

Therefore, the total power dissipated by the MOSFET will be $0.37+1.35+3.675 = 5.4W$. Its temperature can be calculated using the junction to ambient thermal resistance, which is $62C°/W$ according to the datasheet. This means that it will become $5.4W \cdot 62C°/W = 334.8C°$ higher than the ambient temperature. Assuming an ambient temperature of $25C°$, the MOSFET will reach a temperature of $359.8C°$. This is probably too hot and and therefore a switching frequency of 10 kHz will be used if a voltage of 100V is used. Following the same procedure as above results in a MOSFET temperature of $110C°$. In any case, a heatsink will be helpful.

Concerning the other specifications of the MOSFET, it can be seen that it is quite overrated but since it was available in the lab, it was still chosen. Components that are already in the lab are preferred, since it saves a lot of ordering and shipping time. The turn-on time of the MOSFET is very low, however, the turn-off time is a lot larger. This difference can be compensated in the gate driver circuit. The gate charge is the amount of charge that needs to be delivered and removed from the gate to turn the MOSFET on and off. Therefore, the gate charge determines the switching time. This will be further explored in the next section.

3.2.2. Gate driver circuit

The MOSFET is driven by a PWM signal, which is provided by the microcontroller. However, the microcontroller has limited output voltage and current. The low voltage of only 3.3V is a problem because the threshold voltage of the MOSFET is maximally 3.5V [22]. If the gate-source voltage would be 3.3V, the MOSFET would not turn fully on. Moreover a high gate-source voltage is preferred to create a larger channel between drain and source to allow a higher current to flow.

Secondly, the microcontroller can only supply a small current. The pico has a maximum current of 50 mA [23]. This is a problem since the gate charge of 330 nC needs to be supplied to or drawn from the gate to turn the MOSFET on or off. A small current will take longer to supply this charge, therefore a higher current will result in less switching time.

For these two reasons a gate driver circuit is required. This is a circuit that is driven by the PWM signal from the microcontroller. It basically amplifies the signal resulting in a higher voltage and higher current pwm signal.

A gate driver circuit is either a self-built circuit or a dedicated integrated circuit. The integrated circuit was chosen, since it is faster, smaller and easier to use. Moreover, it can provide galvanic isolation between the control side and the power side. This protects the microcontroller and possibly the user against high voltages. Another advantage is the isolated ground at the high side of the gate driver. This makes implementing the high-side switch easier, since the isolated ground can be directly connected to the source of the MOSFET. This ensures the gate-source voltage will not be influenced by the source to ground voltage.

The selected gate driver integrated circuit is the STGAP2SICS from STMicroelectronics (Figure 3.6). Its specification are listed in Table 3.3.

Table 3.3: *Specifications of the STGAP2SICS [24]*

Quantity	Specification
Maximum output voltage	26V
Maximum output current	4A
Propagation delay	75 ns
Galvanic isolation	6000V

Figure 3.6: *STGAP2SICS*

This circuit was also available in the lab, so it could be used immediately.

The gate driver integrated circuit needs some additional components to function correctly. It requires some capacitors, two gate resistors and external supplies. Two variants exist, unipolar and bipolar. The difference is the negative voltage of the high-side supply. In the unipolar case, the gate-source voltage varies between 0 and 15V. The bipolar circuit provides an additional supply that pulls down the gate-source voltage from 15V down to -5V. These two supplies are VH and VL in Figure 3.7 that shows the bipolar configuration.

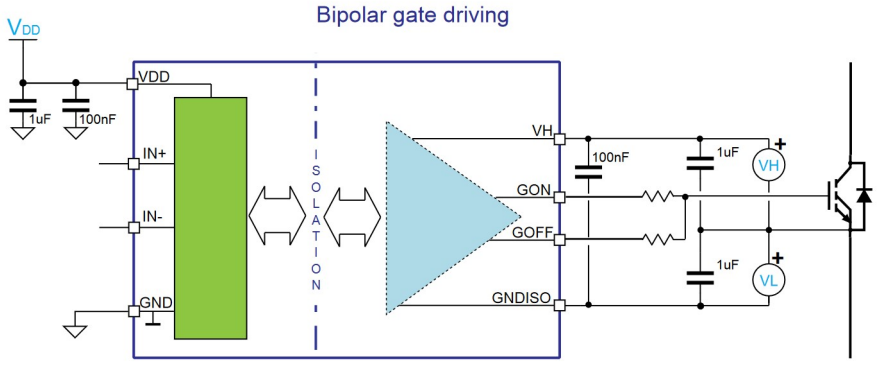


Figure 3.7: *Gate driver circuit [24]*

The bipolar circuit was chosen over the unipolar circuit since it pulls down the gate-source voltage more below the threshold voltage, which ensures the MOSFET is turned-off more strictly. This prevents possibly false turn-ons. The supplies to the gate driver are provided by the low-side supply, which is set to 5V, through the MGJ2D241505SC, a DC to DC (boost) converter. This converter provides a -5V, 15V and ground output isolated from the input [25].

The purpose of the capacitors is to filter high frequency noise and ripples from the supplies and to store and deliver impulse current [24]. The values for the capacitors are taken from the datasheet and can be seen in Figure 3.7.

The gate resistors are used to determine the switching time. As stated above, the gate charge needs to be supplied to the gate to turn the switch on and off. The higher the gate current, the faster the switching. Therefore it would seem that a smaller resistor value or no resistor at all is the best choice. However, this is not the case, since a sufficiently large resistor is needed to suppress oscillations from the resonance between the source inductance and gate capacitance [26]. The optimum value of the gate resistor can be calculated, but this requires measuring the parasitic parameters. An easier method is starting from 10Ω and lower the resistor value until a slightly underdamped waveform is obtained [27]. This will prevent significant oscillations while maintaining a low switching time.

The selected gate driver has separate output for turn on and turn off, so two gate resistors are required. The circuit from Figure 3.7 was built on a breadboard with the components described above. A function generator was used to create a PWM signal with an amplitude of 5V. Then the described method was applied to obtain the resistor values. It was found that the turn-on resistor should be 1.2Ω and the turn-off resistor should be 1Ω. With these resistor values, the gate-source voltage has the following waveform.

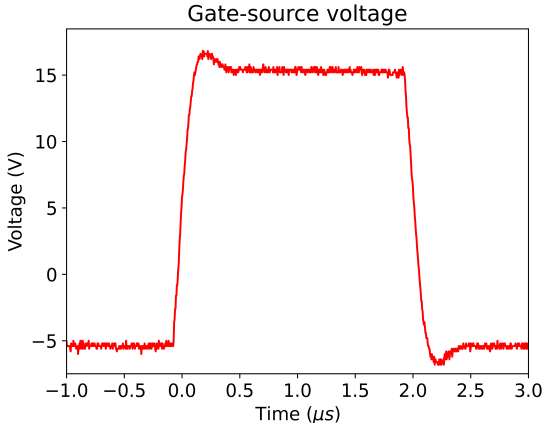


Figure 3.8: *Measured gate-source voltage*

The resistors were chosen to create a slightly underdamped waveform, as can be seen in the figure. Finally a pull down resistor was added between the gate and the source to make sure the gate-source voltage is zero when the circuit is turned off. The value of this pull-down resistor is 10 $k\Omega$.

3.2.3. Other components

The first version of the circuit has three more components, which are the low DC supply of 5V to power the gate driver circuit, the high DC supply and the current-limiting resistor. The DC power supplies are supplies from the lab. The high-side supply creates the DC voltage that is switched on and off to create the desired output.

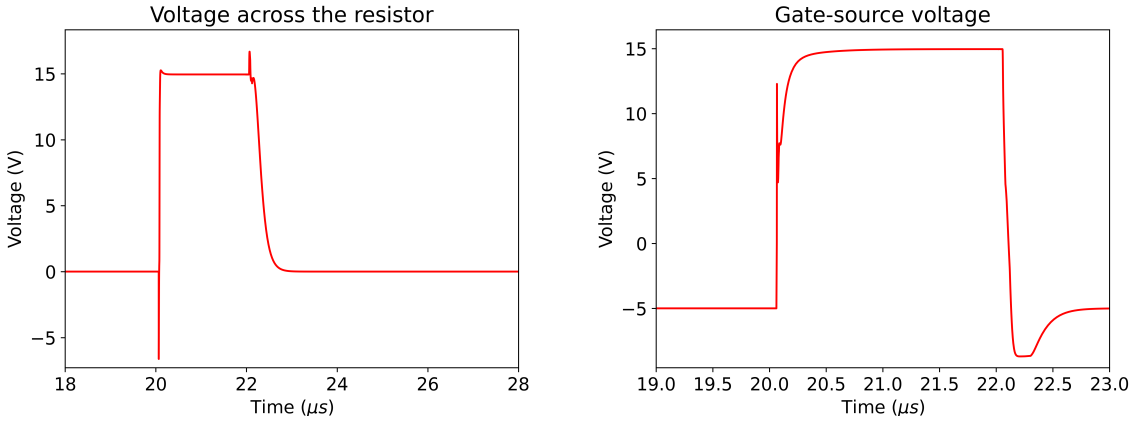
The current-limiting resistor determines the maximum current that can flow through the circuit. Based on Table 2.1, a current of 10A has been chosen for now as experiments will have to determine which value actually works the best. Since the HS150 power resistor of 10Ω from ARCOL was already available in the lab, it has been decided to use this one for the moment [28]. According to Ohm's law, this means the voltage needs to be 100V. This gives a power of 1 kW during a discharge. However, the duty cycle still needs to be taken into account, leading to a lower value for the power

dissipation. This resistor has a power rating of 150W, meaning roughly a duty cycle of 15% would be acceptable since $1kW \cdot 15\% = 150W$. To take a margin, it is decided to use a duty cycle of 10%.

3.3. Results of the first circuit

3.3.1. Simulation results

Now that the basic circuit is designed, it was simulated using LTSpice. The selected MOSFET has a spice model, however, the gate driver does not. A similar model was used as a replacement. The specifications were set to the requirements from Table 2.2, except for the high-side DC source, which was set to only 15V. The reason for using 15V during the measurements is because of safety. The results of the simulation are plotted in Figure 3.9.



(a) *Voltage across the resistor in the simulation*

(b) *Gate-source voltage in the simulation*

Figure 3.9: *Simulation results of the first circuit*

The results seem to be quite as expected. In Figure 3.9a, voltage across the resistor can be seen. It has some small errors, which are the undershoot at turn on, and overshoot at turn off. The reason for these errors is probably that there are two spikes in the gate-source voltage, as can be seen in Figure 3.9b. These errors were not present in the measured gate-source voltage in Figure 3.8, so it is likely that the spikes across the resistor will not happen as well. They probably originate from the fact that the gate driver used in the simulation is different from the one used in the experiments.

3.3.2. Measurement results

Since the simulation results are as desired up to an acceptable extent, the circuit was built and tested. The implementation of the circuit can be seen in Figure 3.10. The gate driver circuit is left out of this figure.

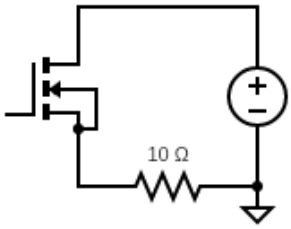


Figure 3.10: *Implementation of the basic circuit*

For safety reasons, the DC power supply is set to 15V instead of 100V during this test as said above. The measured voltage across the resistor is plotted in Figure 3.11.

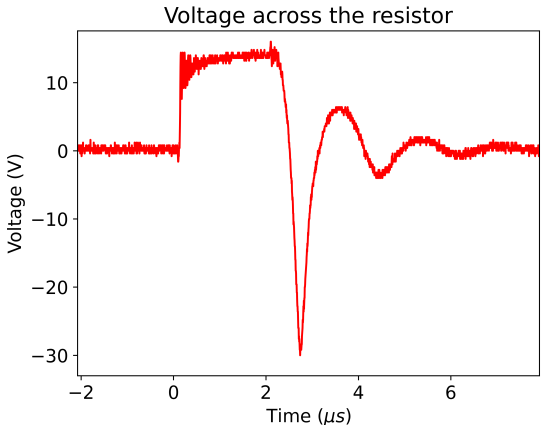


Figure 3.11: *Measured result of the first circuit*

The measurements are quite different from the simulation results in Figure 3.9a. A large oscillation can be seen when the switch turns off. This behaviour suggests that there is an inductive load, since that would generate a large negative voltage when its current is cut off. Moreover, it would also resonate with the capacitances of the MOSFET. To check this, the power resistor was measured with an LCR-meter, which showed that it has a parasitic inductance of 6.4 μH. When this inductance is added in series with the resistor, the simulation shows a similar result as Figure 3.11, as plotted in Figure 3.12.

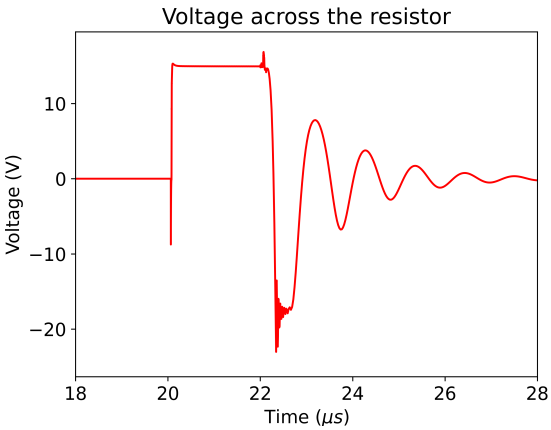


Figure 3.12: *Simulation result with the parasitic inductance*

The simulation still shows the negative spike at turn on, which is not present in the measurements. Both show the oscillation, but the period of the oscillation is a lot shorter in the simulation. The measured voltage has a period of 1.8 μs, while the simulation result has a period of 1.1 μs. This is quite a large difference, so the simulation has to be improved.

3.3.3. Improving the simulation

The oscillation is caused by resonance between the inductance of the resistor and the capacitances of the MOSFET. The period of LC resonance is known to be given by $T = 2\pi\sqrt{LC}$. Since the period in the simulation is too small, the capacitance used in the model for the MOSFET might be too low.

In the MOSFET LTSpice model, the internal capacitances have to be adjusted as these are voltage-dependent [29]. In Figure 3.13, the internal capacitances of a MOSFET can be seen.

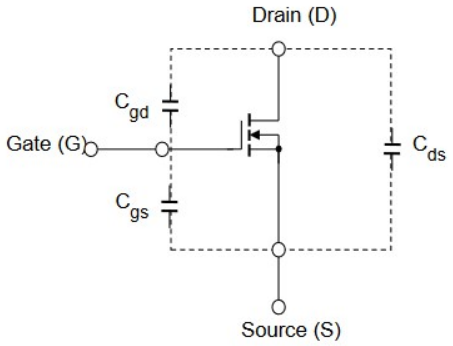


Figure 3.13: *The internal capacitances of a MOSFET [29]*

In the datasheet, the input, output and gate-drain capacitance are given as a function of the drain-source voltage. Using the relations,

$$C_{iss} = C_{gs} + C_{gd} \tag{3.6}$$

$$C_{oss} = C_{ds} + C_{gd} \tag{3.7}$$

where C_{iss} and C_{oss} stand for the input and output capacitances respectively, it is found that C_{gd} =1.3 nF, C_{gs} =6.9 nF and C_{ds} =13.5 nF at V_{ds} =15V. Now these new values for the capacitances were put

into the model for the MOSFET and the circuit from Figure 3.10 was simulated again. The results are plotted in Figure 3.14.

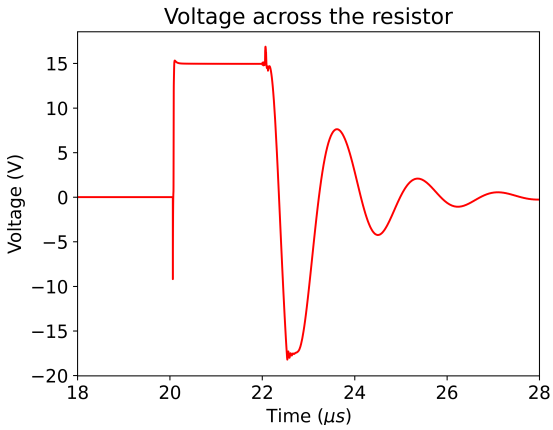


Figure 3.14: *Simulation results with adjusted capacitances*

The result still looks like Figure 3.12, however, the period seems to match the measurements better. It was measured to be 1.8 μs, so the simulation is more accurate. The negative peak has decreased a bit and still does not match the measurement, which indicates the simulation is still not complete. However, the results are good enough since the oscillation is quite large. In the next section the oscillation will be removed.

3.4. Removing the oscillation

3.4.1. Freewheel diode

The most common solution to compensate an inductive load is a freewheel diode [30]. This is a diode that is placed anti-parallel to the load as seen in Figure 3.15.

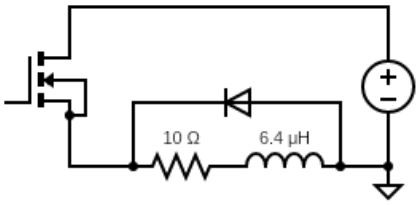
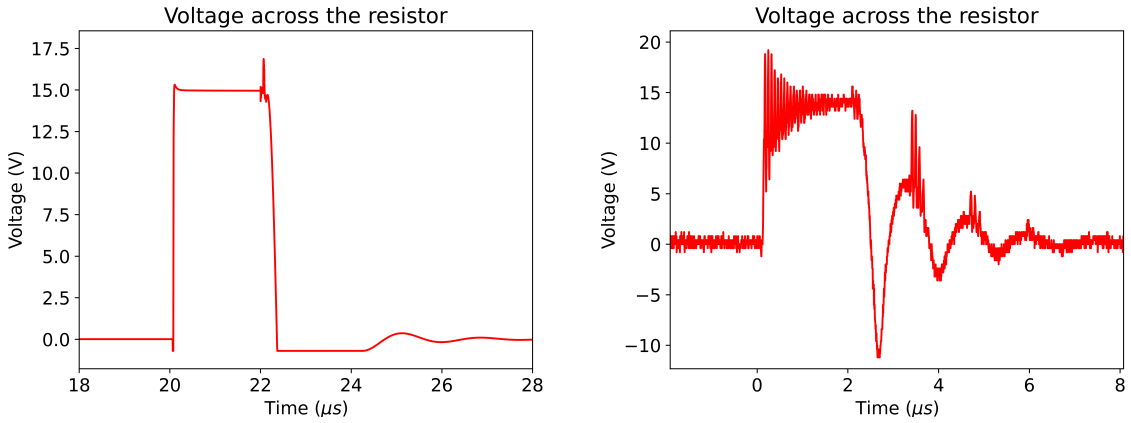


Figure 3.15: *Implementation of a freewheel diode*

The diode is reversed biased when the switch is closed, so it does not influence the circuit then. When the switch is opened the current from the inductor can flow through the diode so it can discharge safely. This diode was implemented using a Shottky diode, the C3D02060F by Wolfspeed. It is fast and can handle voltages up to 600V and currents up to 120A [31]. Using the C3D02060F, the circuit from Figure 3.15 was simulated, and the results are plotted in Figure 3.16a. Since the result is very good (no visible oscillations), the circuit was built and measured. The measurements are plotted in Figure 3.16b



(a) *Simulation results* **(b)** *Measurement results*

Figure 3.16: *Results of the freewheel diode solution*

The results do not match at all. The simulation shows a nice PWM signal. It has a small spike, but that is an acceptable error. However, the measurement still shows a large oscillation. The amplitude has dropped from -30V to -10V, but it is still not acceptable. This is probably the case because the wires connecting the diode and the MOSFET have parasitic inductance as well. When these inductances are added, the circuit looks like this:

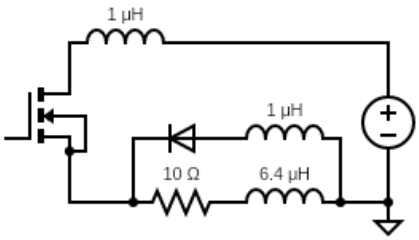


Figure 3.17: *Circuit with the parasitic inductances*

The values of the inductances are set to 1 μH. Since these inductances depend a lot on the placement of the wires, they vary a lot. A LCR-meter was used to measure the inductance of a single cable. The inductance was highly variant, but the value was roughly around 1 μH. This circuit was now simulated, and the results can be seen in Figure 3.18.

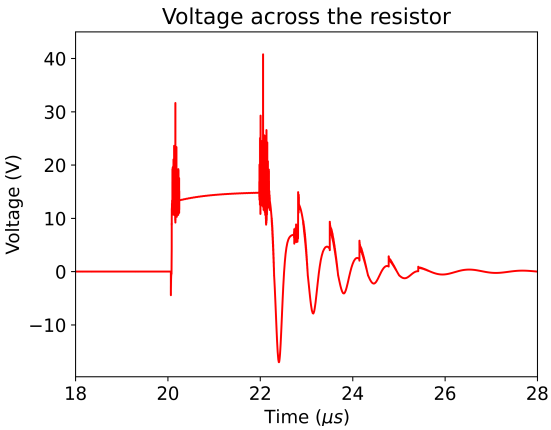


Figure 3.18: *Simulation results with the parasitic inductances*

The simulation mostly matches the measurement in Figure 3.16b quite well. The oscillation periods are both 1.8 μs and the negative peaks approximately the same. Furthermore, the spikes on top of the oscillations are visible in the simulation. The oscillation at turn-on is present in the simulation, however, it has a higher amplitude and decays faster. The main difference is the large spike of 40V at turn-off in the simulation. This is not present at all in the measurement. The reason for this is probably that the model for the gate driver is different and that the simulation circuit does not include all parasitics.

Up until now all circuits have been built on a standard soldering board. This of course creates a lot of parasitics. When the custom PCB is used, the performance will probably be a lot better. To test this, the inductor behind the diode was decreased to 100 nH (wire parasitic). The result is shown in Figure 3.19.

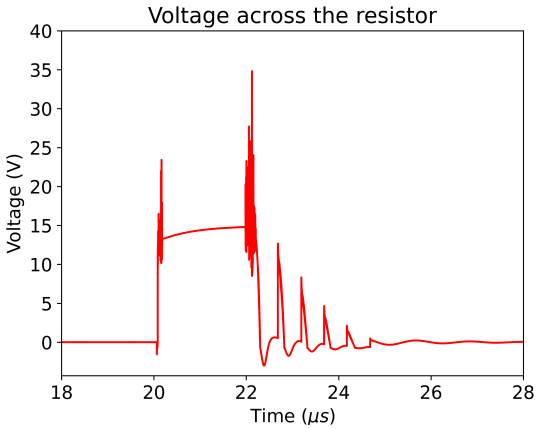


Figure 3.19: *Simulation results with a smaller inductance*

The result is a bit better than before, especially if the large positive spike at turn-off is ignored as before, since it has not been present in the measurements. The negative spikes are gone, but the positive spikes on top of the oscillations remain at approximately 12V, just as observed in the measurements of Figure 3.16b. Since most parasitics will smaller with the PCB, it is expected that the results will probably be better as well.

3.4.2. PCB

The circuit was built on the PCB and tested. The high-side supply is still 15V, the 5V on the low side is supplied by the linear voltage regulator described in Subsection 4.2.1. The PWM signal is provided by the microcontroller as described in Section 4.3. The result is plotted in Figure 3.20.

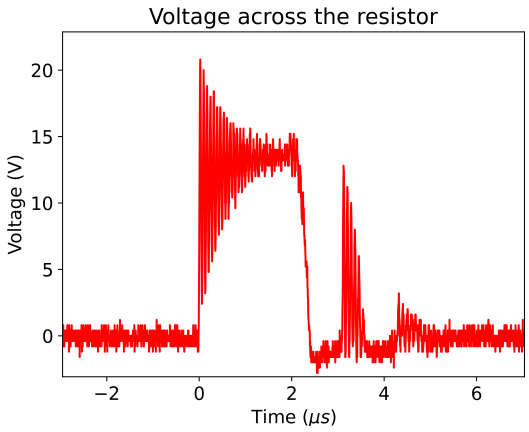


Figure 3.20: *Measurements on the PCB*

The measurement is a lot better than the result obtained from the standard soldering board (Figure 3.16b), and resembles Figure 3.19 quite well. This confirms the assumption that the inductance from the wire behind the freewheel diode was indeed the cause of the significant oscillations in Figure 3.18.

The result is still not good, as there is still a large spike of about 13V about 1 μs after turn-off, and also a lot of high frequency noise in the output. A capacitor of 10 μF was added in parallel to the DC supply to filter out this noise. The capacitor was placed directly on the contact points on the PCB of the voltage source, as seen in Figure 3.21. The circuit was measured again, and the result is plotted in Figure 3.22.

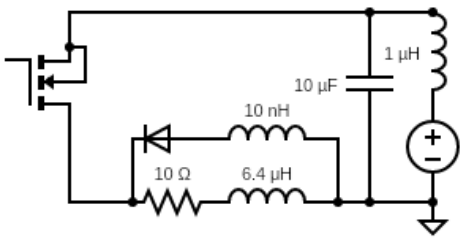


Figure 3.21: *The final circuit*

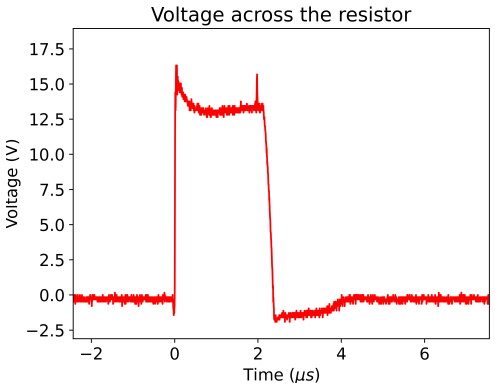


Figure 3.22: *Measurement with filtering capacitor*

The result is much better, almost all the high-frequency oscillations are gone. The positive spike has also been removed, and only a small undershoot is visible. This small shoot has actually always already been present but not really visible due to the large oscillations. It is due to the voltage drop across the freewheel diode that is present during the short period of time it conducts the current from the power resistor [32].

The only remaining problem is that the voltage drops to around 13V after it reaches 15V at turn-on. This can be explained by the inductor present directly after the voltage source as visible in Figure 3.21. As the switch turn-on, the current increases through this inductor, resulting in a positive voltage drop across it. This means that there will be less than 15V across the power resistor. The result is still acceptable as this can be compensated by turning up the DC supply.

4
Microcontroller

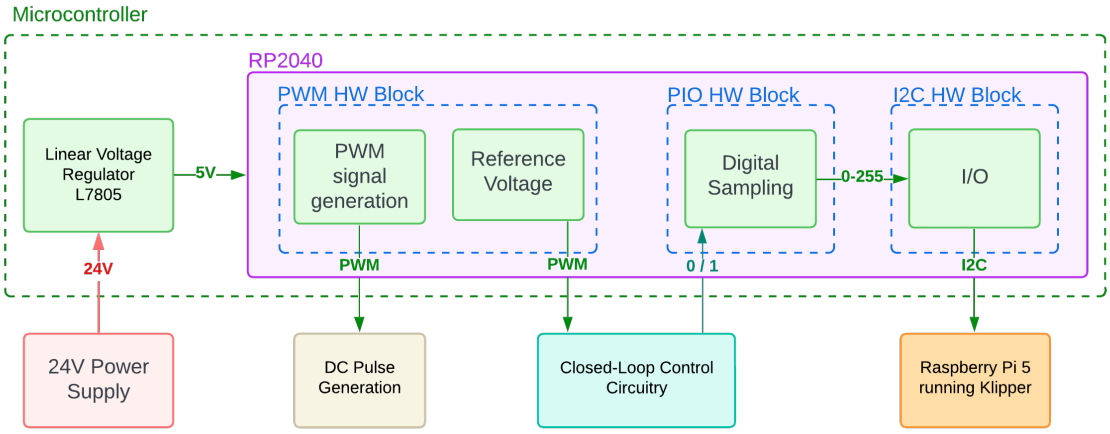


Figure 4.1: *Schematic overview of microcontroller working*

Everything done in this project could be regulated by dedicated fixed hardware to generate the necessary signals to drive the project. However, this approach locks in the parameters and restricts the use of more complex functionality. This limitation was recognized from the beginning of the power supply design. The solution was to use a microcontroller to create and process all the signals needed for the power supply. Consequently, requirement **EDM_PS_ST_3** called for the use of a microcontroller. This chapter explores all aspects of the microcontroller, with a schematic view provided in Figure 4.1. First, the choice of microcontroller will be examined, followed by an explanation of how it will be powered. Finally, the different hardware blocks will be discussed with in-depth explanations for the choices made.

4.1. Microcontroller choice

There are multiple vendors and versions of microcontrollers that could be used for the project. The likes of Arduino, ESP32 and STM32 were all compared of which the RP2040 (Figure 4.2) microcontroller on the Raspberry Pi Pico board came out on top (Figure 4.3), which is referred to as RPI Pico from now on. The key specifications can be found in Table 4.1. The performance for this package for around 4 euros with the possibility for wifi and Bluetooth 6.5 euros is unmatched. Additionally, the documentation for this chip is phenomenal in the form of starting guides [33], microcontroller datasheets [34], C/C++ software development kit [35], MicroPython documentation [36] and examples for every aspect of the microcontroller [37].

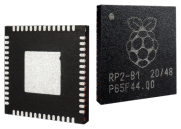


Figure 4.2: *RP2040 microcontroller [38]*

Table 4.1: *Specifications Raspberry Pi Pico [23]*

Specification	Value
Processor	Dual-core ARM Cortex-M0+
Clock Speed	Up to 133 MHz
SRAM	264kB
Flash Memory (Onboard)	2MB
GPIO Pins	26
ADC Channels	3 (12-bit)
UARTs	2
SPI Ports	2
I2C Ports	2
USB Support	Yes (native USB 1.1)
Input Voltage	1.8V - 5.5V

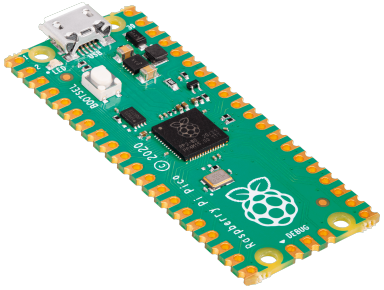


Figure 4.3: *Raspberry Pi Pico [39]*

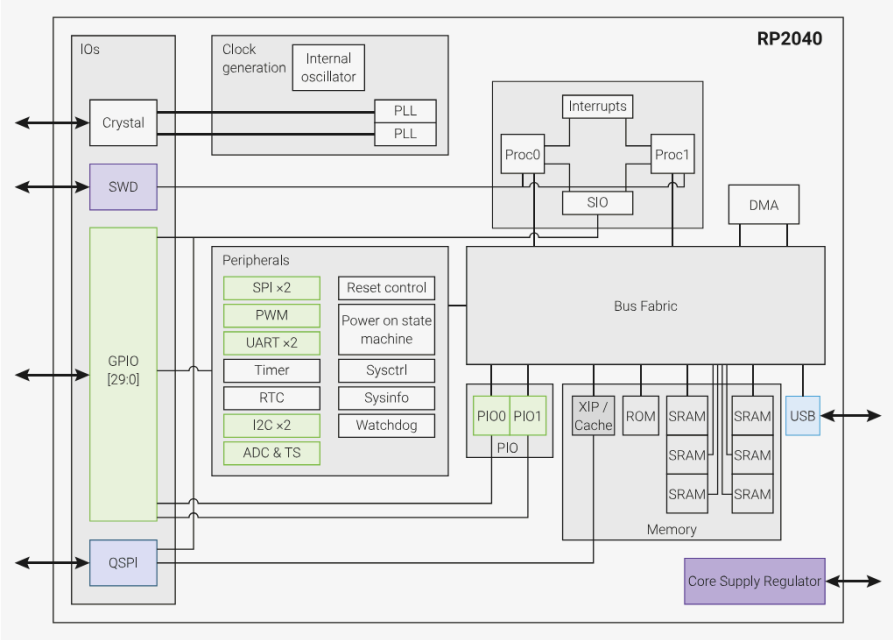


Figure 4.4: *RP2040 hardware block diagram [40]*

4.1.1. Programming language

The RPI pico has the ability to run C/C++ using the pico-sdk [41], C++ using the Arduino IDE [42], MicroPython [43] and embedded rust [44]. The C/C++ route is the fastest official way to code for the RP2040, and a lot of support and documentation is available to code this way in combination with the pico-sdk. This software development kit is used to directly talk to the dedicated hardware blocks (of which an overview can be found in Figure 4.4), without having to work with separate registers. Another way to use C++ is using the Arduino IDE that, although not officially supported, has a familiar interface for people who previously worked with Arduino using libraries and small workarounds. MicroPython is a much more beginner-friendly way to program the RP2040, but it has a quite substantial performance penalty as the program gets interpreted instead of compiled. Embedded Rust is a more modern compiled language with the most important feature being memory safe, while still being as performant as the ”native” C/C++. Although Embedded Rust is getting more traction, the support and documentation are not quite far enough for it to justify learning a new programming language for a 10-week project.
Although MicroPython was the favourite due to its simplicity, the performance of MicroPython is not

enough for this use case (see Subsection 4.4.1). Previous experiences with the RPI Pico and the Arduino IDE found that some critical hardware interfaces are missing for this project. Thus the C/C++ with the pico-sdk was found to be the best way to program the RP2040 microprocessor.

4.2. Powering the microcontroller

The two sources responsible for powering the whole power supply are a 24V input and a high-voltage DC source. The 24V input is used for the logic part of the system, which includes the microcontroller. According to the specifications of the RPI Pico in Table 4.1, the input voltage needed on the VSYS line ranges from 1.8V to 5.5V. The electrical specifications from the datasheet [23] state that the power required for the microcontroller under full load does not exceed 100mA in the absolute worst case.

A small additional current can be expected since the 3.3V rail of the Pico is used for some other parts of the power supply on the low side. These devices mainly use this 3.3V as a reference signal and are only involved in digital operations. Considering that the Pico is not using both cores to their fullest potential at all times, it can be assumed that the average current draw is under 100 mA, which is a significant factor in decision-making.

4.2.1. Linear voltage regulator

The first option explored was to use a linear voltage regulator, such as the L7805 [45] (displayed in Figure 4.5), to convert the 24V input to 5V for the microcontroller. The specifications for the L7805, found in Table 4.2, are more than sufficient for the simple task of powering the microcontroller. It is simple to integrate and provides a very stable output voltage over a wide range of input voltages. However, due to the workings of the linear voltage regulator, the voltage drop multiplied by the current is lost as heat. In this case, it would be approximately $\frac{P_{out}}{P_{in}} = \frac{V_{out}}{V_{in}} = \frac{5}{24} \approx 20\%$ efficient if it is assumed that $I_{in} = I_{out}$. For the voltage drop of 19V and an average current of 100 mA, this would result in a 1.9W power loss in the voltage regulator. Despite this inefficiency, pairing the regulator with a small heatsink would be fully acceptable, requiring only two external capacitors.

Table 4.2: *L7805 specifications [45]*

Specification	Value
Type	Linear voltage regulator
Input voltage range	7V - 35V
Output voltage	5V
Output current (maximum)	1.5A
Output voltage tolerance	± 0.2V
Efficiency	Low

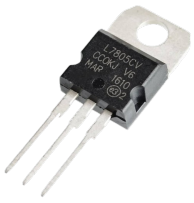


Figure 4.5: *L7805*

4.2.2. Switching regulator

A better option would be to use a switching regulator, such as the L7983 [46] (fixed 5V variant) which can be seen in Figure 4.6 with the specifications in Table 4.3. Instead of dissipating excess voltage as heat, the switching regulator operates by rapidly switching the power on and off at a high frequency. This method results in much higher efficiency, around 85% at 100mA [46]. However, these improved efficiency numbers come at the cost of a more complex design and the need for more components.

Due to its small package size, the L7983 is only feasible for use on a PCB, which limits testing options before final system implementation. Additionally, the switching operation of the chip can cause electromagnetic interference (EMI) to nearby components, potentially disrupting their operation if not properly managed. Despite these challenges, the chip offers additional functionality compared to the linear voltage regulator, such as an enable function, soft start, and adjustable frequency.

Table 4.3: *L7983 specifications [46]*

Specification	Value
Type	Synchronous step-down switching regulator
Input voltage range	3.5V to 60V
Output voltage	Fixed 5V
Output current (maximum)	300mA
Switching frequency	Programmable (200kHz to 2.2MHz)
Efficiency	High

Figure 4.6: *L7983*

4.2.3. Choice and performance

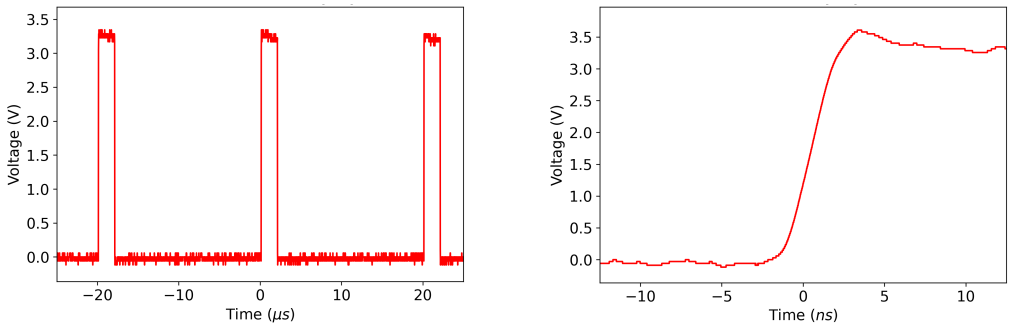
Due to the limited time available for testing, the safer ”guaranteed to work” option of the linear voltage regulator was chosen. The switching regulator could be an interesting upgrade for the next iteration, offering a more compact design and the potential to utilize the extra features mentioned.

When testing the output of the linear voltage regulator, it was found to have a ripple of less than 1 mV at an average of 4.954V under normal use with all components connected and using the recommended capacitance values from the datasheet. These results are more than sufficient to run the microcontroller without any issues.

4.3. PWM generation

The RPI Pico has 8 PWM hardware blocks, each capable of driving two PWM output signals on all GPIO pins, at a theoretical maximum frequency of 62.5 MHz when running at the base clock speed of 125 MHz. This hardware can generate the square wave needed for the power supply to function, allowing for real-time adjustments of frequency and duty cycle. The exact implementation of the PWM hardware is detailed in section 4.5 of the C SDK Documentation [34].

The requirements specify a frequency range of 1 kHz to 50 kHz with a duty cycle of 10% to 90% (**EDM_PS_F_10** and **EDM_PS_F_11**), a capability demonstrated in Figure 4.7a. When zoomed in, the rise time is approximately 3 ns, as shown in Figure 4.7b.



(a) *Displaying 50kHz 10% duty cycle PWM capability*

(b) *50kHz 10% duty cycle PWM zoomed*

Figure 4.7

4.4. Closed-loop control

The microcontroller needs to constantly measure the voltage waveform as part of a closed-loop system. As fully explained in Chapter 5, the voltage drops when an arc is formed. The microcontroller must detect the timing of these arcs and communicate this timing to the device moving the electrode. The scope of the power supply subgroup, and thus this report, is to determine the timing of the arcs and provide a method to digitally share this information with the speed controller, as specified in **EDM_PS_ST_4**.

4.4.1. Polling ADC

The first and most obvious approach is to use the onboard analog-to-digital converter (ADC) of the RP2040 to periodically sample the voltage. As a test of its usability, the ADC was polled using MicroPython for a 1kHz signal at a 50% duty cycle. Polling involves repeatedly calling the `read_u16()` function in a while loop and capturing the results using Listing B.11. It was found that the sample rate for this method is around 50k samples per second. According to the datasheet, the maximum sample rate of the ADC is around 500k samples per second. Thus, MicroPython is too slow to achieve these speeds.

The same program was then written in C, utilizing Direct Memory Access (DMA) (see chapter 2.5 of C SDK Documentation [34]). With DMA, the maximum sample rate of 500k samples per second was reached using Listing B.12. This is already vastly better than the MicroPython version, but still not deemed fast enough for the precise timing needed for closed-loop control. At 50kHz, there would be 10 samples per period, giving a resolution of 2 μs, which can be the entire pulse at a 10% duty cycle (**EDM_PS_F_11**). This resolution hinders the ability to accurately observe how the pulse width itself changes depending on the width of the discharge.

4.4.2. FFT

Another option under consideration is to take an FFT of multiple pulses. This approach is based on the principle that decreasing the duty cycle will increase the period of the sinc function [47]. However, this method assumes that the pulses are consistent every time, which is not the case for the irregular process of EDM where sparks occur randomly (see Subsection 3.1.2). Detecting the small changes required for this method necessitates a higher sampling rate than the 500kHz of the ADC.

4.4.3. Integration

Another approach explored is to once again take hundreds of peaks and calculate the area to obtain a general idea of the average duty cycle of the pulse time. However, this option has not been explored in detail because during this time, another much better option was found, which will be discussed in detail in Subsection 4.4.4.

4.4.4. PIO

As part of the RP2040 microcontroller architecture, there are multiple hardware blocks dedicated to specific hardware-accelerated tasks (see Figure 4.4). Among these hardware blocks are the two Programmable Input Output (PIO) blocks. These blocks allow users to program clock-perfect programs using 9 different assembly instructions, also called PIOASM. A program can be written to check if a GPIO pin changes from high to low or from low to high.

A program which analyzes a PWM signal [48] was adapted for this project. It checks if there is a change in one clock cycle and increases a counter in the next until the GPIO pin changes. This effectively allows for ”digital” sampling every 2 clock cycles, resulting in a sample rate of 62.5MHz at the base clock of 125MHz. While overclocking is an option [49], it is considered a future endeavour and is not within the scope of the current project.

The duty cycle can then be calculated by converting the number of clock cycles to time and performing quick calculations to determine the pulse width. This method does require sacrificing analog values and only accepts digital signals. However, it offers the ability to analyze every pulse individually, avoiding the need to average multiple pulses due to sampling frequency limitations.

4.4.5. Reference voltage PWM

Due to the limitation of only being able to sample digitally, as discussed in Subsection 4.4.4, additional circuitry is required. The full circuitry will be discussed in detail in Chapter 5. It is important to note that a voltage reference is necessary for this purpose. Since this power supply is designed for flexibility in

extensive testing, it must accommodate a full range of input parameters (see Table 2.3). Consequently, it is not possible to have a fixed voltage at which the GPIO switches from 0 to 1 and vice versa.

The RP2040 has hysteresis, also commonly referred to as a Schmitt trigger [50], turned on by default on all GPIO pins. Through experimentation, using Listing B.10, threshold values were determined, which can be found in Table 4.4. To maintain flexibility in determining at which voltage the switch from a logical 0 to 1 occurs, a signal from the microcontroller needs to be generated as a reference. This is accomplished by creating a PWM signal combined with a low-pass filter to produce a voltage. The duty cycle of the PWM signal determines the reference voltage, where 100% duty cycle yields 3.3V, 50% yields 1.65V, and so on. A detailed analysis of the analog part of the circuit and its characteristics can be found in Subsection 5.3.3.

Table 4.4: *Hysteresis of GPIO RP2040*

Operation	Voltage
0 -> 1	1.571V
1 -> 0	1.258V

4.5. I2C implementation

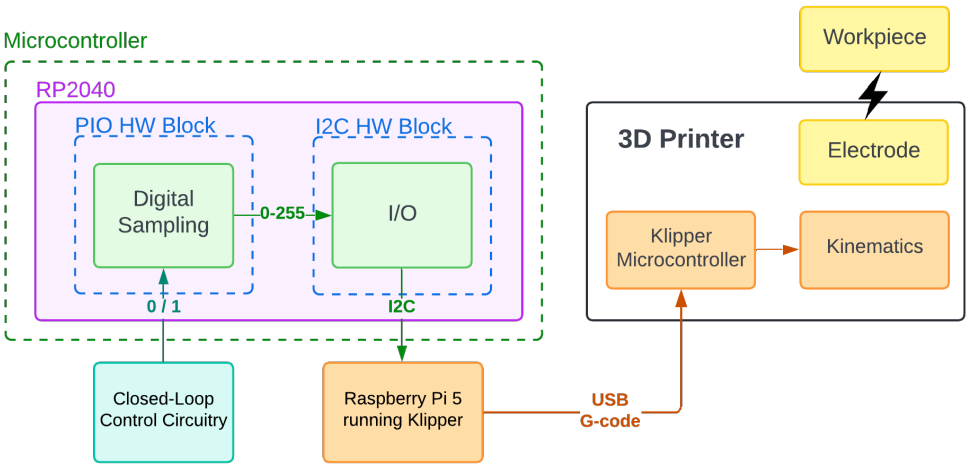


Figure 4.8: *Schematic Overview I2C implementation with speed control device*

For closed-loop control, having a data signal from the microcontroller to the device controlling the speed of the electrode is essential. While there are various methods to transmit such information, the Inter-Integrated Circuit (I2C) digital protocol was chosen for its ease of use, support for hardware acceleration, and its balance between speed and signal cables.

I2C operates on the master-slave principle, with the master (in our case, the speed control device) controlling the data flow. This protocol is also standard for many sensors, making it an ideal choice as the power supply essentially acts as a sensor to the outside world.

4.5.1. Speed control device

For context, an Ender 3 Pro running a modified Klipper firmware [51] via a Raspberry Pi 5 (RPI 5) is used to control the kinematics of the electrode. A detailed explanation of this setup can be found in the

report of the electrode and dielectric subgroup [6], which is responsible for the kinematics part of the machine as well.

The power supply is directly connected to the RPI 5 using the SDA and SCL data lines, with an extra wire for a shared ground as a reference. In this setup, the RPI 5 acts as the master and controls the flow of data, while the power supply operates in slave mode. The power supply reacts to commands sent from the RPI 5 (Listing B.4). These commands can involve changing the parameters of the power supply or requesting the current status of the electrode sparks. This is achieved by polling the microcontroller when necessary.

The microcontroller sends one byte back in response, where a value of 0 indicates a dead short between the electrode and the workpiece, and a value of 255 indicates that no spark occurred. Values in between provide an indication of how far the electrode is from the workpiece, allowing experimentation to find the optimum value. However, these experiments fall outside the scope of the power supply subgroup's responsibilities.

4.6. Implementation on PCB

The implementation on the PCB is relatively straightforward. An optimal approach would be to integrate the RP2040 directly onto the main PCB and add the necessary supporting hardware. However, for this project, such integration is deemed unnecessary, although it presents an interesting challenge for a potential second version. Instead, the RPI Pico board with the RP2040 (as seen in Figure 4.3) is mounted on the PCB using standard pin headers.

This board includes not only the RP2040 but also flash memory, a micro USB port, oscillators, and supporting power supply hardware. While this method sacrifices some compactness, it offers the advantage of rapid interchangeability.

During testing, the linear voltage regulator discussed in Subsection 4.2.1 did not require any additional heatsink. This was concluded after using the scientifically sound method of touching the component with a finger without burning it.

5
Closed-loop control

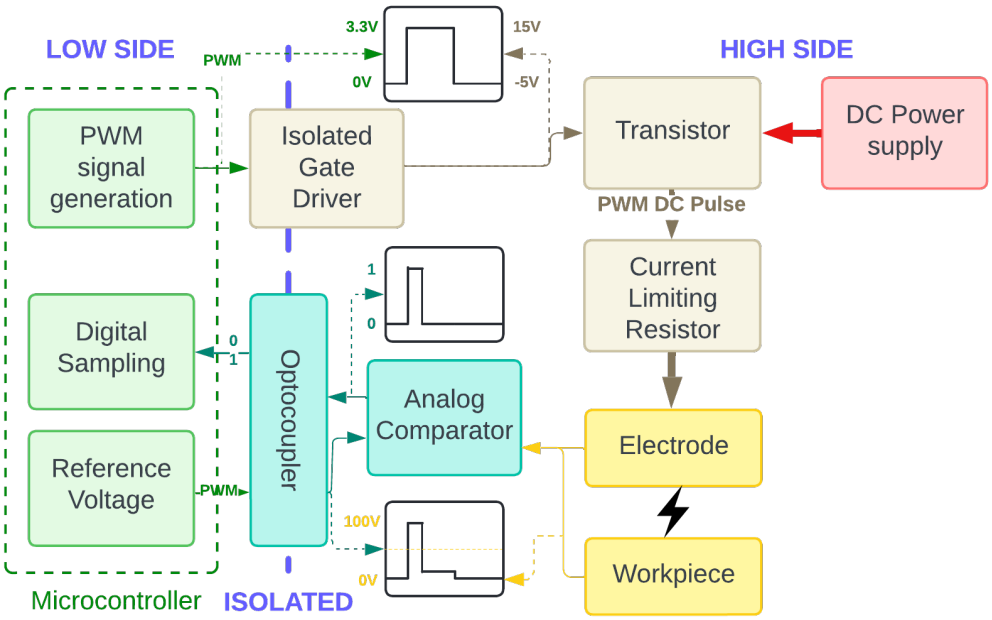


Figure 5.1: *Schematic overview of closed-loop control*

Electric discharge machining works best when the electrode is at an optimal distance from the workpiece. The electrode must move at a certain speed to continuously remove material. If the speed is too high, the electrode may touch the workpiece and fuse together, while moving too slowly will drastically increase machining time.

A key objective of this project is to optimize the final workpiece result while improving the machining speed by maximizing the electrode's speed. There is a sweet spot for the pulse frequency and duration. The power supply needs to sense this optimum point and communicate it to the speed regulator. The method for achieving this on the microcontroller side is discussed in Section 4.4.

This chapter explores the theory behind the discharge process and explains how to convert an analog signal into a digital signal that the microcontroller can interpret to detect sparks. A schematic overview of the final implementation can be found in Figure 5.1.

5.1. Theory and assumptions

The state of the gap width is reflected in the waveforms of the gap voltage, and this fact is exploited in the closed-loop system. The electric field in the gap can be considered as uniform, for which Equation 5.1 holds:

$$E = Vd \tag{5.1}$$

where V is the voltage and d the gap width.

If the gap width is small, the ignition delay time is smaller due to the fact that the electric field is more intense, leading to faster creation of the plasma channel. If the gap width is large, the opposite holds. This idea is presented in Figure 5.2. This behaviour was verified by the Electrode and Dielectric subgroup, of which the result can be found in Figure 5.3.

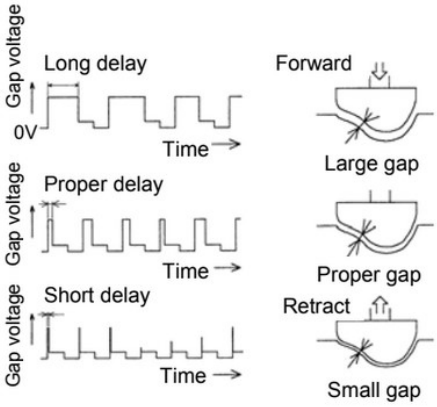


Figure 5.2: *Principle of the closed-loop control [3]*

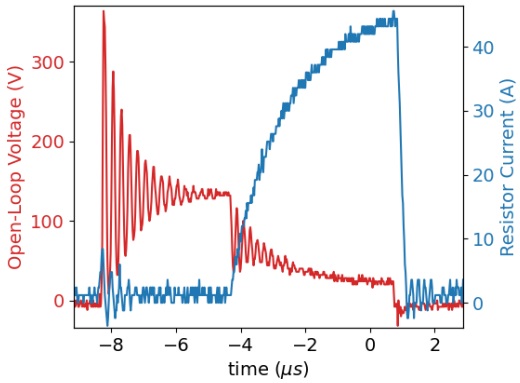


Figure 5.3: *Measurement of ignition delay time is possible [6]*

5.2. Spark sensing

As mentioned before, it is essential to sense the presence and speed of sparks during the EDM process. This can be done by either measuring the voltage over the current-limiting resistor or the open loop voltage across the electrode and workpiece. The current-limiting resistor method involves measuring the voltage across a power resistor to indirectly measure the voltage across the electrode and the workpiece. This approach has the added benefit of measuring the current through the resistor, and consequently, through the electrode and workpiece.

The basic principle is that when there is no short, the full voltage is across the electrode and the workpiece. When a spark creates a short, a conductive channel forms, allowing current to flow. This current is limited by the resistor, causing an increase in voltage across the resistor proportional to the current flow. The voltage across the current-limiting resistor, electrode, and workpiece is constantly being turned off and on. When a spark occurs, the voltage shifts from being across the electrode and workpiece to the current-limiting resistor.

The main problem with using the voltage across the current-limiting resistor is the lack of signal when no sparks are created. This impedes the microcontroller from effectively analyzing the pulses, as it constantly tries to measure the duty cycle to gauge the distance. This issue is solved by measuring the open-loop voltage between the electrode and workpiece, making this the preferred method for monitoring the spark timing.

5.3. Circuit design

The open loop voltage is analog and needs to be converted into a workable digital signal, specifically a 3.3V digital signal for the microcontroller. During the PCB design process, it was determined to be safer to separate the ground of the microcontroller side (low side) from the ground of the High DC power supply (high side). This separation adds complexity to the system, requiring more complicated methods to transfer signals and power between the two sides. This section discusses the components used to achieve this separation and functionality.

5.3.1. Power supply

A separate isolated DC-to-DC converter is needed to create a signal for the closed-Loop control system. An observant reader might have noticed there is already a DC-to-DC converter used for powering the gate driver circuitry. The exact same DC-to-DC converter (namely the MGJ2D241505SC as in Figure 5.4 of which the specifications are given in Table 5.1) is used for the closed-Loop control system, but it could not be shared because the gate driver power supply has its 0V reference at the source for correct operation. This is not the reference voltage needed for the other circuitry, which requires the high DC source ground as its reference.

By using a separate isolated DC-to-DC converter, we ensure that the grounds of the low side and high side remain isolated, maintaining system safety and integrity. The details of the specific components and their integration into the circuit will be discussed in subsequent sections.

Table 5.1: *MGJ2D241505SC specifications [52]*

Feature	Specification
Type	Isolated DC-DC converter
Output power (max)	2 W
Isolation voltage	5200 V
Typical input voltage	24 V
Output 1 voltage	15 V
Output 2 voltage	-5 V
Efficiency (Typical)	80.5%
Package type	SIP (Single in-line package)

Figure 5.4: *MGJ2D241505SC*

5.3.2. Analog comparator

An analog comparator is used to compare its input to a reference voltage level set by the microcontroller. A schematic view of the final version of this part of the power supply can be found in Figure A.10.

First, as the highest voltage the power supply should handle is 150V (**EDM_PS_F_8**), this needs to be reduced to 3.3V, which is the working voltage for all the logic signals in the power supply. This is accomplished with a simple resistive divider using 100kΩ and 2.2kΩ resistors, creating a $150V \cdot \frac{2.2k\Omega}{100k\Omega+2.2k\Omega} \approx 3.22V$ signal that scales linearly with the voltage across the electrode and workpiece. Although this introduces a path between a potential 150V and GND, it only dissipates $\frac{150V}{102.2k\Omega} = 1.47mW$, which is well within the tolerance of even basic 0.25W resistors.

This scaled voltage is then compared with a reference voltage (further explained in Subsection 5.3.3). Depending on whether the scaled voltage is above or below the threshold, the comparator will output a logical 1 or 0 at the required 3.3V. For this project, the analog comparator TLV3501AID as in Figure 5.5 was chosen due to its excellent fit for the required specifications. As detailed in Table 5.2, it has a low propagation delay and operates at the needed 3.3V.

It is important to note that while the fast propagation delay is beneficial, the equality of the rise and fall times is more crucial. This is because the width of the pulse, rather than the speed at which it arrives at the microcontroller, is important. With the TLV3501AID, the difference between the rise and fall times is never less than 0.5 ns, which is significantly lower than the microcontroller's sampling period of $\frac{1}{62.5MHz} = 16ns$.

The final circuit on the PCB for this comparator circuitry, including the reference voltage signal circuitry, can be found in Figure A.10.

Table 5.2: *TLV3501AID specifications [53]*

Specification	Value
Type	High-speed comparator
Propagation delay (max)	4.5 ns
Input/output voltage range	Rail-to-rail
Supply voltage	2.7 V to 5.5 V
Hysteresis	Yes (6mV)
Common mode rejection ratio (CMRR)	55 dB (typical)
Power supply current (max)	3.2 mA
Package	8-pin SOIC

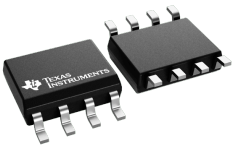


Figure 5.5: *TLV3501AID*

5.3.3. PWM reference level

The analog comparator discussed in Subsection 5.3.2 requires a reference voltage to function correctly. This reference voltage can be set by the microcontroller based on the required detection level. As explained in Subsection 4.4.5, the design's wide range of operational conditions necessitates adjustable supply voltage levels, resulting in various reference voltages required for the analog comparator.

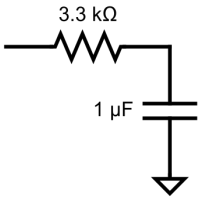


Figure 5.6: *RC circuit*

To achieve this, the microcontroller generates a PWM signal to set the reference voltage. This PWM signal is then converted to an analog voltage using a simple RC circuit as seen in Figure 5.6. The RC circuit smooths out the PWM signal, producing a stable analog voltage that serves as the reference for the comparator. This setup allows for flexibility in setting the detection threshold, accommodating the design's varying operational conditions.

The R and C values need to be chosen so that the system can quickly respond to changes in the PWM signal while minimizing voltage ripple. These are mainly determined by $\tau = RC$, where a higher τ gives a slower response but a lower voltage ripple. Minimizing voltage ripple should be the main design priority. The equations for the steady-state part after $V_C \approx DV_{source}$ are described in Equation 5.2 and 5.3.

$$\Delta V_C = V_{C,high} - V_{C,low} \tag{5.2}$$

$$V_{C,high} = V_{Source} + (V_{C,low} - V_{Source})e^{-\frac{t_{high}}{RC}} \qquad \text{with } t_{high} = \frac{D}{f_{PWM}} \tag{5.3}$$

An approximation of the voltage ripple can be made if the voltage ripple is low by setting $V_{C,low} = DV_{Source}$ for $D \approx 0.5$, which gives Equation 5.4.

$$\Delta V_c \approx 0.5 \cdot V_{Source}(1 - e^{-\frac{(0.5)/f_{PWM}}{RC}}) \tag{5.4}$$

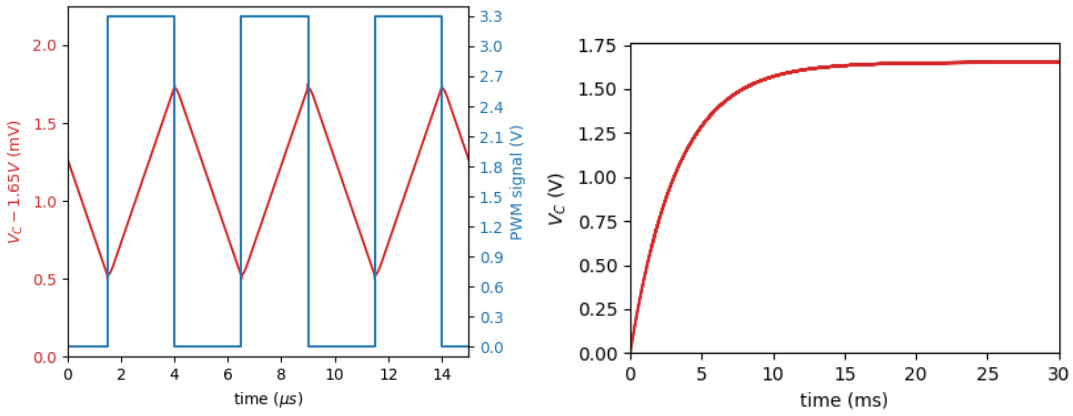
The voltage ripple depends on the R, C, and f_{PWM} values. The capacitor value was chosen to be 1 μF to maintain consistency with other capacitors in the circuit. The frequency f_{PWM} for the calculations was set at 200 kHz as a reasonable value. Although this frequency can be adjusted by the microcontroller, the calculations are based on this baseline. Therefore, only the resistor value needed to be determined. Given the hysteresis for the analog comparator is 6 mV, a maximum voltage ripple of 1.5 mV for $D = 0.5$ was chosen as a safety measure. Using Equation 5.4, it was calculated that the minimum resistor value should be at least 2.75 kΩ. Consequently, a 3.3 kΩ resistor was selected, which gives:

$$\Delta V_C = 1.65(1 - e^{-\frac{(0.5)/200{,}000}{3.300\times1\cdot10^{-6}}}) \approx 1.25\text{mV}$$

This calculation was verified with simulation results found in Figure 5.7a. The time it takes for this circuit to reach a steady voltage level is given by:

$$5\tau = 5 \cdot 3{,}300 \cdot 1 \cdot 10^{-6} = 16.5ms$$

This was also verified by simulation in Figure 5.7b.



(a) *Spice simulation verifying voltage ripple*

(b) *Spice simulation verifying settling time*

Figure 5.7: *Simulations using R = $3.3k\Omega$, C = $1\mu F$, $f_{PWM} = 200kHz$ and D = 0.5*

5.3.4. Optocoupler

As a safety precaution, the microcontroller logic operates on a low side, while the higher voltage circuitry operates on a high side. These two sides need to be electrically isolated, meaning no direct connections can be made between their ground planes. As illustrated in Figure 5.1, there is an isolation barrier where the isolated gate driver and the optocoupler transmit the necessary signals between the low and high sides.

The optocoupler is crucial for conveying the PWM reference signal from Subsection 5.3.3 from the low side to the high side, as well as for transmitting the output of the analog comparator from the high side to the low side. This bidirectional data transmission can be achieved using two optocouplers. However, for this design, a single bidirectional optocoupler was chosen for its compactness and suitability for the application. Both sides utilize 3.3V logic, with the low side powered by the 3.3V supply on the RPi Pico and the high side powered by the supply discussed in Subsection 5.3.1.

Specifically, the ACSL-7210-06RE, a high-speed CMOS bidirectional optocoupler, was used (Figure 5.8 and Table 5.3). Unlike traditional photodiodes, the CMOS technology in this optocoupler employs a CMOS photodiode, resulting in faster operation. It has a minimal pulse width distortion of only 10 ns, ensuring precise pulse width measurements. While this component is ideal for the analog comparator's output signal, it might be considered overkill for transmitting the PWM reference signal.

Table 5.3: *ACSL-7210-06RE specifications [54]*

Feature	Specification
Type	CMOS optocoupler
Number of channels	2 (bidirectional)
Data rate	25 MBd
Isolation voltage	3750 VRMS
Package	SOIC-8

Figure 5.8: *ACSL-7210-06RE*

5.4. Actual results

After manufacturing the PCB, the real-life working of the system was put to the test. One of the initial problems encountered was that the 3.3V line on the high side was not actually 3.3V. This issue

was traced to the DC-DC converter described in Subsection 5.3.1, which had a non-negligible source resistance that had not been accounted for in the design. This was easily fixed by reducing the resistance values and accounting for the source resistance, ultimately creating the 3.3V necessary for logic reference and power for the optocoupler and comparator.

A feature of the PCB is that the most relevant nodes in the circuit are broken out into 2.54 mm headers, which are exclusively used for testing purposes. The first test measurement involved checking the difference between the rise and fall time delays through the optocoupler. The delay was measured at $0.5 \cdot V_{logic} = 1.65V$. The rise time delay was found to be 37.56 ns, as displayed in Figure 5.9. Similarly, the fall time delay was 39.92 ns, as shown in Figure 5.10, resulting in a difference of $39.92\ ns - 37.56\ ns = 2.36\ ns$. This, combined with the comparator delay of $\leq 4.5\ ns$, yields a total delay of $\leq 6.86\ ns$, which is less than half a clock cycle for the digital sampling circuit discussed in Subsection 4.4.4.

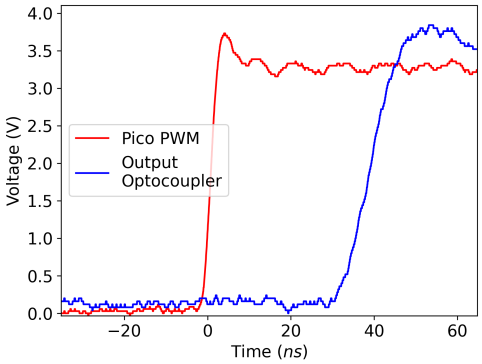


Figure 5.9: *Rise delay (37.56 ns)*

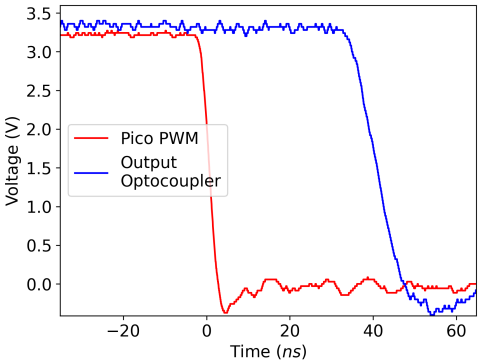


Figure 5.10: *Fall delay (39.92 ns)*

To test the comparator, a sawtooth waveform was generated using a signal generator to observe the switching point. This revealed a small flaw in the design of the analog circuitry. The hysteresis of 6 mV is not high enough for the comparator to fully reject noise (see Figure 5.11), resulting in rapid switching of its output when a small noise source is present, as evident in the sawtooth waveform shown in Figure 5.12. The theoretical noise in the reference signal is only about 1.25 mV, which could not be verified because the noise floor of the oscilloscope used was higher than this. Thermal noise from the resistors should not affect the results, as those voltages are orders of magnitude lower than the measured noise. The signal generator and the measuring equipment primarily caused the noise. This noise was greatly reduced in the digital signal when the measuring equipment was disconnected, verifying the circuit's functionality.

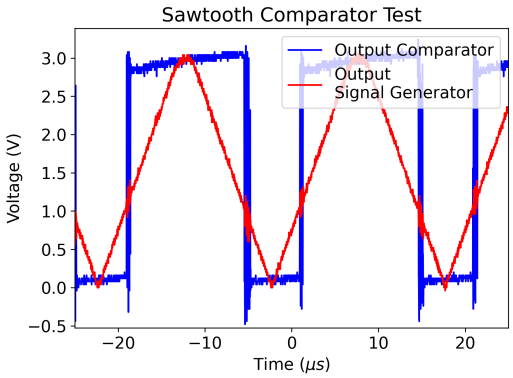


Figure 5.11: *Comparator output for 50kHz sawtooth with reference at* $\approx 1.05V$

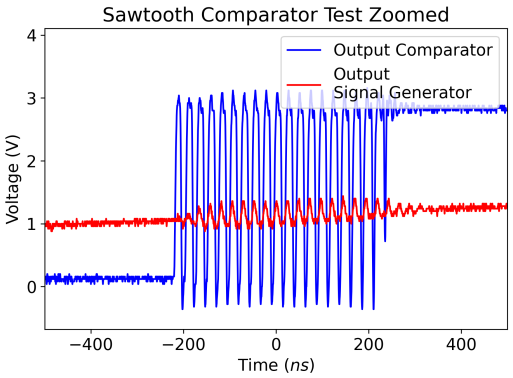


Figure 5.12: *Comparator output for 50kHz sawtooth with reference at* $\approx 1.05V$ *zoomed in at one of the rising edges of the comparator's output*

In Chapter 6, it is discussed that without the measuring equipment, the system performs well in the actual application, and none of the observed problems are present. The optocoupler effectively filters out most of this behavior, as shown in Figure 5.13 and Figure 5.14. The same delays observed earlier were also found in this case.

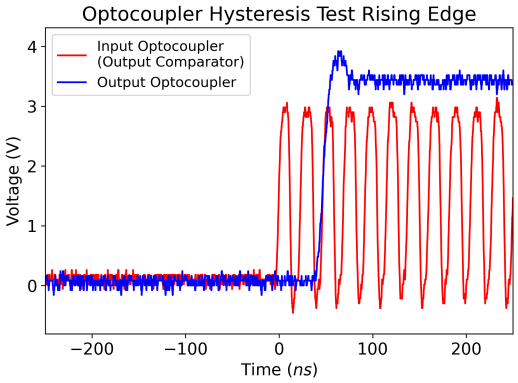


Figure 5.13: *Optocoupler filtering out most of the hysteresis problem on rising edge*

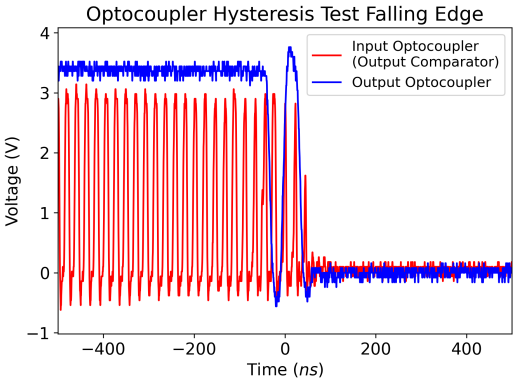


Figure 5.14: *Optocoupler filtering out most of the hysteresis problem on falling edge*

6
Experimental results

A couple of things were changed compared to the final circuit in Chapter 3. Firstly, the pulse width is increased to 10 μs, since it was found that this gives better sparks [6]. Secondly, the current-limiting resistor is changed to 3.3Ω because the performance of the EDM is a lot better when the current is higher [6]. This is done by putting three of the 10Ω power resistors in parallel. The extra cables give a bit more parasitic inductance, but the inductances of the resistors are also in parallel, so the inductance should drop to a third of the original value of 6.4μH. The latter was measured to be around 2 μH, which is as expected. The downside of a lower resistance is less damping of the oscillations. Since the lower inductance did not compensate this effect fully the filtering capacitor was increased to 47 μF. The new circuit can be seen in Figure 6.1.

6.1. High voltage tests

The circuit could now be tested at higher voltages. It was first tested at 30V since that is the highest voltage the sources in the lab can produce without using a converter and the filtering capacitor can only handle 35V. The result is plotted in Figure 6.2.

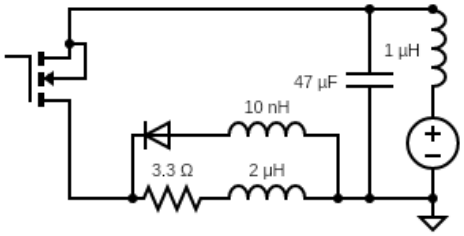


Figure 6.1: *The new circuit*

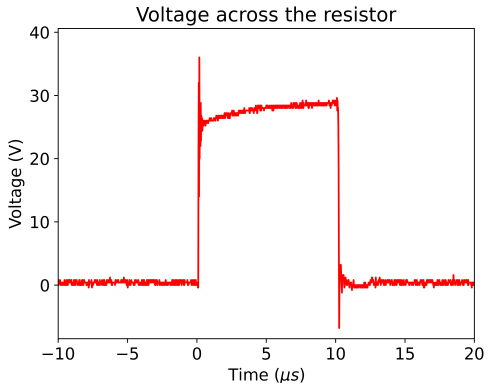


Figure 6.2: *Measurement at 30V*

The result is quite good, the noise is still reduced. However, the lower resistance causes the turn-on spike to be a lot higher compared to Figure 3.22. The peak is around 35V, so at 100V it should be approximately 130V, which is acceptable.

Since the circuit works properly, it is ready to be tested at 100V. This voltage is created using a converter and a rectifier as voltage sources producing 100V directly could not be made available. In this report, this circuitry has been considered as a black box as the Electrode and Dielectric subgroup has been responsible for this circuit [6]. All components were already able to withstand this high voltage, except for the filtering capacitor, which could only handle 35V. Therefore a larger capacitor was selected, this one can handle 350V with the same capacitance as before. The output voltage was again measured, and the result is plotted in Figure 6.3.

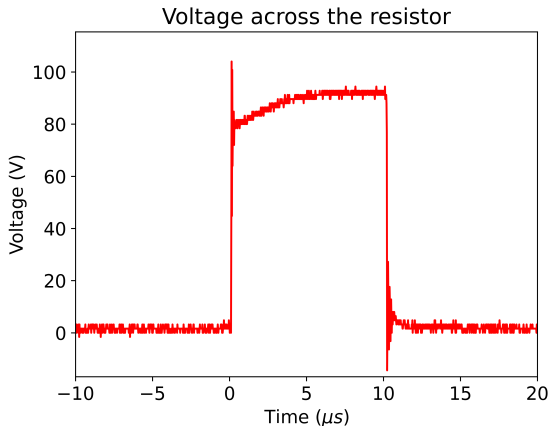


Figure 6.3: *Measurement at 100V*

The result is better than expected since the peak is only around 105V instead of 130V. The main difference is the fact that the voltage settles at a bit above 90V, which means that almost 10V is lost due to parasitic resistances and inductances. This effect is also present in Figure 6.2, but it is less noticeable in that figure. Fortunately, this can be compensated easily by setting the voltage at 110V or even higher, so it is no real problem.

6.1.1. Connecting the electrode

Since the basic circuit works very well, the electrode and the workpiece were connected between the resistor and ground. For testing purposes, the electrode was shorted to the workpiece. For safety reasons, the voltage was first set back to 30V. The result is shown in Figure 6.4.

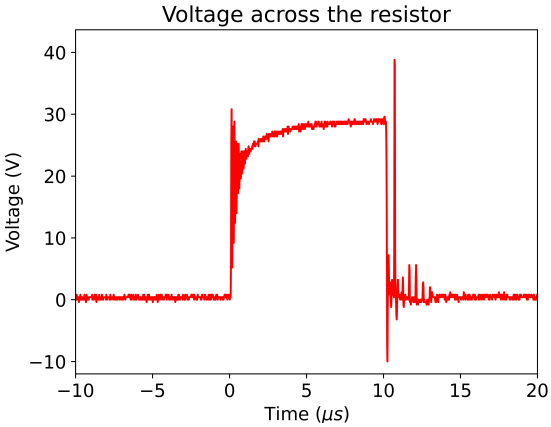
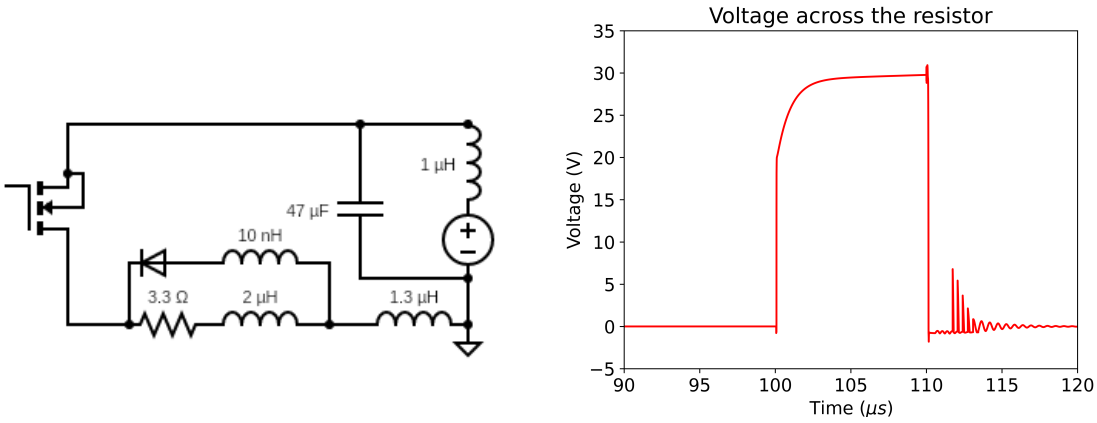


Figure 6.4: *Measurement with electrode at 30V*

The spike at turn-on is lower than before, but there are very large spikes after turn-off again. These are probably caused by the wires used to connect the electrode and workpiece to the circuit. The inductance of these wires was measured to be 1.3 μH. This is now included in the circuit, as shown in Figure 6.5a. This circuit was simulated at 30V, the results are plotted in Figure 6.5b.

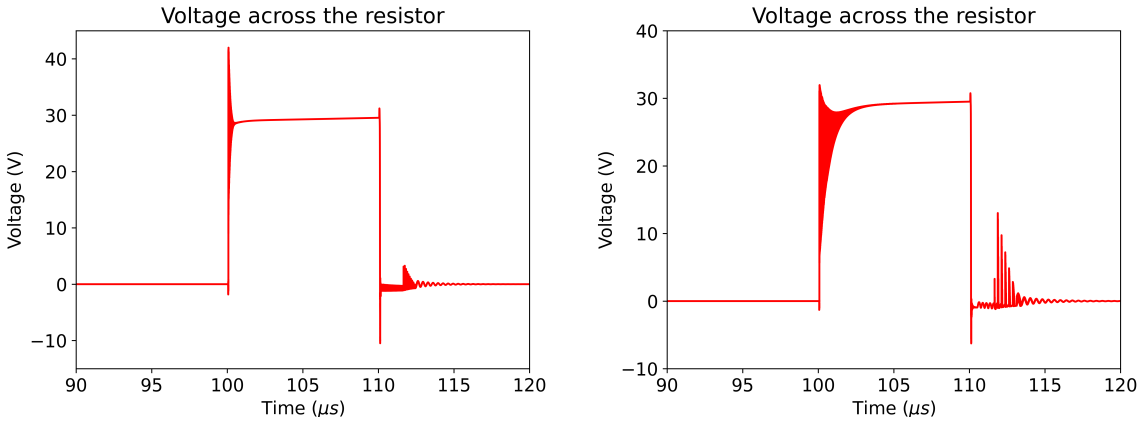


(a) *Circuit with extra cable inductance*

(b) *Simulation results*

Figure 6.5: *Model and simulation result of the electrode cable*

The simulation result partly matches the measurements from Figure 6.4. The falling slope is very similar, but the spike at turn-on is missing and the spikes at turn-off are a lot smaller, only 7V compared to almost 40V. This means that the simulation model is not complete. Power resistors often have parasitic capacitances as well [55]. When a capacitor of 1 pF is added in parallel to the resistance the simulation is more accurate. Two simulations were run to show this, the results are shown in Figure 6.7. The simulation without the electrode can be seen in Figure 6.6a and with the electrode in Figure 6.6b.



(a) *Simulation without electrode*

(b) *Simulation with electrode*

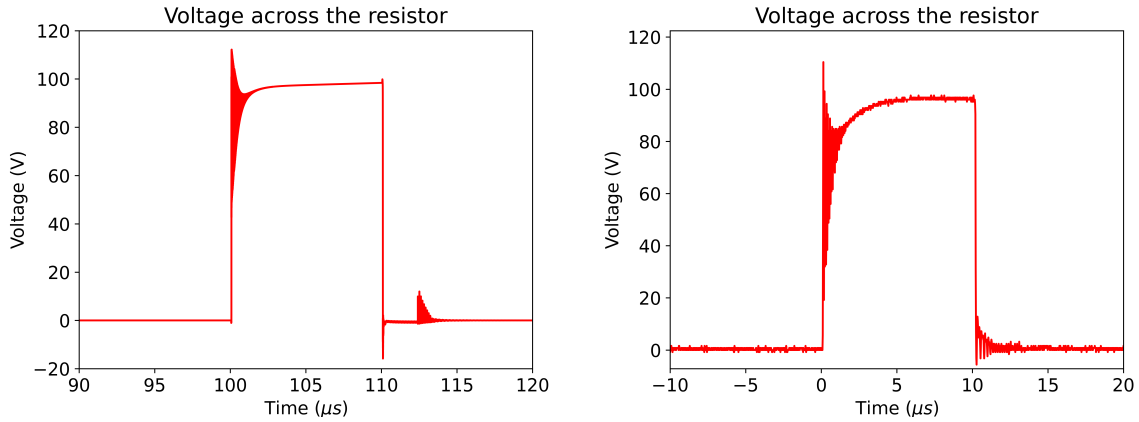
Figure 6.6: *Improved simulation results*

These results are a lot more accurate than before. Especially the simulation without the electrode matches the measurement in Figure 6.4 very well. The simulation with the electrode is more accurate too, as it now shows the spike at turn-on and the small oscillations as well. However, the spikes after turn-off are still a lot smaller than the measurement. This could be the case because the wire inductance has not been measured correctly, or the response time of the diode has not been modelled correctly. However, the result has improved as the largest spike after turn-off is now 13V instead of 7V.

The most obvious solution to the spikes is adding another filtering capacitor. However, this is not optimal, since a capacitor in parallel to the gap should be avoided. The reason for this is that the

sparks will be weaker if the capacitor is not fully charged, so strong sparks can only be formed when the capacitor is full. This slows down the frequency and therefore the material removal rate a lot.

If the spikes in Figure 6.4 would scale linearly, they should become around 130V when the voltage is increased to 100V. This would still give good performance, so a capacitor would not be necessary. To check this, the simulation was run at 100V. The result is plotted in Figure 6.7a.



(a) *Simulation result at 100V*

(b) *Measurement at 100V*

Figure 6.7: *Results at high voltage*

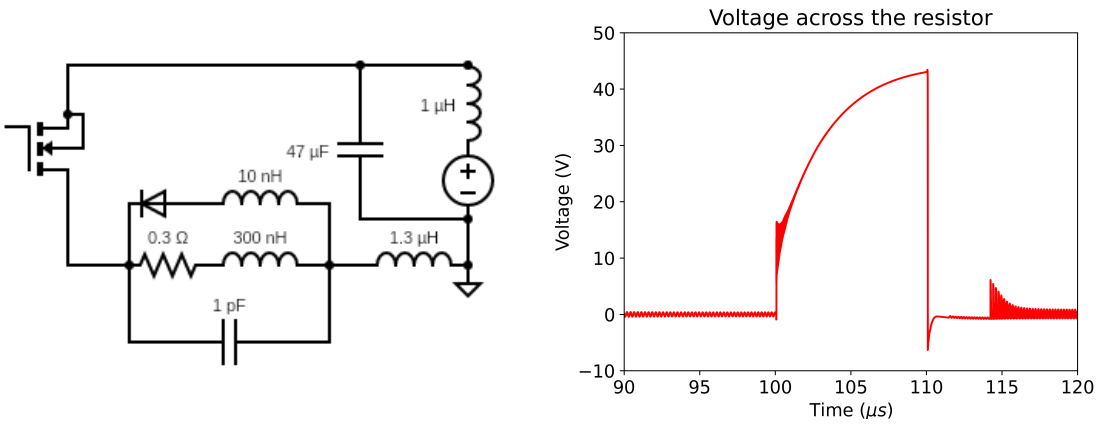
Surprisingly, the spikes after turn-off are still around 15V. This means that the spikes do not increase when the voltage is increased. To check whether this is the case in reality as well, the circuit with the shorted electrode was again measured. The voltage was increased slowly from 30V to 100V to make sure that the spikes would not blow up anything. The result is shown in Figure 6.7b.

The output is very good. The spike at turn-on is higher than the simulation result, but the spikes after turn-off are almost completely gone. To find an explanation for the first, more investigation has been done in the simulation. It turns out that the origin of these spikes primarily comes from the parasitic inductance of the electrode and workpiece wire. The capacitor discharges when the NMOS is on, and charges when it is off. At the boundary of these events, a small resonance in the capacitor voltage could be observed due to the wire inductance, filter capacitance and current-limiting resistance. This causes the diode current to oscillate as well after turn-off, weakening the freewheel effect. At times when it becomes zero, the freewheel effect is completely gone, resulting in the voltage spikes from the inductances. It turns out that at higher voltages like 100V, the resonance disappears. This might be explained by the fact that a higher input voltage charges the capacitor faster, reducing the time it spends in a state sensitive to resonance.

6.1.2. Low-resistance implementation

During testing the resistance was lowered even further, since a high current gives stronger sparks. Eventually, the resistance was set to only about 0.4Ω. This was accomplished by putting four small power resistors of 1.5Ω in parallel, which lowers the inductance to a value of around 300 nH. These resistors have a power rating of 20W to 30W. If the voltage would still be 100V, this would result in currents up to 250A. This is above the maximum rating of the switch [22] and it would result in high power dissipation in the resistors, so the voltage should be lowered. Therefore the voltage was set to 60V. Because the current through the parasitic inductance of the electrode cables will be very high, a freewheel diode was added to ensure the inductance can discharge safely. The circuit is shown in Figure 6.8a. To determine whether the circuit can handle this, the circuit was simulated with this

resistance and inductance. The result is plotted in Figure 6.8b.



(a) *Circuit with lower resistance* **(b)** *Simulation result at 60V*

Figure 6.8: *Simulation model and results of the low-resistance implementation*

The voltage is quite low now since the parasitic impedances are significant. Moreover, the waveform is different, since the induced voltage across the inductance of the wires will decay more slowly due to the lower resistance. Also, the resistance has now become quite low, comparable to some other components' resistances, causing the voltage drop across those to be larger as well relative to the power resistor's resistance. Because of this, it will take a bit longer to achieve dielectric breakdown, but this is not a problem since the sparks will be very intense. The maximum voltage is around 45V, which means that the peak current is approximately 115A. This is far below the maximum ratings of the switch, so the circuit should be able to handle it. Therefore the circuit was built and tested, and the setup can be seen in Figure 6.9 (compare with Figure 1.5). The converter and rectifier from the Electrode and Dielectric subgroup are visible in the middle. In the appendix, close-up figures of the PCB can be found.

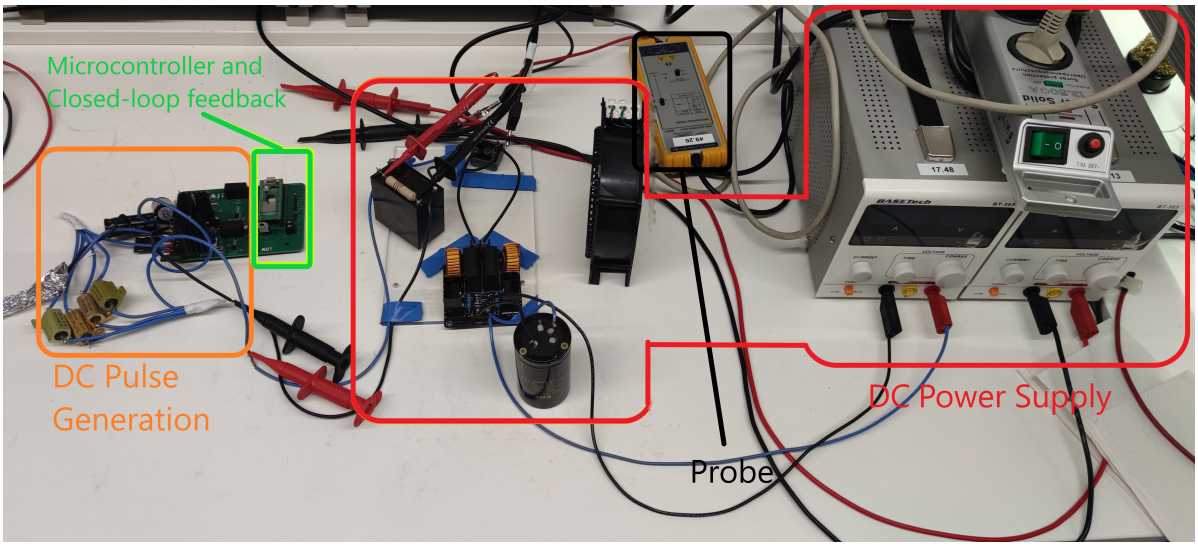


Figure 6.9: *Setup of the power supply scope*

The measurement of the output voltage can be seen in Figure 6.10.

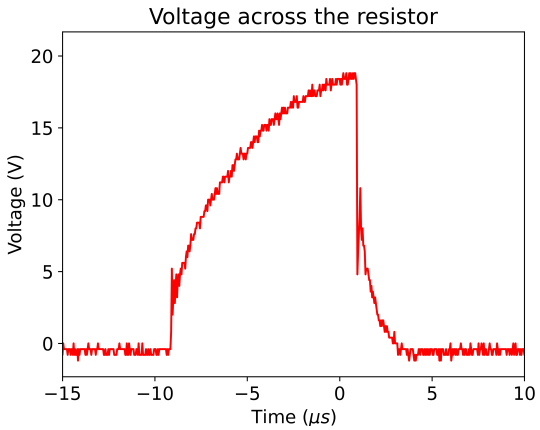


Figure 6.10: *Measurement at 60V with low resistance*

The waveform of the measurement is similar to the simulation, however, the voltage is a lot lower, the maximum value is around 18V. Apparently, the impedances in reality are higher than in the simulation model. This is not a big problem, since this measurement was taken while the electrode was shorted to the workpiece. When the gap is open the voltage will be higher, since this was also the case before, as seen in Figure 6.11 and Figure 6.12. Therefore the voltage will be high enough to cause breakdown with very intense sparks since the current will be very high.

6.2. Closed-loop feedback

With the DC pulse system consistently producing sparks at higher voltages, the closed-loop feedback system designed in Chapter 5 can be tested. Everything was verified to be working correctly, and it performed even better than expected.

One consequence of using only a transistor to switch the voltage on and off is that the voltage over the electrode floats when the switch is off. When in the air, the voltage remains high as it has nowhere to discharge, giving the microcontroller a constant logical high. However, when the electrode just touches the deionized water, the floating potential slowly dissipates, causing the voltage to drop gradually, as shown in Figure 6.11. This voltage drop is most likely due to the DC current from electrolysis and occurs much faster the closer the electrode is to the workpiece, as seen in Figure 6.12. This change can be attributed to the reduced gap width, resulting in decreased gap resistance, as described in Equation 6.1 [56].

$$R = \frac{\rho l}{A} \tag{6.1}$$

where ρ is the deionized water's conductivity, l the gap width and A the cross-sectional area of the electrode.

Although this will not help the system regulate the speed of the electrode when it is sparking, it can indicate how far the electrode needs to move to get close enough to the workpiece, thereby saving setup time. When sparks are created, the voltage almost instantly drops to zero, which is successfully measured by the microcontroller, as discussed in Subsection 4.4.4. This feature is currently implemented primarily for short circuit detection (**EDM_PS_F_6**), but with additional time, it could be expanded to analyze each pulse more closely and regulate the speed even more precisely.

This confirms the functionality of the closed-loop feedback system, which has also been verified to communicate successfully with the speed regulator using the I2C connection.

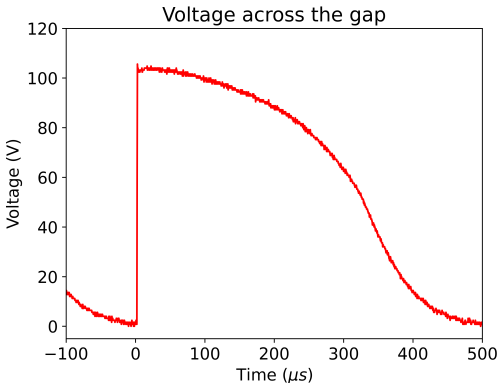


Figure 6.11: *2 kHz 2% duty cycle with electrode far away from the workpiece*

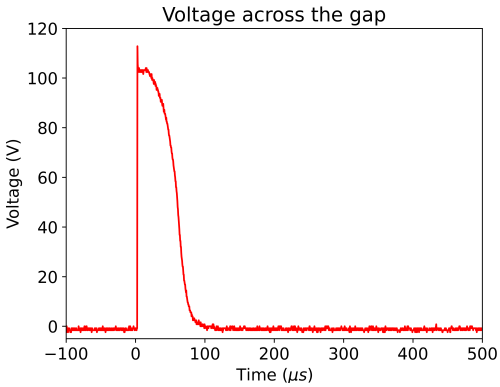


Figure 6.12: *2 kHz 2% duty cycle with electrode close to (but not touching) the workpiece*

6.2.1. Test of basic EDM

The basic EDM was tested using the setup shown in Figure 6.13. The 3D printer was not turned on during this test, so the electrode stood still. Instead, the plastic box containing the workpiece was moved by hand to create a large hole in less than five minutes. This hole is shown in Figure 6.14.

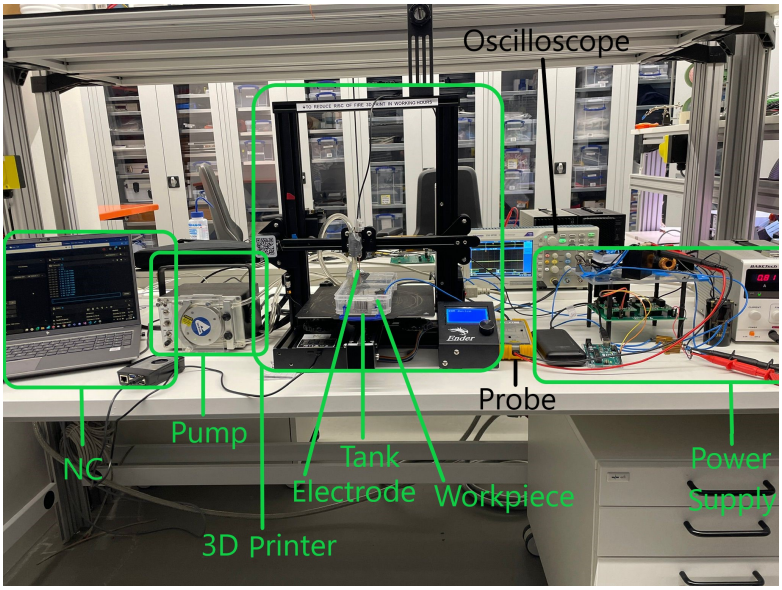


Figure 6.13: *Real-life implementation of the EDM setup*

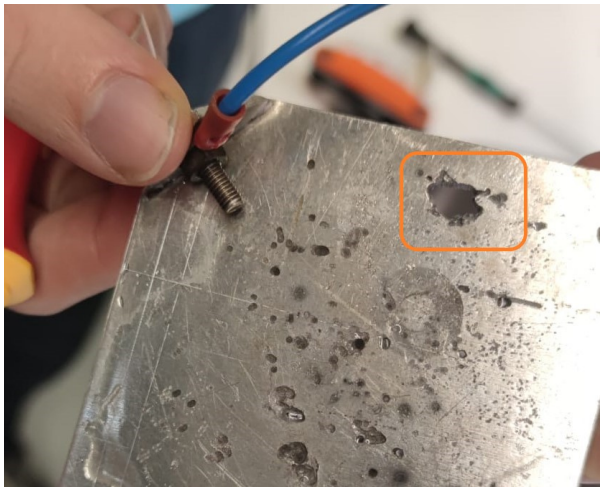


Figure 6.14: *Hole drilled by hand*

6.3. Electromagnetic interference

The main reason the feedback loop is not more involved than the short detection is due to the significant challenges encountered during testing. These challenges arose from switching around a hundred amps on and off at high frequencies. The electromagnetic interference (EMI) generated by the high current changes in the cables and/or the sparks caused the connection between the 3D printer's control circuitry and the Raspberry Pi 5 (used for communicating the GCode and feedback signal) to drop.

Numerous precautions were taken to mitigate these issues, such as using shielded USB cables, removing non-essential cables to the control circuitry, twisting wires carrying high current and shielding them, shielding the workpiece area, and physically distancing sensitive electronics. These measures resulted in the test setup shown in Figure 6.15. Despite these efforts, the setup still caused an unstable connection, hindering further optimization tests. Likely, using separate stepper drivers and microcontrollers to replace the necessary stock 3D printer hardware would solve some of these problems, but this is beyond the scope of this project.

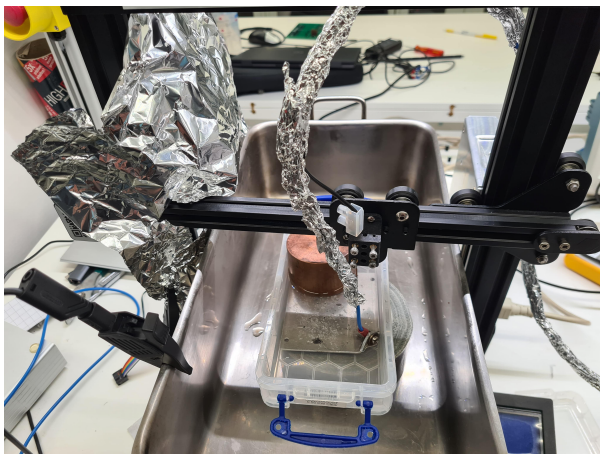


Figure 6.15: *Test setup with EMI measures*

6.3.1. Wireless communication feedback system

Part of removing all potentially problematic cable connections involved eliminating the ”long” I2C cables, which could disrupt operation. Fortunately, a version of the microcontroller board (Pico W [57]) features a wireless modem (CYW43439) capable of WiFi and Bluetooth. This inspired the idea to replace the I2C connection with a wireless one. The Raspberry Pi 5 already uses a wireless network to communicate the web interface to its users. Therefore, the RPI Pico W can directly connect to the same network to communicate with the RPI 5.

This communication is achieved by hosting a Transmission Control Protocol (TCP) server on the Pico (see Listing B.6) and having Klipper (the software controlling the 3D printer and the connection point for the closed-loop system) act as the TCP client to facilitate two-way communication. A schematic view of this setup replacing I2C is given in Figure 6.16. This TCP server is run on a separate processing core to prevent it from blocking the microprocessor during operation.

Although this solution did not resolve the EMI problem, it enabled the power supply to be physically separated from the printer controller. Additionally, it makes the data to and from the power supply accessible to all devices on the network, rather than being limited to the specific RPi 5 used for controlling the 3D printer.

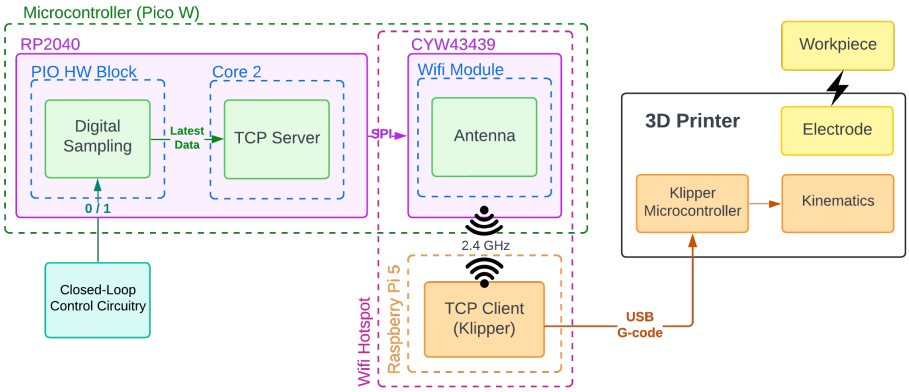


Figure 6.16: *Schematic Overview Wifi implementation replacing I2C connection*

6.4. GCode compatibility

As part of **EDM_PS_ST_5**, the system should be controllable using GCode. This means that the frequency and duty cycle of the DC pulses and the reference voltage for the closed-loop system should be controllable via GCode. The same TCP connection from Subsection 6.3.1 can be used to handle these commands. The Electrode and Dielectric subgroup parsed Klipper GCode commands into the correct string format (e.g., freq=xxxx,duty=xx) [6], which is then sent via the TCP connection. The microcontroller receives this data and sets the parameters to the correct values, fulfilling the GCode Compatibility requirement. This approach significantly increases the feasibility of testing multiple parameters quickly and efficiently, eliminating the need to hard code these values and recompile the microprocessor each time.

7 Conclusion

The objective of this thesis is to guide the user through the development process of the power supply for an Electric Discharge Machine (EDM). The design process began by establishing requirements as a foundation for development. The primary objective was to ensure proper transistor logic and the generation of clean high-voltage pulses. This proved to be more challenging than initially anticipated. After testing multiple configurations using diodes and capacitors to compensate for parasitic inductances in the circuit, this objective was achieved. As a result, requirements **EDM_PS_F_2** through **EDM_PS_F_7** were met. The circuit was then implemented on a PCB (**EDM_PS_ST_2**) with three different ground pours (**EDM_PS_ST_1**).

As this board is intended for parameter optimization, it requires parameters that can be adjusted. All parameters are controllable by a microcontroller (**EDM_PS_ST_3**), ensuring rapid adjustments without the need for hardware modifications (**EDM_PS_F_8** through **EDM_PS_F_11**). This resulted in consistent pulses (**EDM_PS_P_1**). The main contribution of this thesis is the implementation of a closed-loop system and its documentation (**EDM_PS_ST_4**). This is achieved through a microcontroller that digitally detects the ignition delay time of sparks. It requires supporting hardware to convert the analog signal to digital and to serve as an isolated bridge between the signal side and the high-voltage side of the PCB.

Every subsystem has been designed, tested, and is fully ready as a platform for parameter optimization, meeting all the initially defined requirements. The remaining requirements, not previously mentioned, were implemented during integration with the other subgroup. This integration proceeded smoothly, except for electromagnetic interference issues causing disconnects, which hindered further testing and optimization. All critical parameters can be easily adjusted using GCode and transmitted wirelessly. The resulting spark performance of the system significantly exceeds that of the previously used half-bridge design, while also being notably smaller and safer, marking the first version of the power supply as a resounding success.

7.1. Discussion

Despite the addition of several parasitic components to the simulation, the results still differ somewhat from the measurements. There are two main reasons for this. Firstly, the model used for the gate driver is not that of the selected gate driver, but a similar one, since there was no SPICE model of the STGAP2SICS available on the internet. Secondly, not all parasitics have been taken into account, either because the simulation became too difficult or because determining these parasitics was beyond the scope of this project. However, significant reductions in differences between the simulation and the measurements have been achieved by enhancing the simulation circuit to a degree where the results are sufficiently accurate.

The delivery of the PCB significantly impacted our planning, necessitating adjustments. Fortunately, this time could be utilized for conducting additional simulations to gain further insights into the circuit for DC pulse generation. However, this meant that integration with the other subgroup could not be further optimized. The electromagnetic interference also greatly hindered progress, as valuable time was spent trying to solve this problem instead of optimizing the closed-loop control system.

One of the requirements, particularly the necessity for short detection (**EDM_PS_F_6**), was moved to be integrated into the microcontroller closed-loop system, as it inherently incorporates such a feature.

Given that the power supplies used are all external, the power switch (**EDM_PS_F_1**) and the emergency switch (**EDM_PS_F_7**) should not have been requirements of the EDM power supply. However, they are likely already present in these power supplies.

The hysteresis of the analog comparator should have been designed to be more noise-resistant from the beginning. It is most likely that the measuring system and test setup caused the signal to oscillate, but it would have been sensible to increase this to around 50 mV for added safety.

In hindsight, SMD resistors should have been used instead of through-hole components. This would have greatly reduced the design's size. Also, the size of the PCB is bigger than it needs to be due to most nets being broken out to a pin header for testing purposes.

The isolated DC-DC converter used to generate the 3.3V signal is not ideal, as it is specifically designed for gate drivers. This resulted in issues due to a non-negligible source resistance, leading to voltage levels different from what was expected.

7.2. Future work

Improvements can be pursued in several areas, starting with the creation of a SPICE model for the STGAP2SICS. This would enhance simulation accuracy, facilitating future design extensions.

On the PCB front, lessons were learned to optimise external connector placement for enhanced usability. Eliminating probe points in favor of more compact solutions can further improve space utilization. Integration of additional components directly onto the PCB, such as the microcontroller, would enhance the overall design's elegance and functionality. Likewise, incorporating the DC source and associated circuitry directly onto the PCB would make a more polished, production-ready product.

Moreover, cost optimization is another avenue for improvement. Time was more valuable which influenced the selection of certain components, exploring more budget-friendly alternatives for larger-scale production could be beneficial.

Overall, these enhancements aim to refine the design, increase performance, improve space efficiency, and pave the way for future scalability and cost-effectiveness when this power supply is not a testing platform.

The kinematics setup should also be revised with more robust EMI protection with purposely written code specifically for this type of machine. The 3D printer setup should also be looked at to see if improvements can be made in precision in the realm of micro stepping for stepper motors and overall mechanical precision.

Bibliography

[1] K. Ho and S. Newman, “State of the art electrical discharge machining (edm),” *International Journal of Machine Tools and Manufacture*, vol. 43, no. 13, pp. 1287–1300, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0890695503001627

[2] Massachusetts Institute of Technology, “What is plasma?” [Online]. Available: https://www.psfc.mit.edu/vision/what_is_plasma

[3] M. Kunieda, B. Lauwers, K. Rajurkar, and B. Schumacher, “Advancing EDM through Fundamental Insight into the Process,” *CIRP Annals*, vol. 54, no. 2, pp. 64–87, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007850607600201

[4] A. Descoeudres, “Characterization of electrical discharge machining plasmas,” p. 137, 2006. [Online]. Available: http://infoscience.epfl.ch/record/84931

[5] PocoGraphite, “Edm technical manual.” [Online]. Available: https://www.edmtechman.com/about.cfm?pg=2&chap=2#a1

[6] T. Qualm, S. Mahmoud, and R. Helmer, “Design and prototyping of an electrostatic discharge machining (edm) device,” TU Delft, Tech. Rep., 2024, unpublished report.

[7] S. Saha, “Experimental investigation of the dry electric discharge machining (dry edm) process,” Ph.D. dissertation, 01 2008.

[8] F. Han, S. Wachi, and M. Kunieda, “Improvement of machining characteristics of micro-edm using transistor type isopulse generator and servo feed control,” *Precision Engineering*, vol. 28, no. 4, pp. 378–385, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141635904000121

[9] BaxEDM, “Baxedm bx18 manual rev08.” [Online]. Available: https://www.baxedm.com/wp-content/uploads/2023/05/BaxEDM-BX18-Manual-Rev08.pdf

[10] T. Muthuramalingam and B. Mohan, “Influence of discharge current pulse on machinability in electrical discharge machining,” *Materials and Manufacturing Processes*, vol. 28, no. 4, pp. 375–380, 2013. [Online]. Available: https://doi.org/10.1080/10426914.2012.746700

[11] A. Equbal and A. K. Sood, “Electrical discharge machining: An overview on various areas of research,” *Manufacturing and Industrial Engineering*, vol. 13, 09 2014.

[12] J. Xu, K. Wang, Y. Liu, and Q. Zhang, “Evaluation of energy consumption and carbon emission in edm,” *The International Journal of Advanced Manufacturing Technology*, vol. 132, no. 3, pp. 1511–1524, 05 2024.

[13] M. Jahan, Y. Wong, and M. Rahman, “A study on the quality micro-hole machining of tungsten carbide by micro-edm process using transistor and rc-type pulse generator,” *Journal of Materials Processing Technology*, vol. 209, no. 4, pp. 1706–1716, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924013608003440

[14] M. Zahiruddin and M. Kunieda, “Comparison of energy and removal efficiencies between micro and macro edm,” *CIRP Annals*, vol. 61, no. 1, pp. 187–190, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000785061200008X

[15] 5.3 product verification. [Online]. Available: https://www.nasa.gov/reference/5-3-product-verification/#hds-sidebar-nav-6

[16] R. Robotics. Powercore-v1.0-hardware: Cad electronics for powercore v1. [Online]. Available: https://github.com/Rack-Robotics/Powercore-V1.0-Hardware

[17] B. Bhattacharyya and B. Doloi, "Chapter four - machining processes utilizing thermal energy," in *Modern Machining Technology*, B. Bhattacharyya and B. Doloi, Eds. Academic Press, 2020, pp. 161–363. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128128947000049

[18] M. Xu, R. Wei, C. Li, and T. J. Ko, "High-frequency electrical discharge assisted milling of inconel 718 under copper-beryllium bundle electrodes," *Journal of Manufacturing Processes*, vol. 85, pp. 1116–1132, 01 2023.

[19] B. Fleming, *The EDM How-to Book*. Fleming Publications, 2005. [Online]. Available: https://books.google.nl/books?id=X6JRAgAACAAJ

[20] T. Muthuramalingam and B. Mohan, "Performance analysis of iso current pulse generator on machining characteristics in edm process," *Archives of Civil and Mechanical Engineering*, vol. 14, no. 3, pp. 383–390, Sep 2014. [Online]. Available: https://doi.org/10.1016/j.acme.2013.10.003

[21] P. Electronics, "Power electronics - mosfet power losses," 2018. [Online]. Available: https://www.youtube.com/watch?v=KSOHwVoxpzg

[22] Infineon Technologies AG, "650 V CoolMOS™ C6 Power Transistor," 2011. [Online]. Available: https://www.infineon.com/dgdl/Infineon-IPW65R037C6-DS-v02_00-en.pdf?fileId=db3a3043337a914d0133877a719210a9

[23] R. P. Ltd, "Raspberry pi pico datasheet," 2024. [Online]. Available: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf

[24] STMicroelectronics, "STGAP2SICS Datasheet," 2022. [Online]. Available: https://www.st.com/resource/en/datasheet/stgap2sics.pdf

[25] Murata Power Solutions, "MGJ2D241505SC Datasheet," 2017. [Online]. Available: https://octopart.com/datasheet/mgj2d241505sc-murata+power+solutions-49448198

[26] L. Balogh, "Fundamentals of MOSFET and IGBT Gate Driver Circuits," Texas Instruments Incorporated, Tech. Rep. SLUA618A, 2017. [Online]. Available: https://www.ti.com/lit/ml/slua618a/slua618a.pdf

[27] Biricha, "Mosfet gate drive resistor selection - part 1: Turn on," 2023. [Online]. Available: https://youtu.be/fsgKpAq2gd0?si=gyF-juyRZ3qYoNkL

[28] ARCOL, "Hs aluminium housed resistors," Tech. Rep. [Online]. Available: https://www.farnell.com/datasheets/2661022.pdf

[29] "Power MOSFET Electrical Characteristic," Toshiba, Tech. Rep., 2023. [Online]. Available: https://toshiba.semicon-storage.com/info/application_note_en_20230209_AKX00063.pdf?did=13415

[30] Analog Devices, "Switching Inductive Loads With Safe Demagnetization," 2017. [Online]. Available: https://www.analog.com/en/resources/technical-articles/switching-inductive-loads-with-safe-demagnetization.html

[31] Wolfspeed, "C3D02060F, 3rd Generation 600 V, 2 A Silicon Carbide Schottky Diode," 2023. [Online]. Available: https://assets.wolfspeed.com/uploads/2023/11/Wolfspeed_C3D02060F_data_sheet.pdf

[32] T. Instruments, “Driving inductive loads with power switches,” 2019. [Online]. Available: https://www.ti.com/video/6018730150001

[33] R. P. Ltd, “Getting started with raspberry pi pico,” 2024. [Online]. Available: https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf

[34] ——, “Rp2040 datasheet,” 2024. [Online]. Available: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf

[35] R. Pi, “Raspberry pi c sdk documentation,” Raspberry Pi Foundation, Tech. Rep., 2024. [Online]. Available: https://www.raspberrypi.com/documentation/microcontrollers/c_sdk.html

[36] R. P. Ltd, “Raspberry pi pico python sdk,” 2024. [Online]. Available: https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-python-sdk.pdf

[37] R. Pi, “Raspberry pi pico examples,” 2024. [Online]. Available: https://github.com/raspberrypi/pico-examples

[38] R. P. Foundation, “Rp2040.” [Online]. Available: https://www.raspberrypi.com/products/rp2040/

[39] Reichelt, “Rasp pi pico raspberry pi pico, rp2040.” [Online]. Available: https://www.reichelt.nl/nl/de/raspberry-pi-pico-rp2040-cortex-m0-microusb-rasp-pi-pico-p295706.html

[40] R. Pi, “Hardware design with rp2040,” Raspberry Pi Foundation, Tech. Rep., 2020. [Online]. Available: https://datasheets.raspberrypi.com/rp2040/hardware-design-with-rp2040.pdf

[41] ——, “Raspberry pi pico sdk source code,” 2021. [Online]. Available: https://github.com/raspberrypi/pico-sdk

[42] A. Team, “Arduino ide,” 2019. [Online]. Available: https://github.com/arduino/arduino-ide

[43] D. P. George, “Micropython,” 2014. [Online]. Available: https://github.com/micropython/micropython

[44] R. E. W. Group, “Rust embedded,” 2024. [Online]. Available: https://github.com/rust-embedded

[45] STMicroelectronics, “L78 series - voltage regulators,” Tech. Rep., 2018. [Online]. Available: https://www.st.com/resource/en/datasheet/l78.pdf

[46] ——, “L7983 - 60 v, 300 ma synchronous step-down switching regulator with 10 µa quiescent current,” Tech. Rep., 2020. [Online]. Available: https://www.st.com/resource/en/datasheet/l7983.pdf

[47] Stanford University, “Ee261 - the fourier transform and its applications,” 2007, accessed: 2024-05-29. [Online]. Available: https://see.stanford.edu/materials/lsoftaee261/book-fall-07.pdf

[48] GitJer, “PwmIn,” 2024, Part of the Some_RPIPico_stuff repository. [Online]. Available: ”https://github.com/GitJer/Some_RPI-Pico_stuff/tree/main/PwmIn”

[49] L. Upton, “Don’t try this at home: Overclocking rp2040 to 1ghz.” [Online]. Available: https://www.raspberrypi.com/news/dont-try-this-at-home-overclocking-rp2040-to-1ghz/

[50] T. Instruments, “Application brief: Understanding schmitt triggers,” Texas Instruments, Tech. Rep. SCEA046A, 2022. [Online]. Available: https://www.ti.com/lit/ab/scea046a/scea046a.pdf

[51] K. O’Connor and the Klipper Community, *Klipper: 3D Printer Firmware*, 2024, accessed: 2024-05-26. [Online]. Available: https://www.klipper3d.org/

[52] L. Murata Manufacturing Co., “Kdc-mgj2 high frequency chip inductors [datasheet],” Murata Manufacturing Co., Ltd., Tech. Rep., 2022. [Online]. Available: https://www.murata.com/products/productdata/8807029997598/kdc-mgj2.pdf

[53] T. Instruments, “Tlv3501 - 100-ma, low-dropout regulator with shutdown,” Tech. Rep., 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/tlv3501.pdf

[54] Broadcom, “Acsl-7210 dual channel bidirectional optocoupler,” Broadcom Inc., Tech. Rep., 2019. [Online]. Available: https://docs.broadcom.com/doc/AV02-4235EN

[55] EEPower.com, “Resistor capacitance.” [Online]. Available: https://eepower.com/resistor-guide/resistor-fundamentals/resistor-capacitance/

[56] L. Forschner, E. Artmann, T. Jacob, and A. K. Engstfeld, “Electric potential distribution inside the electrolyte during high voltage electrolysis,” *The Journal of Physical Chemistry C*, vol. 127, no. 9, pp. 4387–4394, 2023. [Online]. Available: https://doi.org/10.1021/acs.jpcc.2c07873

[57] R. P. Ltd, “Raspberry pi pico w datasheet,” 2024. [Online]. Available: https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf

[58] “Kicad.” [Online]. Available: https://www.kicad.org/

[59] T. van den Akker. (2024) Edm powersupply. [Online]. Available: https://github.com/Tim-van-den-Akker/EDM_Powersupply

[60] ——. (2024) Edm microcontroller. [Online]. Available: https://github.com/Tim-van-den-Akker/EDM_Microcontroller

A

PCB layout

A.1. Full PCB schematic and render

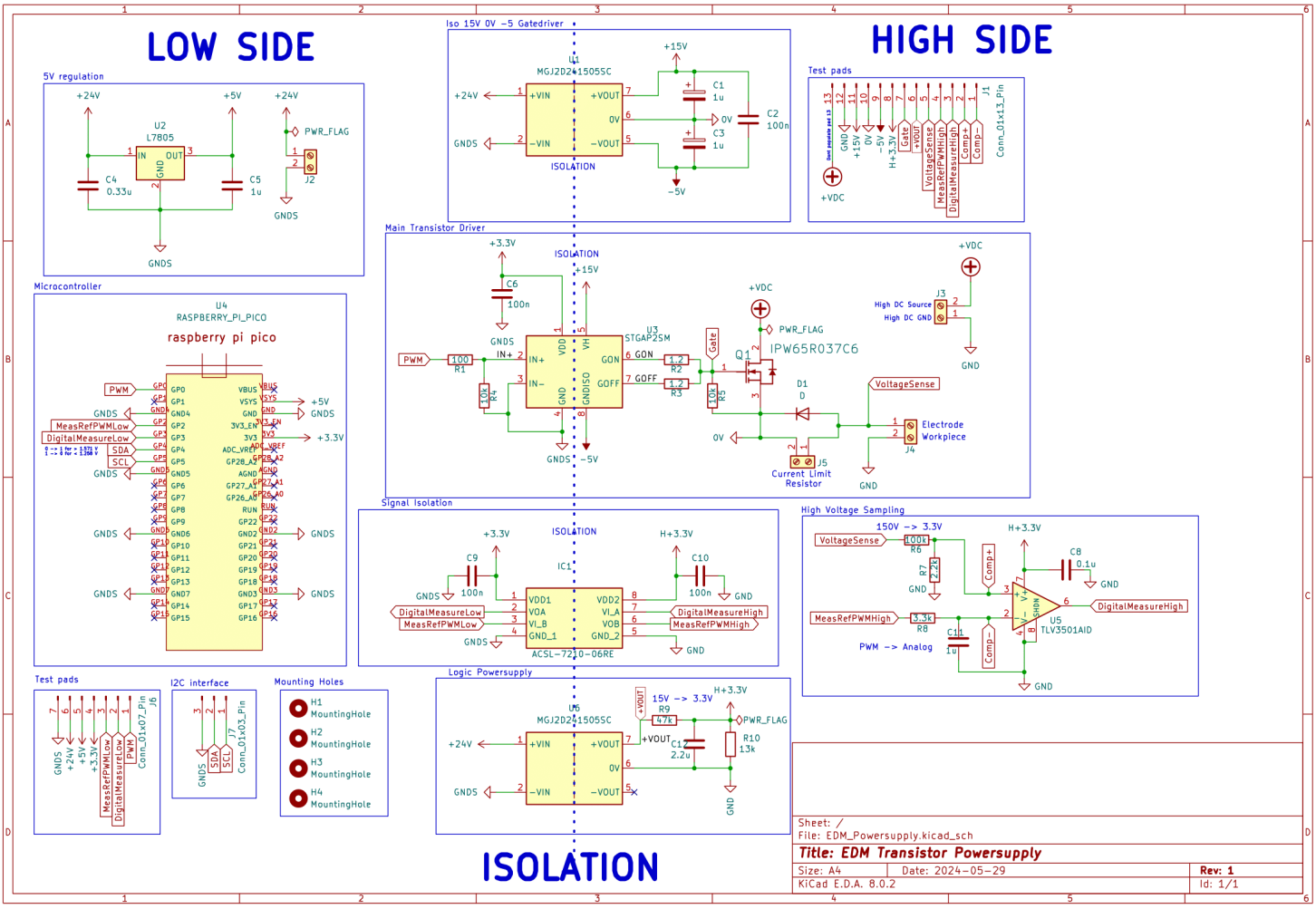


Figure A.1: *Full Schematic EDM Power Supply*

The requirements specify that the design must be implemented on a PCB (**EDM_PS_ST_2**). The separate subsystems have been developed using the open-source PCB software KiCad [58] from the outset. The complete design files, along with their history, are available on GitHub [59]. Figure A.1 provides the full schematic, while Section A.2 highlights the individual subsystems.

Figure A.2: *3D model of EDM Power Supply PCB*

In Figure A.2 a graphical version of the final PCB is shown. This is also made entirely in KiCad and uses some models from the libraries grabcad.com and snapeda.com.

A.2. Individual Schematics

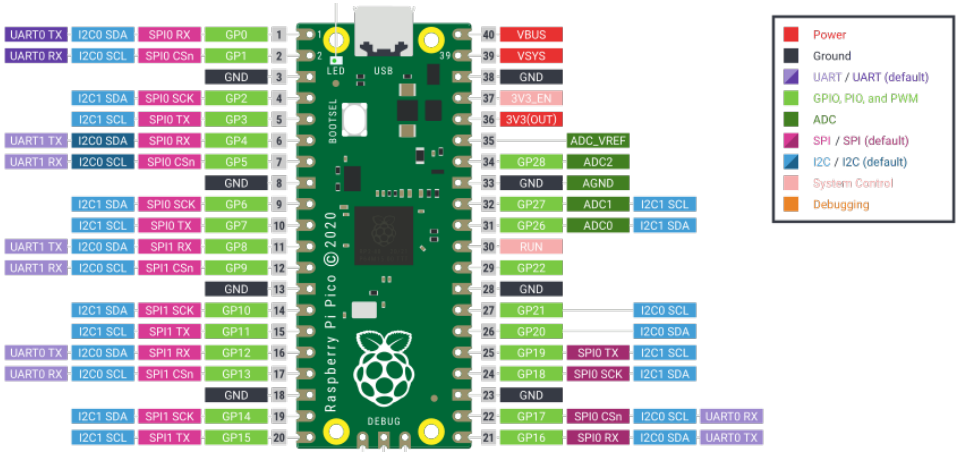


Figure A.3: *Raspberry Pi Pico Pinout [23]*

In Figure A.3, the pinout of the Raspberry Pi Pico is provided, indicating its functionality. For the first version of the PCB, GPIO 0 through 5 are utilized, along with the power functions. This pinout serves to guide the reader regarding the utilization of the RPI Pico in this design and potential future updates.

In the remainder of this chapter, a more detailed view is given of the schematics of the different subsystems. All these schematics are made using the recommended values from the datasheets. Parts specifically designed for this power supply are discussed in the relevant chapters.

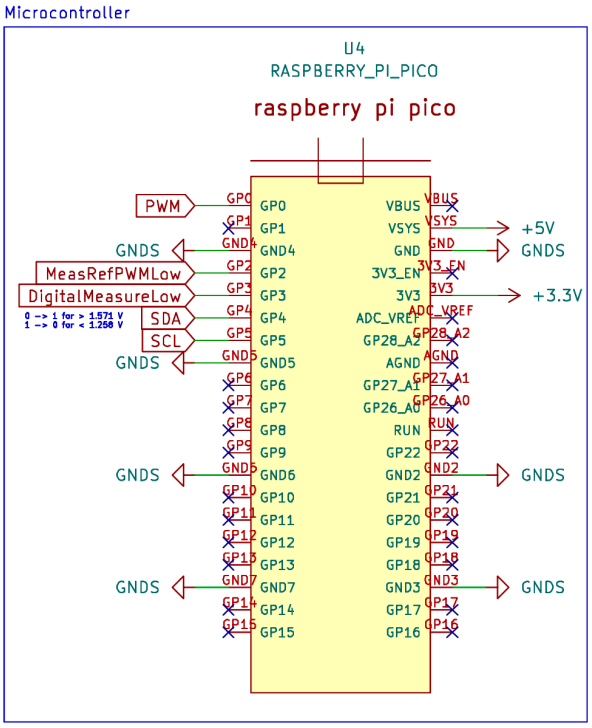


Figure A.4: *PCB Schematic Microcontroller*

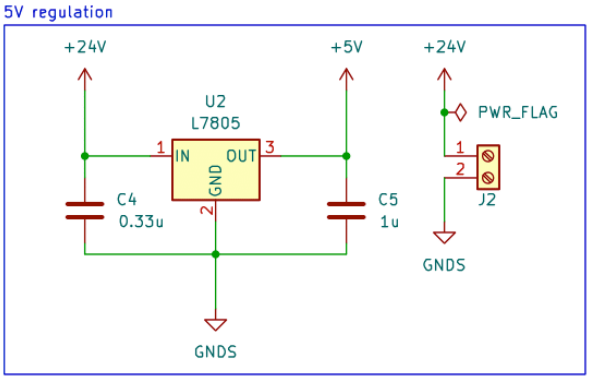


Figure A.5: *5V regulation circuit and 24V input connector*

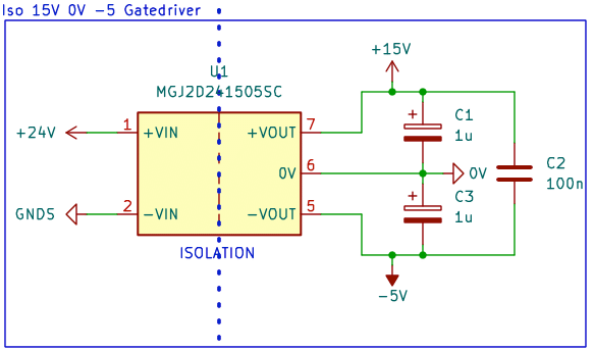


Figure A.6: *Transistor Gate Driver Power Supply*

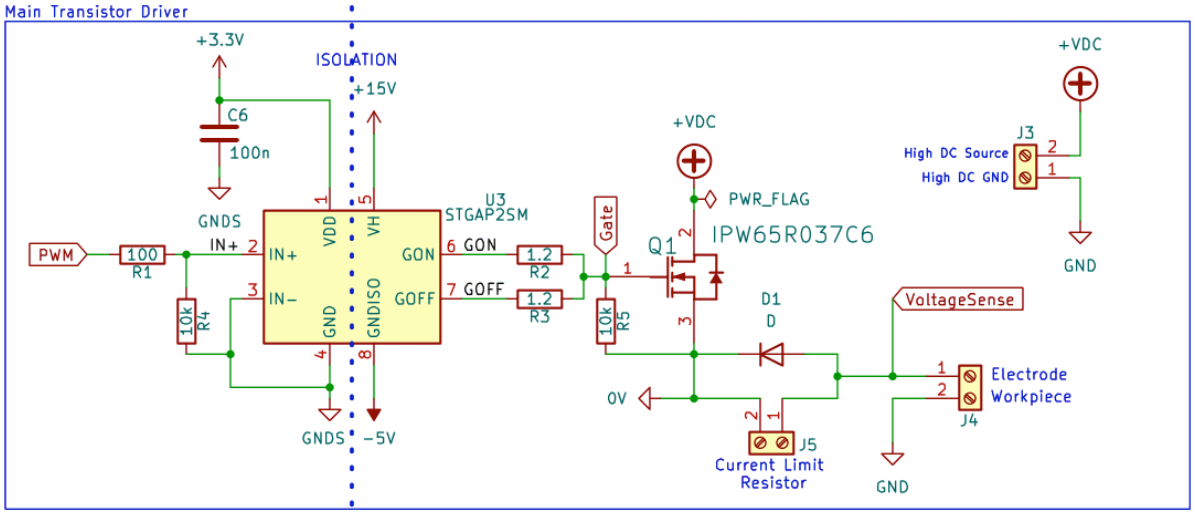


Figure A.7: *Main Transistor Logic*

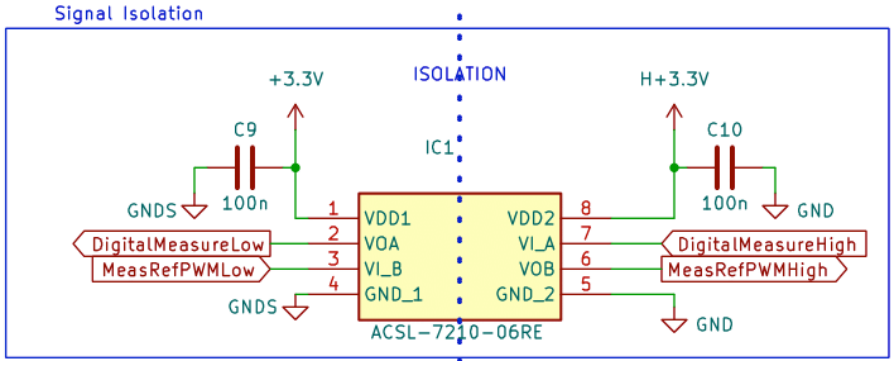


Figure A.8: *Optocoupler Circuitry*

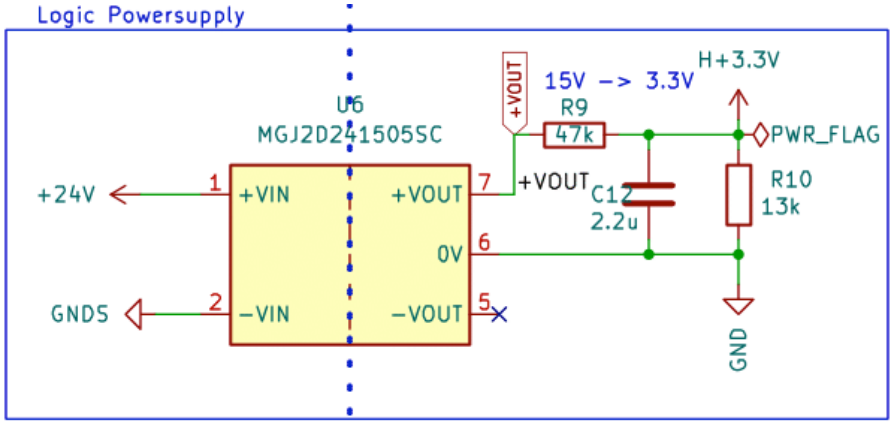


Figure A.9: *Power Supply for High Side 3.3V*

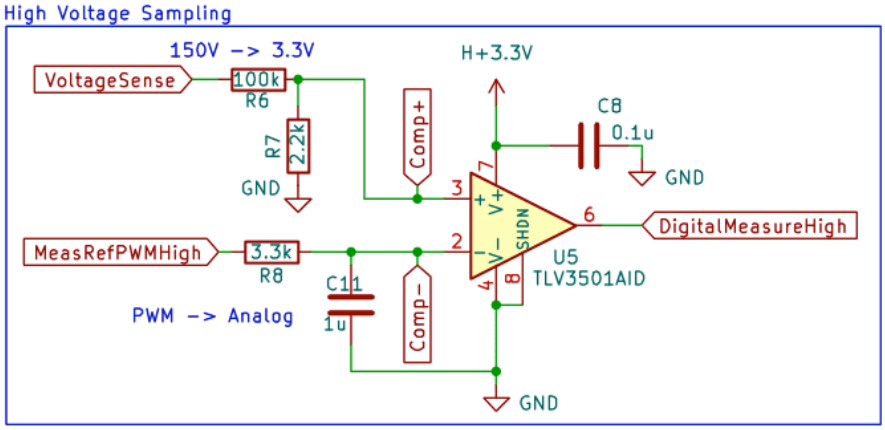


Figure A.10: *Analog Comparator Circuitry*

B

Code for Raspberry Pi Pico

This chapter presents the latest version of the code used on the Raspberry Pi Pico, along with some test scripts developed during the project. The complete design files and their history are available on GitHub [60]. Most of the code leverages the Pico-SDK [41] to access the hardware directly.

B.1. Latest Production Code

```
#include <stdio.h>
#include <string.h>
#include <time.h>

#include ”pico/stdlib.h”
#include ”hardware/dma.h”
#include ”hardware/adc.h”
#include ”hardware/pwm.h”
#include ”hardware/gpio.h”
#include ”hardware/irq.h”
#include ”hardware/i2c.h”
#include ”pico/multicore.h”
#include ”pico/i2c_slave.h”

// WIFI
#include ”pico/cyw43_arch.h”
#include ”lwip/pbuf.h”
#include ”lwip/tcp.h”


#include ”pulse_analyzer.pio.h”

// PIN SETUP FINAL PCB
#define PWM 0
#define REF_PWM 2
#define DIG_MEAS 3
#define SDA 4
#define SCL 5
#define LED_PIN 25

// WIFI SETUP
#define WIFI_SSID ”PiSetup”
#define WIFI_PASSWORD ”raspberry123”

#define TCP_PORT 4242
#define DEBUG_printf printf
#define BUF_SIZE 1

uint BASE_CLK_FREQ = 125000000;
uint GLOBAL_FREQ = 2000;

float LATEST_DUTY;
bool SHORTED = false;

uint8_t DATA_SEND;
```

Listing B.1: *Dependency import and code Setup*

```
void setup_pwm() {
    // Tell GPIO 0 and 1 they are allocated to the PWM
```

```
    gpio_set_function(PWM, GPIO_FUNC_PWM);
    gpio_set_function(REF_PWM, GPIO_FUNC_PWM);

    // Find out which PWM slice is connected to GPIO 0 (it's slice 0)
    uint slice_num = pwm_gpio_to_slice_num(PWM);
    uint ref_slice_num = pwm_gpio_to_slice_num(REF_PWM);

    uint PWM_FREQ = 2000;
    uint PWM_DUTY = 2;

    uint REF_PWM_FREQ = 100000;
    uint REF_PWM_DUTY = 40;

    uint wrap_value = BASE_CLK_FREQ / PWM_FREQ; // Calculate wrap value
    uint ref_wrap_value = BASE_CLK_FREQ / REF_PWM_FREQ; // Calculate wrap value

    pwm_set_wrap(slice_num, wrap_value - 1); // Set wrap value (subtract 1 because it's 0-
    indexed)
    pwm_set_wrap(ref_slice_num, ref_wrap_value - 1); // Set wrap value (subtract 1 because it
    's 0-indexed)

    // Set channel 0A and 1A output high for a proportion of the period
    pwm_set_chan_level(slice_num, 0, wrap_value * PWM_DUTY / 100);
    pwm_set_chan_level(ref_slice_num, 0, ref_wrap_value * REF_PWM_DUTY / 100);

    // Set the PWM running
    pwm_set_enabled(slice_num, true);
    pwm_set_enabled(ref_slice_num, true);
    /// \end::setup_pwm[]
}

void change_pwm(int freq, float duty) {
    uint wrap_value = BASE_CLK_FREQ / freq;
    uint slice_num = pwm_gpio_to_slice_num(0);
    // Disable the PWM before configuring
    pwm_set_enabled(slice_num, false);
    pwm_set_wrap(slice_num, wrap_value - 1);
    pwm_set_chan_level(slice_num, PWM, wrap_value * duty / 100);
    pwm_set_enabled(slice_num, true);
}

void change_ref_pwm(float voltage) {
    // convert voltage to duty cycle where 0V is 0% and 3.3V is 100%
    uint REF_PWM_FREQ = 100000;
    uint REF_PWM_DUTY = voltage / 3.3;

    uint wrap_value = BASE_CLK_FREQ / REF_PWM_FREQ;
    uint slice_num = pwm_gpio_to_slice_num(0);
    // Disable the PWM before configuring
    pwm_set_enabled(slice_num, false);
    pwm_set_wrap(slice_num, wrap_value - 1);
    pwm_set_chan_level(slice_num, PWM, wrap_value * REF_PWM_DUTY);
    pwm_set_enabled(slice_num, true);
}
```

Listing B.2: *PWM initialization and functions to change duty cycle*

```
class pulse_analyzer
{
public:
    pulse_analyzer()
    {
        // pio 0 is used
```

```
        pio = pio0;
        // state machine 0
        sm = 0;
        // configure the used pins
        pio_gpio_init(pio, DIG_MEAS);
        // load the pio program into the pio memory
        uint offset = pio_add_program(pio, &pulse_analyzer_program);
        // make a sm config
        pio_sm_config c = pulse_analyzer_program_get_default_config(offset);
        // set the 'jmp' pin
        sm_config_set_jmp_pin(&c, DIG_MEAS);
        // set the 'wait' pin (uses 'in' pins)
        sm_config_set_in_pins(&c, DIG_MEAS);
        // set shift direction
        sm_config_set_in_shift(&c, false, false, 0);
        // init the pio sm with the config
        pio_sm_init(pio, sm, offset, &c);
        // enable the sm
        pio_sm_set_enabled(pio, sm, true);
    }

    // read_dutycycle (between 0 and 1)
    float read_dutycycle(void)
    {
        read();
        // printf("Pulse width: %d\n", pulsewidth);
        // printf("Period: %d\n", period);
        // printf("Duty cycle: %f\n\n", ((float)pulsewidth / (float)period));
        return ((float)pulsewidth / (float)period);
    }

private:
    // read the period and pulsewidth
    void read(void)
    {
        // clear the FIFO: do a new measurement
        pio_sm_clear_fifos(pio, sm);
        // wait for the FIFO to contain two data items: pulsewidth and period
            while (pio_sm_get_rx_fifo_level(pio, sm) < 2){
            }
        // read pulse width from the FIFO
        pulsewidth = pio_sm_get(pio, sm);
        // read period from the FIFO
        period = pio_sm_get(pio, sm) + pulsewidth;
        // the measurements are taken with 2 clock cycles per timer tick
        pulsewidth = 2 * pulsewidth;
        // calculate the period in clock cycles:
        period = 2 * period;
    }
    // the pio instance
    PIO pio;
    // the state machine
    uint sm;
    // data about the PWM input measured in pio clock cycles
    uint32_t pulsewidth, period;
};
```

Listing B.3: *PIO based on [48]*

```
static const uint I2C_SLAVE_ADDRESS = 0x17;
static const uint I2C_BAUDRATE = 400000; // 400 kHz

static const uint I2C_SLAVE_SDA_PIN = PICO_DEFAULT_I2C_SDA_PIN; // 4
```

```
169 static const uint I2C_SLAVE_SCL_PIN = PICO_DEFAULT_I2C_SCL_PIN; // 5
170
171 static void i2c_slave_handler(i2c_inst_t *i2c, i2c_slave_event_t event) {
172     uint8_t data_send = (uint8_t) (LATEST_DUTY);
173     switch (event) {
174     case I2C_SLAVE_RECEIVE: // master has written some data
175         printf("Received: %i\n", i2c_read_byte_raw(i2c));
176         // Nothing is yet being done with the received data
177
178     case I2C_SLAVE_REQUEST: // master is requesting data
179         printf("Requested\n Sending %d\n", data_send);
180         i2c_write_byte_raw(i2c, data_send);
181         break;
182     case I2C_SLAVE_FINISH: // master has signalled Stop / Restart
183         break;
184     default:
185         break;
186     }
187 }
188
189 static void setup_slave() {
190     gpio_init(I2C_SLAVE_SDA_PIN);
191     gpio_set_function(I2C_SLAVE_SDA_PIN, GPIO_FUNC_I2C);
192     gpio_pull_up(I2C_SLAVE_SDA_PIN);
193
194     gpio_init(I2C_SLAVE_SCL_PIN);
195     gpio_set_function(I2C_SLAVE_SCL_PIN, GPIO_FUNC_I2C);
196     gpio_pull_up(I2C_SLAVE_SCL_PIN);
197
198     i2c_init(i2c0, I2C_BAUDRATE);
199     // configure I2C0 for slave mode
200     i2c_slave_init(i2c0, I2C_SLAVE_ADDRESS, &i2c_slave_handler);
201 }
```

Listing B.4: *I2C setup and handler*

```
203 void wifi_init() {
204     if (cyw43_arch_init()) {
205         printf("failed to initialize\n");
206         return;
207     }
208
209     cyw43_arch_enable_sta_mode();
210
211     printf("Connecting to Wi-Fi...\n");
212     if (cyw43_arch_wifi_connect_timeout_ms(WIFI_SSID, WIFI_PASSWORD, CYW43_AUTH_WPA2_AES_PSK,
         10000)) {
213         printf("failed to connect\n");
214         cyw43_arch_deinit();
215         return;
216     }
217
218     printf("Connected to Wi-Fi\n");
219 }
```

Listing B.5: *WIFI initialization and connection*

```
221 typedef struct TCP_SERVER_T_ {
222     struct tcp_pcb *server_pcb;
223     struct tcp_pcb *client_pcb;
224     uint8_t buffer_sent[BUF_SIZE];
225     uint8_t buffer_recv[BUF_SIZE];
226     int sent_len;
```

```
227     int recv_len;
228 } TCP_SERVER_T;
229 
230 static TCP_SERVER_T* tcp_server_init(void) {
231     TCP_SERVER_T *state = static_cast<TCP_SERVER_T*>(calloc(1, sizeof(TCP_SERVER_T)));
232     // printf("tcp_server_init");
233     return state;
234 }
235 
236 static err_t tcp_server_close(void *arg) {
237     // printf("tcp_server_close");
238     TCP_SERVER_T *state = (TCP_SERVER_T*)arg;
239     if (state->client_pcb) {
240         tcp_close(state->client_pcb);
241         state->client_pcb = NULL;
242     }
243     if (state->server_pcb) {
244         tcp_close(state->server_pcb);
245         state->server_pcb = NULL;
246     }
247     return ERR_OK;
248 }
249 
250 static err_t tcp_server_send_data(void *arg, struct tcp_pcb *tpcb) {
251     // printf("tcp_server_send_data");
252     TCP_SERVER_T *state = (TCP_SERVER_T*)arg;
253     memset(state->buffer_sent, DATA_SEND, BUF_SIZE);
254     printf("Sending %d\n", state->buffer_sent[0]);
255     state->sent_len = 0;
256     return tcp_write(tpcb, state->buffer_sent, BUF_SIZE, TCP_WRITE_FLAG_COPY);
257 }
258 
259 static err_t tcp_server_sent(void *arg, struct tcp_pcb *tpcb, u16_t len) {
260     // printf("tcp_server_sent");
261     TCP_SERVER_T *state = (TCP_SERVER_T*)arg;
262     state->sent_len += len;
263     return tcp_server_send_data(arg, tpcb);
264 }
265 
266 static err_t tcp_server_recv(void *arg, struct tcp_pcb *tpcb, struct pbuf *p, err_t err) {
267     // printf("tcp_server_recv");
268     TCP_SERVER_T *state = (TCP_SERVER_T*)arg;
269 
270     int freq = 0;
271     float duty = 0.0;
272 
273     if (err == ERR_OK && p != NULL) {
274         // create string
275         char *data = (char*)calloc(p->len + 1, sizeof(char));
276         memcpy(data, p->payload, p->len);
277 
278         const char *freq_pos = strstr(data, "freq=");
279         if (freq_pos) {
280             // Move past "freq="
281             freq_pos += 5;
282             freq = atoi(freq_pos);
283         }
284 
285         // Find "duty="
286         const char *duty_pos = strstr(data, "duty=");
287         if (duty_pos) {
288             // Move past "duty="
289             duty_pos += 5;
```

```
            duty = atof(duty_pos);
        }

        change_pwm(freq, duty);
        printf("Received: %s\n", data);
        printf("Frequency: %d\n", freq);
        printf("Duty cycle: %f\n", duty);


        state->recv_len = p->len;
        // pbuf_free(p);
        return ERR_OK;
    }
    return ERR_OK;
}

static err_t tcp_server_accept(void *arg, struct tcp_pcb *client_pcb, err_t err) {
    // printf("tcp_server_accept");
    TCP_SERVER_T *state = (TCP_SERVER_T*)arg;
    state->client_pcb = client_pcb;
    tcp_arg(client_pcb, state);
    tcp_sent(client_pcb, tcp_server_sent);
    tcp_recv(client_pcb, tcp_server_recv);
    return tcp_server_send_data(arg, client_pcb);
}

static bool tcp_server_open(void *arg) {
    // printf("tcp_server_open");
    TCP_SERVER_T *state = (TCP_SERVER_T*)arg;
    struct tcp_pcb *pcb = tcp_new_ip_type(IPADDR_TYPE_ANY);
    if (!pcb || tcp_bind(pcb, IP_ANY_TYPE, TCP_PORT) != ERR_OK) {
        return false;
    }
    state->server_pcb = tcp_listen(pcb);
    tcp_arg(state->server_pcb, state);
    tcp_accept(state->server_pcb, tcp_server_accept);
    return true;
}
```

Listing B.6: *TCP initialization and request handlers based on [37]*

```
// Core 1 Main Code
void core1_entry() {
    pulse_analyzer pulse_analyzer_instance; // Instantiate an object of the pulse_analyzer
    class

    // get current time
    absolute_time_t time_since_last_spark = get_absolute_time();
    while (1){

        LATEST_DUTY = pulse_analyzer_instance.read_dutycycle();
        if (LATEST_DUTY < 0.0001) {
            // turn led on
            SHORTED = true;
            time_since_last_spark = get_absolute_time();
            cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 1);
        } else if (absolute_time_diff_us(time_since_last_spark, get_absolute_time()) <
    100000){
            SHORTED = true;
            cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 1);
        }
        else {
            SHORTED = false;
```

```
            cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 0);
        }
        DATA_SEND = (uint8_t) SHORTED;
    }
}
```

Listing B.7: *Second Core code polling the duty cycle measured*

```
int main(){

    stdio_init_all();
    setup_pwm();
    setup_slave();
    sleep_ms(10);
    wifi_init();
    sleep_ms(10);

    // initialise GPIO (Green LED connected to pin 25)
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);
    gpio_put(LED_PIN, 0);

    multicore_launch_core1(core1_entry);

    TCP_SERVER_T *state = tcp_server_init();
    if (!state) {
        return 1;
    }
    if (!tcp_server_open(state)) {
        return 1;
    }

    while (1)
    {
        tight_loop_contents();
    }
    free(state);

}
```

Listing B.8: *Main function with initialization of all functionality and main loop*

```
; https://github.com/GitJer/Some_RPI-Pico_stuff/tree/main/PwmIn
.program pulse_analyzer

; algorithm:

; loop:
;    reset y, the 'timer' for the pulsewidth (high period)
;    reset x, the 'timer' for the low period. (high + low = period)
;    wait for a 0, then wait for a 1: this is the rising edge
;    loop:
;       decrement y timer
;       test for falling edge
;    y timer value is the pulse width (actually, (0xFFFFFFFF - y)*2/125MHz is the pulse width
    )
;    loop:
;       test for rising edge
;       decrement x timer
;    x timer is the low period (actually, (0xFFFFFFFF - x)*2/125MHz is the low period)
;    push both y and x to the Rx FIFO

.wrap_target
```

```
start:
    mov y ~NULL         ; start with the value 0xFFFFFFFF
    mov x ~NULL         ; start with the value 0xFFFFFFFF
    wait 0 pin 0        ; wait for a 0
    wait 1 pin 0        ; wait for a 1, now we really have the rising edge
timer_hp:               ; loop for high period
    jmp y-- test        ; count down for pulse width
    jmp start           ; timer has reached 0, stop count down of pulse, restart
test:
    jmp pin timer_hp    ; test if the pin is still 1, if so, continue counting down
timer_lp:               ; loop for low period
    jmp pin timerstop   ; if the pin has become 1, the period is over, stop count down
    jmp x-- timer_lp    ; if not: count down
    jmp start           ; timer has reached 0, stop count down of low period, restart
timerstop:
    mov ISR ~y          ; move the value ~y to the ISR: the high period (pulsewidth) (0
    xFFFFFFFF-y)
    push noblock        ; push the ISR into the Rx FIFO
    mov ISR ~x          ; move the value ~x to the ISR: the low period (0xFFFFFFFF-x)
    push noblock        ; push the ISR into the Rx FIFO
.wrap
```

Listing B.9: *PIOASM code for digital sampling at 62.5MHz [48]*

B.2. Various Test Scripts

```
int main(){
    stdio_init_all();

    // initialise GPIO (Green LED connected to pin 25)
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);
    gpio_put(LED_PIN, 1);

    gpio_init(GPIO0_PIN);
    gpio_set_dir(GPIO0_PIN, GPIO_IN);

    while (1)
    {
        gpio_put(LED_PIN, gpio_get(GPIO0_PIN));
    }

}
```

Listing B.10: *Test script to find hysteresis of GPIO pins*

```
from machine import ADC, Pin
import time

adc = ADC(Pin(26))

while True:
    print(adc.read_u16())
```

Listing B.11: *Test script feasibility using ADC in MicroPython*

```
#define CLOCK_DIV 96
#define FSAMP 500000

#define SAMPLES 1000

```

```
// globals
dma_channel_config cfg;
uint dma_chan;
float freqs[SAMPLES];

void setup_adc() {
    // Set up the ADC to sample on pin 26 (GPIO 26)
    adc_gpio_init(26 + CAPTURE_CHANNEL);

    adc_init();

    adc_select_input(CAPTURE_CHANNEL);
    adc_fifo_setup(
        true,    // Write each completed conversion to the sample FIFO
        true,    // Enable DMA data request (DREQ)
        1,       // DREQ (and IRQ) asserted when at least 1 sample present
        false,   // We won't see the ERR bit because of 8 bit reads; disable.
        true     // Shift each sample to 8 bits when pushing to FIFO
    );

    // set sample rate
    adc_set_clkdiv(CLOCK_DIV);

    sleep_ms(1000);
    // Set up the DMA to start transferring data as soon as it appears in FIFO
    uint dma_chan = dma_claim_unused_channel(true);
    cfg = dma_channel_get_default_config(dma_chan);

    // Reading from constant address, writing to incrementing byte addresses
    channel_config_set_transfer_data_size(&cfg, DMA_SIZE_8);
    channel_config_set_read_increment(&cfg, false);
    channel_config_set_write_increment(&cfg, true);

    // Pace transfers based on availability of ADC samples
    channel_config_set_dreq(&cfg, DREQ_ADC);
}

void sample(uint8_t *capture_buf) {
  adc_fifo_drain();
  adc_run(false);

  dma_channel_configure(dma_chan, &cfg,
      capture_buf,    // dst
      &adc_hw->fifo,  // src
      SAMPLES,          // transfer count
      true            // start immediately
      );

  adc_run(true);
  dma_channel_wait_for_finish_blocking(dma_chan);
}

// initialise the buffer
    uint8_t cap_buf[SAMPLES];

    for (int j = 0; j < 1000; j++){
        sample(cap_buf);
    }
```

Listing B.12: *Test script feasibility using ADC in C++ with DMA*

C

Pictures of the experimental setup

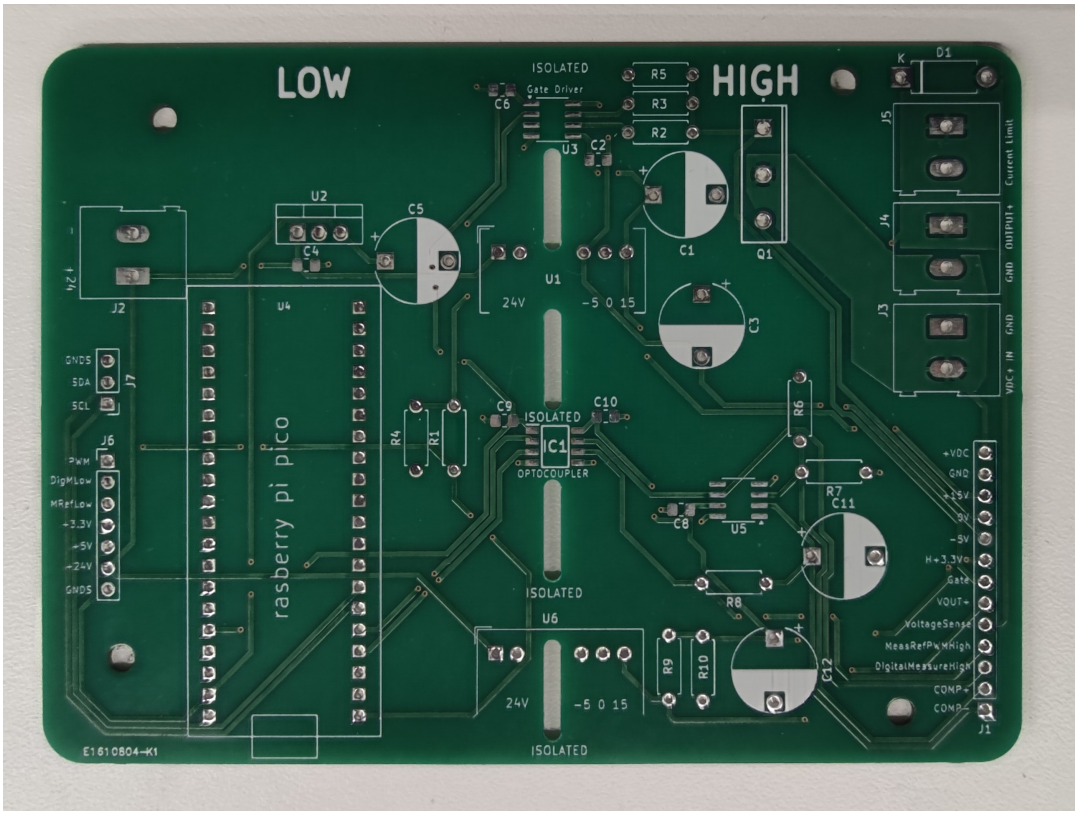


Figure C.1: *Plain PCB*

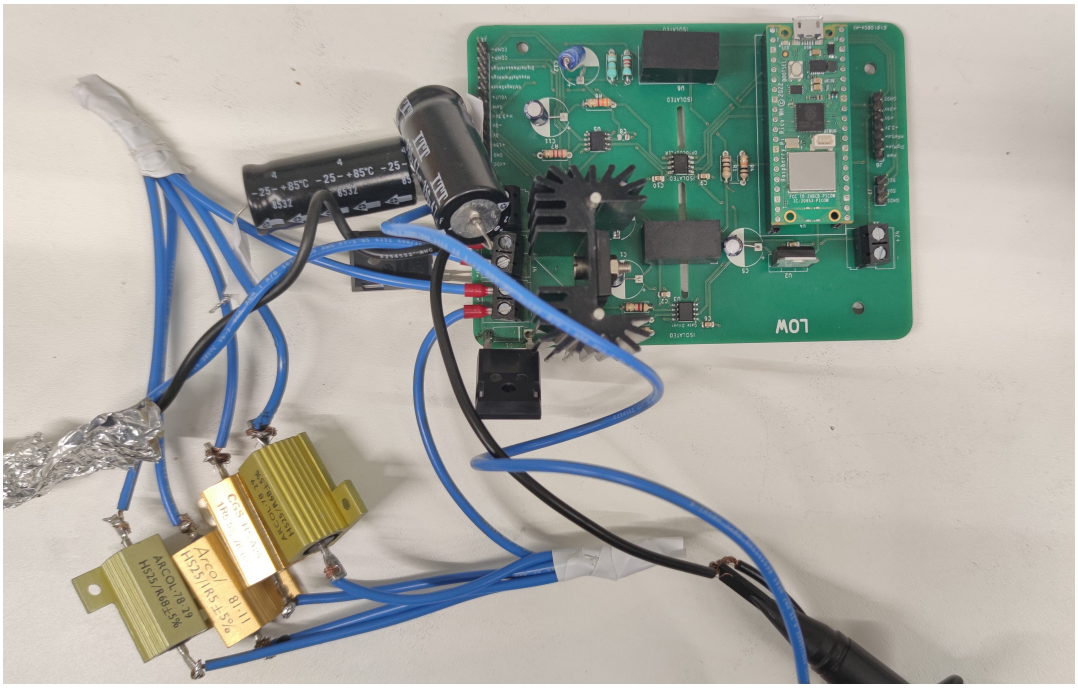


Figure C.2: *The power supply's scope*

In Figure C.1, the PCB without the components soldered on it can be seen, while in Figure C.2 the subsystems of the power supply subgroup's scope are visible, which includes the DC pulse generation circuit (on the PCB and the external resistors in parallel), the Raspberry Pico and closed-loop system (on the PCB).