University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

BACHELOR'S THESIS

Study programme/specialisation: Automation and electronics design	Spring/ ~~Autumn~~ semester, 20.21 Open / ~~Confidential~~
Author: Ali Hamid Jabbour	
Programme coordinator: Supervisor(s): A/Prof Arnfinn A. Eielsen	
Title of bachelor's thesis: Designing and building a power inverter (DC to AC)	
Credits: 20 points	
Keywords: Inverter, STM32, PWM, Si, SiC	Number of pages: ……90…… + supplemental material/other: ……43…… Stavanger, ……30.05.2021…… date/year

Abstract

The goal of this project is to design and build a MOSFET-based DC to AC inverter to be used in the comparison of the silicon and silicon carbide MOSFETs. A digital pulse-width modulation on an STM32 microcontroller is used to produce the inverter control signals.
A theoretical review of suitable power electronics devices is performed, a comparison between Si-based and SiC-based MOSFETs is presented, and the inverter hardware and software design is explained in detail. Hardware design covering the selection of components and calculating their values, the monitoring sensors used, and the PCB layout. Software design detailing the code needed to perform the control and monitoring tasks.
The design is implemented into a PCB and is combined with the microcontroller. Some tests are performed and discussed to evaluate the prototype.

Sammendrag

Målet med dette prosjektet er å konstruere og bygge en MOSFET-basert DC til AC inverter som skal brukes for å sammenlige silicon og silicon carbide MOSFETs. En digital pulsbredde modulering på en STM32 mikrokontroller er brukt for å produsere inverterens kontorll signaler. Det er gjennomført en teoretisk gjennomgang av passende kraft elektronisk komponenter, en sammenligning mellom Si-basert og SiC basert MOSFETs og konstruksjonen av inverterens maskinvare og programvare er forklart i detalj. Maskinvarens forklaring inneholder valg av de ulike komponentene og beregningene av deres verdier, de overvåkende sensorene som er brukt og PCB utlegget. Programvare forklaringen viser til koden som gjorde det mulig å utføre kontroll og overvåknings oppgavene. Konstruksjonen er implementert inn i en PCB og er kombinert sammen med mikrokontrolleren. Noen tester er gjennomført og diskutert for å evaluere prototypen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The increased environmental concerns and the rising prevalence of photovoltaic renewable energy generation, which produces direct current, the introduction of large-scale energy storage systems replacing inefficient peaking power plants, and the accelerated adoption of electric vehicles. As these changes in power generation and demands are distributed in nature and span the whole power spectrum from low to high power applications, it has increased the need for more efficient and reliable power inverters.

1.2 Objectives

The main goal of this work is to design and build a MOSFET based inverter that is able to use both Si and SiC MOSFETs; a secondary goal is to evaluate the performance of the inverter and show the benefits of using SiC MOSFETs as power devices regarding efficiency, size, and cost, reliability, and complexity.

1.3 Scope

The purpose of this work is educational in nature, and the result is not intended to be used in any certain application; therefore, the power rating of the inverter was decided out of practical considerations such as safety and available tools and equipment at the lab. Since 50 V is the maximum voltage allowed to be used when working alone in the lab and the maximum current for the lab equipment at this voltage is 6 A, which gives a maximum power of 300 watts; thus the inverter needs to be able to handle such power with some margin.

1.4 Organization

This work is organized in the following chapters:

- **Chapter 2.** Provides a brief description of inverter types, control schemes, and circuit topologies, semiconductors types, materials, and characteristics.
- **Chapter 3.** Goes into the implemented inverter circuit in detail and provides a basis for the design choices.
- **Chapter 4.** Explains the microcontroller code used in this work, giving the necessary context from its reference manual, and it shows the computer program used to communicate with the microcontroller.
- **Chapter 5.** Shows the combined hardware/software product of this work and contains the results from all the performed tests under different parameter variations.
- **Chapter 6.** Evaluates the obtained results, identifies strong points and the shortcomings, and provides suggestions for improvements and further development.

Chapter 2

Theory

This chapter provides the theoretical framework of the problem at hand, various considerations and possible solutions. It lists the different types of inverters, semiconductor materials and devices, the advantages and disadvantages of each, and lays the groundwork for the design phase. The bulk of this chapter is retrieved mainly from the book ***Power electronics*** with emphasis on the information relevant to this work [21].

2.1 Power converter definition and classifications

Power converters process and control the flow of electric energy by supplying voltages and currents in a form that is optimally suited for user loads. It utilizes power semiconductor devices controlled by signal electronics (integrated circuits) and possibly energy storage elements such as inductors and capacitors. These converters can be classified according to how the semiconductor are switched into:

- Line frequency (naturally commutated) converters, where the power semiconductor devices are turned on and phase locked to the line voltage waveform at a frequency of

50 or 60 Hz.

- Switching (forced-commutated) converters, where the power semiconductor devices are turned on and off at much higher frequencies than the line frequency. Although, the inverter output may be either dc or have a frequency comparable to the line frequency.
- Resonant and quasi-resonant converters, where the controllable switches turn on and/or turn off at zero voltage and/or zero current.

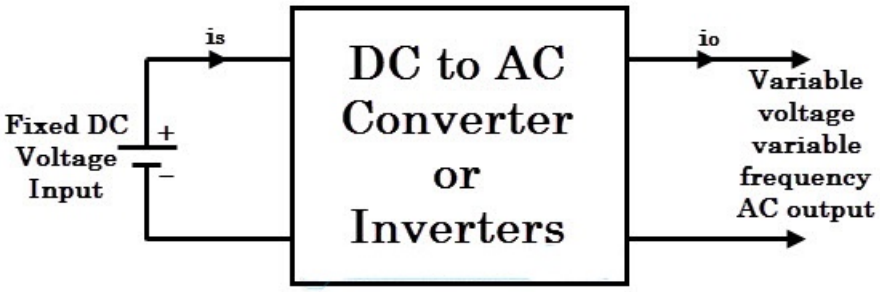


Figure 2.1.1: DC/AC inverter [21]

These inverters are further divided depending on the number of output phases into *single-phase* and *three-phase* inverters; a single-phase inverter will be implemented in this work, all the following sections are written with the assumption of a single-phase inverter without explicitly stating it. Based on the form (frequency) on the two sides, converters can be divided into: ac/ac, ac/dc, dc/dc and dc/ac which is the topic of this bachelor [21].

2.1.1 Applications

An inverter is needed when the power source have a direct current and the device using the power or the transmission medium is using alternating current. Therefore the inverters had traditionally two main areas of application:

- As an interface between a battery or a fuel cell power source and AC equipment. Also in grid-ties and off the grid photovoltaic systems.
- Electric motor speed control, as in industrial applications and more recently in electric vehicles.

2.1.2 Switch-mode dc/sinusoidal ac inverters

These inverters are used in ac motor drives and ac power sources where the objective is to produce a sinusoidal ac output whose magnitude and frequency can both be controlled. These can be either a voltage or a current source depending on the DC input source; voltage source inverters (VSIs) are far more common and has a wide range of applications. The type of VSIs that this work is interested in is the *Pulse width modulated (PWM) inverter*, where it is possible to control the magnitude and frequency of the output voltage, there are several techniques used to pulse modulate the shape of the ac voltage as close as possible to a sine wave. The most common technique is sinusoidal PWM which will be used in this work and explained in more details in the next section [21].

2.1.3 Sinusoidal pulse width modulation (SPWM)

Pulse width modulation means the variation of the duty cycle ($D = \frac{T_{on}}{T}$) of a periodic signal. In inverter circuits the PWM is slightly complex, since we would like the inverter output to be sinusoidal with controllable magnitude and frequency. In order to produce a sinusoidal output voltage waveform at a desired frequency, a sinusoidal control signal at the desired frequency called *Fundamental frequency* f_1 is compared with a triangular waveform. The frequency of the triangular waveform establishes the inverter switching frequency called f_s and is generally kept constant along with its amplitude 2.1.2-a. We need to recognize that the output voltage of the inverter V_o will contain voltage components at harmonic frequencies of f_1. The relations between the frequency and amplitude of the control and triangular signal

determines the amplitude of V_o and its harmonic frequencies contents. These relations are described by amplitude modulation ratio m_a and m_f.

$$\begin{aligned} m_a &= \frac{(\hat{V}_{control})}{\hat{V}_{tri}} \\ m_f &= \frac{f_s}{f_1} \end{aligned} \tag{2.1}$$

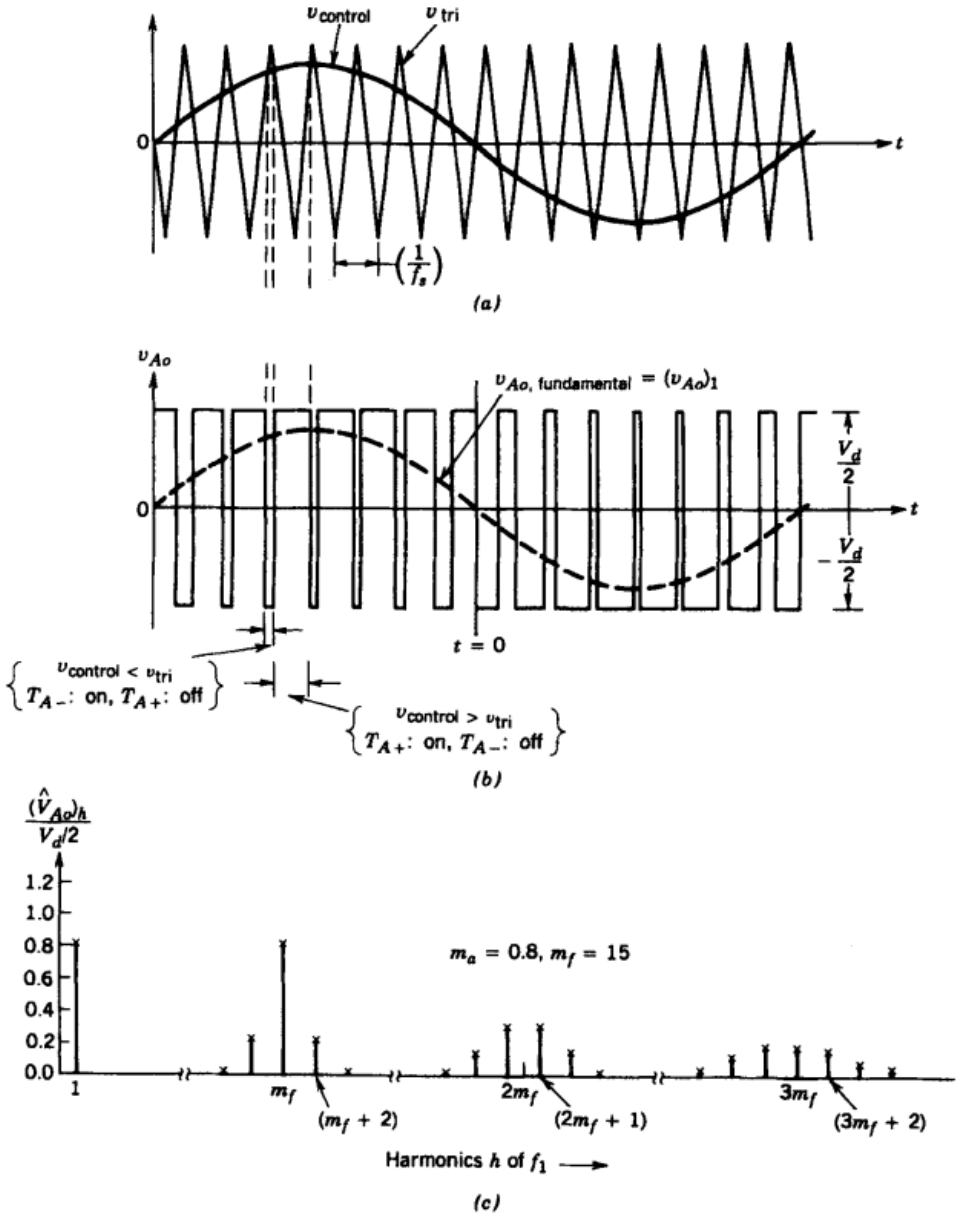


Figure 2.1.2: Sinusoidal pulse width modulation [21].

Running the risk of oversimplifying, we can say that m_a controls the output voltage amplitude:

$$\frac{V_d}{2} < (\hat{V}_o)_1 < \frac{4}{\pi}\frac{V_d}{2}$$

and m_f controls at what frequencies the harmonics will show up, as they appear as side bands, centered around the switching frequency and its multiples, that is, around harmonics m_f $2m_f$, $3m_f$, and so on 2.1.2-c. This general pattern holds true for all values of m_a in the range 0-1, and this will limit the amplitude of V_o to $\frac{V_d}{2}$ 2.1.3.

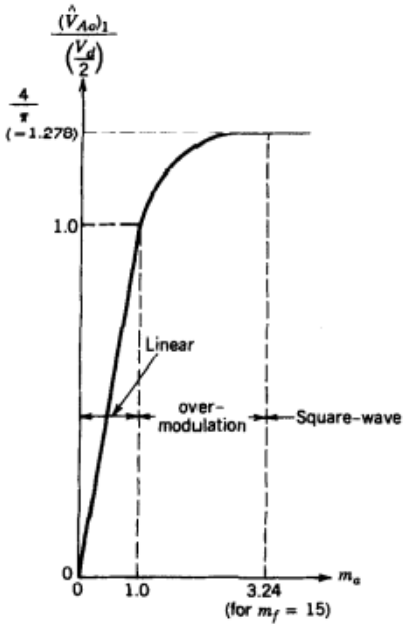


Figure 2.1.3: Voltage control by varying m_a [21].

The selection of the switching frequency is guided by the relative ease in filtering harmonic voltages at high frequencies, it is desirable to use as ***high*** a switching frequency as possible, except for one significant drawback: ***Switching losses*** in the inverter switches increase proportionally with the switching frequency f_s. Another consideration is keeping the switching frequency outside of the audible range (less than 6 kHz or greater than 20 kHz) [21].

2.1.4 Single-phase inverter topologies

The two most common inverter topologies are half-bridge and full-bridge. Their simplicity and ease of implementation can be the reason for their popularity. Both topologies allows for multi levels design, which improves the harmonic contents of the output voltage. Single level full bridge topology will be used in this work as the goal is to evaluate the switching element itself and not the design.

2.1.4.1 Half-bridge

In a half-bridge inverter shown in 2.1.4 two equal capacitors are used in series across the dc input and their junction is at a mid-potential, with a voltage $\frac{V_d}{2}$ for each capacitor. This topology requires only two switching elements which makes it the cheapest and simplest way to make an inverter. Only one of these can be on at the same time, which allows for two different voltage levels at the output of the inverter: $\frac{V_d}{2}$ and $-\frac{V_d}{2}$.

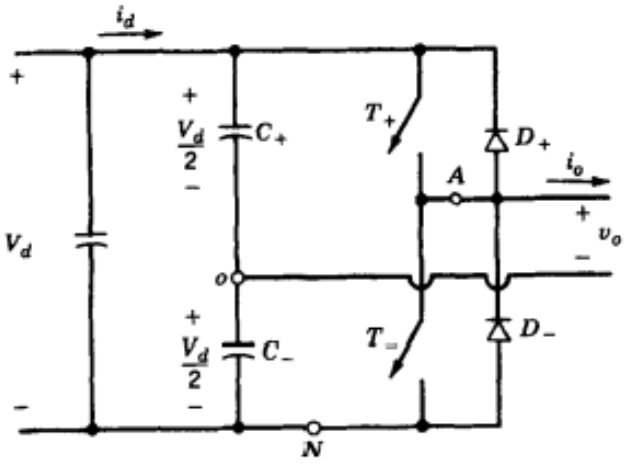


Figure 2.1.4: Half-bridge inverter [21].

2.1.4.2 Full-bridge

The full-bridge inverter is similar to the half bridge-inverter, but it has an additional leg to connect the neutral point to the load as shown in 2.1.5. This means that four switching elements are required, two of them are on at the same time and they are responsible for half of the output voltage waveform, the two branches outputs are 180° out of phase. This topology allows for three different voltage levels at the output of the inverter V_d, $-V_d$ and 0. With the same dc input voltage, the maximum output voltage of the full-bridge inverter is twice that of the half-bridge inverter. This implies that for the same power, the output current and the switch currents are one-half of those for a half-bridge inverter. Which give full-bridge topology a significant advantage over half-bridge at higher power ratings.

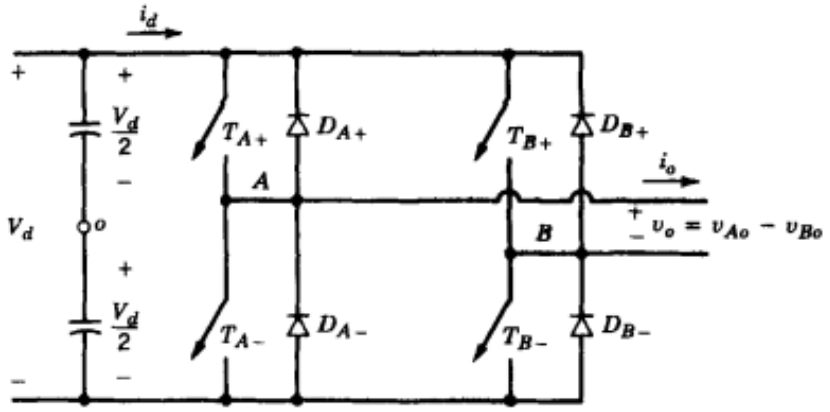


Figure 2.1.5: Full-bridge inverter [21].

2.2 Power semiconductor switches types

The main building block of an inverter or any type of power electronics device is the semiconductor switch. The power capabilities, ease of control and cost of semiconductor switches determines where the power electronics devices can be applied. Before talking about the different types of semiconductors, a basic understanding of the ***P-N junction*** is required,

since all semiconductors are some different combination of p-n junctions.
A p-n junction is a boundary formed when an n-type region in a silicon crystal is adjacent to or abuts a p-type region in the same crystal. The "P" (positive) side contains an excess of holes, which is created by diffusing donors impurities (boron) into p-type silicon, while the "N" (negative) side contains an excess of electrons, which is created by diffusing acceptor impurities (phosphorus) into an n-type silicon crystal. This allows electrical current to pass through the junction only in one direction. Presently available power semiconductor devices can be classified into three groups according to their degree of controllability:

1. Diodes. On and off states controlled by the power circuit.
2. Thyristors. Latched on by a control signal but must be turned off by the power circuit.
3. Controllable switches. Turned on and off by control signals, which includes several device types bipolar junction transistors (BJTs), metal-oxide- semiconductor field effect transistors (MOSFETs), gate turn off (GTO) thyristors, and insulated gate bipolar transistors (IGBTs).

The following characteristics in a controllable switch are desirable:

1. Block arbitrarily large forward and reverse voltages with zero current flow when Off (Small leakage current in the off state).
2. Conduct arbitrarily large currents with zero voltage drop when on (Low on-state voltages and resistances) to minimize on-state power losses.
3. Switch from on to off or vice versa instantaneously when triggered (Short turn-on and turn-off times). This will permit the device to be used at high switching frequencies.
4. Vanishingly small power required from control source to trigger the switch. This will simplify the control circuit design.

5. Capability to withstand rated voltage and rated current simultaneously while switching. This will eliminate the need for external protection (snubber) circuits across the device.

6. Large dv/dt and di/dt ratings. This will minimize the need for external circuits otherwise needed to limit dv/dt and di/dt in the device so that it is not damaged.

7. Large power dissipation capability.

In spite of significant progress in the development of power devices, there are none available that simultaneously have all of these properties. In all device types, there are a trade-off between breakdown voltages and on-state losses. There is also a trade-off between on-state losses and switching speeds. In the next sections, the structure and i-v characteristics of relevant semiconductor devices will be listed and will go into power MOSFETs in more details, both the silicon and silicon carbide variant [21].

2.2.1 Diodes

Figure 2.2.1 shows the circuit symbol for the diode and its steady-state i-v characteristic. When the diode is forward biased, it begins to conduct with only a small forward voltage across it, which is on the order of 1 V. When the diode is reverse biased, only a negligibly small leakage current flows through the device until the reverse breakdown voltage is reached. In normal operation, the reverse-bias voltage should not reach the breakdown rating. One important property to look at is the reverse-recovery time Trr, at turn-off, the diode current reverses for a certain amount of time, before falling to zero. This reverse-recovery (negative) current is required to sweep out the excess carriers in the diode and allow it to block a negative polarity voltage.

Depending on the application requirements, various types of diodes are available:

1. Schottky diodes. These have low forward voltage drop, are limited in their blocking

voltage capabilities to 50- 100 V.

2. Fast-recovery diodes. Used in high-frequency circuits in combination with controllable switches where a small reverse-recovery time is needed. At power levels of several hundred volts and several hundred amperes.

3. Line frequency diodes. The on-state voltage of these diodes is designed to be as low as possible and as a consequence have larger Trr, which are acceptable for

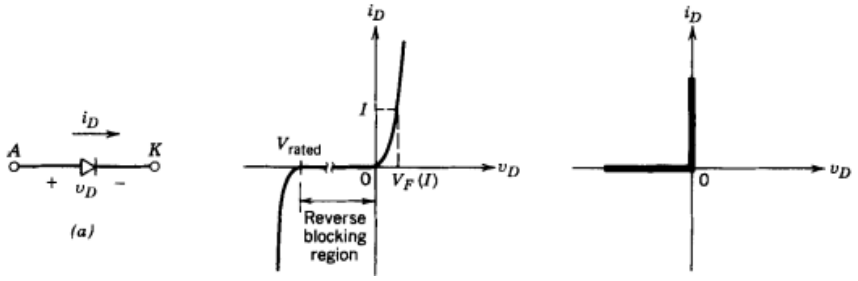


Figure 2.2.1: The diode current-voltage characteristics and circuit symbol [21].

2.2.2 Power BJT

A power bipolar junction transistor has a vertically oriented four-layer structure of alternating p-type and n-type doping such as the npn and pnp transistor shown in figure 2.2.2. The transistor has three terminals, labeled collector, base, and emitter. Npn transistors are much more widely used than pnp transistors as power switches. The vertical structure is preferred for power transistors because it maximizes the cross-sectional area through which the current in the device is flowing. This minimizes the on-state resistance and thus the power dissipation in the transistor. In addition, having a large cross-sectional area minimizes the thermal resistance of the transistor, thus also helping to keep power dissipation problems under control. The n+ region that terminates the drift region serves as the collector contact to the outside world. The thickness of the drift region determines the breakdown voltage of the transistor and thus can range from tens to hundreds of micrometers in extent. The base thickness is made as small

as possible in order to have good amplification capabilities. However, if the base thickness is too small, the breakdown voltage capability of the transistor is compromised. Thus, base thicknesses in power devices are a compromise between these two competing considerations and are typically several micrometers to a few tens of micrometers in thickness, compared with the small fraction of a micrometer in thickness for logic-level transistors [21].

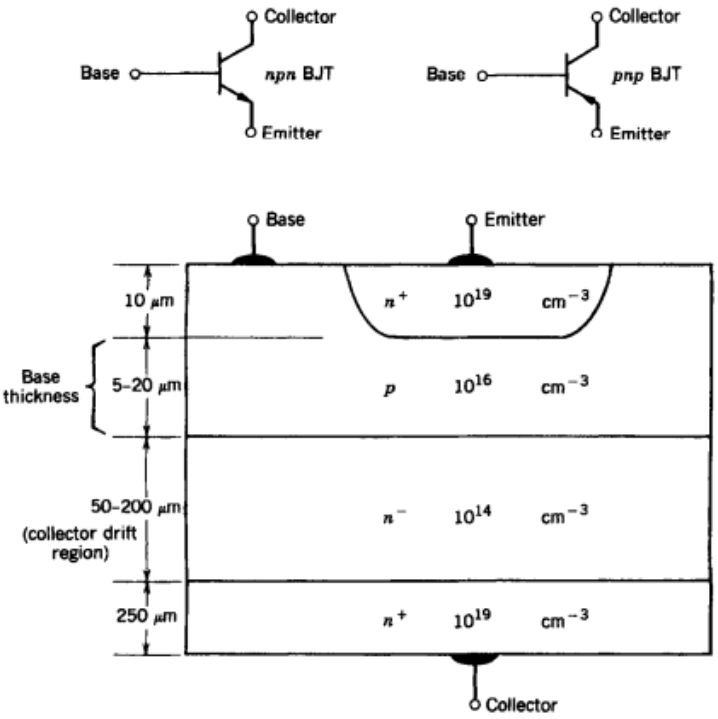


Figure 2.2.2: Vertical cross section of a BJT and circuit symbol [21].

Figure 2.2.3 shows the i-v characteristics of a typical npn power transistor. It can be noticed that there is a maximum collector-emitter voltage that can be sustained across the transistor when it is carrying substantial collector current BV_{SUS}, the collector- emitter breakdown voltage when the base is open circuited BV_{CEO}. The region labeled primary breakdown is due to conventional avalanche breakdown of the C-B junction and the attendant large flow of current. This region of the characteristics is to be avoided because of the large power dissipation that clearly accompanies such breakdown. The region labeled second breakdown

must also be avoided because large power dissipation also accompanies it, particularly at localized sites within the semiconductor [21].

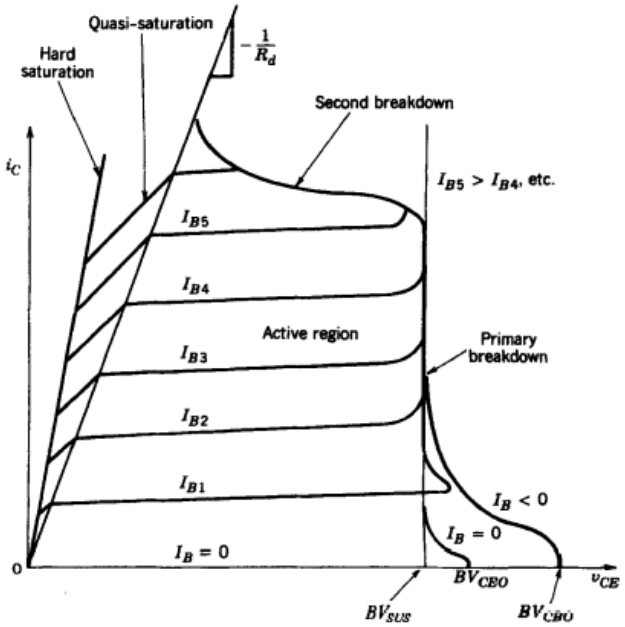


Figure 2.2.3: The BJT current-voltage characteristics [21].

2.2.3 Power MOSFETs

A power MOSFET has the vertically oriented four-layer structure of alternating p-type and n-type doping shown in figure 2.2.4. The $n^+pn^-n^+$ structure is termed an enhancement mode n-channel MOSFET. A structure with the opposite doping profile can also be fabricated and is termed a depletion mode p-channel MOSFET. The p-type middle layer is usually termed the body and is the region where the channel is established between source and drain. The n- layer is the drain drift region which determines the breakdown voltage of the device. At first glance, it would appear that there is no way that current can flow between the drain and source terminals of the device because one of the p-n junctions (either the body-source junction or the drain-body junction) will be reverse biased by either polarity of applied voltage

between the drain and source. There can be no injection of minority carriers into the body region via the gate terminal because the gate is isolated from the body by a layer of silicon dioxide termed the gate oxide, which is a very good insulator and, hence, there is no BJT operation. However, an application of a voltage that biases the gate positive with respect to the source will convert the silicon surface beneath the gate oxide into an n-type layer or channel, thus connecting the source to the drain and allowing the flow of appreciable currents [21].

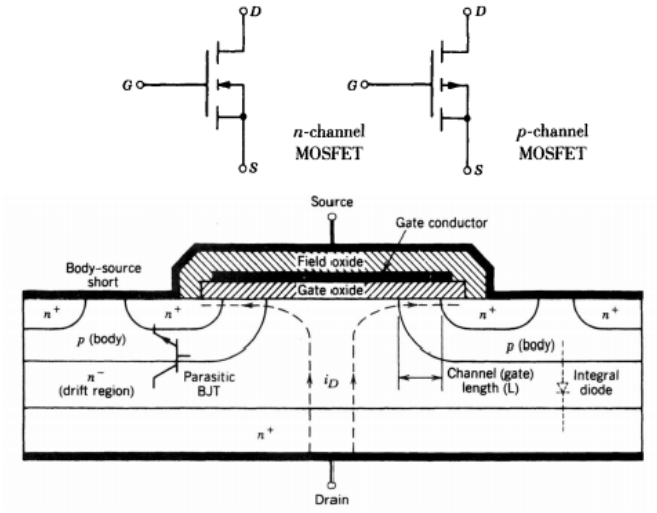


Figure 2.2.4: Vertical cross section of a MOSFET and circuit symbol [21].

The MOSFET, like the BJT, is a three-terminal device where the input, the gate in the case of the MOSFET, controls the flow of current between the output terminals, the source, and drain. The source terminal is usually common between the input and output of a MOSFET. The output characteristics, drain current i_D as a function of drain-to-source voltage V_{DS}, with gate-to-source voltage V_{GS}, as a parameter, are shown in figure 2.2.5 for an n-channel MOSFET.

In power electronic applications, the MOSFET is used as a switch to control the flow of power to the load, the MOSFET is in cutoff when the gate-source voltage is less than the threshold voltage $V_{GS(th)}$, which is typically a few volts in most power MOSFETs, the device is an open

circuit and must hold off the power supply voltage applied to circuit. This means that the drain-source breakdown voltage BV_{DSS}, must be larger than the applied drain- source voltage to avoid breakdown and the attendant high power dissipation. When breakdown occurs, it is due to the avalanche breakdown of the drain-body junction.

When the device is driven by a large gate-source voltage, it is driven into the ohmic region where the drain-source voltage $V_{DS(on)}$ is small. In this region the power dissipation can be kept within reasonable bounds by minimizing $V_{DS(on)}$ even if the drain current is fairly large. In the active region the drain current is independent of the drain- source voltage and depends only on the gate-source voltage.

Overall the transfer curve of a power MOSFET is quite linear, in contrast to the parabolic transfer curve of the logic-level device.

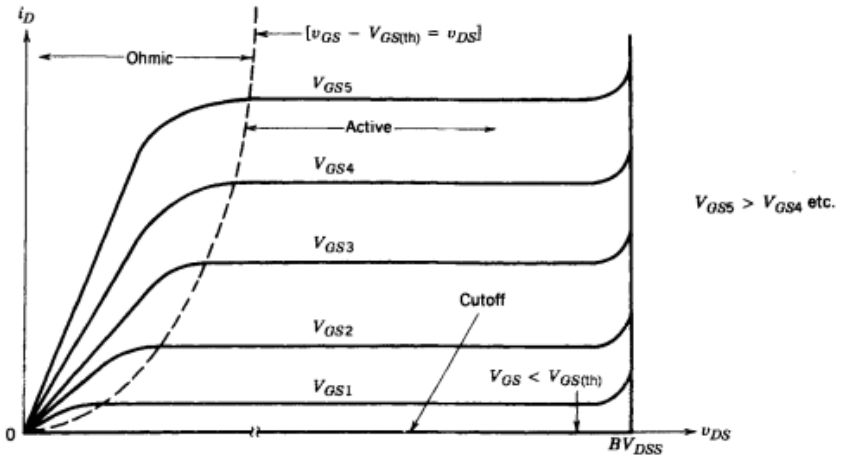


Figure 2.2.5: The MOSFET current-voltage characteristics [21].

2.2.4 Power IGBTs

IGBT stands for insulated gate bipolar transistors, the vertical cross section of a generic n-channel IGBT is shown figure 2.2.6. This structure is quite similar to that of the vertical diffused MOSFET shown in 2.2.4. The principal difference is the presence of the p+ layer that forms the drain of the IGBT. This layer forms a p-n junction (labeled J_1, in the figure),

which injects minority carriers into what would appear to be the drain region of the vertical MOSFET. The IGBT does retain the extension of the source metallization over the body region that is also used in power MOSFETs. A circuit symbol for an n-channel IGBT is shown in figure 2.2.7 This symbol is essentially the same as that used for an n-channel MOSFET, but with the addition of an arrowhead in the drain lead pointing into the body of the device, indicating the injecting contact. Some prefer to consider the IGBT as basically a BJT with a MOSFET gate input and, thus, to use the modified BJT symbol for the IGBT, this symbol device has a collector and emitter rather than a drain and source [21].

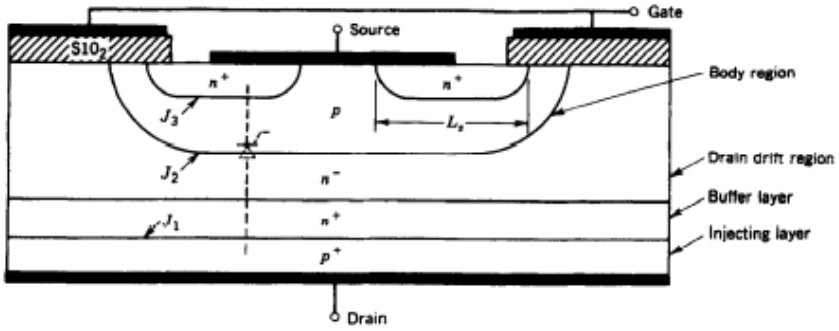


Figure 2.2.6: Vertical cross section of a IGBT [21].

Figure 2.2.7 shows the i-v characteristics of an n-channel IGBT. In the forward direction they appear qualitatively similar to those of a logic-level BJT except that the controlling parameter is an input voltage, the gate-source voltage, rather than an input current. The junction labeled J_2 in figure 2.2.6 blocks any forward voltages when the IGBT is off. The reverse-blocking voltage indicated on the i-v characteristic can be made as large as the forward-blocking voltage if the device is fabricated without the n^+ buffer layer. Such a reverse-blocking capability is useful in some types of ac circuit applications. The junction labeled J_1, in figure 2.2.6 is the reverse-blocking junction. However, if the n^+ buffer layer is used in the device construction, the breakdown voltage of this junction is lowered significantly, to a few tens of volts, and the IGBT no longer has any reverse-blocking capability. The transfer curve $i_D - v_{GS}$ shown in figure 2.2.7 is identical to that of the power MOSFET. The curve is reasonably linear over

most of the drain current range, becoming nonlinear only at low drain currents where the gate-source voltage is approaching the threshold. If v_{GS} is less than the threshold voltage $v_{GS(th)}$, then the IGBT is in the off state. The maximum voltage that should be applied to the gate-source terminals is usually limited by the maximum drain current that should be permitted to flow in the IGBT [21].

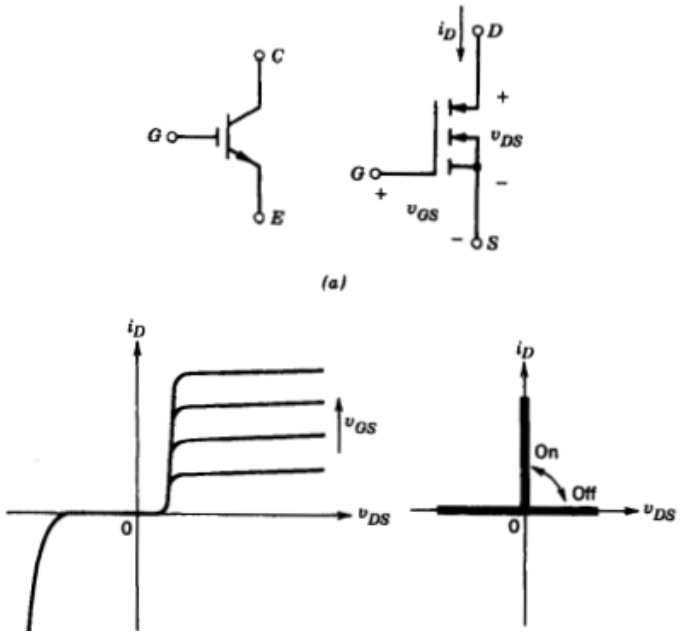


Figure 2.2.7: The IGBT current-voltage characteristics and circuit symbol [21].

2.2.5 Summary

Only a few definite statements can be made in comparing these devices since a number of properties must be considered simultaneously. It should be noted that in addition to the improvements in these devices, new devices are being investigated. The progress in semiconductor technology will undoubtedly lead to higher power ratings, faster switching speeds, and lower costs, which will be discussed later in this chapter when talking about silicon carbide. Figure 2.3.2 shows a summary of power device capabilities regarding power capacity and switching speeds.

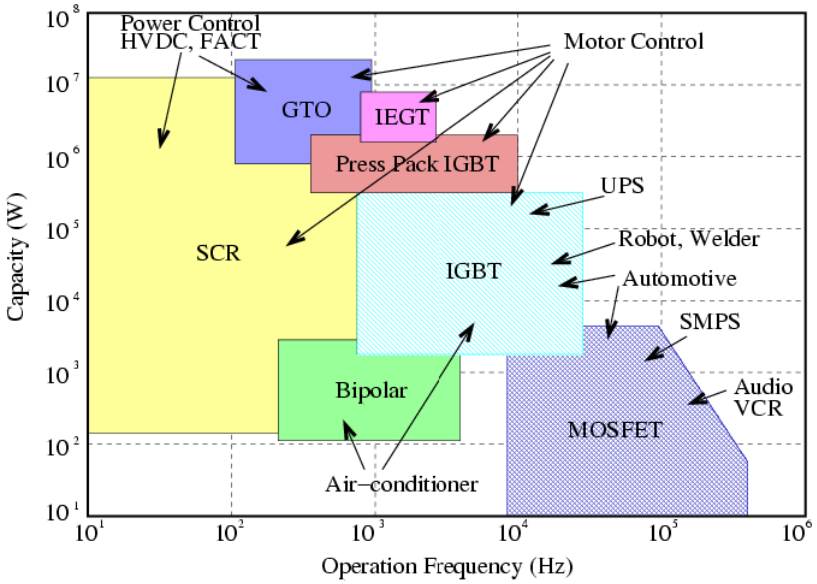


Figure 2.2.8: Summary of power semiconductor device capabilities and applications [20].

2.3 Semiconductors materials

Silicon was the material of choice for making power devices since mid 20th century, but there was always an interest in finding other materials that can overcome the shortcomings of silicon. There mainly two semiconductor materials that are currently situated to replace silicon in power devices, silicon carbide (***SiC***) and gallium nitride (***GaN***). Figure 2.3.1 shows these materials properties in comparison with silicon and how they would impact the power devices made using them.

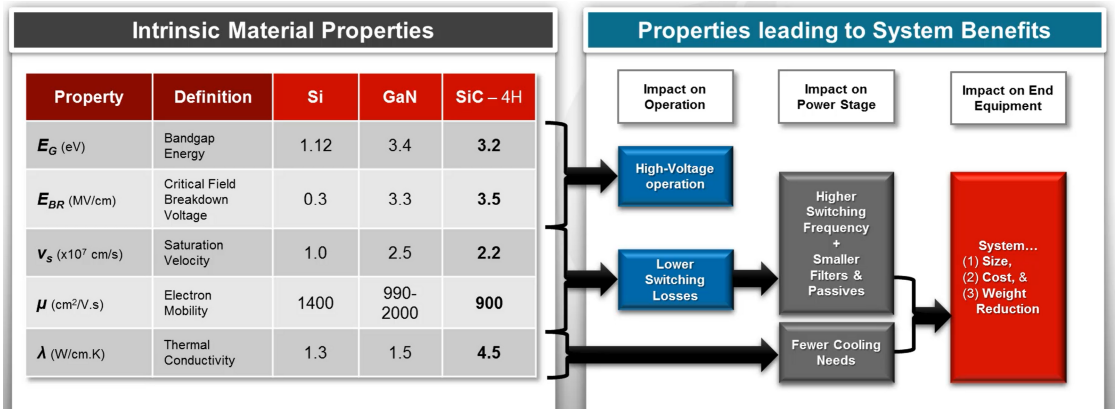


Figure 2.3.1: Semiconductor materials properties compression and power devices benefits [10].

Silicon carbide, as an intrinsic material, has a higher bandgap, significantly higher breakdown voltage, slightly lower electromobility, but almost twice saturation velocity and almost three times the thermal conductivity as silicon as compared to silicon.

A higher bandgap energy and breakdown voltage translates into robust high voltage operation. Higher saturation velocity allows for faster settling, which in turn results in lower switching loss. Lower losses result in higher system efficiency, and also enable the system to operate at higher switching frequencies. Higher switching frequencies help in proportionally reducing the size of the filters in passives, such as inductors, capacitors, transformers, used in a power system.

Higher thermal conductivity results in fewer cooling needs. All these advantages finally resolved in three system level benefits. First, system size reduction. Second, system cost reduction. And third, system weight reduction. Accordingly, all power systems where size, weight, and cost are important performance metrics. Silicon carbide turns out to be a perfect material to develop power devices, especially power MOSFETs switches [10].

Gallium nitride, like silicon carbide, offers similar advantages over silicon, but power devices

producers has not been able to make it commercially viable yet, on the other hand SiC MOSFETs has been available since 2003 and currently on the fourth iteration of them, and they are bid to replace silicon in all power electronics applications in the future as producers improve manufacturing process and devices properties further. Figure 2.3.2 shows the applications that have already adopted silicon carbide MOSFETs and other future adoption candidates.

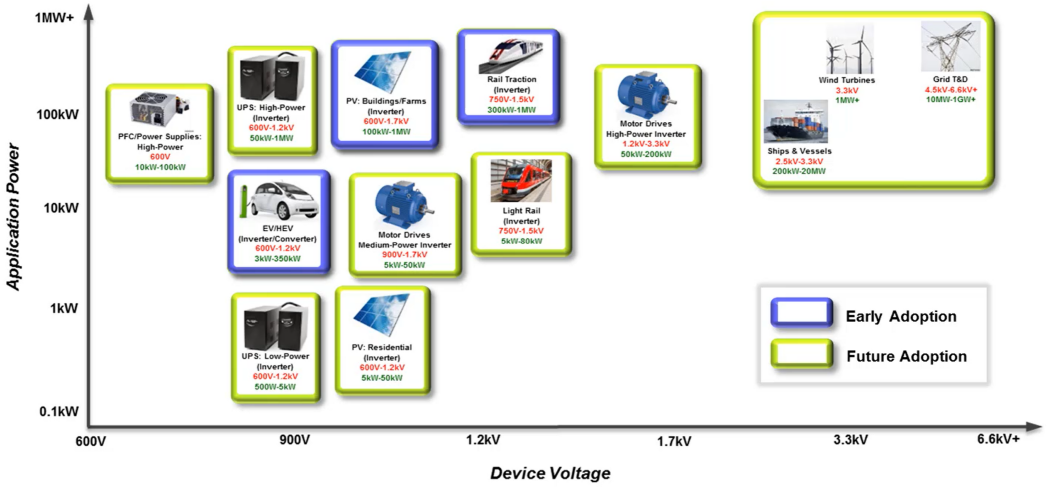


Figure 2.3.2: SiC applications [10].

The applications where silicon carbide MOSFETs are getting adopted first in low volume are highlighted in violet, which are, rail traction, solar for commercial farms and EV traction inverter and DC-DC modules. Rail traction and EV benefit immediately from system size and weight reduction and also benefit from higher temperature operation capability of silicon carbide as compared to silicon IGBTs [10].

Figure 2.3.3 shows how using silicon carbide and gallium nitride opens up the possibility of using high speed switching MOSFETs in higher power applications that has not been possible before, where IGBT/GTO previously dominated, with all the aforementioned benefits that come with high speed switching.

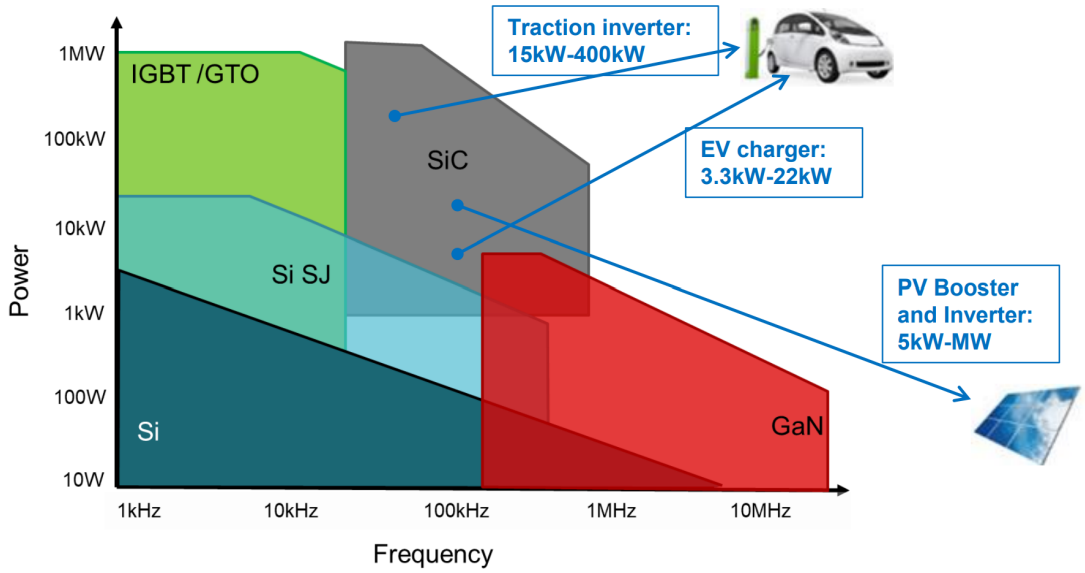


Figure 2.3.3: Power - Switching speed chart [10].

2.4 MOSFET power losses

In accordance with the objectives of this work MOSFETs will be used as power devices, and since efficiency is a major aspect of the inverter performance, power losses in MOSFETs should be discussed and understood. There are four sources of losses in a MOSFET:

1. MOSFET conduction losses.
2. MOSFET switching losses.
3. MOSFET gate losses.
4. General gate losses.

2.4.1 MOSFET conduction losses

Conduction losses for a MOSFET in a basic switching circuit as the one shown in figure 2.4.1 can be calculated using the equation 2.2 for Q1 or Q2.

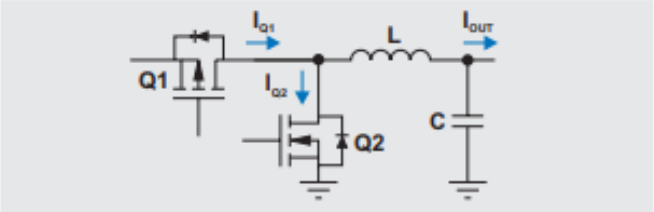


Figure 2.4.1: MOSFET conduction losses [15].

$$\begin{aligned} P_{con} &= R_{DS(on)} \times I^2_{QSW(RMS)} \\ &= R_{DS(on)} \times \frac{V_{out}}{V_{in}} \times (I^2_{out} + \frac{I^2_{Ripple}}{12}) \end{aligned} \tag{2.2}$$

Note that R is the $R_{DS(on)}$ of the selected MOSFET, I is the root-mean-square (RMS) current through the MOSFET, and that neither of these is a function of switching frequency [15].

2.4.2 MOSFET switching losses

Figure 2.4.2 shows how switching losses for a MOSFET and when they happen during the turn-on switching process. The losses happen because a current flows through the MOSFET while the voltage across it is still quite high, this period is divided to t1 and t2 in 2.4.2. As shown in the figure, I_D (the drain current) starts to flow when V_{GS} (gate-to-source voltage) reaches V_{th} (gate-to-source threshold voltage) while there is no change in V_{DS} (drain-to-source voltage). This period is called t1. When V_{GS} reaches V_{plat} (Miller plateau), the current stabilizes and V_{DS} starts to decline, this ends when V_{DS} reaches its final value around zero and the turning on process is concluded when V_{GS} reaches the desired value, usually 10

V for silicon power MOSFETs. The turn off process has similar graph but without the Miller plateau, thus a different period of time where current and voltage exist simultaneously across the MOSFET.

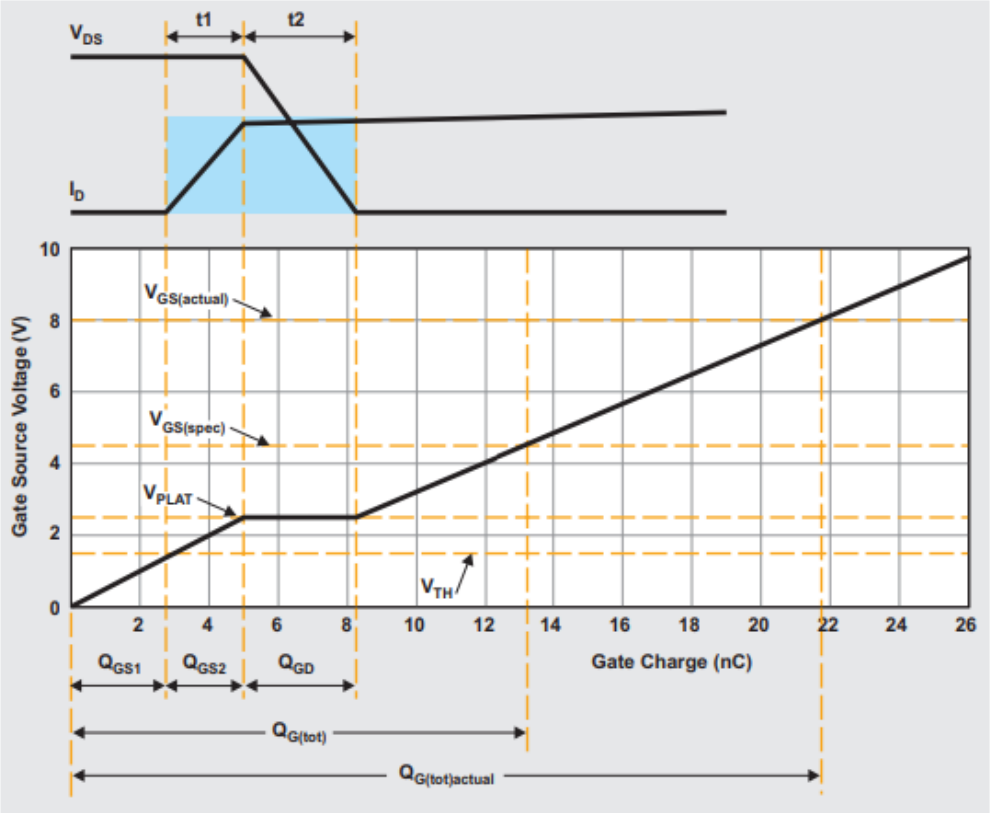


Figure 2.4.2: MOSFET switching losses [15].

Switching losses during turn on can be calculated using the equation 2.3.

$$\begin{aligned} E_{on} &= E_{t1} + E_{t2} \\ &= (V_{DS} \times \frac{I_D}{2}) \times t1 + (\frac{V_{DS}}{2} \times I_D) \times t2 \end{aligned} \tag{2.3}$$

Where V_{DS} is the input voltage, I_D is the load current, t1 ($t1 = \frac{Q_{GS2}}{I_G}$) is dependant on Q_{GS2} and t2 ($t2 = \frac{Q_{GD}}{I_G}$) is dependant on Q_{GD} and both are dependant on I_G, Q_{GS2} and

Q_{GD} are determined by the MOSFET characteristics, while I_G is determined by the MOSFET driver and gate resistance and limited by how much the MOSFET can handle [15], this will be discussed at later point when talking about the MOSFET driver. The equation for losses during turn off is similar 2.4, where $t = \frac{Q_{GD}}{I_G}$.

$$E_{off} = (\frac{V_{DS}}{2} \times \frac{I_D}{2}) \times t \tag{2.4}$$

These switching losses (E_{on} and E_{off}) obviously happens each time the MOSFET switches on/off, and consequently:

$$P_{SW} = (E_{on} + E_{off}) \times f_{SW} \tag{2.5}$$

Equations 2.3, 2.4 and 2.5 shows that the switching losses becomes higher as V_{DS}, I_D and f_{SW} goes higher.

2.4.3 MOSFET gate losses

Switch-MOSFET gate losses are caused by the energy required to charge the MOSFET gate during turn on, which is lost during turn off. That is, the $Q_{G(Tot)}$ at the gate voltage of the circuit. This can be shown in figure 2.4.3 with the imaginary capacitor which approximate how the MOSFET gate-source terminals act. This losses are also switching frequency dependant, which is shown in equation 2.6.

$$P_{Gate} = Q_{G(Tot)} \times V_G \times f_{SW} \tag{2.6}$$

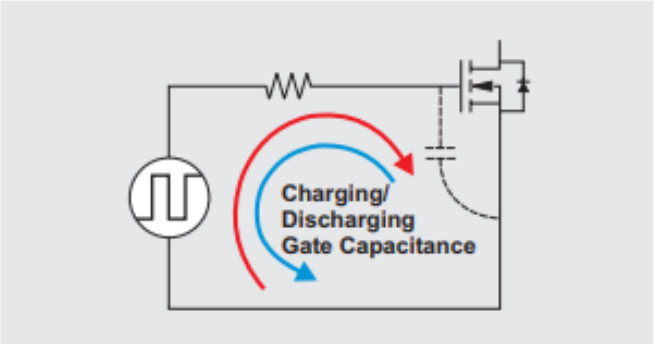


Figure 2.4.3: MOSFET gate losses [15].

2.4.4 General gate losses

Some of the gate-drive losses do not occur on the MOSFET, rather occur at the gate-driver IC as shown in figure 2.4.4. This losses are like the MOSFET gate losses are switching frequency and $Q_{G(Tot)}$ dependant, in addition, they are R_{GHI} and R_{GLO} which are the driver turn on and turn off resistances respectively. These relations are shown in equation 2.7.

$$P_{Drv} = \frac{V_{G\_Drv} \times Q_{G(Tot)} \times f_{SW}}{2} \times \left(\frac{R_{GHI}}{R_{GHI} + R_G + R_{GI}} + \frac{R_{GLO}}{R_{GLO} + R_G + R_{GI}}\right) \tag{2.7}$$

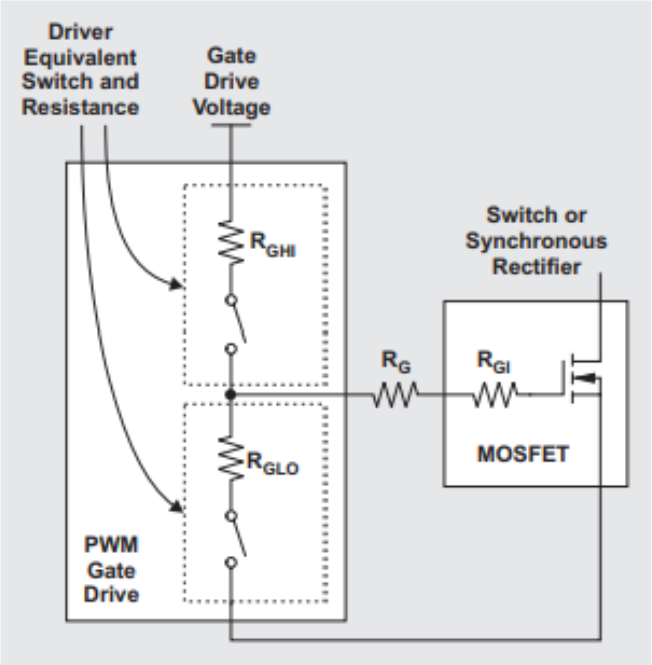


Figure 2.4.4: MOSFET general gate losses [15].

2.4.5 Summary

As explained in the previous sections, some of the MOSFET losses are switching frequency and it is useful to plot how the losses change in relation to switching frequency for a typical MOSFET as shown in 2.4.5. It is worthy to note that larger MOSFETs with lower $R_{DS(on)}$ provide lower conduction losses at the cost of higher gate capacitances $Q_{G(Tot)}$, which results in higher switching losses, gate losses and general gate losses. There is no known method for calculating a "best" $Q_{G(Tot)}$ and $R_{DS(on)}$ in a given situation, although figure-of-merit (FOM) numbers are typically mentioned in data sheets as (FOM = $R_{DS(on)} \times Q_{G(Tot)}$) which try to combine the two very important MOSFET performance metrics into one [15].

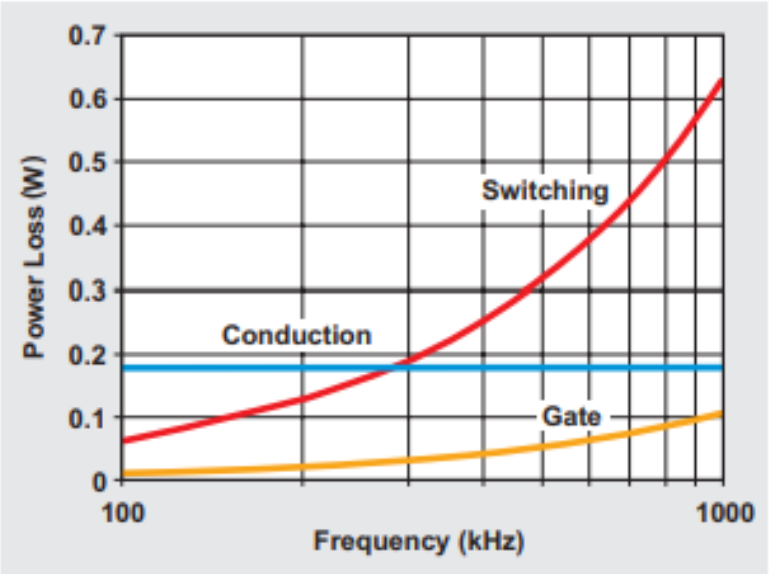


Figure 2.4.5: MOSFET losses in relation to switching frequency for a typical MOSFET[15].

Chapter 3

Hardware Design

This chapter explains the design choices made in the inverter circuit and their reasons; it divides the circuit into blocks based on function and provides the values for the different components that have been used. In addition, it explains the PCB layout and the reasoning behind it describes the production process and production test results.

3.1 Inverter circuit design

The first step is to decide on the topology of the inverter, as discussed in theory chapter 2.1.4, half-bridge, full-bridge, and multi-level inverters were considered (multi-level was deemed complex and unnecessary). The benefits of using full-bridge topology outweigh the drawbacks; therefore, it was the chosen topology.

Evaluating the performance of the inverter can be done using external measuring devices, multi-meters, and oscilloscopes, but it can be useful to be able to monitor key measurements from the circuit with integrated sensors, and it is needed for the output control being worked in the complementary control project; thus it was decided to build sensing elements into the inverter circuit design.

The switching power elements need a control signal; this signal can be generated with the help of a microcontroller; the microcontroller used in this work is the STM32 F3 implemented in STM32F3DISCOVERY kit [23]. This microcontroller was chosen because of its familiarity with the author and ease of use.

Before starting with the design process, a good amount of simulation work has been done to better understand the full-bridge topology, control signal, and the effect of the switching frequency on the output voltage. This work is included in appendix B.

Figure 3.1.1 shows the overall design of the inverter how the different parts interact with each other. In the next sections, each part will be discussed in detail.

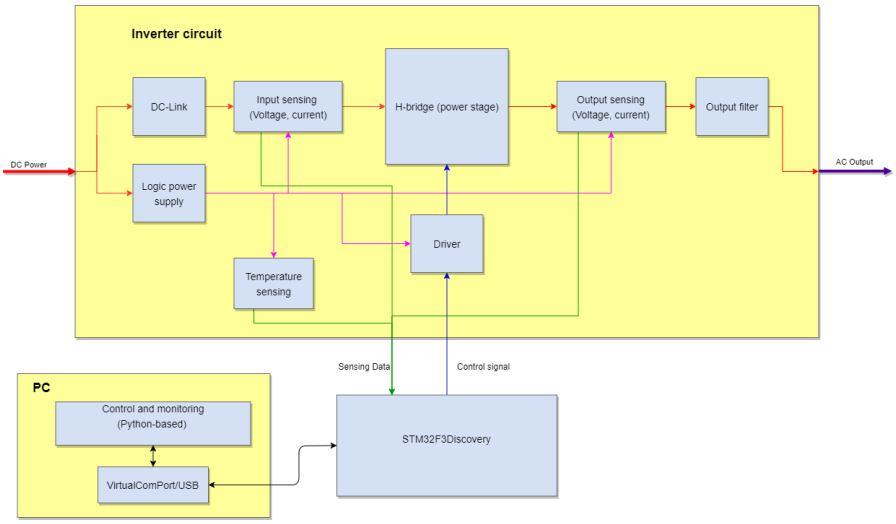


Figure 3.1.1: Overall inverter design.

The design will be done in ***Altium Designer***, the design files, one schematic with two variants and a complementary PCB design and bill of materials of the two variants, in appendix A.

3.1.1 Power stage H-bridge (full bridge)

The first and most important component that should be decided on is the power switching device, and in this work, a silicon MOSFET and a silicon carbide MOSFET are needed. First, the power parameters of the MOSFET must be considered, which are the drain-source breakdown voltage V_{DS} and the drain current I_D. The SiC MOSFET ***IMW65R072M1H 650 V CoolSiC M1 SiC Trench Power Device*** from ***infineon*** [5] is chosen, as the name suggests this MOSFET has a $V_{DS} = 650$ V, which is the lowest available V_{DS} for SiC based MOSFET, and it is much higher the needed 50 V, the drain current that a MOSFET can handle at a certain voltage should be read from a figure in the data-sheet called ***Safe operating area (SOA)***, there is two versions of this figure in the data-sheet of this MOSFET, one at $T_c = 25°C$ and one at $T_c = 80°C$, the figure with $T_c = 80°C$ is used since it provides a more realistic use scenario, this figure is included here 3.1.2, and I_D should be read at V_{DS} =50 V and at the $100\mu s$ line since 10kHz is the lowest switching frequency to be used. This gives an $I_D \approx 9$ A; this gives a power rating of 450 watts, which is more than the 300 watts needs.

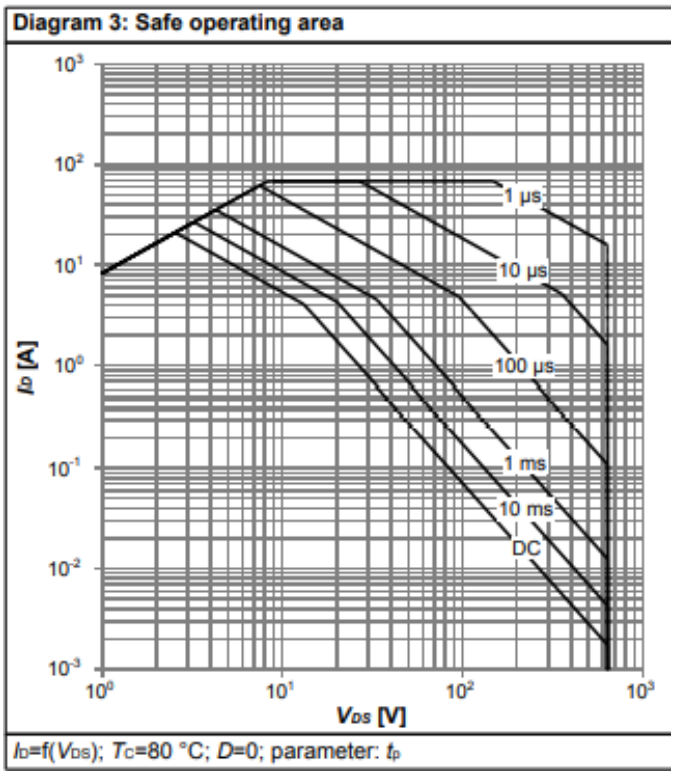


Figure 3.1.2: Safe operating area (SOA) for IMW65R072M1H 650 V CoolSiC M1 SiC at $T_c = 80°C$ [5].

In a similar way, a suitable silicon MOSFET it chosen, the ***IPW60R090CFD7 600V CoolMOS CFD7 Power Transistor*** from ***infineon*** [4], as the name suggests this MOSFET has a $V_{DS} = 600$ V, and it is much higher the needed 50 V, and in a similar way reading from the ***SOA*** from the data-sheet [4], we find $I_D \approx 18$ A, this gives a power rating of 900 watts, which is more than the 300 watts needs.

It is appropriate to compare these two MOSFETs here and to point out the key differences; this compression is shown in table 3.1.1

Parameter	Symbol	IMW65R072M1H	IPW60R090CFD7
Max gate source voltage	V_{GS}	-5/23 V	-20/20 V
Drain-source on-state resistance Tj=25°C	$R_{DS(on)}$	72 mΩ	69 mΩ
Drain-source on-state resistance Tj=150°C	$R_{DS(on)}$	94 mΩ	157 mΩ
Gate charge total	Q_g	22 nC	51 nC
Thermal resistance, junction - case	R_{thJC}	1.3 °C/W	1.0 °C/W
Price at the time of writing	$	7.83 $	15.66 $

Table 3.1.1: IMW65R072M1H CoolSiC M1 SiC and IPW60R090CFD7 CoolMOS CFD7 comparison [5] and [4].

The table shows that these MOSFETs have a similar $R_{DS(on)}$ at Tj=25°C, but the SiC MOSFET is significantly better at Tj=150°C, which will translate to lower conduction losses at higher temperatures. It also shows that the thermal resistance for the SiC MOSFET is slightly worse, but that might not be important as the SiC MOSFET is expected to produce less heat in general. The gate charge total for the SiC MOSFET is almost 2.5 times less than the Si MOSFET, which means fewer switching losses, thus enabling higher switching speeds at the same loss conditions. When it comes to price, the SiC MOSFET costs almost double the Si MOSFET, though the value proposition needs to be considered in the context of the whole system cost, which will be discussed later.

Figure 3.1.3 shows 4 MOSFETs arranged in a full-bridge topology with accompanying freewheeling diodes, which is not strictly necessary when using MOSFETs; the large SOA of the MOSFET means that in most situations, snubber circuits are not needed for the device [21], though they were implemented to see their impact on performance. Some gate-related structures are shown in the figure and will be discussed later.

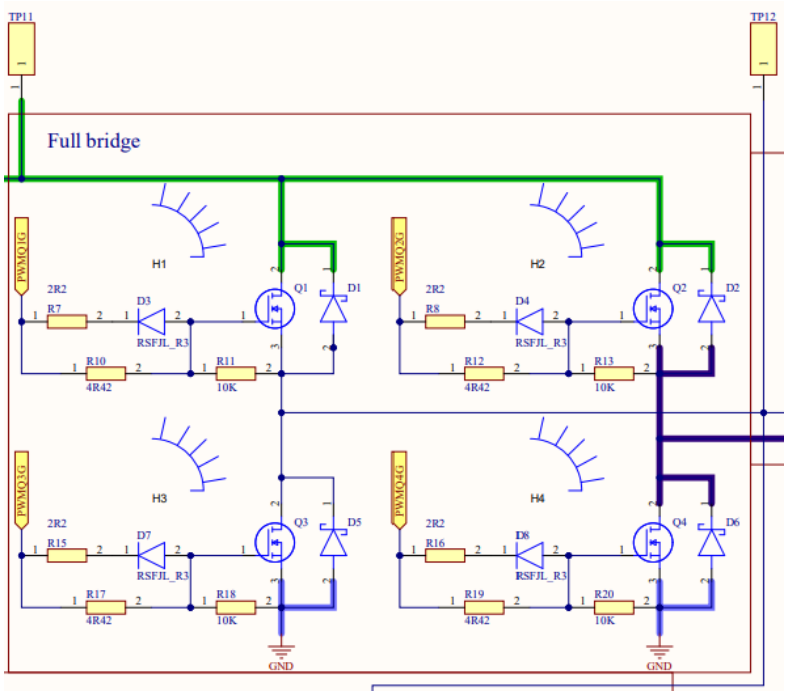


Figure 3.1.3: Full bridge schematics.

3.1.2 Driver ICs

It is possible in some situations to control the power devices using the signal from the microcontroller directly, but the microcontroller can not provide enough voltages and currents to switch the power device correctly; therefore, the so-called driver ICs are often needed to ensure the power devices switch in an optimal way, by providing the needed gate voltages and current in the correct way to minimize losses and to protect the power devices also.

Therefore, choosing the correct driver IC is critical for the correct and safe functioning of the power device. In the next paragraphs, some key attributes of the driver IC will be explained and why they need to be considered.

The first attribute to look at is the output supply voltage which should be higher than the recommended turn on V_{GS}, in our case, more than 18 V and more than 10 V for the SiC and Si MOSFETs, respectively. The second attribute is the source/sink current, which needs

to be high enough to ensure the driver delivers enough currents to achieve the shortest turn on and off times the MOSFET is capable of. Using a lacking driver would result in limiting the MOSFET switching speed, thus increasing the switching times and increasing losses; how this happens is explained in 2.4.2.

It is worth mentioning that SiC MOSFETs have the reputation that they are hard to drive; this was somewhat correct with the first and second generation of SiC MOSFETs, the problem was mainly that they required higher V_{GS} to turn on than their Si counterparts, and higher currents to achieve higher switching speeds, in addition, the drivers that were available when the SiC MOSFETs where introduced was optimized for Si MOSFETs. The situation is much better now with the lower V_{GS} values for the fourth and third generation of the SiC MOSFETs, and with the advances in the driver ICs, that became better suited to drive the SiC MOSFETs as they become more and more widespread.

The choice of the driver IC in this work was guided by ***infineon*** recommendation and implemented with the help of the application note ***Using the EiceDRIVER™ 2EDi product family of dual-channel functional and reinforced isolated MOSFET gate drivers*** [2]. The chosen driver IC is ***2EDS8265H***, which have the capability of driving the SiC and Si MOSFETs used in this work, i.e. max V_{GS} at 20 V, source/sink current at 4/8 A and up to 10 MHz PWM switching frequency. The EiceDRIVER™ 2EDi is a family of fast dual-channel isolated MOSFET gate-driver ICs providing functional (2EDFx) or reinforced (2EDSx) input-to-output isolation by means of coreless transformer (CT) technology. Due to high driving current, excellent common-mode rejection, and fast signal propagation, 2EDi is particularly well suited for driving medium- to high-voltage MOSFETs (CoolMOS™, OptiMOS™, CoolSIC™) in fast-switching power systems [6].

Figure 3.1.4 shows the implementation of the ***2EDS8265H***, this driver is used to control Q1 and Q3 MOSFET in High-low driver configuration; the driver gets the control signal from the microcontroller, the external components, Shunt resistor R24, Bypass capacitors (input C27 and output C28) and Bootstrap circuit (C26, R25, and D10) were chosen according to the application note [2], Gate resistances (R7, R10, R11 and D3 shown in figure 3.1.3) were chosen according to the application note about gate resistors [3], the calculations and

reasoning for the values are included in appendix C.1.

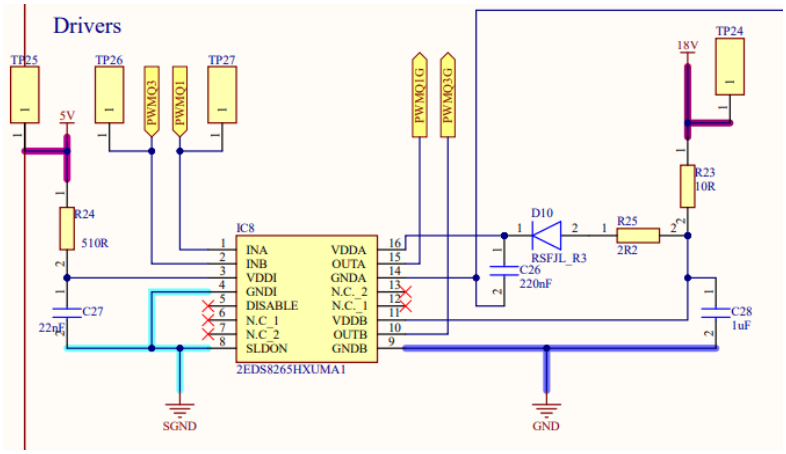


Figure 3.1.4: Driver schematics.

Other features the driver IC have that are worth mentioning are UVLO and DTC.
UVLO stands for Undervoltage Lockout function, which ensures that the output can be switched to its high level only if the gate driver supply voltage exceeds the UVLO threshold voltage. Thus it can be guaranteed that the power switch transistors always stay within their Safe Operating Area (SOA). Otherwise, a too low driving voltage could cause the power MOSFET to enter its saturation (ohmic) region with potentially destructive power dissipation [6], (the UVLO for the used driver is 8 V).
DTC stands for Dead Time Control function, which increases the propagation delay of the rising output voltage by a time t_{DT}. This feature is used in half-bridge applications to prevent the switches from a shoot-through current due to overlapping or jitter on the PWM signals [6].

3.1.3 Sensing

The values that will be measured are the input voltage and current, output voltage and current, and the temperature near one of the high side MOSFETs and near one of the low

side MOSFETs. Voltage and current measurements will allow for the input and power output to be calculated, thus giving an idea about the efficiency of the inverter. The temperature of the MOSFETs will indicate the amount of losses that are happening since these losses are radiated as heat; another use case for the temperature is the possibility to implement a temperature-dependent kill switch to turn off the inverter and prevent damage to the components. The measurements from these sensors are sent to the microcontroller, where they are transformed to digital values and sent to the monitoring software on the computer, as it is shown in the overall system design figure 3.1.1. An important factor to consider when choosing the sensing methods and sensors is the capabilities of the analog to digital converter (ADC) present in the microcontroller, the ADC in STM32 F3 have a voltage range between 0 and 3 V, thus limiting the analog voltage coming from the sensors to this range.

3.1.3.1 DC Sensing

Voltage sensing is done using a simple and suitable voltage divider with resistor values of $1M\Omega$ and $22k\Omega$, which is expected to give values between 0 and 1.1 V, well within the ADC range, corresponding to DC voltage between 0 and 50 V. Current sensing is achieved using the ***Allegro ACS722*** current sensor IC [16], it is a small package, economic and precise solution for AC or DC current sensing in industrial, commercial, and communications systems. The device consists of a precise, low-offset, linear Hall sensor circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which is sensed by the integrated Hall IC and converted into a proportional voltage. The internal resistance of this conductive path is 0.65 $m\Omega$ typical, providing low power loss [16].

The specific part chosen for this design is ***ACS722LLCTR-05AB-T2***, which has the highest sensitivity (264 mV/A) and covers a current range between -5 and 5 A, which is more than the desired 3 A, the output voltage from the current sensor is expected to vary in the range between 1.65 and 2.45 V. This component is implemented in the design with the suitable input bypass capacitors and with the possibility of a low pass filter to be fitted on the output

if needed as shown in figure 3.1.5.

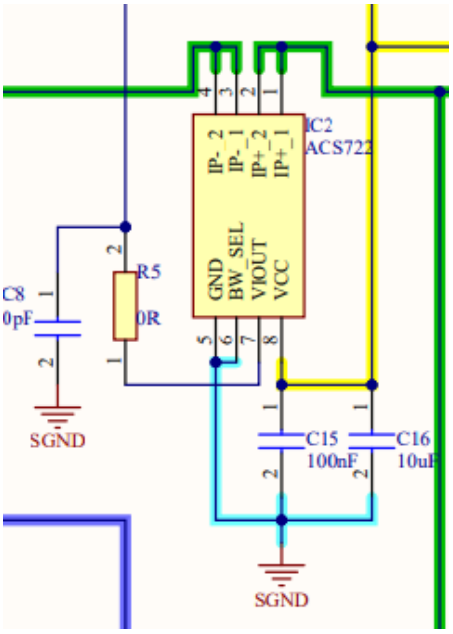


Figure 3.1.5: DC current sensor schematics.

3.1.3.2 AC Sensing

AC voltage and current sensing are done in a similar way to the DC sensing, with the difference of using a voltage offset on the exception of introducing of voltage bias of 1.65 V to the instrumentation amplifier connected to the AC voltage sensing, in order to offset the voltage from $[-1.1 - 1.1]V$ to $[0.55 - 2.75]V$, which is suitable for the microcontroller ADC. The output from the current sensor is expected to vary in the range between 0.85 and 2.45 V.

3.1.3.3 Temperature sensing

A generic and suitable temperature sensor is chosen, ***TMP35GRTZ***, which is a low voltage, precision centigrade temperature sensors. It provides a voltage output that is linearly proportional to the Celsius (Centigrade) temperature. The TMP35 does not require any ex-

ternal calibration to provide typical accuracies of ±1°C at +25°C, and ±2°C over the -40°C to +125°C temperature range [8].

This component is implemented in the design with the suitable input bypass capacitors and with the possibility of a low pass filter to be fitted on the output if needed, as shown in figure 3.1.6.

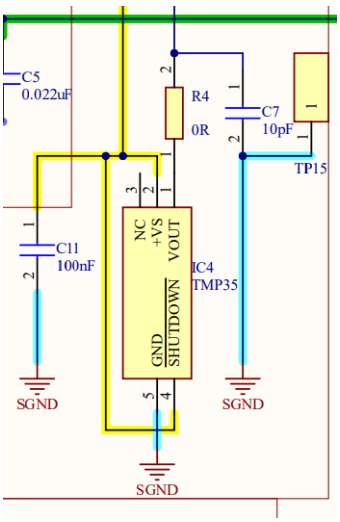


Figure 3.1.6: Temperature sensor schematics.

3.1.3.4 Instrumentation amplifier

In order to get accurate measurements, it is necessary to minimize the effect of the microcontroller on the sensor output voltages. For this purpose, a suitable instrumentation amplifier is used, ***LT1789-1***, which is micropower, precision instrumentation amplifiers that are optimized for single-supply operation from 2.2V to 36V. The gain is set with a single external resistor for a gain range of 1 to 1000. The output can handle capacitive loads up to 400pF in any gain configuration while the inputs are ESD protected up to 10kV (human body) [7]. This component is implemented in the design with the suitable input bypass capacitors and with no gain resistor, resulting in a gain of 1, as shown in figure 3.1.7.

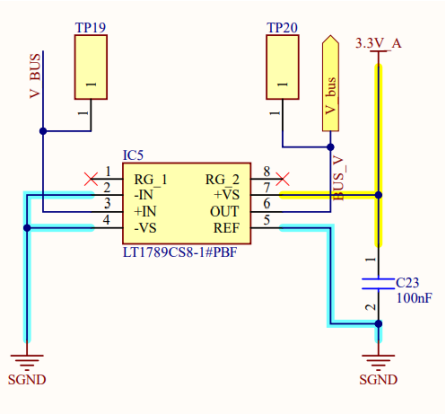


Figure 3.1.7: Instrumentation amplifier schematics.

3.1.4 Power supply

The integrated circuits used in this design were chosen to have wide supply voltage ranges, thus minimizing the need for different supply voltages; nonetheless, the design contains the following supply voltages:

- Analog 3.3 v, for the current sensors, temperature sensors, and the instrumentation amplifiers.
- Analog 1.65 v, as a bias voltage for one of the instrumentation amplifiers.
- Digital 5 V, for the analog voltage regulator and the MOSFET driver ICs logic side.
- Digital 10 or 18 V, for the MOSFET driver ICs power supply side and the second switching regulator.

To produce this voltages two switching regulators, ***LM22675-5.0*** [12], were used for the digital voltages and one linear analog voltage reference IC for the two analog voltages, ***REF2033*** [13]. These component are implemented in the design with the suitable input and output bypass capacitors as shown in figure 3.1.8.

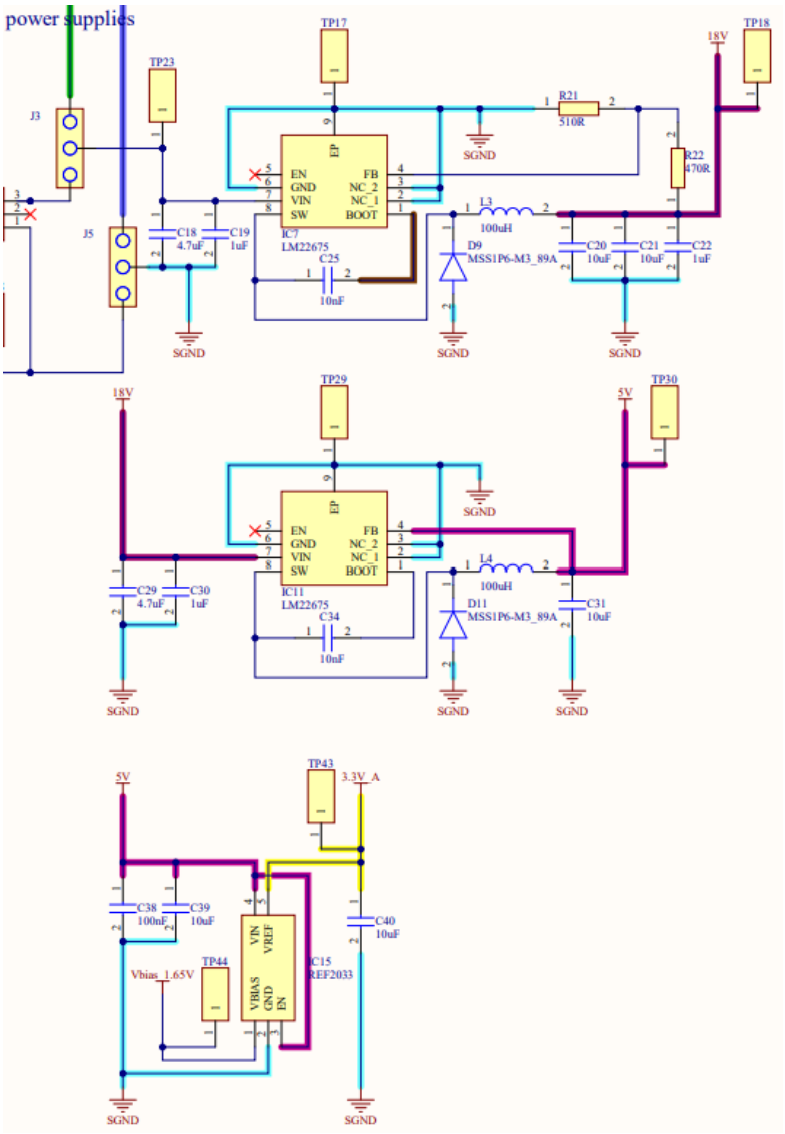


Figure 3.1.8: Power supplies schematics.

Rough current calculations were done to make sure that the power draw from the different ICs will not exceed the capabilities of the chosen power supply ICs.

3.1.4.1 Switching voltage regulator

The ***LM22675-5.0*** switching regulator provides all of the functions necessary to implement an efficient high voltage step-down (buck, y used to convert a higher dc voltage to a lower dc voltage) regulator using a minimum of external components. This easy-to-use regulator incorporates a 42 V N-channel MOSFET switch capable of providing up to 1 A of load current. Excellent line and load regulation along with high efficiency (> 90%) are featured. A switching frequency of 500 kHz allows for small external components and a good transient response. Built-in soft-start (500 µs, typ) saves external components. The LM22675 also has built-in thermal shutdown, and current limiting to protect against accidental overloads [12]. The aforementioned features coupled with a wide input voltage range 4.5 to 42 V and adjustable output voltage make this component a good choice for this design. As shown in figure 3.1.8 ***LM22675-5.0*** is used as IC7 and IC8. Where IC8 is used to step down the voltage from 18 V or 10 V for the SiC and Si variants respectively to 5 V, thus eliminating the need for a voltage divider and the feedback pin (FB) is connected to the output directly, as the internal Loop Compensation for the -5.0 option is optimized for output voltages of 5 V. While IC7 is used to step down the input voltage to 18 V or 10 V for the SiC and Si variants respectively, thus requiring a voltage divider on the output to get the desired voltages. The voltage divider values are calculated according to the data-sheet guidelines for 10 V and 18 V output voltage; these calculations are included in appendix C.2.

3.1.5 DC Link

The bus link (DC link) capacitor is used in DC to AC inverters to decouple the effects of the inductance from the DC voltage source to the power bridge in hard switched pulse width modulated (PWM) inverter that such as the one used in this work. The bus link capacitor provides a low impedance path for the ripple currents associated with a hard switched inverter. The ripple currents are a result of the output inductance of the load, the bus voltage, and the PWM frequency of the inverter [22].

This ripple currents and voltages affect the power source of the inverter in a negative way, and a properly sized DC link capacitor to minimize these effects. Since the PWM frequency is a contributing factor to this ripple effect, the ability to use higher switching speeds with the SiC MOSFET will result in a reduction in the DC link capacitor size depending on the switching frequency chosen. These calculations for both the Si and SiC variants are included in appendix C.3.

3.1.6 Output filter

The output filter is an LC low pass filter that filters out the switching frequency and the switching frequency harmonics. The calculations for the values of L and C for both the Si and SiC variants are included in appendix C.4.

3.1.6.1 Inductor design

The primary role of the inductor in the output filter is to filter out the switching frequency harmonics. The design of an inductor, amongst other factors, depends on the calculation of the current ripple and choosing a material for the core that can tolerate the calculated current ripple [14].

3.1.6.2 Capacitance selection

The output inductor and capacitor form a low pass filter that filters out the switching frequency. To get good switching frequency attenuation, the cut-off frequency is kept at F_{sw}/10 or lower [14].

3.1.7 Other design elements

Some other design elements worth mentioning:

- Heat sink dimensioning, doing some rough calculations using the planed power rating and switching frequency, it is found that the heat produced from the MOSFET should not exceed 5 W, taking $90°C$ as the desired case temperature for the MOSFET, and ambient temperature of $20°C$, $R_{thJC} = 1.3°C/W$a heat sink is needed with the maximum heat resistance of $12.7°C/W$ (Calculation done using an online calculator [1]). A suitable heat sink with a thermal resistance of $12°C/W$ is chosen [9].
- Since the maximum voltage the switching voltage regulator can handle is 42 V, the design needed to have two input sources, one to supply the power stage and the other to supply the logic and control ICs; an added benefit would be isolating the logic from the power stage.
- An abundance of test points are used to make the troubleshooting process and direct measurements easy, except where it can affect performance, for example, near the gate of the MOSFET where any stray inductances or capacitances can affect the performance negatively.

3.2 Inverter circuit production

After finishing the design schematics, the next step is implementing the circuit to a printed circuit board (PCB); this is accomplished by producing the PCB and soldering the components to it. The available method of producing PCBs at the university is a milling machine that can produce two layers circuit boards, this machine has certain limitations regarding the width of the tracks due to the available milling bits, milling drills, and accuracy of the milling head. Thus, certain rules should be considered when designing the PCB layout.
The software used for the PCB design is, as mentioned before is ***Altium Designer***. Most of the 2D and 3D models of the components used were available on the component sellers' websites and were imported to the design program, or they were created from the data-sheets as Altium Designer allows modifying the elements or creating new symbols and packages.
The basic general design rules are followed when designing the layout, such as:

- As small as possible PCB and shortest tracks possible to reduce cost, minimize parasitic elements and reduce noise susceptibility.
- Mounting most of the components on the top layer to make the soldering process easier and later testing easier also.
- Minimizing the tracks in the bottom layer in order to have a large continuous ground plan on the bottom layer, which will help reduce the noise.
- As wide as possible traces to be able to handle the amount of power going through them.
- Some specific rules linked to the milling machine used such as: the minimum distance between the ICs pins (8-10 mill, 0.2-0.25 mm), the minimum distance between the tracks (8-10 mill, 0.2-0.25 mm), and minimizing the vias to make the soldering process easier since the vias are not plated and require running a wire through them.

Some examples of following these rules will be shown in the next section with some other

design elements and the reasoning behind them. Some of the components of data-sheets include layout recommendations, which were taken into consideration throughout the design process.

3.2.1 PCB Layout

Figure 3.2.1 shows the PCB top layer, with the following color coding: Green positive voltage input (PWD+), light purple negative voltage input (PWR-GND), brown positive output voltage (AC+), dark purple negative output voltage (AC-), cyan logic ground, dark pink is 18/10 V, light pink is 5 V, and yellow is 3.3 V.

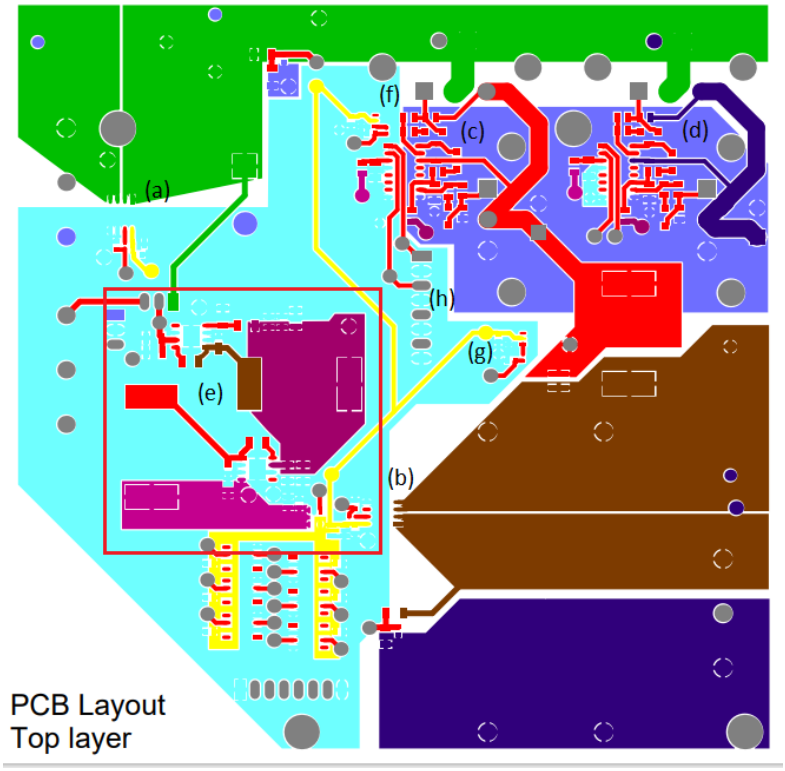


Figure 3.2.1: PCB layout top layer, better resolution found here A.3.

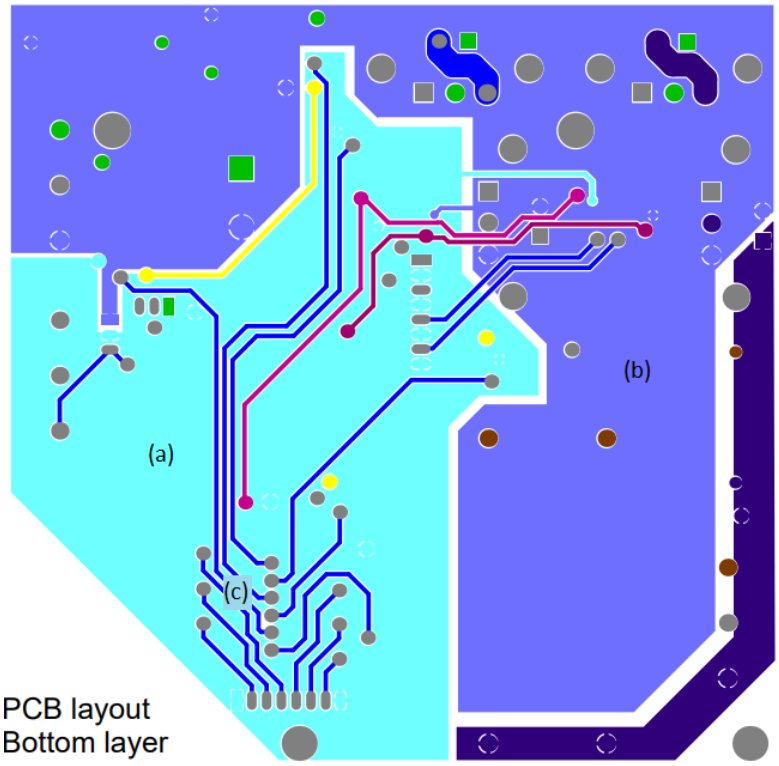


Figure 3.2.2: PCB layout bottom layer, better resolution found here A.4.

Design elements worth mentioning are listed here:

- Wide power stage tracks (Green, red, brown, dark purple).
- Current sensing IC according to layout recommendation to reduce interference with signal side, figure 3.2.1 (a) and (b).
- Driver IC according to layout recommendation to achieve optimal performance, figure 3.2.1 (c) and (d).
- Grouping the power supply for the logic ICs together and separating them from the power stage with ground, figure 3.2.1 (e).
- Temperature sensors near the high side (f) and low side (g) MOSFETs, figure 3.2.1.

- Control signal jumper as close as possible to the driver ICs to reduce interference, figure 3.2.1 (h).
- Large continuous ground planes on the bottom layer, figure 3.2.2 (a) and (b).
- Signal tracks on the bottom layer to protect them from the noise of the switching elements present on the top side, figure 3.2.2 (c).

3.2.2 Mounting components

When the first version of the PCB layout was produced, it was obvious that soldering the original instrumentation amplifier used [11] will prove to be difficult to solder as shown in figure 3.2.3, as the soldering pads are very narrow and thin, which increases the chance of the pad breaking off the PCB material, nonetheless soldering the component was attempted hoping to save time, by not modifying the design and avoiding the need to order new parts, the attempt failed as shown in figure 3.2.3 and valuable time was lost. The problem with soldering stemmed from the fact that the milling machine bit was not able to stick to the specified (8-10 mill, 0.2-0.25 mm), and the minimum milling distance the machine produced was almost (20 mill, 0.5 mm). The desire to use the original instrumentation amplifier was because it was two amplifiers in the same package, which would have reduced the number of components, reducing the cost and making the soldering process easier, which backfired horribly. A new instrumentation amplifier was chosen, and the schematics and layout were modified, and the PCB was reproduced.

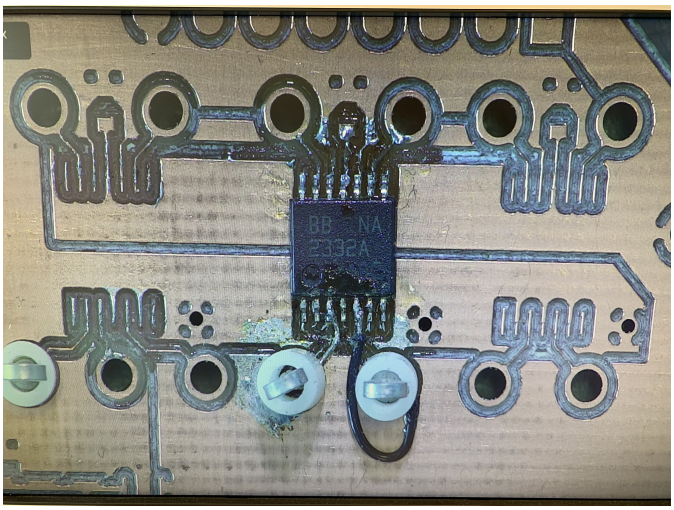


Figure 3.2.3: Old instrumentation amplifier mistake.

Afterward, the mounting of the components was done beginning with the power supply ICs first, to make sure that the correct voltages from the regulators ICs, in order to not to damage the other ICs with over or under voltage, and that's where the second challenge was faced, where a mistake in the schematics lead to that pin 1 (bootstrap) instead of pin 8 (switching) being connected to the inductor, this mistake was costly and took some time before it was discovered, it was corrected by severing the connection between pin 1 and the inductor and connecting a jumper wire from pin 8 to the inductor as shown here in figure 3.2.4, this mistake is corrected in the schematics included in the appendix, but the PCB layout was not updated due to lack of time.

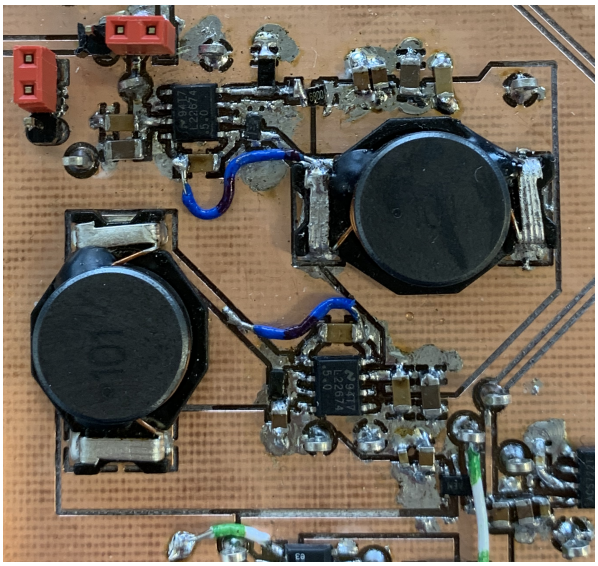


Figure 3.2.4: Switching regulators fix.

After getting the desired voltages from the switching regulators, the analog power supply was mounted, then the new instrumentation amplifiers were mounted; another mistake was discovered when testing them, in a rush to modify the layout design, the ground connection of the instrumentation amplifiers was not connected, this had to be corrected with jumper wires shown in this figure 3.2.5; also some more jumper wires were needed for one of the instrumentation amplifiers to function correctly, these mistakes are corrected in the schematics included in the appendix, but the PCB layout was not updated due to lack of time.

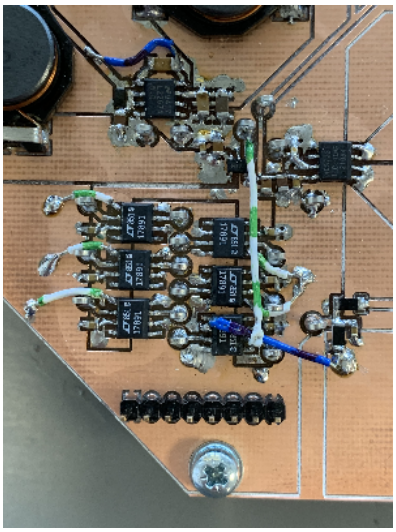
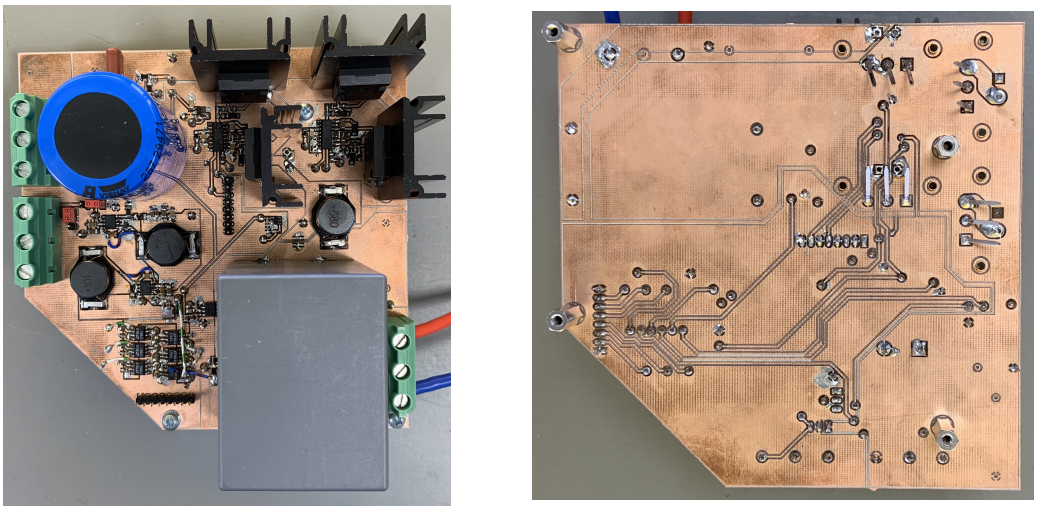


Figure 3.2.5: Instrumentation amplifiers fix.

Unfortunately, another challenge was faced during the mounting process, when attempting to mount the MOSFET driver ICs, these ICs came in the wrong package, the wide variant (WB-DSO16) instead of the narrow variant (NB-DSO16); this mistake happened as a result of using a premade incorrect footprint for the component. The correct component was ordered, but this caused some delay in the project. This mistake is corrected in the included design files in appendix A.

After getting the new components, mounting the rest of the components went smoothly without any further problems; the finished PCB is shown here with the Si variant components is shown in this figure 3.2.6, extra pictures from different angles with close-ups of some of the components are included in appendix A.8.



(a) Top layer

(b) Bottom layer

Figure 3.2.6: PCB with Si components.

Chapter 4

Software Design

This chapter provides an introduction of the microcontroller, its different capabilities, and its implementation in this work. It also explains how the desired functions are achieved, outlines the program flow, gives an overview of the source code, and highlights important parts of it. In addition, it explains the **Python** code used to process and visualize the data that the microcontroller produces. The information presented here lacks the in-depth details of how each module works and is limited to the scope of just documenting the code used; it was done this way because of the lack of time when writing this work.

4.1 Microcontroller

In this work, the STM32F3DISCOVERY is used to control and monitor the inverter circuit. This kit is designed with the STM32 F3 32-bit ARM® Cortex™-M4 mixed-signal MCU, the STM32F303VCT6 microcontroller in particular, and also includes an ST-LINK/V2 embedded debug tool interface, Gyroscope ST MEMS, E-compass with accelerometer ST MEMS, LEDs, pushbuttons, and a USB mini-B connector [23].
This STM32F303VCT6 has ARM™Cortex-M4 32-bit MCU with FPU has 256 KB Flash, 48

KB SRAM, 4 ADCs, two DAC channels, seven comparators, four PGAs, 13 timers, 2.0-3.6 V operation, which makes it suitable for this project. The microcontroller is used in conjunction with Discovery shield [17] in order to make USART communication with a computer over USB possible (Discovery shield has a USB MINI-B connector that is connected to PA02 and PA03 i.e. USART2 tx and rx).

The main modules that are used in this work are GPIO, timers, ADC, and USART; their implementation is explained in the coming paragraphs. The general structure used for implementing these modules is as follows, first enabling the module, second enabling and configuring relevant GPIO modules, third configuring the module, fourth enabling any relevant global interrupts; and finally, enabling the module itself if needed.

The developing environment used is ***STM32CubeIDE***, some standard code is written by the IDE when creating a new project, which takes care of the standard configurations and hardware initialization, the full code that is written by the author of the project, in addition to a zip file of the whole project is included in appendix D.

Figure 4.1.1 shows the program flow chart, a combination of flags and hardware interrupts are used to control the program flow, DMA is used to transfer data whenever possible to free up the CPU time. The code included in this work lacks some of the functions described in the flow chart; these functions are marked with a yellow color.

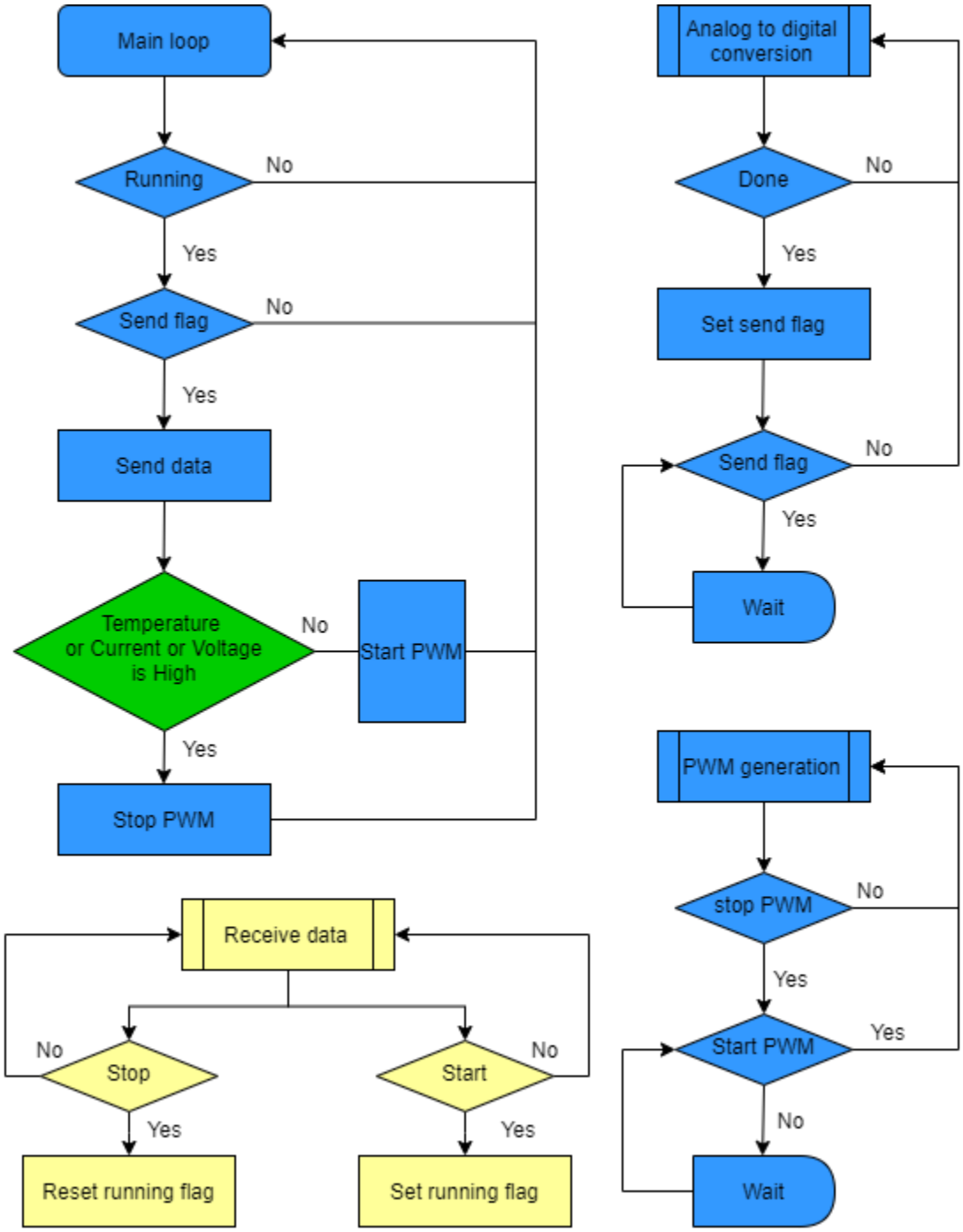


Figure 4.1.1: Microcontroller program flow chart.

4.1.1 GPIO

GPIO stands for General-Purpose Input/Output; most of the other modules explained in this chapter are used with external components; therefore, their respective GPIO pins should be configured accordingly. An example of this is code listing 4.1, where pins 2 and 3 in GPIO module A are configured to function as data sending and receiving pins for the USART2 module. Activating any module also involves enabling the module by turning on the clock signal for the specific module as shown on line 19, where the DMA1 controller and GPIOA modules are enabled by setting their respective pins to one in the RCC APB1ENR (RCC APB1 peripheral clock enable register).

Listing 4.1: GPIO configuration for USART2

```
19          RCC->AHBENR |= RCC_AHBENR_DMA1EN | RCC_AHBENR_GPIOAEN;
20
21          /* GPIO Configuration */
22          GPIO_InitTypeDef GPIO_InitStruct;
23          GPIO_InitStruct.Pin = GPIO_PIN_2 | GPIO_PIN_3;
24          GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
25          GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
26          GPIO_InitStruct.Pull = GPIO_NOPULL;
27          HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

As we see in the code snippet, first we need to create a general GPIO initialization structure on line 22, then choose which pins we need to configure on line 23, choosing the correct alternative function and correct mode on line 24 and line 25, choosing the pull-up or down for the selected pins on line 26 and finally initializing the correct module with the above configuration on line 27.

This process is of configuring a GPIO is common to other peripherals whenever interaction with the outside world is necessary; this process is repeated for each module with some variation according to how they should be configured, which can be figured out by consulting the microcontroller data-sheet [24] and the reference manual [18].

Similarly, other GPIO modules are configured for USART2, TIM2, and ADC4, at the beginning of each respective code file included in appendix D, in addition to code file gpio.c D.3, where the GPIOE is configured for troubleshooting purposes.

4.1.2 DMA

DMA stands for Direct Memory Access, the flexible general-purpose DMA is able to manage memory-to-memory, peripheral-to-memory and memory-to-peripheral transfers. DMA2 controller channel 2 will be used to transfer data from the ADC data register to memory, DMA1 controller channel 7 from memory to USART transmit data register and DMA1 controller channel 6 from USART receive data register to memory. Specific channels from the DMA controllers are connected to specific modules in the microcontroller, there these must be used according to the microcontroller data-sheet [24] and the reference manual [18].
As an example of this is code listing 4.2, where channel 6 and channel 7 of the DMA1 controller is configured for the USART2 module.

Listing 4.2: DMA1 controller configuration for USART2, channel 6 and 7

```
29          //Rx DMA
30          DMA1_Channel6->CNDTR = USART2_RX_BUFF_LEN;
31          DMA1_Channel6->CPAR = (uint32_t)&(USART2->RDR);
32          DMA1_Channel6->CMAR = (uint32_t)Usart2RxBuff;
33          DMA1_Channel6->CCR = DMA_CCR_PL_1 | DMA_CCR_MINC | DMA_CCR_CIRC | DMA_CCR_TEIE | \
34                               DMA_CCR_TCIE | DMA_CCR_EN;
35
36          //Tx DMA
37          DMA1_Channel7->CPAR = (uint32_t)&(USART2->TDR);
38          DMA1_Channel7->CMAR = (uint32_t)Usart2TxBuff;
39          DMA1_Channel7->CCR = DMA_CCR_MINC | DMA_CCR_DIR | DMA_CCR_TCIE | DMA_CCR_TEIE;
```

As we see in the code snippet, on line 30 the size of the data buffer the DMA channel can use for the data received from the USART2 is defined, this value is set in the register called DMA channel number of data register, on line 31 the address of the desired peripheral data register

is defined, this value is set in the register called DMA channel peripheral address register, on line 32 the desired memory address is defined, this value is set in the register called DMA channel memory address register, on line 33 several settings are configured in what is called DMA channel configuration register, these options are:

- Defining the priority of the channel as DMA_CCR_PL_1 (as 10 which means high on a scale from very high to low), so that the sending takes priority over receiving.
- Activating the memory increment mode by setting its respective bit to 1, in order to increment the memory address after each DMA operation.
- Activating the circular mode by setting its respective bit to 1, this will make the DMA controller return to the start of the buffer when reaching the last address in the buffer).
- Enabling the transfer error interrupt by setting its respective bit to 1, in order to raise the interrupt flag whenever there is an error.
- Enabling the transfer complete interrupt by setting its respective bit to 1, in order to raise the interrupt flag when the transfer is complete.
- Finally setting the channel enable bit to 1, so the channel become active.

DMA1 channel 7 is configured similarly except for the following differences:

- Different addresses for the memory and the peripheral.
- The data transfer direction bit is set to 1, to configure in reading from memory mode.
- The size of the memory buffer is not set yet, and rather set on line 62 in data sending shown here in this code listing 4.3.
- The priority bit is left at zero, giving this channel the highest priority.
- Circular mode bit is left at zero, which means that the DMA controller will stop when reaching the last address in the memory buffer.

- finally the channel is left disabled and rather enabled later when the data sending function is called, shown here on line 63 in this code listing 4.3

Listing 4.3: DMA1 channel 7 configuration in USART2 data sending function

```
55 //Putting data in a vector that DMA-transfer sends automatically
56 int8_t usart2_tx( uint8_t const * const data, uint8_t const len ){
57         GPIOE->BSRR = GPIO_BSRR_BS_15;
58
59         if( len > USART2_TX_BUFF_LEN ) return -1;
60         DMA1_Channel7->CCR &= ~DMA_CCR_EN;
61         for(uint8_t i = 0; i < len; i++) Usart2TxBuff[i] = data[i];
62         DMA1_Channel7->CNDTR = len;
63         DMA1_Channel7->CCR |= DMA_CCR_EN;
64
65         GPIOE->BSRR = GPIO_BSRR_BR_15;
66         return 0;
67 }
```

DMA2 channel 2 is used in a similar way to handle the ADC4 data transfer, and this is shown in appendix D.2.

4.1.3 ADC

ADC stands for Analog-to-Digital converter, the ADCs present on the microcontroller has several useful features that are used in this work, such as e resolution between 12 and 6 bit, multi-channel sampling, timer-based operation, and the ability to use a DMA controller. Six channels of the ADC4 are used to convert the voltage values from the inverter circuit sensors to digital values, to be sent to the PC monitoring program and/or used in the microcontroller code.

To do this, ADC4 needs to be configured properly in addition to DMA2 controller to handle the data transfer from the ADC to the memory since we are reading multiple input channels while the ADC has only one data register which is overwritten with new data after each channel conversion. This could have also been done without using the DMA controller by

implementing interrupts with the downside of using too much of the CPU resources. The focus in this section will be only on the ADC configuration since the DMA configuration is discussed earlier in section 4.1.2 and timer configuration will be explained later in section 4.1.5.

After starting up ADC4 and calibrating it in a standard fashion, code listing 4.4 show the further configurations needed to set up the ADC for the desired function.

Listing 4.4: ADC4 configuration

```
78          ADC4->CFGR = ADC_CFGR_EXTEN_0 | 0xE << ADC_CFGR_EXTSEL_Pos | ADC_CFGR_DMAEN | \
79                       ADC_CFGR_DISCEN ;
80                       //EXT14 TIM15_TRGO event;
81                       //RES[1:0]: Data resolution is 00: 12-bit; ADC DMA enable
82          ADC4->SMPR1 =  ADC_SMPR1_SMP3_1 |  ADC_SMPR1_SMP4_1 | \
83                                   ADC_SMPR1_SMP7_1 |  ADC_SMPR1_SMP8_1 ;
84          //110: Tsmp = 181.5/16MHz = 11us for channel 3,4,7 and 8
85          ADC4->SMPR2 =  ADC_SMPR2_SMP12_1 | ADC_SMPR2_SMP13_1;
86          //110: Tsmp = 181.5/16MHz = 11us for channel 12 and 13
87
88          ADC4->SQR1 = ADC_SQR1_L_2 | ADC_SQR1_L_1 | ADC_SQR1_SQ1_0 | ADC_SQR1_SQ1_1 \
89                       | ADC_SQR1_SQ2_2 |ADC_SQR1_SQ3_2 | ADC_SQR1_SQ3_1 | ADC_SQR1_SQ3_0 \
90                       | ADC_SQR1_SQ4_3; // Regular sequence 6 channels, channel 3,4,7 and 8
91          ADC4->SQR2 = ADC_SQR2_SQ5_3 | ADC_SQR2_SQ5_2 | ADC_SQR2_SQ6_3 | ADC_SQR2_SQ6_2 \
92                       | ADC_SQR2_SQ6_0; // channel 12 and 13
93
94          NVIC_EnableIRQ( ADC4_IRQn );
95          NVIC_EnableIRQ( DMA2_Channel2_IRQn );
96          ADC4->CR |= ADC_CR_ADSTART;
```

The following registers are configured:

- ADC Configuration register CFGR: Setting up the ADC to be controlled by and external high trigger (ADC_CFGR_EXTEN_), setting up the trigger to EXT14 which corresponds to trigger out event from TIM15, enabling the DMA to be used, leaving the data resolution bits at 00 in order to have the maximum resolution of 12 bit.
- ADC sample time register 1 SMPR1: determining the sampling time for channel 3, 4,

7 and 8, all the used channels are sett to the same value of 11us, which is enough to charge the ADC capacitor and give the correct value of the respective signal that is connected to the channel.

- ADC sample time register 2 SMPR2: the same is done here as in SMPR1 for channel 12 and 13.
- ADC regular sequence register 1 SQR1: The order of the regular conversion sequence is set up here, by setting total number of channels the sequence contains and the code for the first four channels.
- ADC regular sequence register 2 SQR2: for the rest of the channels.

Then enabling the relevant interrupts and starting the ADC at the end.

4.1.4 USART

USART stands for universal synchronous/asynchronous receiver transmitter, this module can be used to digitally communicate with other devices, and it is used here to send and receive data from a Python program running on a connected computer. This communication is achieved by implementing the previously discussed DMA1 module, and GPIO A module; the code more specific to the USART module is shown in the code listing 4.5.

Listing 4.5: USART2 configuration

```
41          //USART
42          USART2->BRR = 0x10; //16; //Bandwidth rate=32M/(0xF(16))=2Mb
43          USART2->CR3 = USART_CR3_DDRE | USART_CR3_DMAT | USART_CR3_DMAR | USART_CR3_EIE;
44          USART2->CR1 = /*USART_CR1_PCE | USART_CR1_PS |*//* USART_CR1_PEIE |*/ \
45                        USART_CR1_IDLEIE | USART_CR1_TE | USART_CR1_RE | USART_CR1_UE;
```

The following registers are configured:

- USART Baud rate register BRR: the speed of communication over USART2 is set up to 2Mb, which is the maximum rate allowed since the clock the USART2 uses is the system clock divided by 2 (64Mb/2=32Mb), and with 16 as over sampling rate, this gives 32Mb/16=2Mb. It is possible to use such a high speed without any concern for the integrity of the data, because of the short distance the signals travel. This speed is enough for the use case here as it will be shown here when testing the maximum sampling rate for the sensing signals 5.1.1.

- USART Control register 3 CR3: where the DMA for the transmitter and receiver, the action of disabling the DMA on reception error, and error interrupt are enabled.

- USART Control register 1 CR1: where the following options are enabled: IDLE Interrupt Enable (which is raised when USART is not idle and has received data), Transmitter Enable, Receiver Enable and USART Enable. By leaving the following control bits in CR1 as zeros, the following USART settings is achieved by:

  - Bit 28 M1 and Bit12 M0 Word length: M[1:0]= 00: gives 1 Start bit, 8 data bits, n stop bits (leaving Bits 13:12 STOP in CR2 as zeros gives 1 stop bit).
  - Bit 15 OVER8: Oversampling mode: OVER8= 0: gives oversampling by 16.
  - Bit 10 PCE: Parity control enable: PCE= 0: disables parity control.

These various interrupts are used in conjunction with control flags to control and sync the data flow between the DMA and the main function, in both the sending and receiving direction, as in the example shown in code listing 4.6.

Listing 4.6: USART2 interrupt handling for receiving data

```
93 void USART2_IRQHandler( void ){
94 //      GPIOE->BSRR = GPIO_BSRR_BS_12;
95         if( USART2->ISR & ( USART_ISR_FE | USART_ISR_NE | USART_ISR_PE ) ){
96                 USART2->ICR = USART_ICR_NCF | USART_ICR_FECF | USART_ICR_PECF;
97 //              GPIOE->BSRR = GPIO_BSRR_BS_15;
98         }
99         //Making sure that some data is received and saving the data length
```

```
100         else {
101                 USART2->ICR = USART_ICR_IDLECF;
102                 rx_index[1] = rx_index[0];
103                 rx_index[0] = USART2_RX_BUFF_LEN - DMA1_Channel6->CNDTR;
104                 if( rx_index[0] != rx_index[1] ) rx_flag = SET;
105         }
106 //      GPIOE->BSRR = GPIO_BSRR_BR_12;
107 }
```

After checking if any transmission error is detected, error interrupts are cleared (some error handling should have been implemented), on line 101 the idle interrupt is clear, and ***rx_flag*** is set, this flag is to be used in the main function. The complete code for USART2 is included in appendix D.6.

4.1.5 Timer

As mentioned previously, STM32F3DISCOVERY has several timer modules with different features and capabilities; two of these timers are used in this work, TIM15, as a trigger for ADC4 conversation and TIM2 as PWM generator to control the MOSFET driver ICs. Some code snippets from the respective code files are highlighted in the next paragraphs. The complete code for TIM2 is included in appendix D.4.

4.1.5.1 TIM15 as a trigger

TIM15 configuration is quite simple, as shown in code listing 4.7. The timer clock needs to be enabled (line 13), the timer needs to be enabled (line 15), configured in master mode (line 16), setting the clock prescaler and the auto-reload register (line 17 and 18). These settings will result in the timer producing a trigger signal a the frequency 262144Hz, which will start ADC4 conversation.

Listing 4.7: TIM15 configuration as a trigger

```
/*
 * tim15.c
 *
 *  Created on: May 6, 2021
 *      Author: alihj
 */

#include "tim15.h"

// Setting up the timer as a trigger to ADC's
void tim15_init( void ){
        /* Clock enable */
        RCC->APB2ENR |= RCC_APB2ENR_TIM15EN;

        TIM15->CR1 = TIM_CR1_CEN;
        TIM15->CR2 = TIM_CR2_MMS_1;
        TIM15->PSC = 15;
        TIM15->ARR = 7; //fpwm = fclk/div/(psc+1)/(arr+1)=32M/1/(15+1)/(7+1)=262k144
}
```

4.1.5.2 TIM2 as a PWM generator

TIM2 module was chosen as the PWM generating PWM because it is the only timer module on the STM32F3DISCOVERY, that has a 32bit counter register; this was needed when DMA was intended to be used for changing the duty cycle of the PWM signals, the author didn't manage to make it work, the reasons for that will be explained later in the software testing section 5.1.2, the method that ended up working was the update interrupt method, without the need for the 32bit counter register, TIM1 and TIM8 would have been better candidates to generate the PWM signal, because their superior features, most notably the much higher clock speed up to 144kHz compared to up 72 kHz for TIM2 module, which would have allowed for more accurate duty cycle control, other feature include dead-time generation and emergency stop. Due to lack of time, TIM2 was used further.

The control signals needed for the MOSFETs are shown in figure 4.1.2 and the switching status in table 4.1.1.

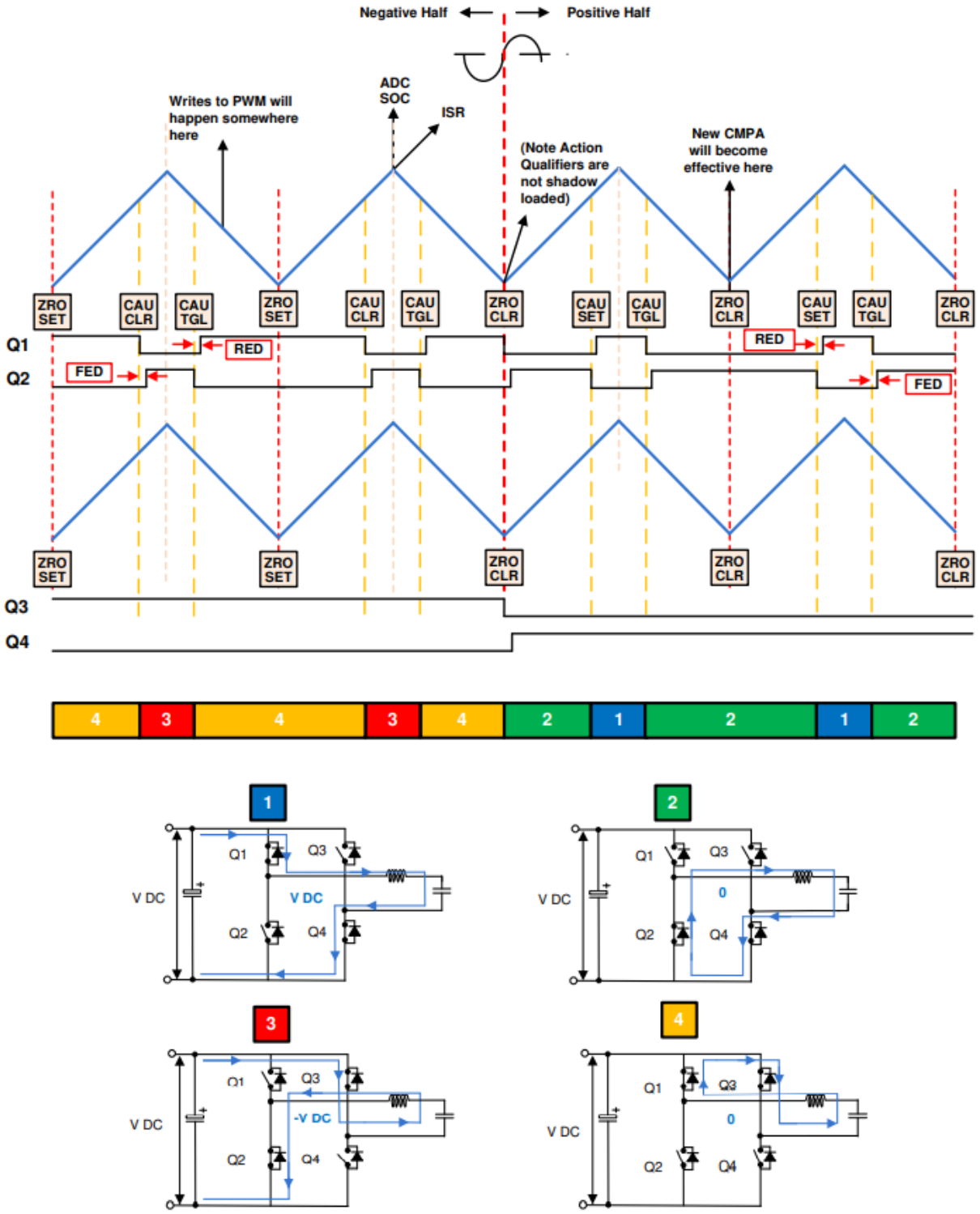


Figure 4.1.2: PWM control signal [14].

Parameter	Q1	Q2 (Q3 in this project)	Q3 (Q2 in this project)	Q4	Voltage at bridge output	State
Positive half cycle	On	Off	Off	On	V_{DC}	1
	Off	On	Off	On	0	2
Negative half cycle	Off	On	On	Off	$-V_{DC}$	3
	On	Off	On	Off	0	4

Table 4.1.1: MOSFETs switching status [14].

This shows the need for four distinct PWM signals, the configuration for TIM2 to produce these signals is shown in code listing 4.8, and the actual duty cycle for each PWM is discussed later in the software testing section 5.1.2.

Listing 4.8: TIM2 configuration as a PWM generator

```
60          // Setting up TIM2 channels counting modes
61          // OC1M 0110: PWM mode 1 - In up counting, channel 1 is active
62          // as long as TIMx_CNT<TIMx_CCR1 else inactive.
63          TIM2->CCMR1 = TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1 | \
64                        TIM_CCMR1_OC2M_2 | TIM_CCMR1_OC2M_1;
65          TIM2->CCMR2 = TIM_CCMR2_OC4M_2 | TIM_CCMR2_OC4M_1 | \
66                        TIM_CCMR2_OC3M_2 | TIM_CCMR2_OC3M_1;
67          // Enabling output ch1 & ch2 & ch3 & ch4
68          TIM2->CCER = TIM_CCER_CC1E | TIM_CCER_CC2E | TIM_CCER_CC3E | TIM_CCER_CC4E;
69          TIM2->PSC = 63; // 1 MHz
70          TIM2->CCR1 = 0;
71          TIM2->CCR2 = 0;
72          TIM2->CCR3 = 0;
73          TIM2->CCR4 = 0;
74          TIM2->ARR = 100 - 1; // 10kHz
75
76          TIM2->DIER = TIM_DIER_UIE; //Enabling update event interrupt
77
78          NVIC_EnableIRQ( TIM2_IRQn );
79          TIM2->CR1 |= TIM_CR1_CEN;
80  }
```

. The following registers needs to be configured:

- TIM capture/compare mode registers CCMR1 and CCMR2 (this controls counting modes): setting it to 0110 for all the channels, which means PWM mode 1 - up counting, this means that the channel x is active as long as TIM2_CNT < TIM2_CCRx else inactive.
- TIM prescaler PSC and TIM auto-reload register ARR: to get a frequency of 10kHz.
- TIM capture/compare registers CCR1, CCR2, CCR3 and CCR4: to 0 (no positive signal will be sent), this will be changed to variable duty cycle values later.
- TIM DMA/interrupt enable register DIER: this way the timer will raise an interrupt every time the counter register is reset, which happens when the counter reaches the auto reload register value.
- TIM capture/compare enable register CCER: setting it to 1111, which enables all TIM2 channels (1,2,3 and 4).
- TIM control register 1 CR1: to enable TIM2.

4.1.6 Main

The main code file contains some IDE automatically generated code that will not be discussed here; the rest of the code will be introduced and discussed in this section. The complete code for Main is included in appendix D.1. Code listing 4.9 shows the include instructions for the definition files for each module used, debugging pins and their associations, and interface pins with their respective input or output function.

Listing 4.9: Microcontroller pins allocation

```
24 /* USER CODE BEGIN Includes */
25 #include "usart2.h"
26 #include "main.h"
```

```
27 #include "adc.h"
28 #include "tim2.h"
29 #include "tim15.h"
30 #include "gpio.h"
31 /*
32  * Debug pins
33  * E8: main while loop
34  * E9: functional communication
35  * E10: ADC4 interrupt
36  * E11: DMA2 channel2 interrupt
37  *
38  * E13: TIM2 interrupt
39  * E14: usart2 rx interrupt
40  * E15: usart2 tx interrupt
41  *
42  * Interface
43  * D8: usart2 tx
44  * D9: usart2 rx
45  *
46  * PD8: ADC4 CH12 input
47  * PD9: ADC4 CH13 input
48  * PD10: ADC4 CH7 input
49  * PD11: ADC4 CH8 input
50  * PB12: ADC4 CH3 input
51  * PB14: ADC4 CH4 input
52  *
53  * PD1: TIM2CH1 PWM Q1 output
54  * PD4: TIM2CH2 PWM Q3 output
55  * PD7: TIM2CH3 PWM Q2 output
56  * PD6: TIM2CH4 PWM Q4 output
57  */
```

Code listing 4.10 shows the variables and flags used in the main function, where the tmp_high equals 1270, which corresponds to $90°C$ as the output from the temperature sensor for $90°C$ is approximately 900 mV, this will be converted to roughly 1270 with a 12bit ADC that have 2.9 V reference voltage, tmp_low equals 11130, which corresponds to $80 \circ C$.

Listing 4.10: Main variables and flags definition

```
69 /* USER CODE BEGIN PV */
```

```
70 FlagStatus rx_flag, tx_flag_1, running;
71 uint16_t adc4_val[ADC4_BUFF_LEN];
72 uint16_t tmp_high=1270, tmp_low=1130; // Roughly 90 C, 80 C respectively
73 /* USER CODE END PV */
```

Code listing 4.11 shows the process of initializing all the used modules by calling their respective init function.

Listing 4.11: Modules initializing functions

```
96   /* USER CODE BEGIN SysInit */
97         // Sensors
98         adc4_init();
99         // ADC4 trigger
100        tim15_init();
101        // PC-Communication
102        usart2_init();
103        // In/out for troubleshooting
104        gpio_init();
105        //PWM generation
106        tim2_init();
107  /* USER CODE END SysInit */
```

Code listing 4.12 shows the main while loop that have been previously shown as a flow chart in figure 4.1.1.

Listing 4.12: Main while loop

```
112  running= SET; // Should be set and reset with a command from the Python program
113  while (1)
114  {
115        /* USER CODE BEGIN WHILE */
116          GPIOE->BSRR = GPIO_BSRR_BS_8;
117          if (running) {
118                  if (tx_flag_1){
119                          tx_flag_1= RESET;
120                          // Send data to PC
121  uint8_t s[16] = {'H', adc4_val[0] >> 8 , adc4_val[0], adc4_val[1] >> 8 , adc4_val[1], \
122  'I', adc4_val[2] >> 8 , adc4_val[2], adc4_val[3] >> 8 , adc4_val[3], \
123  'V', adc4_val[4] >> 8 , adc4_val[4], adc4_val[5] >> 8 , adc4_val[5], '\0'};
124  usart2_tx( s, 16 );
```

```
125     // temperature check
126     if ((adc4_val[0]>tmp_high) | (adc4_val[1]>tmp_high)) {
127             TIM2->CR1 = 0; // Disabling TIM2 to stop the PWM generation
128     }
129     else {
130             if ((!TIM2->CR1) & (adc4_val[0]<tmp_low) & (adc4_val[0]<tmp_low)) {
131                     TIM2->CR1 |= TIM_CR1_CEN; // Enabling TIM2 to start the PWM generation
132             }
133     }
134                     }
135             }
136             GPIOE->BSRR = GPIO_BSRR_BS_8;
137           /* USER CODE END WHILE */
138     }
139 }
```

When the ***tx_flag_1*** gets raised by ***DMA2_Channel2_IRQHandler***, the main while loop performs these two tasks (after resetting ***tx_flag_1***):

1. Take the data from the DMA buffer and arrange it in a vector, then send it to the Python program, using the ***usart2_tx*** function.
2. Check the reading from both of the temperature sensors, disabling the PWM generation if one of the temperatures exceeds the preset value ***tmp_high***, and enabling the the PWM when the temperatures fall below the preset value ***tmp_low***.

4.2 Python program

The planed python monitoring and control program was intended to be able to:

1. Start and stop the control signal from the microcontroller, sending commands on the serial communication port connected to the microcontroller USART2, where the microcontroller enable/disable TIM2, which is responsible for PWM generation.

2. set the switching speed of the MOSFETs, by sending the switching frequency to the microcontroller, where the microcontroller manipulates the auto reload register (ARR) for TIM2, thus changing the period of the switching signal.

3. Receive the raw data from the microcontroller, log it to a file, plot it in real time, do some calculations on the data such as FFT (fast Fourier transform) to analyse the frequency contents of the voltage signal, and get the important inverter performance metric THD (total harmonics disruption), some other calculations to find the efficiency of the inverter by comparing output power to input power.

Figure 4.2.1 shows a rudimentary version of the program interface, the program is at a very early stage of development and is not capable of performing the tasks mentioned above, therefore its code was not included in this work.

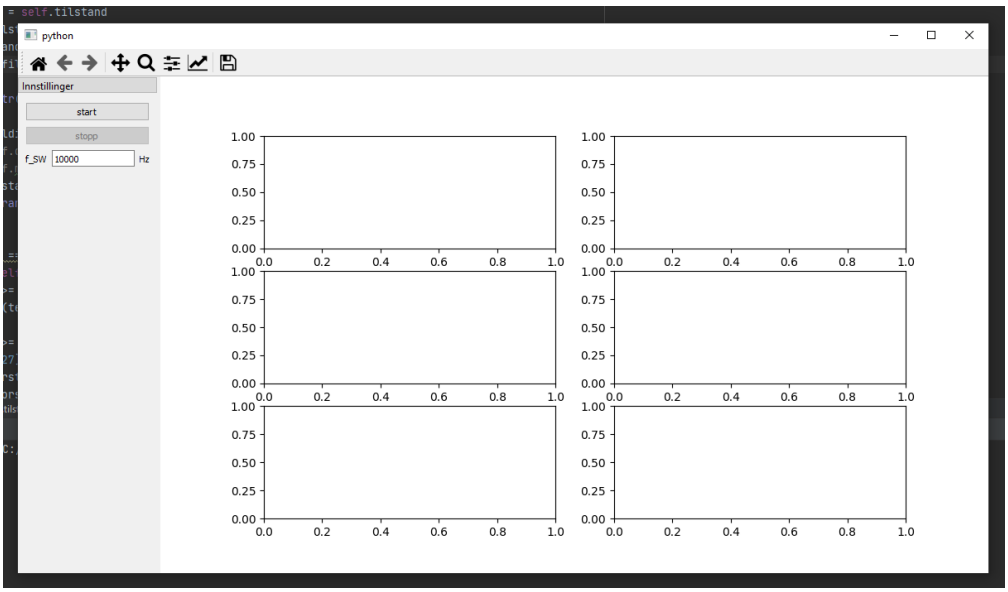


Figure 4.2.1: Python program interface.

Chapter 5

Results and Discussion

This chapter summarizes the obtained hardware and software results, describes the performed tests, reports the results from them, and discusses their adherence to the theoretical expectations.

Figure 5.0.1 shows the final result of this project, the inverter circuit connected to the microcontroller and to a 33 Ω 100 W load.

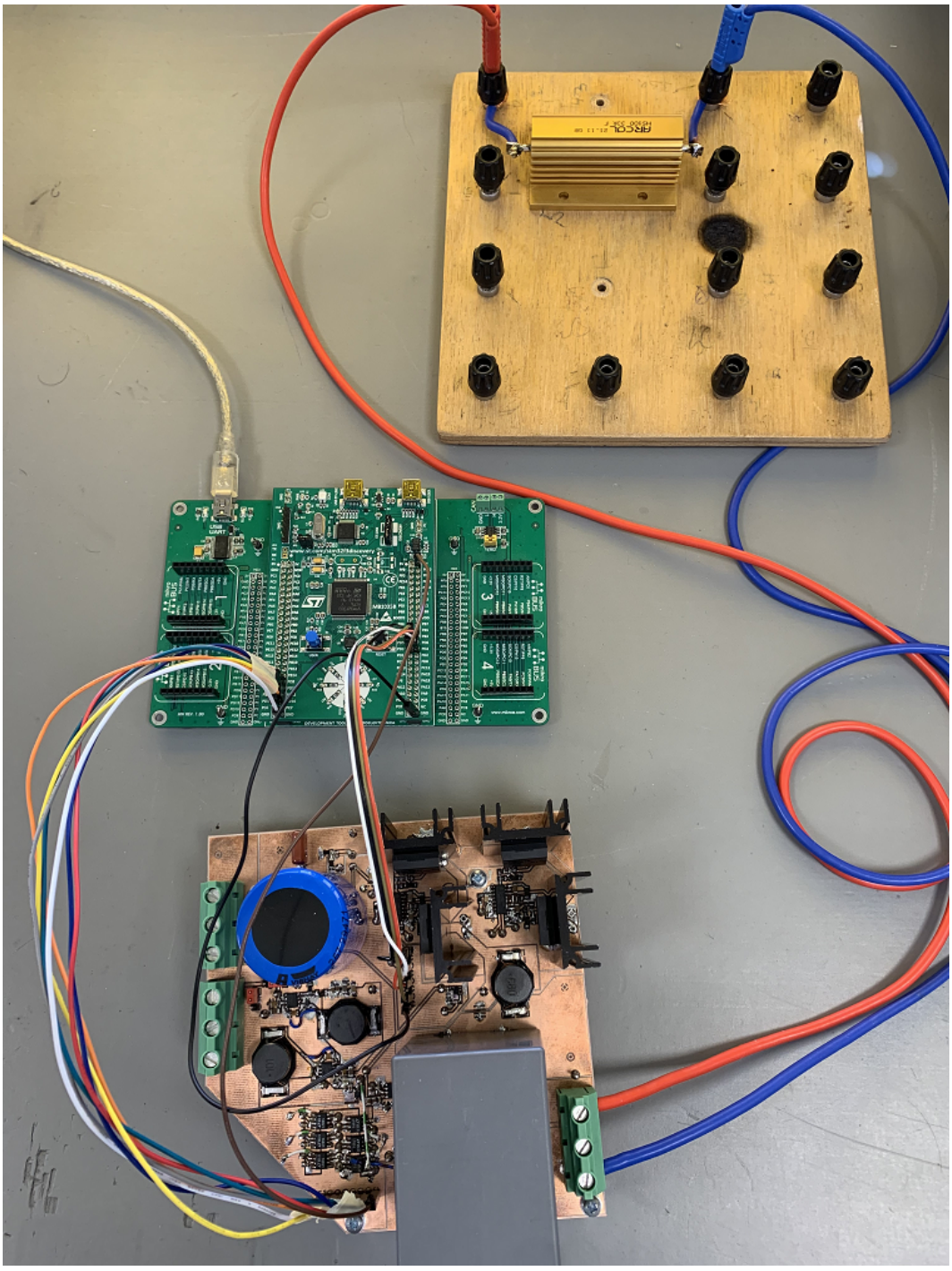


Figure 5.0.1: Final project result.

5.1 Software

In this section, most of the elements of the microcontroller code will be tested, TIM15, ADC4, USART2, and parts of the main loop will be tested together by evaluating the sent data from the microcontroller. TIM2 will be tested to make sure the correct PWM control signals are generated. Unfortunately, performance testing was not performed, and the tests are not optimally documented due to lack of time.

5.1.1 Communication and ADC testing

The test was done by setting dummy known voltages on ADC4 input pins and by connecting an oscilloscope probe to the tx pin of USART2, then setting up the oscilloscope in serial mode S1:**UART/RS232** and configuring the bus as it was set up in the USART module in the microcontroller 4.1.4, number of bits: 8, parity: none, baud rate: 2Mb, polarity: idle high and bit order: least significant bit, as shown in figure 5.1.1.

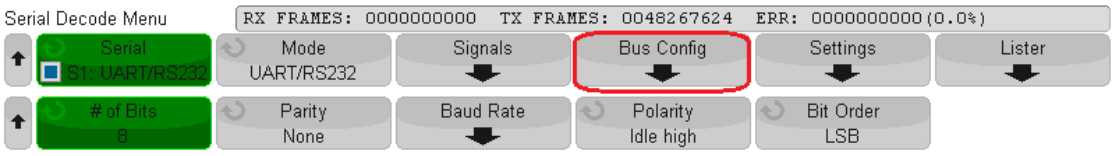


Figure 5.1.1: Oscilloscope serial mode configurations.

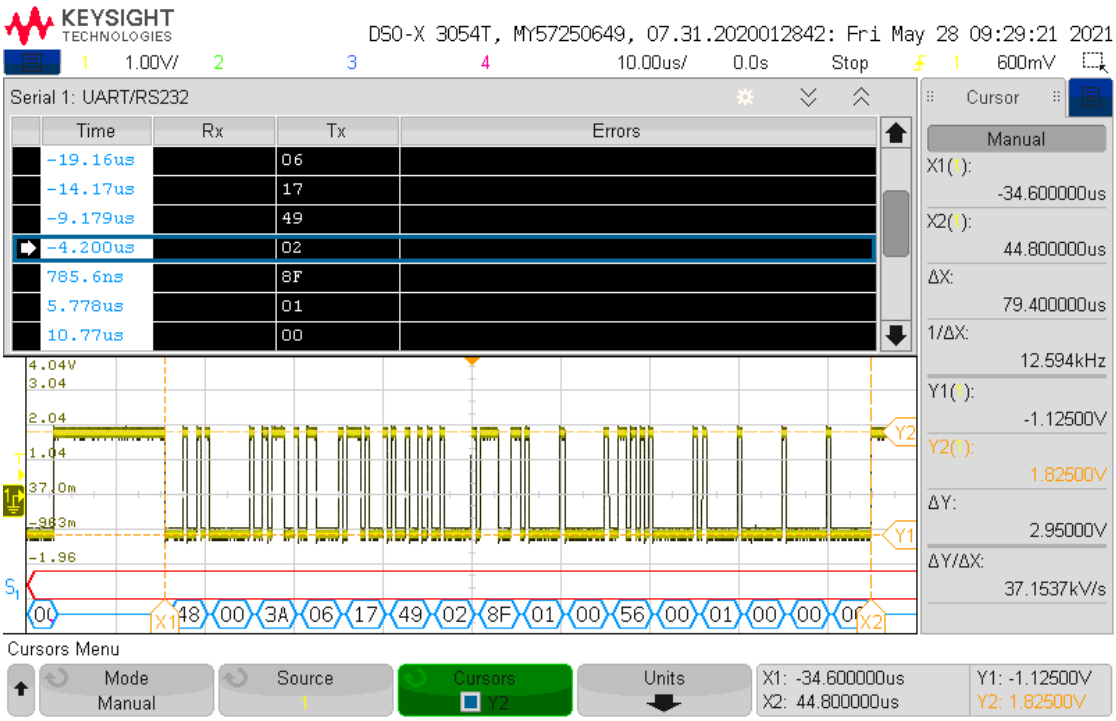


Figure 5.1.2: Serial communication validation.

Figure 5.1.2 shows that sixteen bytes have been successfully transmitted from the microcontroller USART module; also, the values shown by the oscilloscope corresponded to the voltages on ADC4 input pins; this confirms that both TIM15, ADC4, interrupts, and flags are working correctly.

The figure also shows the total time taken to transmit all the data to check if the baud rate is accurate; this simple calculation is done 5.1:

$$baudrate = \frac{N_s \times bps}{\Delta x} \tag{5.1}$$

Where Δx is the time needed to send all the symbols (read from the oscilloscope $79.4\mu s$), N_s is the number of symbols sent (16 symbols as shown on line 124 in code listing 4.12 and also can be counted in the figure), and bps is the number of bits per symbol (from USART configuration 4.1.4 1 start, 8 data, 1 stop = 10 bits). Then we can calculate the baud rate to be $baudrate = \frac{16\times10}{79.4\times10^{-6}} \approx 2Mb$, which further confirms the correct functioning of the

USART module.

It was mentioned previously 4.1.4, that an explanation will be given as to why 2Mb baud rate is sufficient for this project. The reason can be read from figure 5.1.2 where the frequency of the data sent from the microcontroller as a whole is approximately 12.6 kHz; this is called the sampling frequency, which according to the Nyquist theory, will allow the maximum frequency of 6.3 kHz without aliasing, connecting this to the harmonics of the fundamental frequency in the inverter, this will allow for the measurement of up to the 126th harmonic, which is more than enough.

5.1.2 PWM generation testing

As mentioned previously 4.1.5.2, it was difficult to use the DMA controller to produce a variable duty cycle PWM due to the fact that four DMA channels were needed, one for each capture/compare register to be changed; the problems stem from the fact that the DMA channels operations are executed sequentially after each other according to priority, this resulted in different value in each capture/compare register and therefore out of syncing PWM signals where produced.

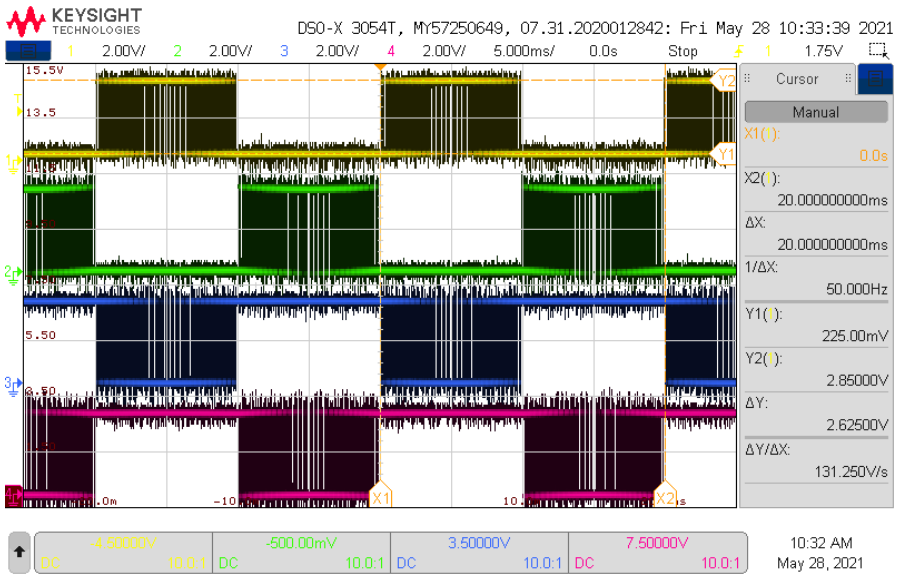
Using the interrupt method gave much better results, as changing all the capture/compare register was done in the same interrupt handler function shown in code listing 5.1. The duty cycle values for the sine wave were calculated in an external program, then arranged in vectors at the correct indexes for each PWM signal, determining which control signal should be high and at what time according to table 4.1.1, this code is shown in appendix D.4 on lines 11-46.

Listing 5.1: Interrupt usage in TIM2 module to produce variable ducy cycle PWM

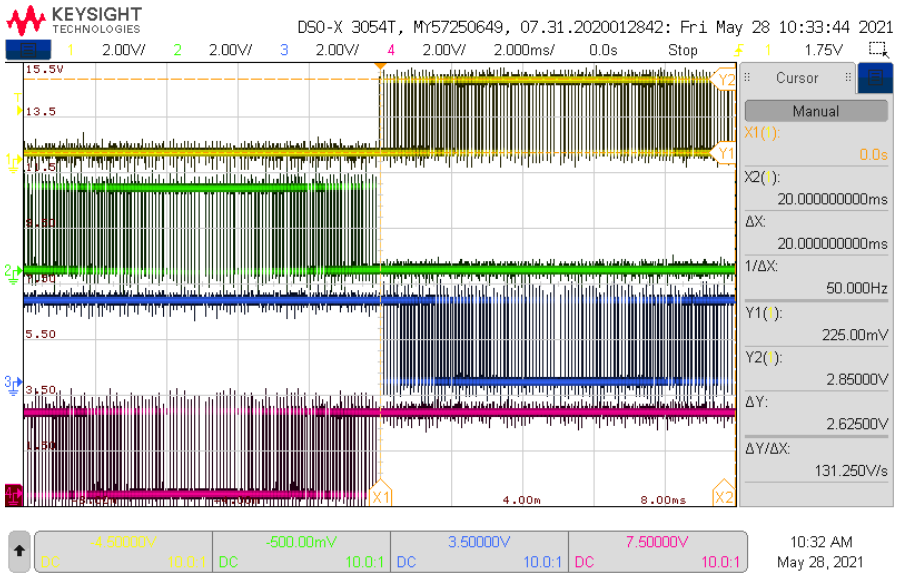
```
82 void TIM2_IRQHandler(void)
83 {
84   static uint16_t index = 0;
85
86   if (TIM2->SR & TIM_SR_UIF)
87   {
88         GPIOE->BSRR = GPIO_BSRR_BR_13;
```

```
89      TIM2->SR=0;
90      TIM2->CCR1 = PWM_BufferQ1[index];
91      TIM2->CCR2 = PWM_BufferQ3[index];
92      TIM2->CCR3 = PWM_BufferQ2[index];
93      TIM2->CCR4 = PWM_BufferQ4[index];
94      index = (index + 1) % (PWM_ELEMENTS);
95      GPIOE->BSRR = GPIO_BSRR_BR_13;
96    }
97  }
```

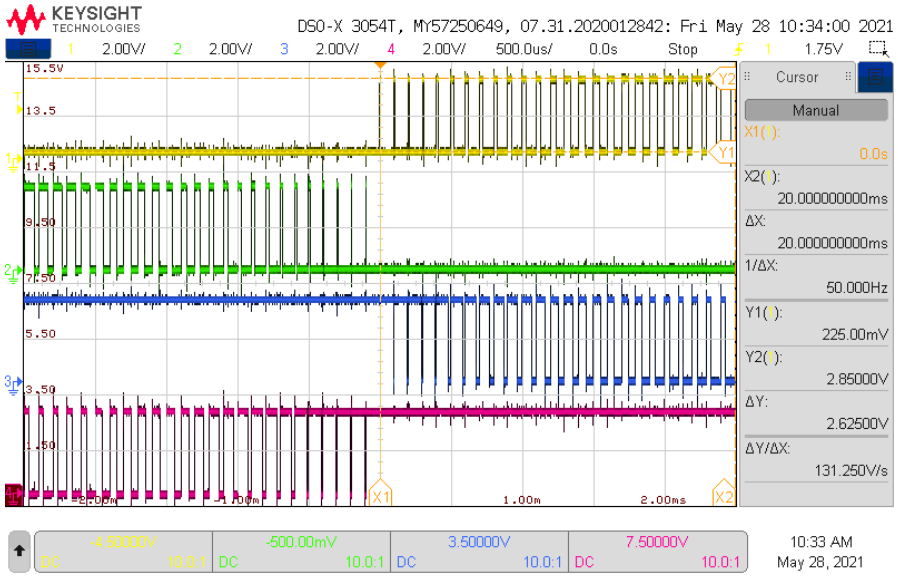
The output from TIM2 output channels is shown at different levels of zooming in figure 5.1.3, where Q1, Q2, Q3, and Q4 control signals are in yellow, green, blue, and magenta, respectively. When looking at the measurements and waveforms in the figure, it can be concluded that TIM2 module is functioning according to specifications in the code file.



(a)



(b)



(c)

Figure 5.1.3: MOSFETs control signals produced by TIM2 module

One final test was done to make sure that that variable duty cycle PWM signal will result in a sinus signal when applied to a suitable filter as the one shown in figure 5.1.4, which is a simple RC low pass filter. Applying the PWM signal on the filter resulted in the oscilloscope picture shown in figure 5.1.5, this confirms that the PWM signal is working like expected, where it can be seen that the two pairs of control signals (Q1 and Q3, Q2 and Q4) are turning on/off in alternating fashion, the pairs (Q1 and Q2), (Q3 and Q4) are never On at the same time, thus not risking a short circuit event.

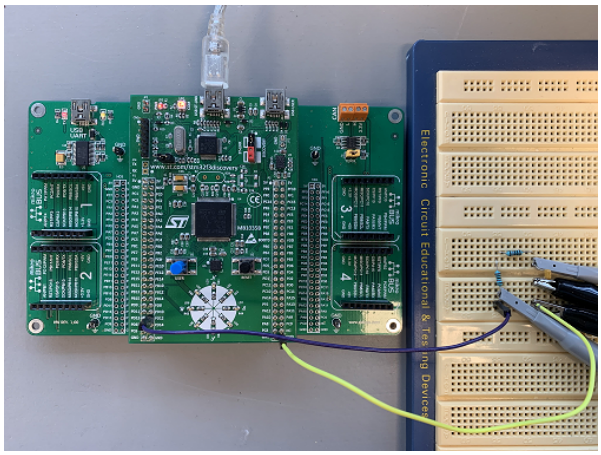


Figure 5.1.4: Microcontroller PWM output connected to RC filter.

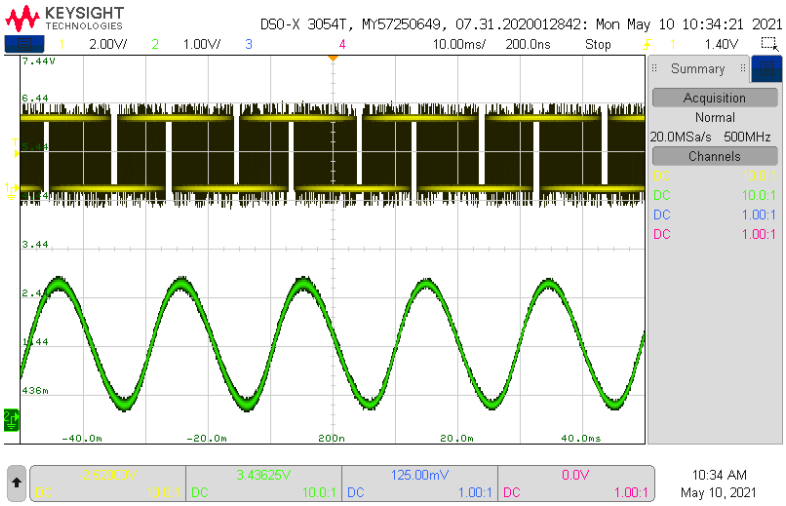


Figure 5.1.5: Microcontroller PWM signal in yellow and filtered signal in green.

5.2 Hardware

5.2.1 Inverter circuit testing

As explained throughout chapter 3, testing was done during the mounting process; whenever measurements differed from the expected values, the issue was investigated and rectified. As a part of a final check, the following tests were preformed on the finished PCB:

1. Supply voltages values. ***Passed***.
2. Ground test points values. ***Passed***.
3. Output from all the sensors against expected values. ***Passed***.
4. Continuity between the jumper pins and corresponding test points. ***Passed***.
5. Discontinuity between ground and all none ground test points. ***Passed***.
6. Driver ICs outputs when a square wave is applied to the control test points. ***Passed***.

These tests should have been documented in a better way, explaining the test methodology and providing the exact values obtained, but that was not possible due to the lack of time.

5.2.2 Inverter output

The operating of the MOSFETs is tested after getting the proper control signals from the microcontroller, setting up the sampling frequency to 10kHz, the input voltage to 20 V, which should produce 0.428 A in the load; these values were chosen to minimize the risk of damaging any components, especially the MOSFETs and their drivers, in case the control signals where incorrect. The oscilloscope channel 1 was connected to the AC+ test point (TP12), and channel 2 was connected to the AC- test point (TP13). This test was done before mounting

the DC-link capacitor.

Figure 5.2.1 shows the inverter output waveform, this waveform has the shape of sinus and the frequency of 50Hz with a $V_{rms} \approx 9V$, although being very noisy and at a lower rms value than the expected 14.4 V, but with voltage spikes reaching 20/-20 V as expected, and in general shows the expected waveform of a filtered PWM generated sinus. This result could have happened for any number of reasons such as: incorrect control signals, driver ICs malfunction, the output LC filter not functioning properly, MOSFETs not turning on completely, and the most obvious reason stated at the beginning of this test, the absence of the DC-link capacitor, especially that the power supply voltage and current values were very erratic, ranging between 13-18 V for the voltage and between 0.32-0.75 A for the current.

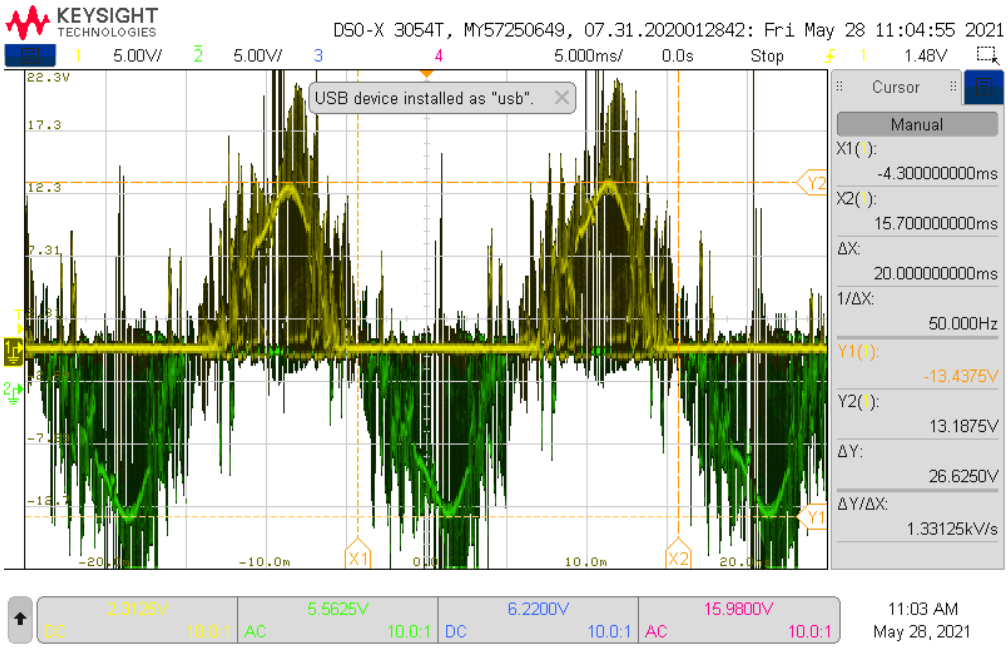


Figure 5.2.1: Inverter output wave form.

Unfortunately, after mounting the DC-link capacitor, the input supply was connecting incorrectly, which damaged the logic power supply ICs and possibly other ICs. There was not enough time to investigate and fix the damage. Sadly, no further testing was done possible. Thus, eliminating the possibility of improving the inverter output waveform or testing the

accuracy of the measurements from the sensing circuitry.

5.3 Discussion

The inverter system seems to perform most of the tasks it was designed for; due to the limited amount of testing, it is not possible to conclude the degree of success. Regarding hardware design, the produced circuit had some design mistakes, both in the schematics, layout, and acquirement, as discussed in hardware design chapter 3, other mistakes were made during production and soldering, the aforementioned mistakes were successfully corrected as they were discovered, it is possible that the hardware design has other mistakes that have not been discovered due to lack of functional testing, but it can be concluded that the basic design principles are sound.
Regarding software design, as the tests show, the implemented functions in the microcontroller code operate as intended; some originally planned functions were not fully developed, such as safety measures, receiving commands from the Python program, and changing the switching frequency on the fly. The program required a good amount of troubleshooting and fine-tuning in order to operate in a satisfactory manner. Implementing the rest of the functions should be a straightforward process without too many hiccups. Little work was done on the Python program side.
A combination of mistakes and challenges lead to none optimal results, these are listed here:

- The main issue was the author's over-ambition, and confidence, the goals for this project were unrealistic based on the knowledge base and experience the author had; this resulted in a lot of time being used to acquire theoretical knowledge, leaving little time to do the practical part of the project.
- Most of the work has been done independently, and it takes a lot for the author to ask for guidance, in combination with minimal involvement of the supervisor; this resulted in further time constraints.

- The several hardware issues that plagued this project can be attributed in part to the fact that the author was working alone, limiting the possibility of someone else checking the work, in addition to some factors related to the tools and equipment being sub optimal, making the production and soldering process even more difficult.
- Poor prioritizing and getting sidetracked to work on the sensing code instead of focusing on getting the power stage working first.
- The classic mistake of doing tests during development and not documenting them was made.

5.3.1 Further development

The following areas require more work:

- Develop and finish the microcontroller code.
- Troubleshoot and get the circuit to an acceptable working condition.
- Use the circuit to do the Si and SiC comparison.

Chapter 6

Conclusion

The author has learned a lot about power electronics and all the details concerning their operation, obtained valuable experience in circuit design and production, and learned a lot about microcontroller programming, limitations, and troubleshooting.

A somewhat working MOSFET based DC to AC inverter prototype has been designed and built, a microcontroller code has been developed to control and monitor the circuit. The system passed a number of preliminary tests and produced a promising result at the end, which falls slightly short of the planned main objective and far away from the secondary objective. This result shows the system's capability of producing the desired alternating current output from the direct current input. Although, this output has a long way before being satisfactory to be used in comparing the Si and SiC MOSFETs. The SiC MOSFETs were not mounted to the circuit, but the design should be capable of using them as desired.

The theory chapter and the comparison done in the hardware design chapter between the Si and SiC MOSFETs shows the potential benefits from using the SiC variant in the inverter design; such benefits include lower power losses and higher switching frequency capabilities; thus, improvements in efficiency and quality of the output, reduction in size and number of passive components, on the other hand, the SiC variant is more expensive. These potential benefits were not confirmed in practice in this project.

References

[1] Heat sink calculator. https://www.heatsinkcalculator.com/heat-sink-thermal-resistance-calculator.html.

[2] Infineon Technologies AG. An 1805 pl52 1806 095202 using the eicedriver 2edi product family of dual-channel functional and reinforced isolated mosfet gate drivers. https://www.infineon.com/dgdl/Infineon-GateDriverICs_EiceDRIVER_2EDi_Using_the_EiceDRIVER_2EDi_family-AN-v01_00-EN.pdf?fileId=5546d46267354aa001675a431da84a41, May 2015. Retrived: 13. May 2021.

[3] Infineon Technologies AG. Application note an2015-06 eicedriver gate resistor for power devices. https://www.infineon.com/dgdl/Infineon-EiceDRIVER-Gate_resistor_for_power_devices-ApplicationNotes-v01_00-EN.pdf?fileId=5546d462518ffd8501523ee694b74f18, Desember 2015. Retrived: 13. May 2021.

[4] Infineon Technologies AG. Imw65r072m1h 650 v coolsic m1 sic trench power device. https://www.infineon.com/dgdl/Infineon-IPW60R090CFD7-DS-v02_00-EN.pdf?fileId=5546d46261ff57770162002f34a82a8f, February 2015. Retrived: 13. May 2021.

[5] Infineon Technologies AG. Imw65r072m1h 650 v coolsic m1 sic trench power device. https://www.infineon.com/dgdl/Infineon-IMW65R083M1H-DataSheet-v02_

00-EN.pdf?fileId=5546d4627862c3e501786e1ef0e93c85, October 2019. Retrived: 13. May 2021.

[6] Infineon Technologies AG. Eicedrive 2edi product family. https://www.infineon.com/dgdl/Infineon-2EDS9265H-DataSheet-v02_05-EN.pdf?fileId=5546d462712ef9b70171832eb2cf1f37, Mars 2020. Retrived: 13. May 2021.

[7] Linear Technology Corporation. Lt 0510 rev c miropower, single supply rail-to-rail output instrumentation amplifiers. https://www.analog.com/media/en/technical-documentation/data-sheets/1789fc.pdf, 2002. Retrived: 13. May 2021.

[8] Analog Devices. Low voltage temperature sensors tmp35/tmp36/tmp37. https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf, May 2015. Retrived: 13. May 2021.

[9] FISCHER ELEKTRONIK. Extruded heatsinks for pcb mounting sk 145/37,5 sts-220. http://www.farnell.com/datasheets/1734934.pdf. Retrived: 13. May 2021.

[10] Suxuan Guo. How to drive sic mosfet.... the right way !! https://training.ti.com/how-drive-sic-mosfet-right-way?cu=1134585, October 2017. Retrived: 13. May 2021.

[11] Texas Instruments. Low-power, single-supply, cmos instrumentation amplifiers. https://www.ti.com/lit/ds/sbos216b/sbos216b.pdf?ts=16221458779888&ref_url=https%253A%252F%252Fwww.google.com%252F, October 2006. Retrived: 13. May 2021.

[12] Texas Instruments. Lm22675/-q1 42 v, 1 a simple switcher step-down voltage regulator. https://www.ti.com/lit/ds/symlink/lm22675-q1.pdf?ts=1621804761879&ref_url=https%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252FLM22675-Q1, November 2014. Retrived: 13. May 2021.

[13] Texas Instruments. Ref20xx low-drift, low-power, dual-output, vref and vref / 2 voltage references. https://www.ti.com/lit/ds/symlink/ref2033.pdf?ts=1621803263709&

ref_url=https%253A%252F%252Fwww.google.com%252F, July 2018. Retrived: 13. May 2021.

[14] Texas Instruments. Tidm-hv-1ph-dcac voltage source inverter reference design. https://www.ti.com/lit/ug/tiduay6e/tiduay6e.pdf?ts=1621879586451&ref_url=https%253A%252F%252Fwww.google.com%252F, March 2020. Retrived: 13. May 2021.

[15] George Lakkas. Mosfet power losses and how they affect power-supply efficiency. *Analog Applications Journal*, AAJ(1Q):22–26, 2016.

[16] Allegro MicroSystems. High accuracy, galvanically isolated current sensor ic. https://www.allegromicro.com/~/media/Files/Datasheets/ACS722-Datasheet.ashx, June 2014. Retrived: 13. May 2021.

[17] MikroElektronika. *STM32F3 Discovery Shield Manual*, 1 edition, 2013.

[18] MikroElektronika. *Reference manual STM32F303xB/C/D/E, STM32F303x6/8, STM32F328x8, STM32F358xC, STM32F398xE advanced ARM®-based MCUs*, 8 edition, Januray 2017. Doc ID 022558 Rev 8.

[19] Ehsan Ali Buriro Munwar Ayaz Memon, Ghullam Mustafa Bhutto. Sizing of dc-link capacitor for a grid connected solar photovoltaic inverter. https://www.semanticscholar.org/paper/Sizing-of-dc-link-capacitor-for-a-grid-connected-lowast-Bhutto/dd51b3f5322ce1cd16cc083dc552332539adbddb, 2020. Retrived: 13. May 2021.

[20] Jong-Mun. Park. Evolution of power semiconductor devices. http://www.iue.tuwien.ac.at/phd/park/node14.html, October 2004. Retrived: 13. May 2021.

[21] Ned Mohan; Tore M. Undeland; William P. Robbins. *POWER ELECTRONICS; Converters, Applications, and Design*. John Wiley & Sons, Inc., 3 edition, 2003.

[22] Michael Salcone and Joe Bond. Selecting film bus link capacitors for high performance inverter applications. https://ieeexplore.ieee.org/document/5075431?arnumber=5075431, May 2009. Retrived: 13. May 2021.

[23] STMicroelectronics. *UM1570 User manual*, 3 edition, February 2013. Doc ID 023594 Rev 3.

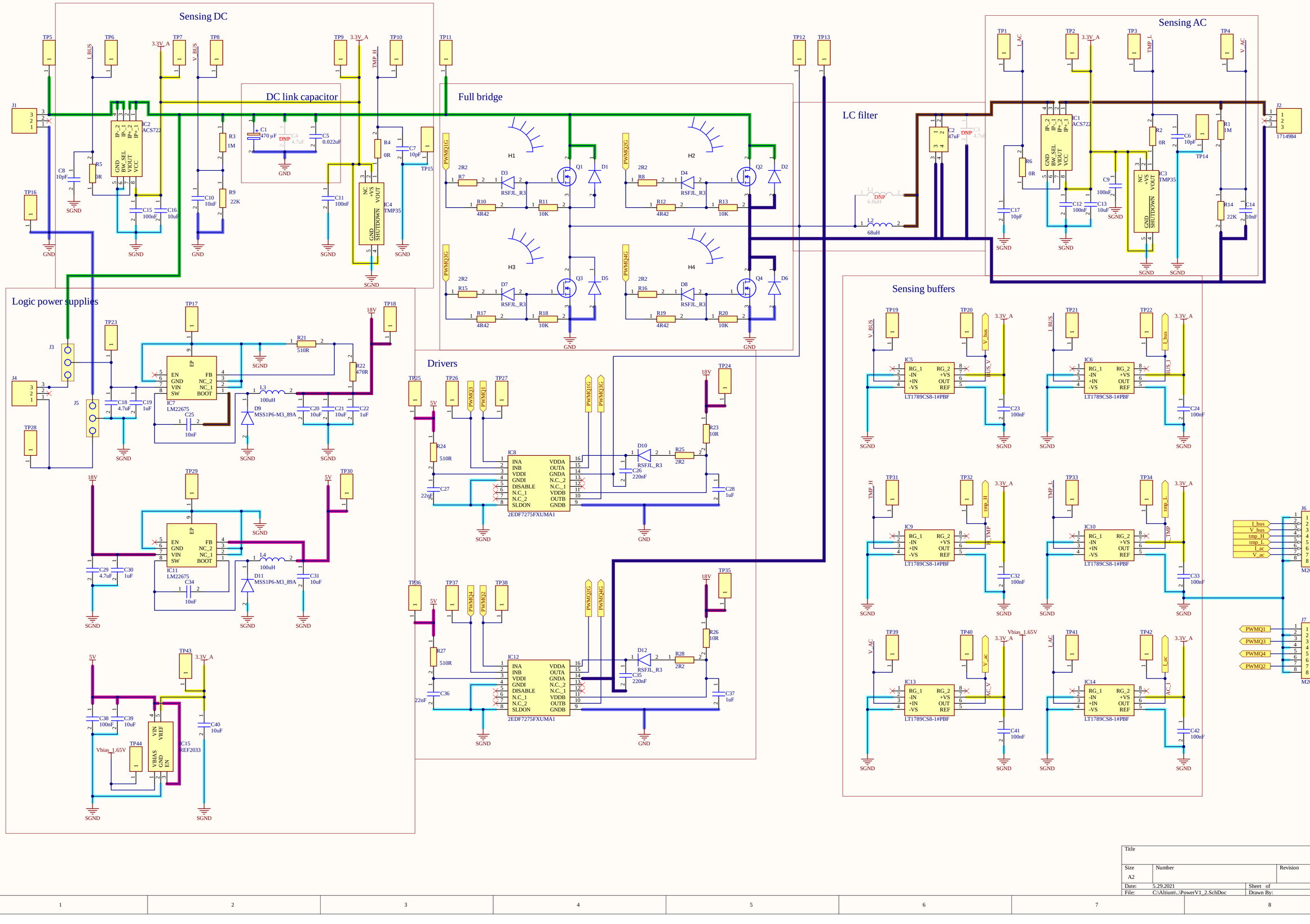
[24] STMicroelectronics. *STM32F303xB STM32F303xC Datasheet*, 8 edition, April 2014. Doc ID 023353 Rev 8.

Appendix A

Schematics and PCB layout

This appendix includes the schematics and PCB layout for the inverter.

Si variant schematics



Title			
Size: A2	Number		Revision
Date: 5.29.2021		Sheet of	
File: C:\Altium\..\PowerV1_2.SchDoc		Drawn By:	

SiC variant schematics

Sensing DC

TP5 TP6 TP7 TP8 TP9 TP10 TP11 TP12 TP13
I_BUS
3.3V_A
V_BUS
TMP_H
J1
IC2 ACS722
R3 1M
R4 0R
C7 10pF
TP15
C1 470 µF DNP
C4 4.7uF
C5 0.022uF
R5 0R
C8 10pF
C10 10nF
R9 22K
C11 100nF
IC4 TMP35
C15 100nF
C16 10uF
TP16
GND SGND

DC link capacitor

Full bridge

PWMQ1G PWMQ2G PWMQ3G PWMQ4G
H1 H2 H3 H4
Q1 Q2 Q3 Q4
D1 D2 D5 D6 DNP
D3 D4 D7 D8 RSFJL_R3
2R2 R7 R8 R15 R16
R10 R12 R17 R19 4R42
R11 R13 R18 R20 10K
GND

LC filter

L1 6.8uH
L2 DNP 68uH
T2 47uF DNP
C3 4.7uF

Sensing AC

TP1 TP2 TP3 TP4
LAC
3.3V_A
TMP_L
V_AC
J2 1714984
IC1 ACS722
R6 0R
R2 0R
C6 10pF
TP14
R1 1M
C17 10pF
C9 100nF
C12 100nF
C13 10uF
IC3 TMP35
R14 22K
C14 10nF
SGND

Logic power supplies

TP17 TP18 TP23
18V
J3 J4 J5
IC7 LM22675
R21 510R
R22 1K27
L3 100uH
D9 MSS1P6-M3_89A
C18 4.7uF
C19 1uF
C20 10uF
C21 10uF
C22 1uF
C25 10nF
TP28
SGND

TP29 TP30
5V
IC11 LM22675
C29 4.7uF
C30 1uF
C34 10nF
L4 100uH
D11 MSS1P6-M3_89A
C31 10uF

TP43
C38 100nF
C39 10uF
C40 10uF
TP44
Vbias_1.65V
IC15 REF2033

Drivers

TP25 TP26 TP27
5V
PWMQ3 PWMQ1
R24 510R
C27 22nF
IC8 2EDF7275FXUMA1
D10 RSFJL_R3
R25 2R2
C26 220nF
R23 10R
C28 1uF

TP36 TP37 TP38
PWMQ4 PWMQ2
R27 510R
C36 22nF
IC12 2EDF7275FXUMA1
D12 RSFJL_R3
R28 2R2
C35 220nF
R26 10R
C37 1uF
TP35

Sensing buffers

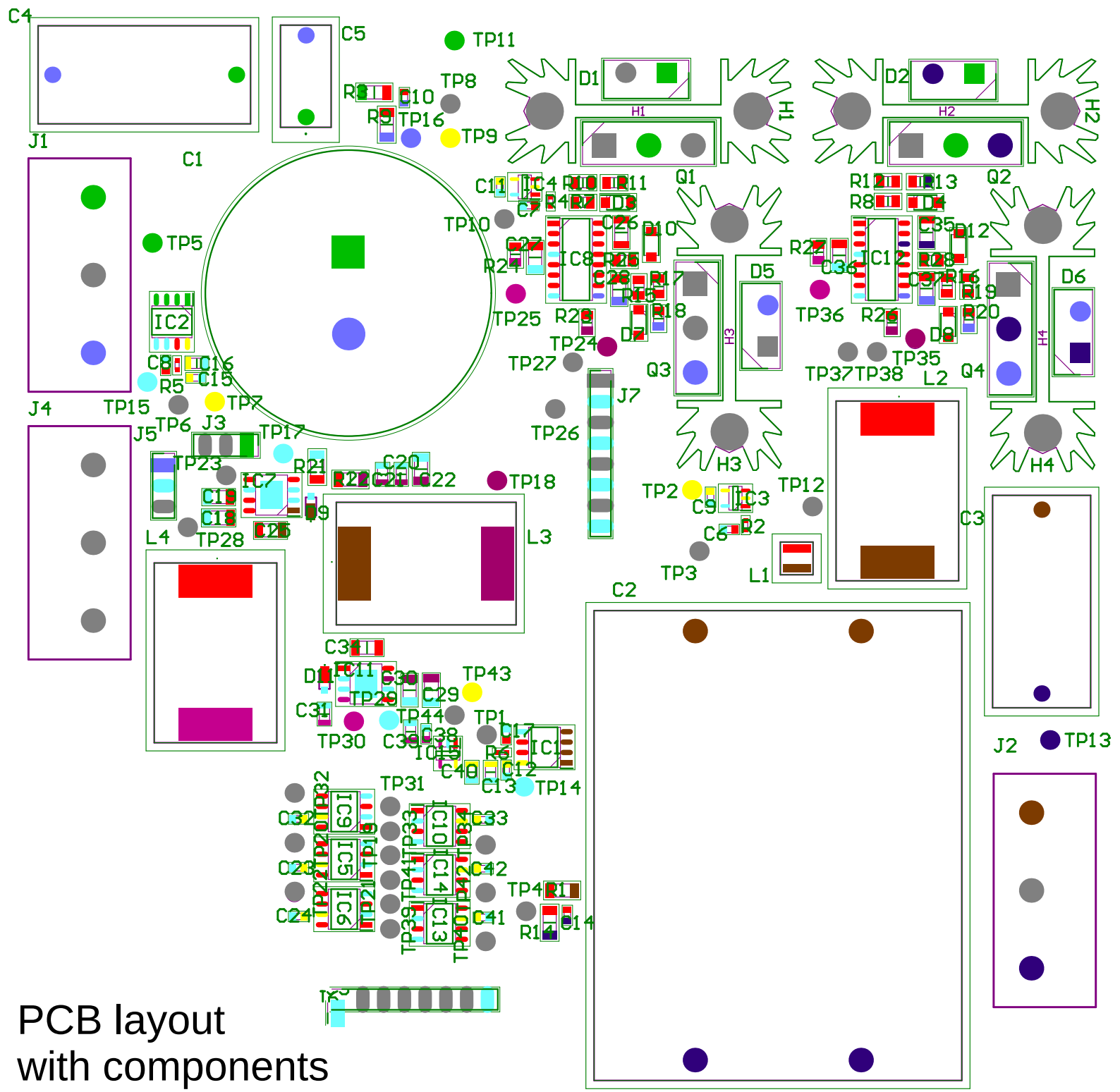
TP19 TP20 TP21 TP22
V_BUS V_bus I_BUS I_bus
IC5 LT1789CS8-1#PBF
IC6 LT1789CS8-1#PBF
C23 100nF
C24 100nF

TP31 TP32 TP33 TP34
TMP_H tmp_H TMP_L tmp_L
IC9 LT1789CS8-1#PBF
IC10 LT1789CS8-1#PBF
C32 100nF
C33 100nF

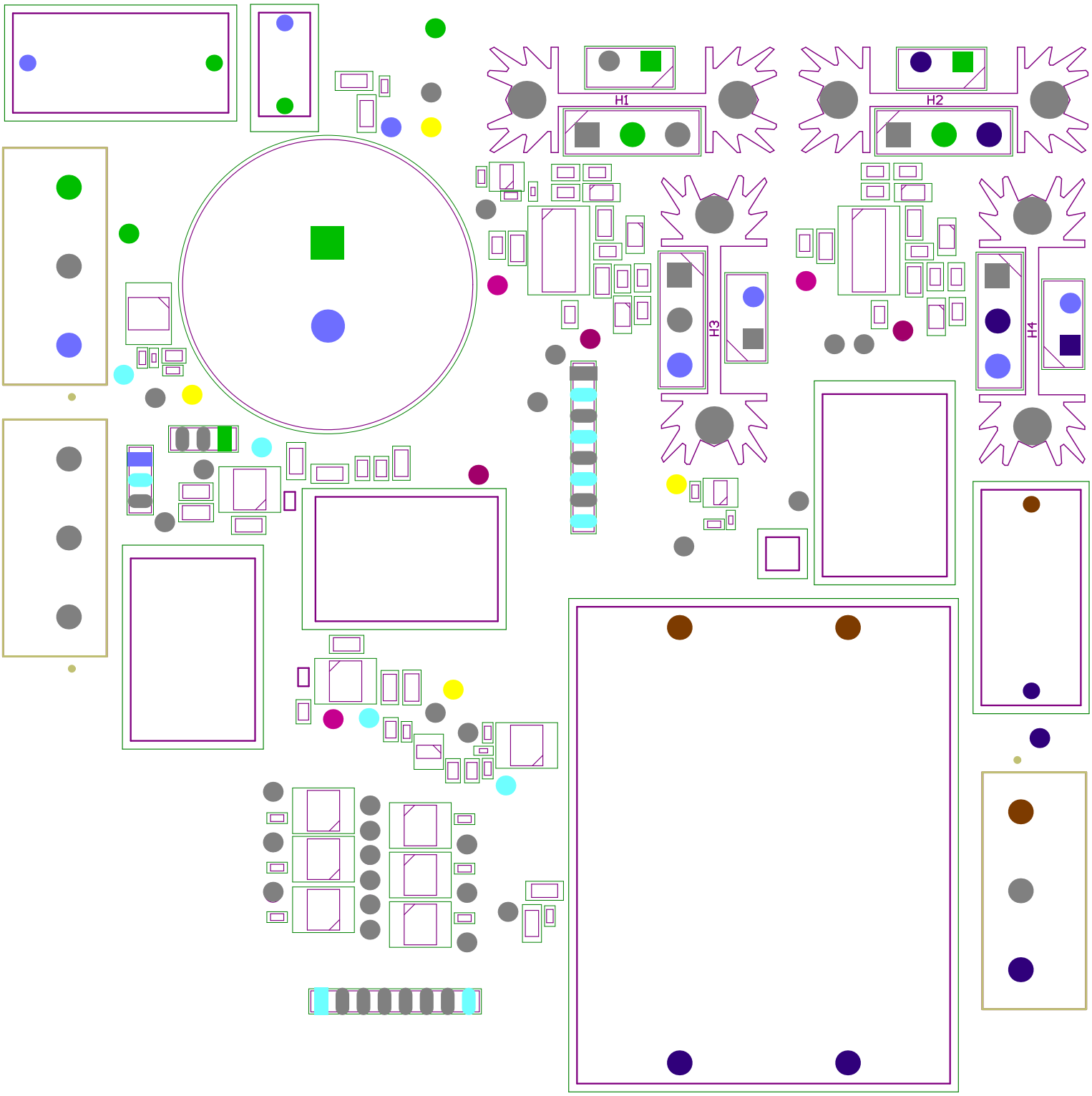
TP39 TP40 TP41 TP42
V_AC V_ac I_AC I_ac
Vbias_1.65V
IC13 LT1789CS8-1#PBF
IC14 LT1789CS8-1#PBF
C41 100nF
C42 100nF

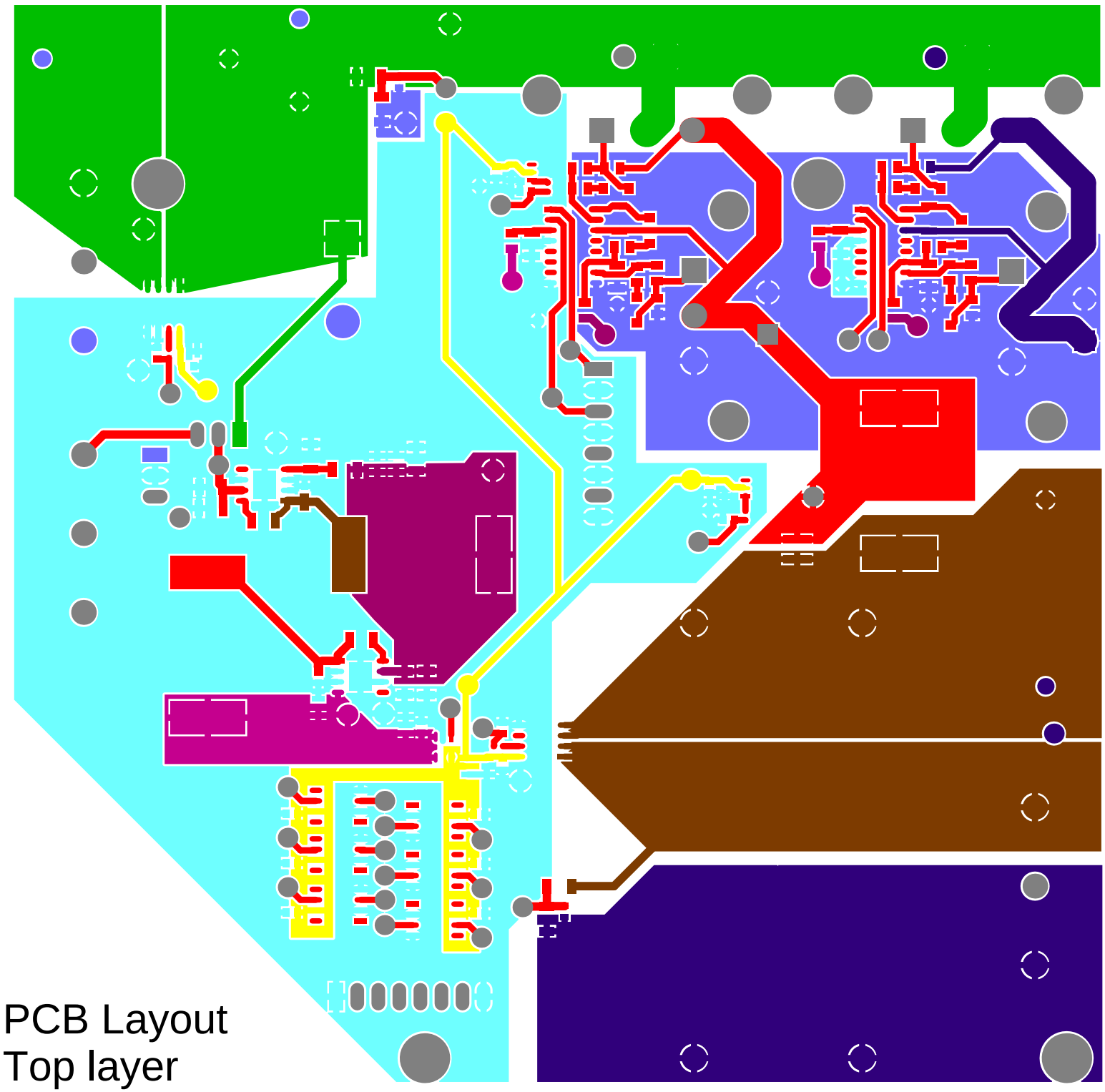
J6 M20-9990845
I_bus V_bus tmp_H tmp_L I_ac V_ac
J7 M20-9990845
PWMQ1 PWMQ3 PWMQ4 PWMQ2

Title			
Size A2	Number		Revision
Date: 5.29.2021		Sheet of	
File: C:\Altium\..\PowerV1_2.SchDoc		Drawn By:	

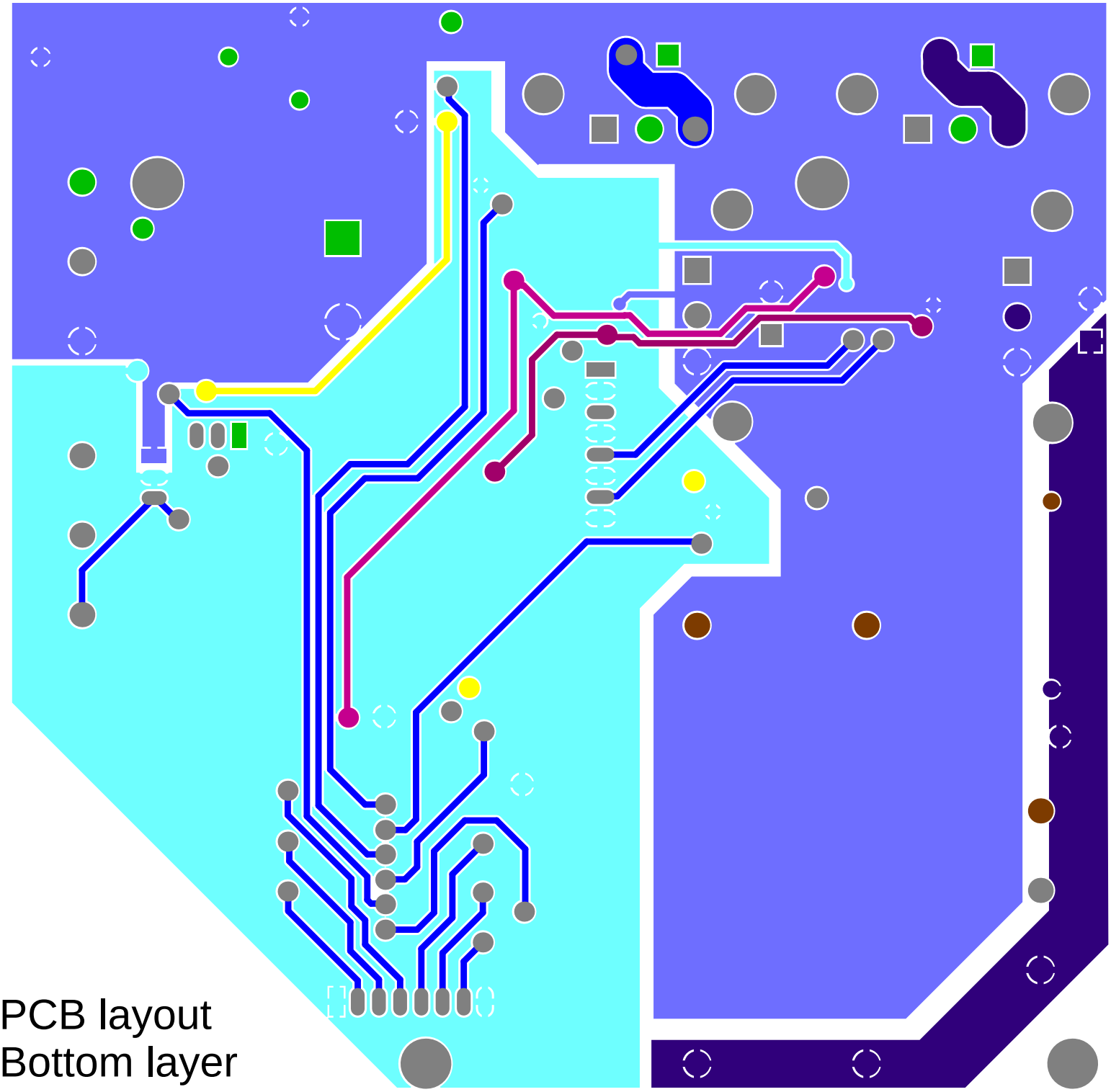


PCB layout
with components

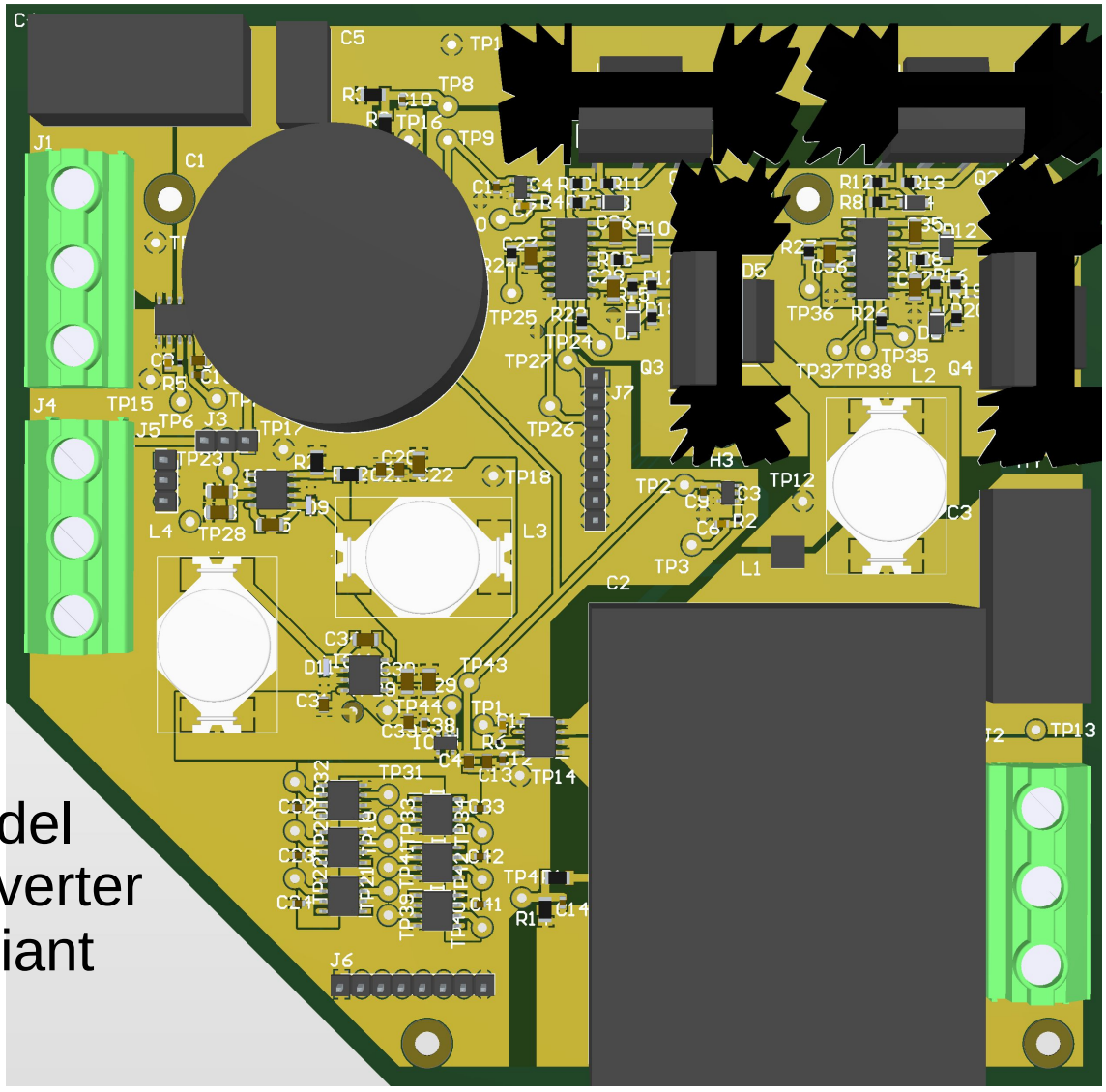

H1
H2
H3
H4



PCB Layout
Top layer



PCB layout
Bottom layer



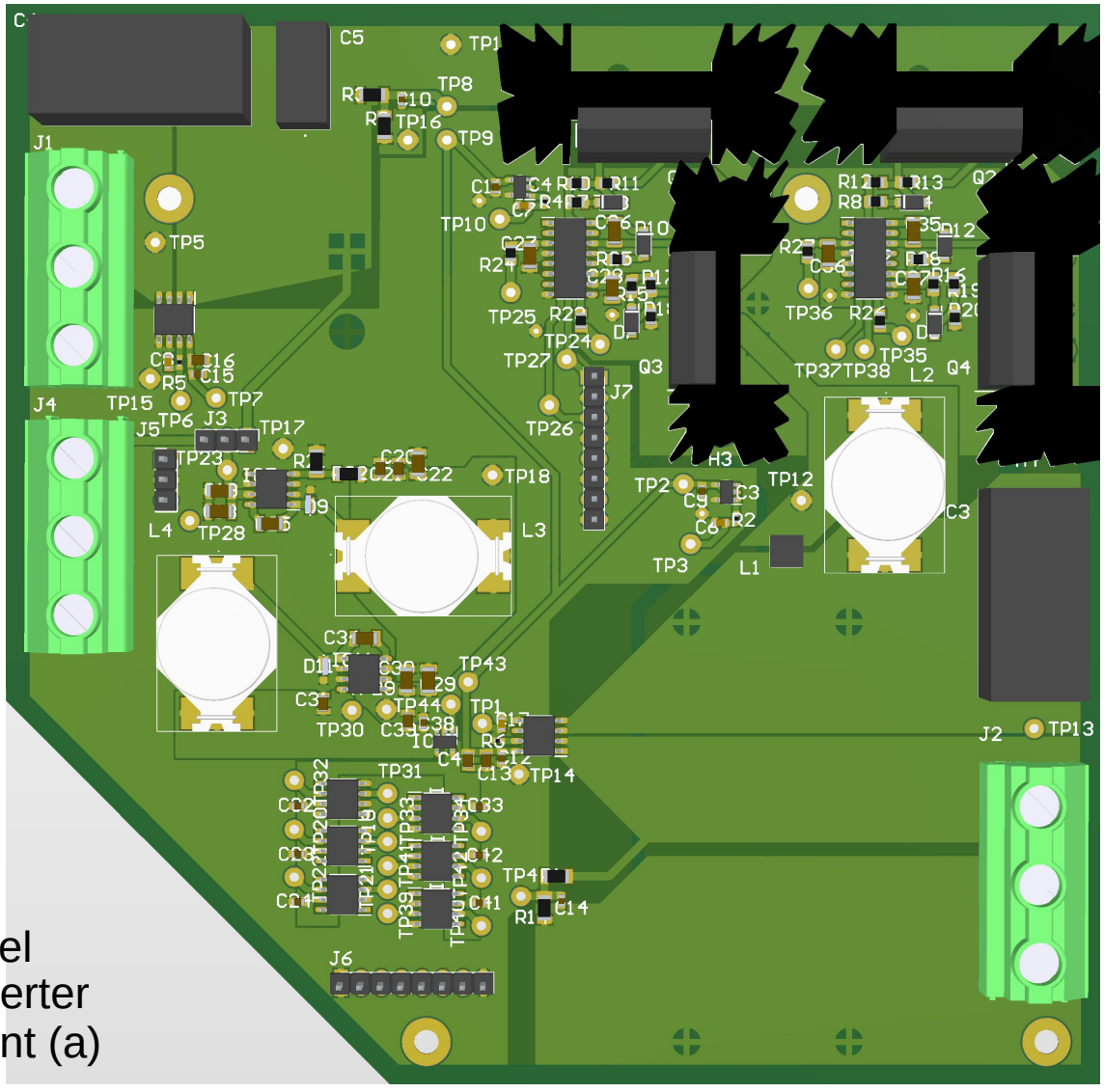
3D model
of the inverter
Si variant

Bill of materials of the inverter Si variant

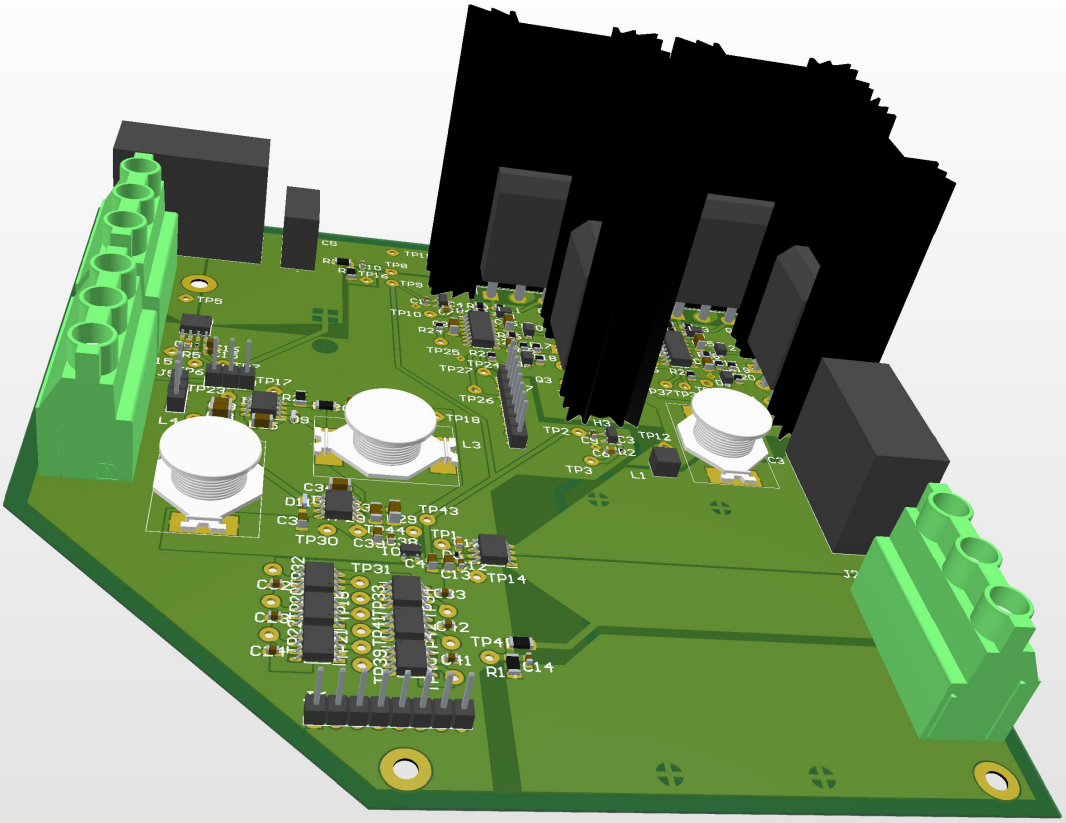
Manufacturer_Part_Number	Designator	Quantity	Description	Comment
MAL225729471E3	C1	1	Electrolytic Capacitor, 470 µF, 500 V, 257 PRM-SI Series, ± 20%, Quick Connect, Snap-In	470 µF
C4AF1EW5470A3AK	C2	1	Film Capacitor	47uF
ECW-F6223HL	C5	1	CAP FILM 0.022UF 3% 630VDC RAD	0.022uF
C0603C100C5GACTU	C6, C7, C8, C17	4	Capacitor	10pF
06035C104KAT2A	C9, C11, C12, C15, C23, C24, C32, C33, C38, C41, C42	11	Capacitor	100nF
C0603X103K1RAC3316	C10, C14	2	Capacitor	10nF
C2012X7R1A106K125AC	C13, C16, C20, C21, C31, C39, C40	7	Capacitor	10uF
1206B475K500CT	C18, C29	2	Capacitor	4.7uF
C1206C105K5RECAUTO	C19, C22, C28, C30, C37	5	Capacitor	1uF
VJ1206Y103JXAMC	C25, C34	2	Capacitor	10nF
CGA5L3X7T2E224K160AA	C26, C35	2	Capacitor	220nF
VJ1206Y223KXXCW1BC	C27, C36	2	Capacitor	22nF
C4D10120A	D1, D2, D5, D6	4	Schottky Diode	C4D10120A
RSFJL R3	D3, D4, D7, D8, D10, D12	6	Diode	RSFJL_R3
MSS1P6-M3/89A	D9, D11	2	Schottky Diode	MSS1P6-M3_89A
	H1, H2, H3, H4	4	HEATSINK TO-218/TO-247 W/PINS 2"	513201B02500G
ACS722LLCTR-10AB-T	IC1, IC2	2	Integrated Circuit	ACS722
TMP35GRTZ-REEL7	IC3, IC4	2	Integrated Circuit	TMP35
LT1789CS8-1#PBF	IC5, IC6, IC9, IC10, IC13, IC14	6	Integrated Circuit	LT1789CS8-1#PBF
LM22675MR-5.0/NOPB	IC7, IC11	2	Integrated Circuit	LM22675
2EDF7275FXUMA1	IC8, IC12	2	Integrated Circuit	2EDF7275FXUMA1
REF2033AIDDCT	IC15	1	Integrated Circuit	REF2033
1714984	J1, J2, J4	3	Connector	1714984
61300311121	J3, J5	2	Connector	61300311121
M20-9990845	J6, J7	2	Connector	M20-9990845
SDR2207-680KL	L2	1	Inductor	68uH
SDR2207-101KL	L3, L4	2	Inductor	100uH
IPW60R090CFD7XKSA1	Q1, Q2, Q3, Q4	4	MOSFET Si (N-Channel)	IPW60R090CFD7XKSA1
CRCW12061M00FKEAC	R1, R3	2	Resistor	1M
MC1206S4F0000T5E	R2, R4, R5, R6	4	Resistor	0R
CRCW08052R20FKEAHP	R7, R8, R15, R16, R25, R28	6	Resistor	2R2
RCS120622K0JNEA	R9, R14	2	Resistor	22K
RC0805FR-074R42L	R10, R12, R17, R19	4	Resistor	4R42
AC0805FR-0710KL	R11, R13, R18, R20	4	Resistor	10K
CRCW0805510RFKEAHP	R21, R24, R27	3	Resistor	510R
CRCW1206470RFKEAHP	R22	1	Resistor	470R
CRCW080510R0FKEAHP	R23, R26	2	Resistor	10R
20-313143	TP1, TP2, TP3, TP4, TP5, TP6, TP7, TP8, TP9, TP10, TP11, TP12, TP13, TP14, TP15, TP16, TP17, TP18, TP19, TP20, TP21, TP22, TP23, TP24, TP25, TP26, TP27, TP28, TP29, TP30, TP31, TP32, TP33, TP34, TP35, TP36, TP37, TP38, TP39, TP40, TP41, TP42, TP43, TP44	44	Test Point	20-313143

Bill of materials of the inverter Si variant

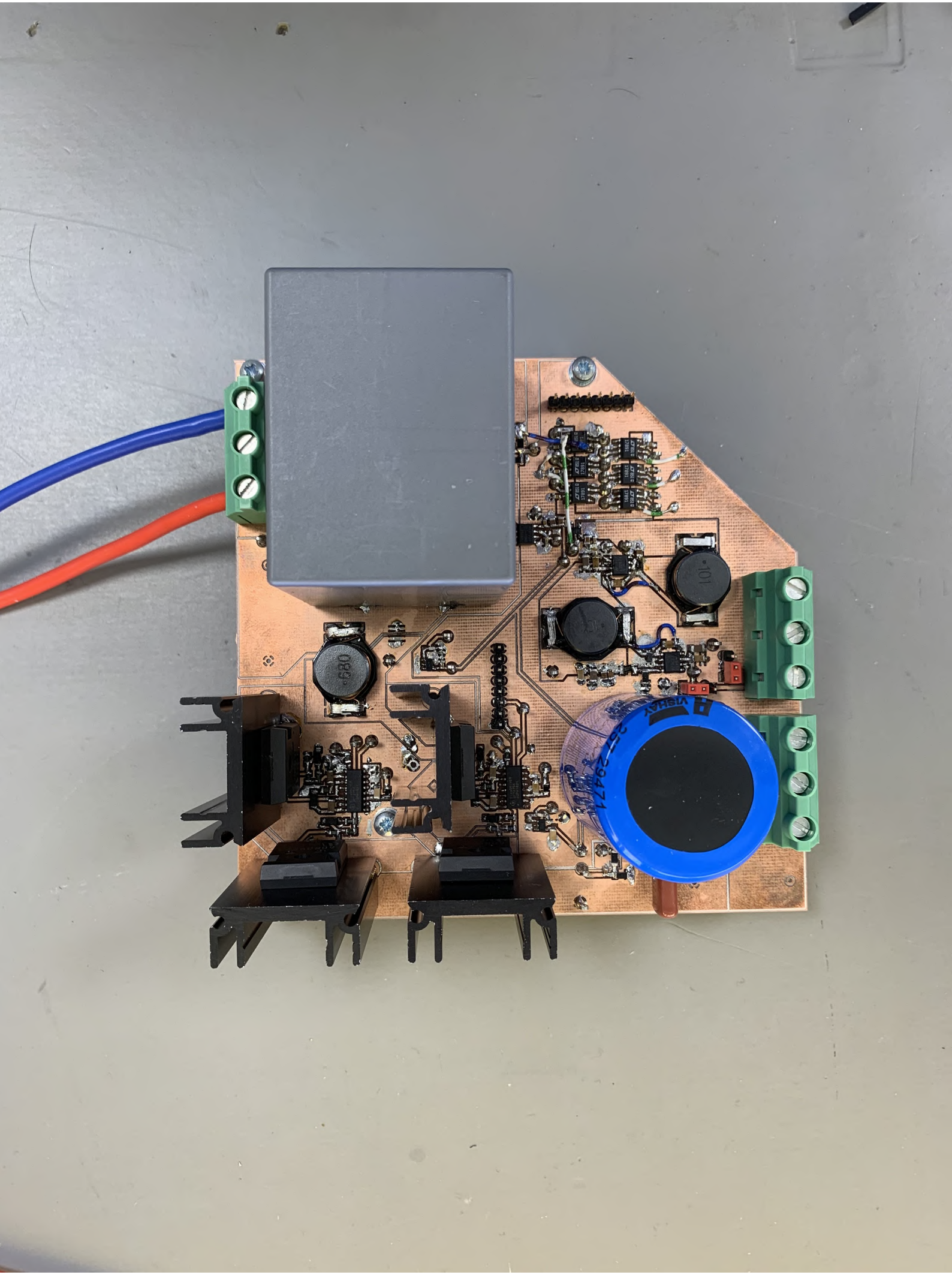
Manufacturer_Part_Number	Designator	Quantity	Description	Comment
ECW-FE2W475J	C3, C4	2	CAP FILM 4.7UF 5% 450VDC RADIAL	4.7uF
ECW-F6223HL	C5	1	CAP FILM 0.022UF 3% 630VDC RAD	0.022uF
C0603C100C5GACTU	C6, C7, C8, C17	4	Capacitor	10pF
06035C104KAT2A	C9, C11, C12, C15, C23, C24, C32, C33, C38, C41, C42	11	Capacitor	100nF
C0603X103K1RAC3316	C10, C14	2	Capacitor	10nF
C2012X7R1A106K125AC	C13, C16, C20, C21, C31, C39, C40	7	Capacitor	10uF
1206B475K500CT	C18, C29	2	Capacitor	4.7uF
C1206C105K5RECAUTO	C19, C22, C28, C30, C37	5	Capacitor	1uF
VJ1206Y103JXAMC	C25, C34	2	Capacitor	10nF
CGA5L3X7T2E224K160AA	C26, C35	2	Capacitor	220nF
VJ1206Y223KXXCW1BC	C27, C36	2	Capacitor	22nF
RSFJL R3	D3, D4, D7, D8, D10, D12	6	Diode	RSFJL_R3
MSS1P6-M3/89A	D9, D11	2	Schottky Diode	MSS1P6-M3_89A
	H1, H2, H3, H4	4	HEATSINK TO-218/TO-247 W/PINS 2"	513201B02500G
ACS722LLCTR-10AB-T	IC1, IC2	2	Integrated Circuit	ACS722
TMP35GRTZ-REEL7	IC3, IC4	2	Integrated Circuit	TMP35
LT1789CS8-1#PBF	IC5, IC6, IC9, IC10, IC13, IC14	6	Integrated Circuit	LT1789CS8-1#PBF
LM22675MR-5.0/NOPB	IC7, IC11	2	Integrated Circuit	LM22675
2EDF7275FXUMA1	IC8, IC12	2	Integrated Circuit	2EDF7275FXUMA1
REF2033AIDDCT	IC15	1	Integrated Circuit	REF2033
1714984	J1, J2, J4	3	Connector	1714984
61300311121	J3, J5	2	Connector	61300311121
M20-9990845	J6, J7	2	Connector	M20-9990845
XAL4030-682MEC	L1	1	Inductor	6.8uH
SDR2207-101KL	L3, L4	2	Inductor	100uH
IMW65R072M1HXKSA1	Q1, Q2, Q3, Q4	4	MOSFET (N-Channel)	IMW65R072M1HXKSA1
CRCW12061M00FKEAC	R1, R3	2	Resistor	1M
MC1206S4F0000T5E	R2, R4, R5, R6	4	Resistor	0R
CRCW08052R20FKEAHP	R7, R8, R15, R16, R25, R28	6	Resistor	2R2
RCS120622K0JNEA	R9, R14	2	Resistor	22K
RC0805FR-074R42L	R10, R12, R17, R19	4	Resistor	4R42
AC0805FR-0710KL	R11, R13, R18, R20	4	Resistor	10K
CRCW0805510RFKEAHP	R21, R24, R27	3	Resistor	510R
CRCW1206470RFKEAHP	R22	1	Resistor	1K27
CRCW080510R0FKEAHP	R23, R26	2	Resistor	10R
20-313143	TP1, TP2, TP3, TP4, TP5, TP6, TP7, TP8, TP9, TP10, TP11, TP12, TP13, TP14, TP15, TP16, TP17, TP18, TP19, TP20, TP21, TP22, TP23, TP24, TP25, TP26, TP27, TP28, TP29, TP30, TP31, TP32, TP33, TP34, TP35, TP36, TP37, TP38, TP39, TP40, TP41, TP42, TP43, TP44	44	Test Point	20-313143

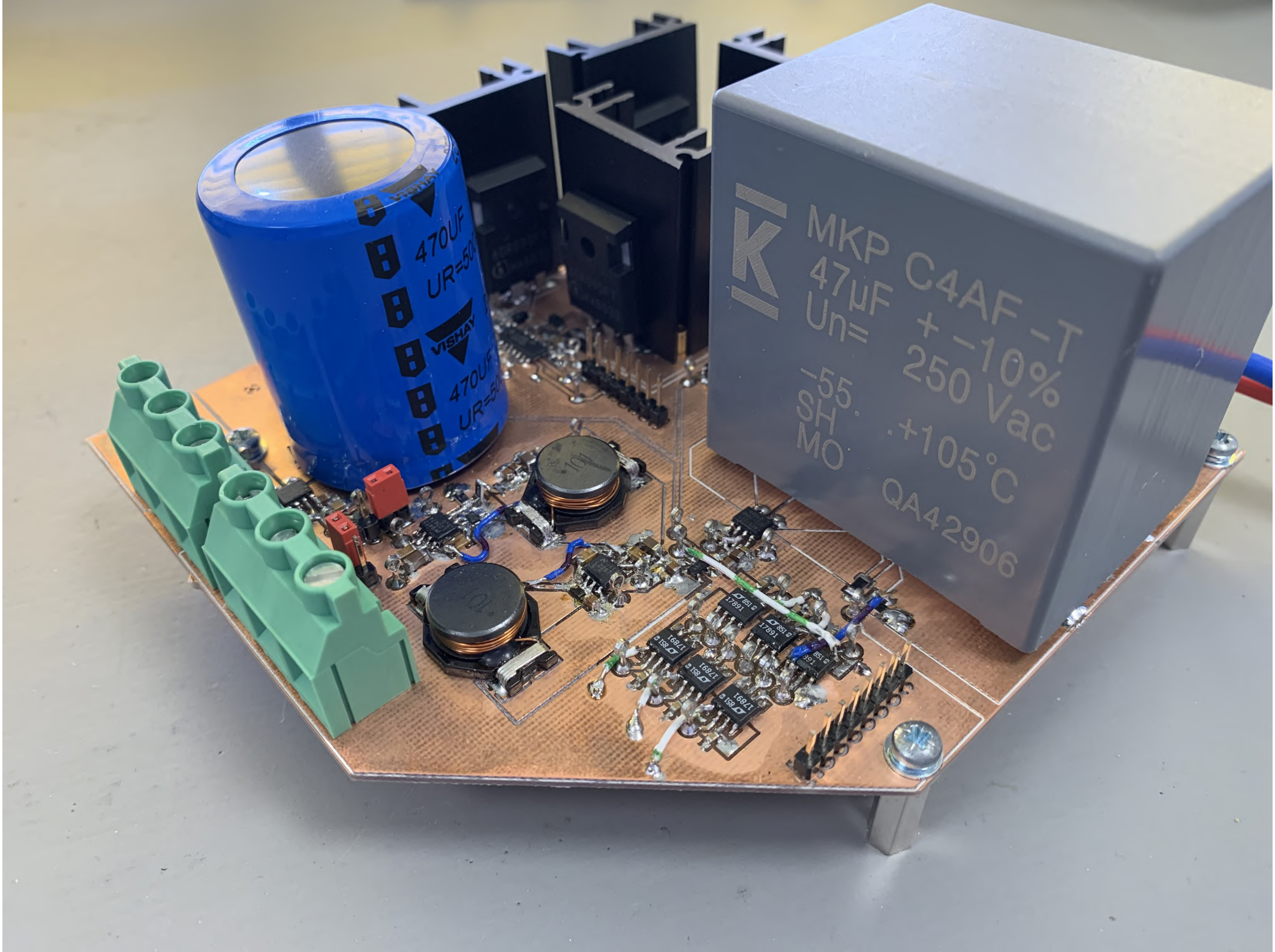


3D model
of the inverter
SiC variant (a)



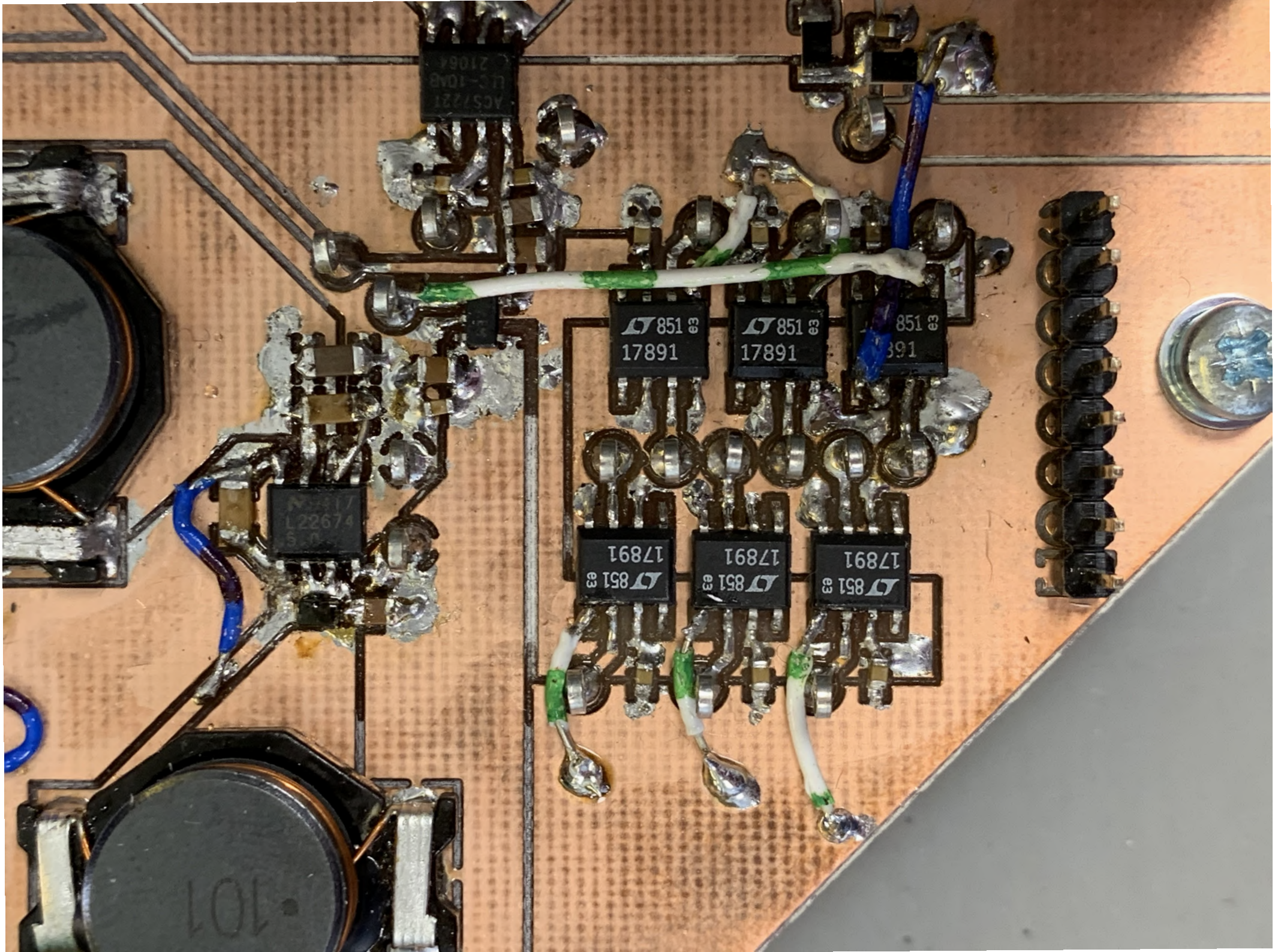
3D model
of the inverter
SiC variant (b)

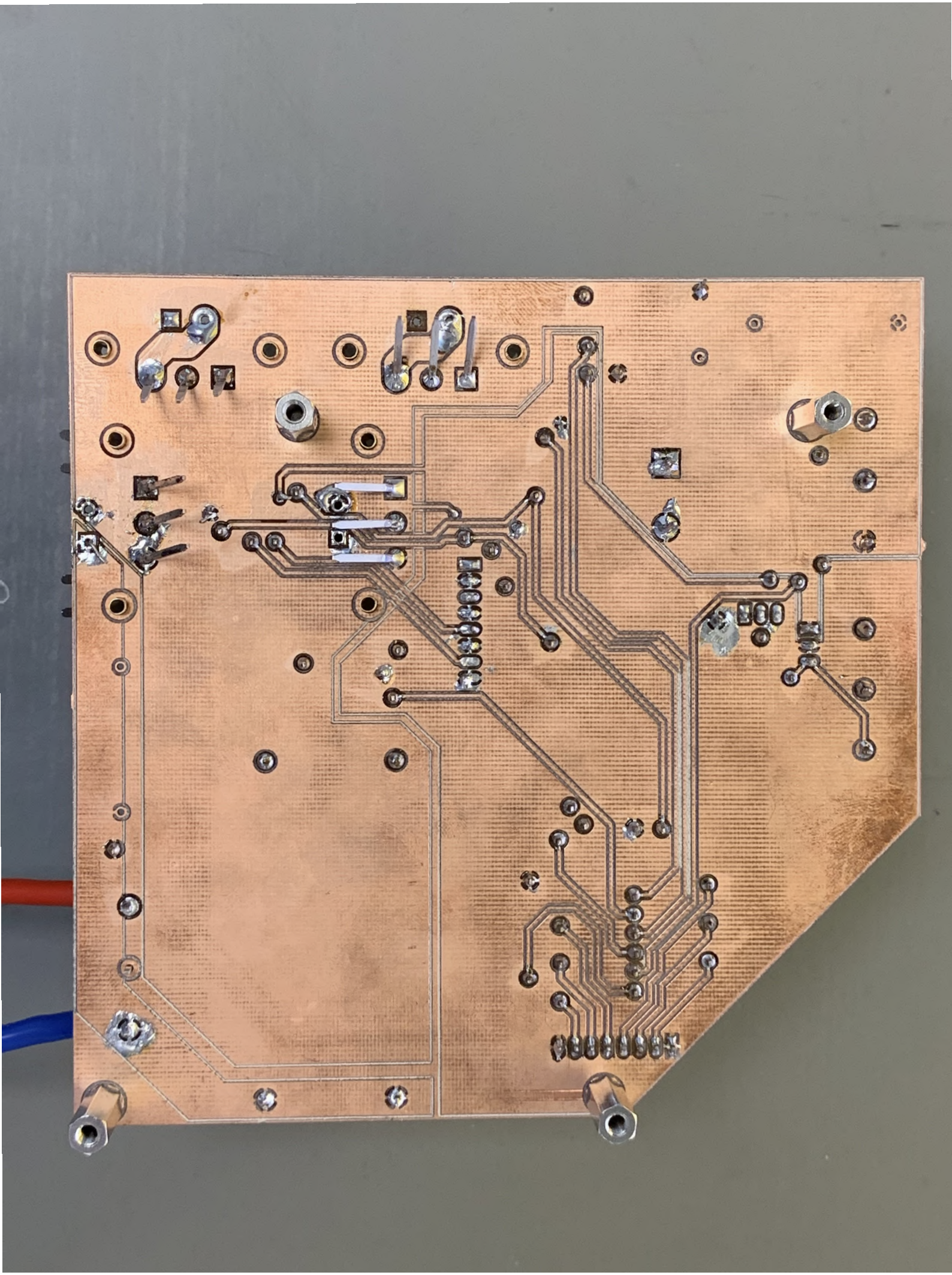



470UF
VISHAY
MKP C4AF -T
47µF +-10%
Un= 250 Vac
-55 .. +105°C
SH
MO
QA42906

EiceDRIVER
2F7275A
02PH1821
851
17891
851
17891
851
17891
851
17891
851
17891
851
17891

101
6800
101


851 e3
17891
851 e3
17891
851 e3
17891
851 e3
17891
851 e3
17891
L22674



Appendix B

Matlab simulation

This appendix includes the Simulink model shown in figure B.0.1fused to understand the full bridge inverter, the model employs a full bridge inverter, PWM signal generator, and an LC output filter.

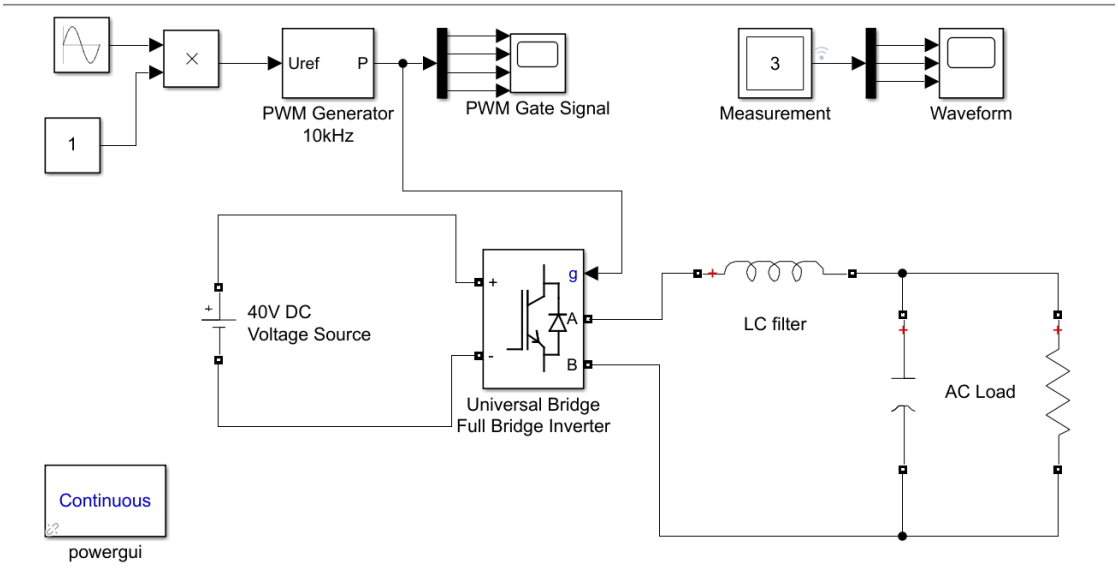


Figure B.0.1: Simulink model.

Figure B.0.2 shows an example of the output of the simulation, that allows for the study of ripple in voltage and current in relation with the PWM frequency.

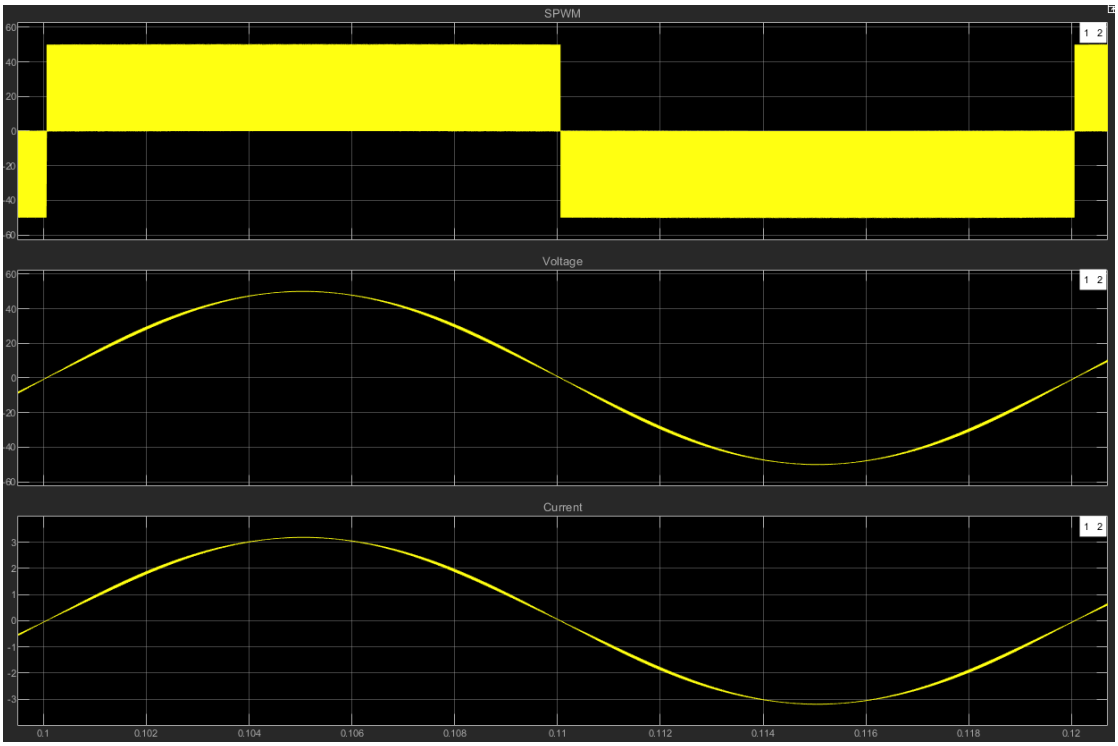


Figure B.0.2: Simulink scope showing the SPWM, Voltage and Current outputs.

Appendix C

Components values calculations

This appendix includes the values calculations for most of the components used in the inverter circuit.

C.1 Driver

The choices made here apply to both of the driver ICs implemented in this work. The included information provided in this section is brief and concise on purpose, since it is will explained in the application note [2] and there is no real benefit of repeating it here.

C.1.1 Shunt resistor dimensioning

The available supply VDD for the driver is 5 V and the switching frequency will not exceed 500 MHz, thus from the application note [2] table 1, a suitable shunt resistor should be between 732 Ω and 453 Ω and the value ***510 Ω*** is chosen in the package size 0805 as recommended in the application note.

C.1.2 Input bypass capacitor

To keep ΔV sufficiently low (e.g. in the few tens of mV range), a minimum Cin of 10 nF is recommended. On the other hand, if the SLDO is activated, Cin should not exceed 22nF due to stability reasons. It is suggested to use a ceramic capacitor in SMD 0805 package with 25 V DC voltage rating [2]. The maximum value of ***22nF*** is chosen in the recommended package size and DC voltage rating.

C.1.3 Output bypass capacitor

A proper dimensioning requires to choose an output bypass capacitance which is at least 20 times larger than the equivalent input capacitance of the MOSFET; a range from 100 nF to 1 µF is common when driving Infineon CoolMOS switches. As for the input bypass capacitance, it is suggested to use a ceramic capacitance in SMD 0805 package with 25 V DC voltage rating [2]. The maximum value of ***1µF*** is chosen in the recommended package size and DC voltage rating.

C.1.4 Bootstrap circuit dimensioning

The bootstrap circuit includes (C26, R25 and D10) as shown in figure 3.1.4, The bootstrap circuit operation is defined by two main periods:

- Charging period: capacitor C26 is charged while the low-side (Q3) switch is ON and the high-side (Q1) switch is OFF. When the Q3 is ON, the diode D10 is forward-biased and the current flows from the voltage source V_{DDB} (+10/18 V) into CB through the bootstrap resistor R25, diode D10, and the low-side switch that finally closes the loop to ground.
- Sourcing period: when the low-side switch is turned off and/or the high-side switch

starts conducting, the source voltage of the HS switch (GNDA) quickly rises until it approaches the full-bridge supply voltage (V_{BUS}). As a consequence, the bootstrap diode D10 gets reverse biased and disconnects the ground supply from C26. The capacitor C26 must be dimensioned to store the energy required to keep the HS switch ON until the next commutation; in this way, a supply voltage of $V_{BUS} + V_{DDB}$ (+10/18 V) is ensured during the sourcing period for proper HS MOSFET driving.

Using equation (10) from [2] copied here C.1 to calculate C26 and R25, using $f_{sw} = 200kHz$ and $D_{Max} = 50\%$as the worst case, then $t_{min} = 2.5\mu s$ and $C26 \times R25 = 0.5\mu$, choosing R25=2.2 gives $C26 \approx 220nF$.

$$t_{min} = \frac{(1 - D_{Max})}{f_{sw}} \geq 5 \times C26 \times R25 \tag{C.1}$$

The diode D10 needs to be able to handle the current through the bootstrap circuit, this current is calculated using the equation (12) and (8), which are combined and copied here C.2.

$$I_{max} = \frac{Q_g + (D_{Max} \times I_{VDDAqu2} \times \frac{1}{f_{sw}})}{(1 - D_{Max})} \times f_{sw} \tag{C.2}$$

Where Q_g is the MOSFET gate charge (51 nC the worst case for the Si MOSFET), D_{Max} is the duty cycle (50%), $I_{VDDAqu2}$ is the quiescent current drawn by the driver during the sourcing period (0.6 mA from the driver data-sheet), f_{sw} is the switching frequency (200kHz worst case). This calculation gives $I_{max} = 21mA$, which is quite small and the diode chosen ***RSFJL_R3*** that has a current rating of 500 mA is capable of handling it.

C.1.5 Gate resistance dimensioning

One of the factors that must be considered when dimensioning the gate resistance is the peak current that the driver can handle to ensure safe operation. Using the equations (1) and (2) from the application note [3] copied here C.3 to find the minimum values allowed for the gate

resistance.

$$I_{G,charging} = \frac{V_{OUTA}}{R_{DSon} + R_{Gon}}$$
$$I_{G,discharging} = \frac{V_{OUTA}}{R_{DSoff} + R_{Goff}} \quad \text{(C.3)}$$

Where $I_{G,charging}$ should be less than Peak Sourcing Output Current 4 A, $I_{G,discharging}$ should be less than Peak Sinking Output Current 8 A, R_{DSon} is High-level (Sourcing) Output Resistance (0.42 Ω, worst case from the data-sheet [6]), R_{DSoff} is Low-level (Sinking) Output Resistance (0.18 Ω, worst case from the data-sheet [6]) and V_{OUTA} is the driver output voltage (18 V, worst case for the SiC variant). This will give a minimum value for the $R_{Gon} = \frac{V_{OUTA}}{I_{G,charging}} - R_{DSon,min} = \frac{18}{4} - 0.42 = 2.9\Omega$ (R10) and $R_{Goff} = \frac{V_{OUTA}}{I_{G,discharging}} - R_{DSoff,min} = \frac{18}{8} - 0.18 = 2.07\Omega$ (R7||R10). Thus, the values of ***R7=4.42 Ω*** and ***R10=2.2 Ω*** are chosen in order to give some safety margin.

C.2 Power supply

For output voltages greater than 5 V, the -5.0 option may provide better loop bandwidth than the -ADJ option, thus the -5.0 option is used in this design. Equation 11 in the data-sheet [12] is used to determine the resistor values in the output divider, which is copied here C.4. Figure C.2.1 is copied from the data-sheet, which shows how the voltage divider is implemented.

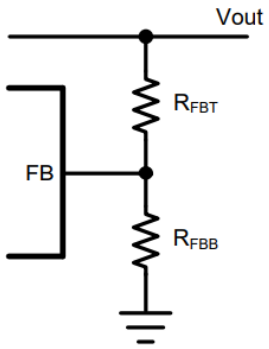


Figure C.2.1: Resistive Feedback Divider [12].

$$R_{FBT} = \frac{R_{FBB} * (V_{out} - 5)}{5 + R_{FBB} * 5 \times 10^{-4}} \tag{C.4}$$

Where the R_{FBB} is the voltage divider bottom resistor and the R_{FBT} is the voltage divider top resistor. A maximum of 2 kΩ is recommended for the -5.0 option. For the -5.0 option, the total internal divider resistance is typically 9.93 kΩ.
Using the information above, the values for the voltage divider is arranged in this table C.2.1.

Output voltage	R_{FBT}	R_{FBB}
10 V	470Ω	512Ω
18 V	1270Ω	512Ω

Table C.2.1: Resistive Feedback Divider Values

C.3 DC link

Using equation (13) from [19], adapting it to our use case here C.5.

$$C_{dc} = \frac{S}{4\pi f_g V_{dc} \overline{V_{dc}}} \tag{C.5}$$

Where C_{dc} the capacitance of dc-link capacitor in farad, S is the rated power of the inverter (Although S= 300 W in this design), f_g is the output voltage frequency (50Hz in this design), V_{dc} is the dc input voltage (50 V in this design) and $\overline{V_{dc}}$ is maximum permissible voltage ripple (8.5% of the V_{dc} = 4.25 V in this design).
Using the information above, $C_{dc} = 2246\mu F$.

This calculated value is quite large but it is the worst case value, and it does not take into consideration that, when using high switching frequency the DC link capacitor need to deliver less current at each turn on, and has time to recover at each turn off. Thus, the values of ***470µF*** and ***4.7µF*** are chosen for Si and SiC variant respectively and they are considered sufficient.

C.4 Output filter

Using equation (10) from [14] to calculate L for the LC output filter, adapting it to our use case here C.6.

$$L = \frac{V_{dc}}{4 f_{sw} \Delta i_{pp,max}} \tag{C.6}$$

Where L the inductance of output inductor in henry, V_{dc} is the dc input voltage (50 V in this design), f_{sw} is the switching frequency (20kHz for Si and 200kHz for SiC), $\Delta i_{pp,max}$ is maximum permissible current ripple in the output current (20% of the output current (3 A) =0.6 A in this design).

Using the information above, the values for the inductor is arranged in this table C.4.1.

Variant	L
Si	$1mH$
SiC	$0.1mH$

Table C.4.1: L Values

The cutoff frequency for the LC low pass filter is given by the equation C.7.

$$\begin{aligned} F_{cutoff} &= \frac{1}{2\pi\sqrt{LC}} \\ C &= \frac{1}{(F_{cutoff} 2\pi\sqrt{L})^2} \end{aligned} \tag{C.7}$$

F_{cutoff} should be 2kHz and 20kHz for Si and SiC variant respectively, this will give the following values for the capacitor is arranged in this table C.4.2.

Variant	C
Si	$6.3\mu F$
SiC	$0.63\mu F$

Table C.4.2: C Values

Since it is quite difficult to find inductors with the value need that also can tolerate the

current ripple at the same time, a compromise was made by using a larger capacitor and a smaller inductor and trying to keep the same cutoff frequency for both variants, running the risk of lower frequency harmonics rejection. Thus, the values of ***C=47µF*** and ***L=68µH*** for the Si variant resulting in $F_{cutoff} \approx 2.8kHz$, and ***4.7µF*** and ***L=6.8µH*** for the SiC variant resulting in $F_{cutoff} \approx 28kHz$.

Appendix D

Microcontroller code files

This appendix includes the full code files that have been written by the author.

And here is the whole STM32CubeIDE project as a zip file  .

D.1 Main function code

Listing D.1: "main.c"

```
1  /* USER CODE BEGIN Header */
2  /**
3    ******************************************************************************
4    * @file           : main.c
5    * @brief          : Main program body
6    ******************************************************************************
7    * @attention
8    *
9    * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
10   * All rights reserved.</center></h2>
11   *
12   * This software component is licensed by ST under BSD 3-Clause license,
13   * the "License"; You may not use this file except in compliance with the
14   * License. You may obtain a copy of the License at:
```

```
15   *                        opensource.org/licenses/BSD-3-Clause
16   *
17   ******************************************************************************
18   */
19 /*InverterMicro1 */
20 /* USER CODE END Header */
21 /* Includes ------------------------------------------------------------------*/
22 #include "main.h"
23 /* Private includes ----------------------------------------------------------*/
24 /* USER CODE BEGIN Includes */
25 #include "usart2.h"
26 #include "main.h"
27 #include "adc.h"
28 #include "tim2.h"
29 #include "tim15.h"
30 #include "gpio.h"
31 /*
32  * Debug pins
33  * E8: main while loop
34  * E9: functional communication
35  * E10: ADC4 interrupt
36  * E11: DMA2 channel2 interrupt
37  *
38  * E13: TIM2 interrupt
39  * E14: usart2 rx interrupt
40  * E15: usart2 tx interrupt
41  *
42  * Interface
43  * D8: usart2 tx
44  * D9: usart2 rx
45  *
46  * PD8: ADC4 CH12 input
47  * PD9: ADC4 CH13 input
48  * PD10: ADC4 CH7 input
49  * PD11: ADC4 CH8 input
50  * PB12: ADC4 CH3 input
51  * PB14: ADC4 CH4 input
52  *
53  * PD1: TIM2CH1 PWM Q1 output
54  * PD4: TIM2CH2 PWM Q3 output
55  * PD7: TIM2CH3 PWM Q2 output
56  * PD6: TIM2CH4 PWM Q4 output
```

```
57  */
58 /* USER CODE END Includes */
59 /* Private typedef -----------------------------------------------------------*/
60 /* USER CODE BEGIN PTD */
61 /* USER CODE END PTD */
62 /* Private define ------------------------------------------------------------*/
63 /* USER CODE BEGIN PD */
64 /* USER CODE END PD */
65 /* Private macro -------------------------------------------------------------*/
66 /* USER CODE BEGIN PM */
67 /* USER CODE END PM */
68 /* Private variables ---------------------------------------------------------*/
69 /* USER CODE BEGIN PV */
70 FlagStatus rx_flag, tx_flag_1, running;
71 uint16_t adc4_val[ADC4_BUFF_LEN];
72 uint16_t tmp_high=1270, tmp_low=1130; // Roughly 90 C, 80 C respectively
73 /* USER CODE END PV */
74 /* Private function prototypes -----------------------------------------------*/
75 void SystemClock_Config(void);
76 /* USER CODE BEGIN PFP */
77 /* USER CODE END PFP */
78 /* Private user code ---------------------------------------------------------*/
79 /* USER CODE BEGIN 0 */
80 /* USER CODE END 0 */
81 /**
82   * @brief  The application entry point.
83   * @retval int
84   */
85 int main(void)
86 {
87   /* USER CODE BEGIN 1 */
88   /* USER CODE END 1 */
89   /* MCU Configuration--------------------------------------------------------*/
90   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
91   HAL_Init();
92   /* USER CODE BEGIN Init */
93   /* USER CODE END Init */
94   /* Configure the system clock */
95   SystemClock_Config();
96   /* USER CODE BEGIN SysInit */
97         // Sensors
98         adc4_init();
```

```
99          // ADC4 trigger
100         tim15_init();
101         // PC-Communication
102         usart2_init();
103         // In/out for troubleshooting
104         gpio_init();
105         //PWM generation
106         tim2_init();
107   /* USER CODE END SysInit */
108   /* Initialize all configured peripherals */
109   /* USER CODE BEGIN 2 */
110   /* USER CODE END 2 */
111   /* Infinite loop */
112   running= SET; // Should be set and reset with a command from the Python program
113   while (1)
114   {
115         /* USER CODE BEGIN WHILE */
116           GPIOE->BSRR = GPIO_BSRR_BS_8;
117           if (running) {
118                   if (tx_flag_1){
119                           tx_flag_1= RESET;
120                           // Send data to PC
121   uint8_t s[16] = {'H', adc4_val[0] >> 8 , adc4_val[0], adc4_val[1] >> 8 , adc4_val[1], \
122   'I', adc4_val[2] >> 8 , adc4_val[2], adc4_val[3] >> 8 , adc4_val[3], \
123   'V', adc4_val[4] >> 8 , adc4_val[4], adc4_val[5] >> 8 , adc4_val[5], '\0'};
124   usart2_tx( s, 16 );
125   // temperature check
126   if ((adc4_val[0]>tmp_high) | (adc4_val[1]>tmp_high)) {
127           TIM2->CR1 = 0; // Disabling TIM2 to stop the PWM generation
128   }
129   else {
130           if ((!TIM2->CR1) & (adc4_val[0]<tmp_low) & (adc4_val[0]<tmp_low)) {
131                   TIM2->CR1 |= TIM_CR1_CEN; // Enabling TIM2 to start the PWM generation
132           }
133   }
134                   }
135           }
136           GPIOE->BSRR = GPIO_BSRR_BS_8;
137         /* USER CODE END WHILE */
138   }
139 }
140 /**
```

```
141   * @brief System Clock Configuration
142   * @retval None
143   */
144 void SystemClock_Config(void)
145 {
146   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
147   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
148
149   /** Initializes the RCC Oscillators according to the specified parameters
150   * in the RCC_OscInitTypeDef structure.
151   */
152   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
153   RCC_OscInitStruct.HSIState = RCC_HSI_ON;
154   RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
155   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
156   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
157   RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL16; //(8Mb/2)*16= 64Mb
158   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
159   {
160     Error_Handler();
161   }
162   /** Initializes the CPU, AHB and APB buses clocks
163   */
164   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
165                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
166   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
167   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1; // SYSCLK not divided; 64Mb
168   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2; // HCLK divided by 2; 32Mb; Timer 64Mb
169   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1; // HCLK not divided; 64Mb
170
171   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
172   {
173     Error_Handler();
174   }
175 }
176
177 /* USER CODE BEGIN 4 */
178
179
180 /* USER CODE END 4 */
181
182 /**
```

```
183   * @brief  This function is executed in case of error occurrence.
184   * @retval None
185   */
186 void Error_Handler(void)
187 {
188   /* USER CODE BEGIN Error_Handler_Debug */
189   /* User can add his own implementation to report the HAL error return state */
190
191   /* USER CODE END Error_Handler_Debug */
192 }
193
194 #ifdef  USE_FULL_ASSERT
195 /**
196   * @brief  Reports the name of the source file and the source line number
197   *         where the assert_param error has occurred.
198   * @param  file: pointer to the source file name
199   * @param  line: assert_param error line source number
200   * @retval None
201   */
202 void assert_failed(uint8_t *file, uint32_t line)
203 {
204   /* USER CODE BEGIN 6 */
205   /* User can add his own implementation to report the file name and line number,
206      tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
207   /* USER CODE END 6 */
208 }
209 #endif /* USE_FULL_ASSERT */
210
211 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

D.2 Analog to digital converter code

Listing D.2: "adc.c"

```
1 /*
2  * adc.c
3  *
4  *  Created on: Apr 25, 2021
5  *      Author: alihj
6  */
7
```

```
8  #include "adc.h"
9
10 extern FlagStatus tx_flag_1;
11 extern uint16_t adc4_val[ADC4_BUFF_LEN];
12
13 int8_t adc4_init( void ){
14         /* Clock enable */
15         RCC->AHBENR |= RCC_AHBENR_ADC34EN | RCC_AHBENR_GPIOBEN \
16                        | RCC_AHBENR_GPIODEN | RCC_AHBENR_DMA2EN;
17
18         /* GPIO Configuration */
19         GPIO_InitTypeDef GPIO_InitStructure;
20         GPIO_InitStructure.Mode = GPIO_MODE_ANALOG;
21         GPIO_InitStructure.Pin = GPIO_PIN_12 | GPIO_PIN_14;
22         GPIO_InitStructure.Pull = GPIO_NOPULL;
23         HAL_GPIO_Init( GPIOB, &GPIO_InitStructure );
24
25         GPIO_InitStructure.Mode = GPIO_MODE_ANALOG;
26         GPIO_InitStructure.Pin = GPIO_PIN_8 | GPIO_PIN_9 | GPIO_PIN_10 | GPIO_PIN_11;
27         GPIO_InitStructure.Pull = GPIO_NOPULL;
28         HAL_GPIO_Init( GPIOD, &GPIO_InitStructure );
29
30         //ADC4 DMA2
31         DMA2_Channel2->CNDTR = ADC4_BUFF_LEN;
32         DMA2_Channel2->CPAR = (uint32_t)&ADC4->DR; //ADC4's data register
33         DMA2_Channel2->CMAR = (uint32_t)adc4_val; //DMA_Memory_BaseAddr
34
35         DMA2_Channel2->CCR = DMA_CCR_PL_1 | DMA_CCR_MINC |DMA_CCR_CIRC | DMA_CCR_TEIE | \
36                              DMA_CCR_TCIE | DMA_CCR_PSIZE_0 | DMA_CCR_MSIZE_0 | DMA_CCR_EN;
37         //PL[1:0] bits(Channel Priority level)HIGH;Memory increment mode;
38         //Circular mode;Transfer error interrupt enable;
39         //PSIZE[1:0] bits (Peripheral size);MSIZE[1:0] bits (Memory size);
40         //Transfer complete interrupt enable;Transfer complete interrupt enable;Channel enable
41
42         ADC34_COMMON->CCR = ADC_CCR_CKMODE_1 | ADC_CCR_CKMODE_0;          //11: clk = HCLK/4 = 16MHz
43         if( ( ADC4->CR & ADC_CR_ADVREGEN_Msk ) == ADC_CR_ADVREGEN_1 ){
44                 //Voltage reference startup
45                 ADC4->CR = 0;
46                 ADC4->CR = ADC_CR_ADVREGEN_0;
47                 //Wait for startup
48                 uint16_t timeout = 1000, tick;
49                 while( ADC4->CR != ADC_CR_ADVREGEN_0 ){
```

```
50                      if( tick > timeout ){
51                              ADC4->CR = ADC_CR_ADVREGEN_1;
52                              return -1;
53                      }
54                      tick++;
55              }
56      } else if( ADC4->CR & ADC_CR_ADEN ) ADC4->CR |= ADC_CR_ADDIS;
57      //Calibration procedure
58      ADC4->CR |= ADC_CR_ADCAL;
59      //Wait for calibration
60      uint16_t timeout = 1000, tick;
61      while( ADC4->CR & ADC_CR_ADCAL ){
62              if( tick > timeout ){
63                      ADC4->CR = ADC_CR_ADVREGEN_1;
64                      return -2;
65              }
66              tick++;
67      }
68      ADC4->CR |= ADC_CR_ADEN;
69      //Wait for enable
70      timeout = 1000, tick = 0;
71      while( !( ADC4->ISR & ADC_ISR_ADRDY ) ){
72              if( tick > timeout ){
73                      ADC4->CR = ADC_CR_ADDIS;
74                      return -3;
75              }
76              tick++;
77      }
78      ADC4->CFGR = ADC_CFGR_EXTEN_0 | 0xE << ADC_CFGR_EXTSEL_Pos | ADC_CFGR_DMAEN | \
79                      ADC_CFGR_DISCEN ;
80                      //EXT14 TIM15_TRGO event;
81                      //RES[1:0]: Data resolution is 00: 12-bit; ADC DMA enable
82      ADC4->SMPR1 =  ADC_SMPR1_SMP3_1 |  ADC_SMPR1_SMP4_1 | \
83                              ADC_SMPR1_SMP7_1 |  ADC_SMPR1_SMP8_1 ;
84      //110: Tsmp = 181.5/16MHz = 11us for channel 3,4,7 and 8
85      ADC4->SMPR2 =  ADC_SMPR2_SMP12_1 | ADC_SMPR2_SMP13_1;
86      //110: Tsmp = 181.5/16MHz = 11us for channel 12 and 13
87
88      ADC4->SQR1 = ADC_SQR1_L_2 | ADC_SQR1_L_1 | ADC_SQR1_SQ1_0 | ADC_SQR1_SQ1_1 \
89                      | ADC_SQR1_SQ2_2 |ADC_SQR1_SQ3_2 | ADC_SQR1_SQ3_1 | ADC_SQR1_SQ3_0 \
90                      | ADC_SQR1_SQ4_3; // Regular sequence 6 channels, channel 3,4,7 and 8
91      ADC4->SQR2 = ADC_SQR2_SQ5_3 | ADC_SQR2_SQ5_2 | ADC_SQR2_SQ6_3 | ADC_SQR2_SQ6_2 \
```

```
92                          | ADC_SQR2_SQ6_0; // channel 12 and 13
93
94          NVIC_EnableIRQ( ADC4_IRQn );
95          NVIC_EnableIRQ( DMA2_Channel2_IRQn );
96          ADC4->CR |= ADC_CR_ADSTART;
97
98          return 0;
99  }
100
101
102 void ADC4_IRQHandler( void ){
103         GPIOE->BSRR = GPIO_BSRR_BS_10;
104         if(ADC4->ISR & ADC_ISR_AWD1) ADC4->ISR = ADC_ISR_AWD1;
105         else {
106         }
107         GPIOE->BSRR = GPIO_BSRR_BS_10;
108 }
109
110 void DMA2_Channel2_IRQHandler( void ){
111         GPIOE->BSRR = GPIO_BSRR_BS_11;
112 //      tx_flag_1 = SET;
113         if(DMA2->ISR & DMA_ISR_TCIF2) {
114                 DMA2->IFCR = DMA_IFCR_CGIF2 | DMA_IFCR_CHTIF2 | DMA_IFCR_CTCIF2;
115                 if(!tx_flag_1) tx_flag_1 = SET;
116         }
117         GPIOE->BSRR = GPIO_BSRR_BR_11;
118 }
```

D.3 General purpose input/output code

Listing D.3: "gpio.c"

```
1  /*
2   * gpio.c
3   *
4   *  Created on: May 10, 2021
5   *      Author: alihj
6   */
7
8  #include "gpio.h"
9
```

```
10 // Enabling the GPIO pins for troubleshooting
11 void gpio_init( void ){
12         /* Clock enable */
13         RCC->AHBENR |= RCC_AHBENR_GPIOEEN;
14         GPIO_InitTypeDef GPIO_InitStructure;
15         GPIO_InitStructure.Pin = GPIO_PIN_8 | GPIO_PIN_9 | GPIO_PIN_10 \
16                         | GPIO_PIN_11 | GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15;
17         GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
18         GPIO_InitStructure.Pull = GPIO_NOPULL;
19         HAL_GPIO_Init(GPIOE, &GPIO_InitStructure);
20 }
```

D.4 Timer code

Listing D.4: "tim2i.c"

```
1  /*
2   * tim2.c
3   *
4   *  Created on: May 26, 2021
5   *      Author: alihj
6   */
7
8  #include "tim2.h"
9
10 // Setting up MOSFET PWM
11 #define PWM_ELEMENTS 200
12
13 const uint16_t PWM_Buffer[PWM_ELEMENTS] = { // Sine Table
14   0,  3,  6,  9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 50, 53, 56,
15  58, 61, 63, 66, 68, 70, 72, 75, 77, 79, 80, 82, 84, 86, 87, 89, 90, 91, 92, 94,
16  95, 96, 96, 97, 98, 98, 99, 99, 99, 99,100, 99, 99, 99, 99, 98, 98, 97, 96, 96,
17  95, 94, 92, 91, 90, 89, 87, 86, 84, 82, 80, 79, 77, 75, 72, 70, 68, 66, 63, 61,
18  58, 56, 53, 50, 48, 45, 42, 39, 36, 33, 30, 27, 24, 21, 18, 15, 12,  9,  6,  3,
19   0,  3,  6,  9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 50, 53, 56,
20  58, 61, 63, 66, 68, 70, 72, 75, 77, 79, 80, 82, 84, 86, 87, 89, 90, 91, 92, 94,
21  95, 96, 96, 97, 98, 98, 99, 99, 99, 99,100, 99, 99, 99, 99, 98, 98, 97, 96, 96,
22  95, 94, 92, 91, 90, 89, 87, 86, 84, 82, 80, 79, 77, 75, 72, 70, 68, 66, 63, 61,
23  58, 56, 53, 50, 48, 45, 42, 39, 36, 33, 30, 27, 24, 21, 18, 15, 12,  9,  6,  3};
24
25  uint16_t PWM_BufferQ1[(PWM_ELEMENTS)];
```

```
26  uint16_t PWM_BufferQ2[(PWM_ELEMENTS)];
27  uint16_t PWM_BufferQ3[(PWM_ELEMENTS)];
28  uint16_t PWM_BufferQ4[(PWM_ELEMENTS)];
29
30 void tim2_init( void ){
31         // Setting up the duty cycle for each MOSFET
32         uint16_t i = 0;
33         for(i=0;i<100;i++)
34         {
35                 PWM_BufferQ1[i] = PWM_Buffer[i];
36                 PWM_BufferQ2[i] = 0;
37                 PWM_BufferQ3[i] = 100-PWM_Buffer[i];
38                 PWM_BufferQ4[i] = 100;
39         }
40         for(i=100;i<PWM_ELEMENTS;i++)
41         {
42                 PWM_BufferQ1[i] = 0;
43                 PWM_BufferQ2[i] = PWM_Buffer[i-100];
44                 PWM_BufferQ3[i] = 100;
45                 PWM_BufferQ4[i] = 100-PWM_Buffer[i-100];
46         }
47
48         /* Clock enable */
49         RCC->AHBENR |= RCC_AHBENR_GPIODEN;
50         RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;
51
52         /* GPIO Configuration */
53         GPIO_InitTypeDef GPIO_InitStruct;
54         GPIO_InitStruct.Pin = GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_7 | GPIO_PIN_6;
55         GPIO_InitStruct.Alternate = GPIO_AF2_TIM2;
56         GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
57         GPIO_InitStruct.Pull = GPIO_PULLDOWN;
58         HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
59
60         // Setting up TIM2 channels counting modes
61         // OC1M 0110: PWM mode 1 - In up counting, channel 1 is active
62         // as long as TIMx_CNT<TIMx_CCR1 else inactive.
63         TIM2->CCMR1 = TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1 | \
64                       TIM_CCMR1_OC2M_2 | TIM_CCMR1_OC2M_1;
65         TIM2->CCMR2 = TIM_CCMR2_OC4M_2 | TIM_CCMR2_OC4M_1 | \
66                       TIM_CCMR2_OC3M_2 | TIM_CCMR2_OC3M_1;
67         // Enabling output ch1 & ch2 & ch3 & ch4
```

```
68          TIM2->CCER = TIM_CCER_CC1E | TIM_CCER_CC2E | TIM_CCER_CC3E | TIM_CCER_CC4E;
69          TIM2->PSC = 63; // 1 MHz
70          TIM2->CCR1 = 0;
71          TIM2->CCR2 = 0;
72          TIM2->CCR3 = 0;
73          TIM2->CCR4 = 0;
74          TIM2->ARR = 100 - 1; // 10kHz
75
76          TIM2->DIER = TIM_DIER_UIE; //Enabling update event interrupt
77
78          NVIC_EnableIRQ( TIM2_IRQn );
79          TIM2->CR1 |= TIM_CR1_CEN;
80  }
81
82  void TIM2_IRQHandler(void)
83  {
84    static uint16_t index = 0;
85
86    if (TIM2->SR & TIM_SR_UIF)
87    {
88          GPIOE->BSRR = GPIO_BSRR_BR_13;
89      TIM2->SR=0;
90      TIM2->CCR1 = PWM_BufferQ1[index];
91      TIM2->CCR2 = PWM_BufferQ3[index];
92      TIM2->CCR3 = PWM_BufferQ2[index];
93      TIM2->CCR4 = PWM_BufferQ4[index];
94      index = (index + 1) % (PWM_ELEMENTS);
95      GPIOE->BSRR = GPIO_BSRR_BR_13;
96    }
97  }
```

Listing D.5: "tim15.c"

```
1  /*
2   * tim15.c
3   *
4   *  Created on: May 6, 2021
5   *      Author: alihj
6   */
7
8  #include "tim15.h"
9
```

```
10 // Setting up the timer as a trigger to ADC's
11 void tim15_init( void ){
12         /* Clock enable */
13         RCC->APB2ENR |= RCC_APB2ENR_TIM15EN;
14 
15         TIM15->CR1 = TIM_CR1_CEN;
16         TIM15->CR2 = TIM_CR2_MMS_1;
17         TIM15->PSC = 15;
18         TIM15->ARR = 7; //fpwm = fclk/div/(psc+1)/(arr+1)=32M/1/(15+1)/(7+1)=262k144
19 }
```

D.5 Universal synchronous/asynchronous receiver transmitter code

Listing D.6: "usart2.c"

```
1  /*
2   * usart2.c
3   *
4   *  Created on: May 2, 2021
5   *      Author: alihj
6   */
7 
8  #include "usart2.h"
9 
10 uint8_t Usart2TxBuff[USART2_TX_BUFF_LEN], Usart2RxBuff[USART2_RX_BUFF_LEN];
11 
12 //Initializing communication with PC over USB:
13 //Setting up DMA-transfer for sending and receiving of data
14 //Configuring the bandwidth rate (BRR) and Usart settings
15 
16 int8_t usart2_init( void ){
17         /* Clock enable */
18         RCC->APB1ENR |= RCC_APB1ENR_USART2EN;
19         RCC->AHBENR |= RCC_AHBENR_DMA1EN | RCC_AHBENR_GPIOAEN;
20 
21         /* GPIO Configuration */
22         GPIO_InitTypeDef GPIO_InitStruct;
23         GPIO_InitStruct.Pin = GPIO_PIN_2 | GPIO_PIN_3;
24         GPIO_InitStruct.Alternate = GPIO_AF7_USART2;
```

```
25          GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
26          GPIO_InitStruct.Pull = GPIO_NOPULL;
27          HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
28
29          //Rx DMA
30          DMA1_Channel6->CNDTR = USART2_RX_BUFF_LEN;
31          DMA1_Channel6->CPAR = (uint32_t)&(USART2->RDR);
32          DMA1_Channel6->CMAR = (uint32_t)Usart2RxBuff;
33          DMA1_Channel6->CCR = DMA_CCR_PL_1 | DMA_CCR_MINC | DMA_CCR_CIRC | DMA_CCR_TEIE | \
34                               DMA_CCR_TCIE | DMA_CCR_EN;
35
36          //Tx DMA
37          DMA1_Channel7->CPAR = (uint32_t)&(USART2->TDR);
38          DMA1_Channel7->CMAR = (uint32_t)Usart2TxBuff;
39          DMA1_Channel7->CCR = DMA_CCR_MINC | DMA_CCR_DIR | DMA_CCR_TCIE | DMA_CCR_TEIE;
40
41          //USART
42          USART2->BRR = 0x10; //16; //Bandwidth rate=32M/(0xF(16))=2Mb
43          USART2->CR3 = USART_CR3_DDRE | USART_CR3_DMAT | USART_CR3_DMAR | USART_CR3_EIE;
44          USART2->CR1 = /*USART_CR1_PCE | USART_CR1_PS |*//* USART_CR1_PEIE |*/ \
45                               USART_CR1_IDLEIE | USART_CR1_TE | USART_CR1_RE | USART_CR1_UE;
46
47          //Enable USART2, TX, RX
48          NVIC_EnableIRQ( USART2_IRQn );
49          NVIC_EnableIRQ( DMA1_Channel6_IRQn );
50          NVIC_EnableIRQ( DMA1_Channel7_IRQn );
51
52          return 0;
53  }
54
55  //Putting data in a vector that DMA-transfer sends automatically
56  int8_t usart2_tx( uint8_t const * const data, uint8_t const len ){
57          GPIOE->BSRR = GPIO_BSRR_BS_15;
58
59          if( len > USART2_TX_BUFF_LEN ) return -1;
60          DMA1_Channel7->CCR &= ~DMA_CCR_EN;
61          for(uint8_t i = 0; i < len; i++) Usart2TxBuff[i] = data[i];
62          DMA1_Channel7->CNDTR = len;
63          DMA1_Channel7->CCR |= DMA_CCR_EN;
64
65          GPIOE->BSRR = GPIO_BSRR_BR_15;
66          return 0;
```

```
67 }
68
69 uint8_t rx_index[2];
70 uint8_t data2[USART2_MAX_LEN];
71
72 //Copying the data from the DMA-vector and returning it
73 uint8_t* usart2_rx( void ){
74         GPIOE->BSRR = GPIO_BSRR_BS_14;
75         int32_t prim = __get_PRIMASK();
76         __disable_irq();
77
78         uint8_t len = (uint8_t)( rx_index[0] - rx_index[1] ) % (uint8_t)USART2_RX_BUFF_LEN;
79         data2[0] = len;
80         for( uint8_t i = 0; i < len; i++ ){
81                 data2[i + 1] = Usart2RxBuff[rx_index[1]];
82                 rx_index[1]++;
83         }
84
85         if( !prim ) __enable_irq();
86         GPIOE->BSRR = GPIO_BSRR_BR_14;
87         return data2;
88 }
89
90 extern FlagStatus rx_flag, tx_flag_1;
91
92 //Interrupt for setting flag for received data and errors
93 void USART2_IRQHandler( void ){
94 //       GPIOE->BSRR = GPIO_BSRR_BS_12;
95         if( USART2->ISR & ( USART_ISR_FE | USART_ISR_NE | USART_ISR_PE ) ){
96                 USART2->ICR = USART_ICR_NCF | USART_ICR_FECF | USART_ICR_PECF;
97 //              GPIOE->BSRR = GPIO_BSRR_BS_15;
98         }
99         //Making sure that some data is received and saving the data length
100         else {
101                 USART2->ICR = USART_ICR_IDLECF;
102                 rx_index[1] = rx_index[0];
103                 rx_index[0] = USART2_RX_BUFF_LEN - DMA1_Channel6->CNDTR;
104                 if( rx_index[0] != rx_index[1] ) rx_flag = SET;
105         }
106 //       GPIOE->BSRR = GPIO_BSRR_BR_12;
107 }
108
```

```
109 //Interrupt for resetting DMA errors for sending and receiving
110 void DMA1_Channel6_IRQHandler( void ){
111 //      GPIOE->BSRR = GPIO_BSRR_BS_13;
112         if(DMA1->ISR & DMA_ISR_TEIF6) {
113                 DMA1->IFCR = DMA_IFCR_CTEIF6;
114         }
115         DMA1->IFCR = DMA_IFCR_CGIF6 | DMA_IFCR_CHTIF6 | DMA_IFCR_CTCIF6;
116 //      GPIOE->BSRR = GPIO_BSRR_BR_13;
117 }
118 void DMA1_Channel7_IRQHandler( void ){
119 //      GPIOE->BSRR = GPIO_BSRR_BS_13;
120         if(DMA1->ISR & DMA_ISR_TCIF7) {
121
122                 DMA1->IFCR = DMA_IFCR_CGIF7 | DMA_IFCR_CHTIF7 | DMA_IFCR_CTCIF7;
123                 ADC4->CR |= ADC_CR_ADSTART; // starting the ADC conversation when the DMA
124                 //is done transferring the data to the USART
125         }
126         DMA1->IFCR = DMA_IFCR_CTEIF7;
127 //      GPIOE->BSRR = GPIO_BSRR_BR_13;
128 }
```