

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Hub USB with individual port control

Francisco Da Silva Tavares



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Vítor Manuel Grade Tavares

October 31, 2024

Resumo

Esta dissertação gira em torno do desenvolvimento de uma solução de controlo centralizada para múltiplas portas USB no âmbito de engenharia de teste. Os desafios existentes na gestão do fornecimento de energia e o fluxo de dados para portas individuais USB levaram à necessidade de desenvolver um sistema eletrónico que servirá como principal mecanismo de controlo destas portas. O objetivo principal passa por eliminar a necessidade de HUB's adicionais, oferecendo uma solução mais eficiente para equipamentos de teste. A solução apresentada permitirá uma independente alternância de energia de cada porta, facilitando a funcionalidade de ligar/desligar e regular o fluxo de dados. Esta inovação não apenas agiliza o processo de teste como também reduz a dependência de componentes externos, contribuindo para uma maior eficácia geral dos equipamentos de teste. Esta dissertação também enfatiza a adaptabilidade da solução, fornecendo uma interface com o utilizador para controlo das portas USB através de LabView ou Python.

Abstract

This dissertation revolves around the development of a centralized control solution for multiple USB ports within the test engineering domain. The existing challenges in managing power supply and data flow for individual USB ports prompted the creation of an electronic system that serves as the core control mechanism. The primary objective is to eliminate the need for additional hubs, offering a more integrated and efficient solution for test fixtures. The solution presented enables independent power toggling for each port, facilitating ON/OFF functionality, and regulates data flow. This innovation not only streamlines the testing process but also reduces dependence on external components, contributing to overall efficiency in test fixtures. The dissertation also emphasizes the adaptability of the solution, providing a user-friendly interface for USB port control through either Python or LabVIEW.

United Nations Sustainable Development Goals

Bosch adheres to the ROHS directive, which seeks to mitigate the risks to human health and the environment associated with the disposal of electronic and electrical waste. This directive achieves its goals by limiting the use of specific hazardous substances in electrical and electronic equipment (EEE), advocating for their replacement with safer alternatives. The restricted substances typically include heavy metals, flame retardants, and plasticizers.

This dissertation supports the Sustainable Development Goals (SDGs) as outlined in Table 1.

Table 1: Project Contribution to Sustainable Development Goals

SDG	Target	Contribution	Key Performance Indicators and Metrics
15	5	In the development of this project, equipment adhering to the ROHS directive was used to mitigate the risks to human health and the environment associated with the disposal of electronic and electrical waste	• Reduction in hazardous waste produced • Percentage of equipment compliant with ROHS directive • Decrease in environmental contamination levels • Number of training sessions on environmental safety conducted

Agradecimentos

Este projeto marca o fim de uma jornada de cinco anos, uma jornada pela qual daria tudo para repetir ou ter a oportunidade de prolongar por mais cinco anos. Ao longo deste percurso, aprendi muito, vivi momentos bons e outros menos bons, fiz amizades e fui acompanhado por pessoas especiais que estiveram ao meu lado mesmo quando tudo parecia atribulado.

Quero expressar minha gratidão, especialmente à minha família, que sempre me forneceu o apoio essencial para concluir este trajeto. Aos meus colegas do departamento do MFE da Bosch de Ovar, agradeço por todos os momentos de lazer e gargalhadas partilhadas, bem como pela ajuda oferecida. Ao meu supervisor, Eng. Luís Lisboa, agradeço pelo apoio prestado neste projeto e pelos conselhos fornecidos que se revelaram importantes neste percurso final.

Ao meu orientador, Prof. Vítor Tavares, deixo meu sincero agradecimento por ter aceite este desafio, por ter concordado em ser meu orientador, por sempre estar disponível quando sua ajuda era essencial e por acreditar em mim.

A todos, um muito obrigado não é suficiente!

Francisco Da Silva Tavares

*“Without commitment, you’ll never start,
but more importantly,
without consistency, you’ll never finish.”*

Denzel Washington

Contents

1	Introdução	1
1.1	Context	1
1.2	Motivation and Goals	1
1.3	Problem and Proposed Solution	2
1.4	Dissertation Structure	3
2	Background and Related Work	4
2.1	Printed Circuit Boards	4
2.2	USB	7
2.2.1	USB 2.0	8
2.2.2	USB 3.0	11
2.2.3	HUB USB	14
2.3	Related Work	15
2.4	Present Contexted in the Company	17
3	Hub Architecture and Design	19
3.1	System architecture	19
3.1.1	Block diagram	19
3.1.2	Eletronic Components Selection	22
3.2	System Design	26
3.2.1	First Layer Data Processing	26
3.2.2	USB 3.0 Data Distribution	33
3.2.3	USB3.0 Connectors	36
3.2.4	USB 2.0 Data Distribution	40
3.2.5	USB2.0 Connectors	42
3.2.6	RS232	44
3.2.7	Power Supply	44
3.3	PCB Design	46
3.3.1	Altium Designer	46
3.3.2	Stackup Layer	46
3.3.3	Defined Differential Impedance	51
3.3.4	Components Placement	53
3.3.5	Routing	55
3.3.6	Polygon Pours	58
3.3.7	Stitching and Shielding Vias	60
3.3.8	Order PCB and Components	62
3.4	Reworking to Overcome Difficulties	63

4	Firmware	66
4.1	Thonny	66
4.2	Code Explanation	66
5	Results and Performance Analysis	72
5.1	USB2.0	72
5.2	USB3.0	73
5.3	RS232	75
6	Conclusions	77
6.1	Future Work	77
	References	79
A	Appendix	82

List of Figures

1.1	Block diagram of the proposed electronic system	3
2.1	Example stack-up drawing	5
2.2	Example stack-up drawing	6
2.3	Example of USB network in industrial context	8
2.4	Scheme of the proposed USB host	8
2.5	USB endpoints between USB Host and devices	9
2.6	USB cable	10
2.7	Example of NRZI encoding	10
2.8	USB3.0 Topology	13
2.9	USB3.0 Cable	13
2.10	Communication layers	14
2.11	Proposed Embedded USB Homecare Internet System	15
2.12	Connection between the USB HUB and PC	16
3.1	Block diagram with the daisy chain configuration	20
3.2	Block diagram with the cascade chain configuration	21
3.3	Advanced block diagram	22
3.4	Final block diagram	26
3.5	TUSB8041 Circuitry and their connections	27
3.6	TUSB8041 Circuitry	28
3.7	TUSB8041 Upstream Circuitry	31
3.8	TUSB8041 Crystal Circuitry	32
3.9	TUSB8020 Circuitry and their connections	33
3.10	TUSB8020 Circuitry	34
3.11	DIP Switch used in TUSB8020	35
3.12	USB3.0 Connectors	36
3.13	Power Switch	37
3.14	Current Limit Threshold	38
3.15	BJT transistors circuitry	39
3.16	USB3.0 Ports Circuitry	40
3.17	TUSB4041 Circuitry and their connections	40
3.18	TUSB4041 Circuitry	41
3.19	DIP Switches used in TUSB4041	41
3.20	USB2.0 Connectors	43
3.21	USB2.0 Connector Circuitry	43
3.22	RS232 Circuitry	44
3.23	Power Supply Circuitry	45

3.24	Multilayer Symmetry	48
3.25	Possible Board Stack-up on a Six-Layer PCB	48
3.26	EuroCircuits Buildup Editor	49
3.27	Stackup defined in Altium Designer	50
3.28	Stackup defined in Altium Designer with description	50
3.29	Differential signaling on a differential-line pair	51
3.30	PCB Design Classification Overview	52
3.31	Differential Impedance parameters in Altium Designer	53
3.32	Message when the target differential impedance is not met	53
3.33	Components Placement on the top layer	54
3.34	Components Placement on the bottom layer	55
3.35	Serpentine Routing	56
3.36	Return Path for differential-line pair Vias with Ground Vias	57
3.37	Ground Vias in differential-line pairs	57
3.38	Polygon Pour in Layer 4	59
3.39	Polygon Pour in Layer 1	59
3.40	Stitching Vias	60
3.41	Shielding Vias	61
3.42	Shitching Vias in IC's grounds	61
3.43	3D View of the PCB	62
3.44	Final PCB	63
3.45	Heat-resistant tape covering the vias	64
3.46	USB extension with data signals exchanged	65
4.1	Firmware Logic	67
4.2	Pin Definition Code	67
4.3	Pin Control and Command Processing	68
4.4	Serial Command Sender for USB Port Control	69
4.5	Serial Port Command Interface	70
4.6	Example of Commands	71
5.1	Pen 2.0 Connected to Computer	72
5.2	Pen 2.0 Connected to the Board	73
5.3	SSD Disk Connected to the Computer	73
5.4	SSD Disk Connected to the Board	74
5.5	SSD Disk Connected to the Board	74
5.6	Putty Settings	75
5.7	Signal Analysis and ASCII Decoding of Transmitted Commands	76

List of Tables

1	Project Contribution to Sustainable Development Goals	iii
2.1	Transfer speeds	12
3.1	Comparison between the different USB HUB based on [11]	24

Abreviaturas e Símbolos

PCB	Printed Circuit Board
SMT	Surface Mount Technology
THT	Through Hole Technology
USB	Universal Serial Bus
I2C	Inter-Integrated Circuit
PIO	Programmable I/O
ICT	In Circuit Test
DUT	Device Under Test
TCK	Test Clock Input
TMS	Test Model Input
TDI	Test Data Input
TDO	Test Data Output
TRST	Test Reset
SCLK	Serial Clock
SS	Slave select
SDA	Serial Data
SCL	Serial Clock
SS	Super Speed
HS	High Speed
FS	Full Speed
GPIO	General Purpose Interface Bus
USB-IF	USB Implementers Forum
YKUSH	YepKit USB Switchable HUB

Chapter 1

Introdução

1.1 Context

The test engineering team at Bosch focuses on creating Test Fixtures by utilizing both internally designed hardware and software, as well as standard equipment available on the market. This approach ensures that the solutions developed are both fast and competitive, meeting the high standards expected by the industry.

A critical requirement for approximately 90% of these fixtures is the ability to switch USB devices on and off. This includes a variety of USB-connected equipment such as USB relay boards, programmers, and debug boards. The capability to control the power state of these devices is essential for efficient testing and development processes, allowing engineers to simulate various operational conditions, conduct thorough diagnostics, and ensure the reliability and performance of the systems under test.

This requirement highlights the importance of versatile and robust USB hub solutions in the overall testing framework. The USB hubs must be capable of handling multiple devices simultaneously, offering precise control over each connected device. This functionality is crucial for the successful execution of complex testing protocols and for maintaining the integrity of the testing environment.

1.2 Motivation and Goals

Bosch Security Systems, a global leader in innovative technology solutions, has proposed a dissertation project that aims to create a controllable USB Hub. Bosch Security Systems focuses on the production of innovative solutions for security and communication systems, fire alarms, electronic displays, and other products for various business units within the Bosch Group. While production is the main activity of this unit, there is also a strong emphasis on development and research.

The actual solution used in the company involves utilizing a system available in the market, YKUSH that stands for Yepkit USB Switchable Hub, that can switch individual devices on or off. However, this system has a significant limitation: it is equipped with only three downstream

ports, which is insufficient for the requirements of the engineering industry. To overcome this constraint, a standard USB hub is integrated within the test fixtures, thereby expanding the number of available ports and enabling control over the power supply to multiple downstream devices.

While this setup achieves the desired functionality, it introduces several disadvantages. The main issue is the lack of centralized control, which makes the system less efficient. Additionally, the extra equipment takes up more space compared to a single, centralized solution. Moreover, this configuration only supports USB communication and is limited to USB 2.0, which restricts its speed and functionality in comparison to more advanced standards like USB 3.0.

The focus of this work lies within the realm of electronic design, specifically in the development of an electronic system that includes the design of a Printed Circuit Board, PCB, and its firmware to exert control over multiple USB ports. The aim is to enhance the functionality of USB ports by providing the ability to manage power supply and control data flow.

The goal of this system is to develop a solution capable of handling both USB 3.0 and USB 2.0 connections, this communication supports speeds up to 5Gbps and 480Mbps respectively, while also supporting the conversion from USB to RS232. The system will feature a total of seven downstream ports for communication: two for USB 3.0, four for USB 2.0, and one for RS232. Besides this ports for communication, two ports USB Type-C will be totally dedicated to power supply.

1.3 Problem and Proposed Solution

As previously mentioned, the limitations of the current solution used in the company include the restricted number of downstream ports, support only for USB 2.0 communication, lack of centralized control and no ability to manage the power of each downstream device. These limitations highlight the need for this project.

The primary objective of this work is to develop a singular device capable of efficiently controlling multiple downstream devices and its power. A key focus is to achieve this without the necessity of additional equipment such as hubs. The aim is to provide users with a versatile solution that can seamlessly manage diverse types of USB ports, addressing the limitations associated with existing options and offering a more streamlined and versatile USB device control system.

The envisioned USB hub goes beyond the typical functionality of expanding USB ports; this electronic system will be designed with the capability of individual control through the USB host, allowing for the switching on or off each port. This USB hub is intended to possess the ability to manage various USB ports, including USB2.0, USB3.0 and RS232 communication.

Specifically tailored for testing scenarios in a production line, the USB hub will be equipped with a microcontroller, ensuring precise control of the power of each downstream USB port. To have this control over the USB ports, a software will have to be developed in order to control the board. With this, the capability to switch on/off the USB devices will be done by the USB host, in this case, a computer.

With this project, the test engineering team will save space inside of the fixtures and will speed up the development and will be able to include in the software used for testing the control of the USB ports instead of calling a command line.

The ability to control the power of each port is a critical feature, especially in the testing industry, where numerous devices are connected to production lines. This functionality not only simplifies device management but also contributes to energy savings, making the system more efficient for industrial applications.

Figure 1.1 illustrates the concept of the system to be developed. It includes one upstream port for connection to the host and seven downstream ports: two for USB 3.0, four for USB 2.0, and one for RS232. The black lines indicate the communication flow, while the red lines represent the power control managed by the microcontroller.

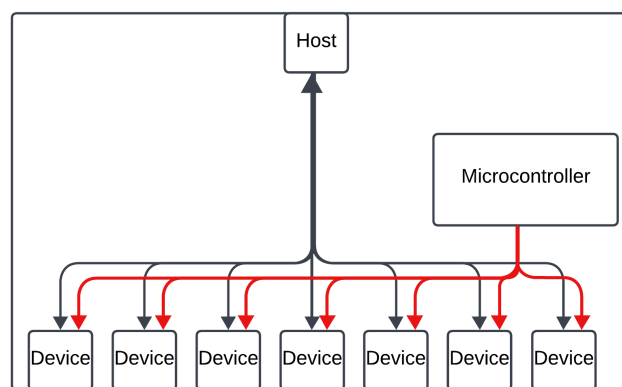


Figure 1.1: Block diagram of the proposed electronic system

1.4 Dissertation Structure

This document is organized to provide a thorough exploration of the topic. It is divided into six main chapters. Chapter 2 examines the fundamental theories and concepts that form the basis of the research, related work and the present context that is used in the company. Chapter 3 elaborates on the methodology and methods used throughout the project, including a detailed analysis of the circuit and the design of the electronic system. Also describes the difficulties that were found and the strategies used to overcome them. Chapter 4 provides an explanation of the firmware developed, followed by Chapter 5, which analyzes the results obtained from this work. Finally, Chapter 6 presents the conclusion and suggests future work that could improve this project.

Chapter 2

Background and Related Work

This chapter examines the essential concepts and advancements in USB hub technology, emphasizing the design and development of PCBs and firmware. It traces the history and importance of these technologies in improving device connectivity and details advanced methods for PCB design and USB hub creation, highlighting their practical uses.

The first section, 2.1, provides a comprehensive overview of printed circuit boards and its historical evolution. In the following section, 2.2, the exploration of USB becomes a focal point with a detailed examination of USB 2.0 and USB 3.0. The discussion on USB extends to include the USB hub, examining its functionality and relevance within the broader context of USB communication. This chapter serves as a foundation for a nuanced understanding of the technological landscape, setting the stage for subsequent chapters in the report.

2.1 Printed Circuit Boards

A printed circuit board functions as an electronic platform, employing copper conductors to establish electrical connections among various components. Essentially, PCBs offer structural support to electronic elements, allowing them to be housed within an enclosure. The design process for a printed circuit board involves a series of steps aligned with manufacturing procedures, integrated circuit packaging, and the configuration of the bare circuit board.

According to Yilmaz [46], the origin of Printed Circuit Boards (PCBs) traces back to the latter part of the 20th century's first quarter. The foundational concepts for PCB technology were rooted in a patent obtained by Cesar Parolini in 1925. However, the blueprint for the contemporary PCB that we recognize today was conceived by Dr. Paul Eisler in 1930 and officially patented in 1943, according with [46]. Initial practical applications of this innovative concept were deferred until the conclusion of World War II. During this post-war period, PCBs found their inaugural use in the United States for constructing sophisticated military equipment, including radars and missile controllers.

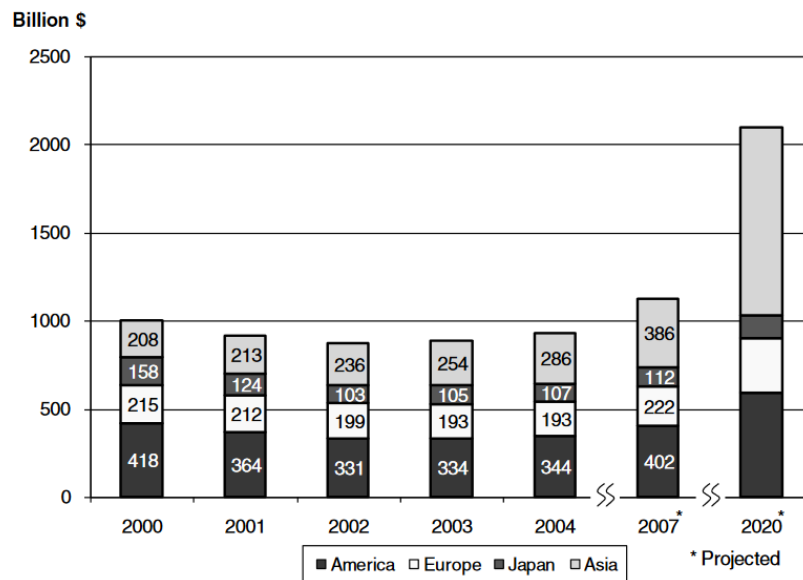


Figure 2.1: Example stack-up drawing, from [46]

A Printed Circuit Board (PCB) serves as a substrate made from paper or glass fabric infused with a resin, typically epoxy, phenolic, or silicone. It comprises one or more layers of metal conductors and insulating material, facilitating the electronic interconnection and mechanical support of electronic components. The simplest iteration is the single-layer, single-sided board, featuring metalized conductors on only one side. Increased complexity and component density are attainable through the production of double-sided and multi-layered boards. In a double-sided configuration, components are assembled on both sides of the substrate [46].

Electronic components are placed onto Printed Circuit Boards and connected according to the designed circuit. There are two primary categories of PCB assembly technologies: SMT and THT.

THT stands for through-hole technology. In this traditional PCB assembly method, the component leads are inserted through the holes, and their extending ends are subsequently cut and soldered.

Surface Mount Technology (SMT) is a contemporary approach to affixing components onto circuit boards, supplanting the traditional through-hole technology for various advantages it provides. In SMT, electronic components are directly mounted onto the surface of the PCB. Automation is a key feature of this technology, employing pick and place machines for the precise positioning of electronic components on the boards [34].

The Figure 2.2 shows both technologies that were listed above.

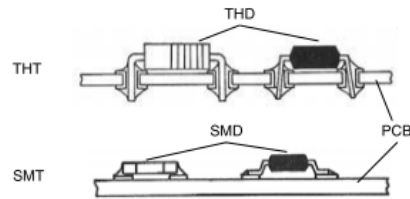


Figure 2.2: Example stack-up drawing, from [46]

Surface Mount Technology presents both advantages and disadvantages in the realm of PCB manufacturing.

On the positive side, SMT enables a more streamlined design for PCBs. Manufacturers can densely pack electronic components onto the circuit board, resulting in a compact and lightweight design preference in the industry. Moreover, SMT is well-suited for high-frequency applications, allowing for increased circuit speeds.

According to the information present in this paper [34], one of the notable benefits of SMT lies in achieving higher component density. This is attributed to the smaller size of electronic components and the absence of drilling mounting holes. SMT efficiently utilizes both sides of the circuit, maximizing available space for component placement.

SMT's contribution to efficient PCB assembly is significant. Leveraging pick and place machines, the technology simplifies and expedites the assembly process. Some of these machines boast an impressive capacity, placing over 136,000 components every hour. Additionally, SMT allows for selective soldering, enabling manufacturers to customize the soldering process for each component.

From a cost perspective, SMT proves to be economical for PCB manufacturers. The components used in SMT are generally more affordable than their through-hole counterparts, making it a budget-friendly option in the manufacturing landscape.

However, SMT is not without its challenges. Repairing SMT components can be intricate due to their small lead size. Additionally, SMT is less suitable for components generating significant heat, as high temperatures can cause, in some cases, solder melting.

Moreover, the SMT process demands skilled operators and expensive automated equipment, posing a potential barrier for some manufacturers. Another concern revolves around the reliability of solder joints, as less solder may compromise their durability—a consideration that weighs on PCB assemblers' minds.

In weighing the advantages and disadvantages, SMT emerges as a technology with significant benefits in terms of size efficiency, assembly speed, and cost-effectiveness. However, careful consideration is required to address challenges related to repair complexity, heat sensitivity, and the demand for skilled operators and specialized equipment.

THT, having been in use before SMT, has its own sets of merits and demerits.

THT boasts a robust mechanical bond, rendering its assemblies resilient in high-stress conditions, whether mechanical or electrical. This durability is particularly advantageous in applications

frequently exposed to stress. The components' ability to withstand wear and tear, thanks to solder joints spanning the board's thickness, further underscores its mechanical strength.

In the realm of fast prototyping, THT takes some advantages. The ease with which THT components can be swapped out makes them ideal for rapid iterations and quick assembly during the early stages of development. This simplicity contributes to the efficiency of prototyping processes.

However, THT has some disadvantages. The requirement for drilling holes comes with increased production costs and extended production times. This drilling process is a notable drawback that impacts the economic feasibility of THT. Moreover, in multilayer circuit configurations, THT introduces limitations. The drilled holes must traverse every layer of the PCB, constraining the available routing clearance. This limitation can affect the flexibility of design in multilayered circuits.

In conclusion, THT mechanical robustness and suitability for prototyping stand out positively. Yet, the drawbacks, such as drilling requirements and limitations in multilayer circuits, underscore the need for careful consideration based on specific manufacturing and design requirements.

2.2 USB

The Universal Serial Bus (USB) stands out as a ubiquitous and transformative innovation in industrial and personal consumer applications. Its widespread adoption in recent years has facilitated the seamless connection of various peripherals, ranging from keyboards and mouse devices to printers, webcams, and more. The simplicity, "plug and play" features, and "hot plug" support of USB contribute to its success, along with the added advantage of supplying power to connected devices.

Beyond the realm of personal computing, USB has found applications in diverse fields, including measurements. Its differential transmission type ensures good noise immunity, and the convenience of not requiring additional power supplies for peripherals is particularly advantageous in both industrial and consumer projects. The cost-effectiveness brought about by the widespread use of USB has further solidified its place in the realm of connectors.

Since its inception in the 1990s, according with [9], USB has evolved into a standard protocol for connecting computers with peripheral devices. From mice, keyboards, and external disk drives to smartphones, tablets, and wearable devices, USB connectors have become the go-to choice for battery charging and data transfer. Over the years, the USB protocol has replaced parallel and serial ports, establishing itself as the preferred method for interconnecting peripheral devices with computer hosts [9].

As the protocol versions advance and develop, USB has achieved impressive data rates, reaching up to 10 Gbit/s and 20 Gbit/s in USB 3.1 and USB 3.2, respectively, based on [9]. This continual evolution underscores USB's role as a dynamic and high-performance connectivity solution.

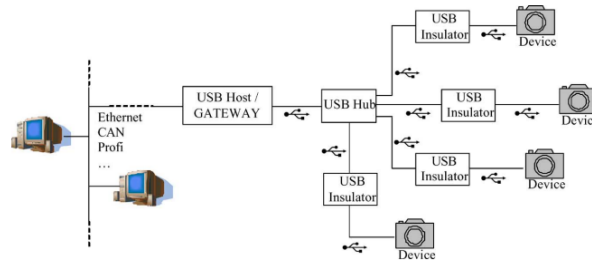


Figure 2.3: Example of USB network in industrial context, from [9]

In the example shown in Figure 2.3 we have an USB Host making the bridge between Ethernet and USB, and for that, it was created a USB Host solution that consists of a microcontroller-based system [9].

Figure 2.4 illustrates an example of a USB hub where an 8-bit microchip microcontroller is employed to manage both the interaction with the sensor network and the connection to the high-level network. In this setup, the microcontroller facilitates communication between the various connected devices.

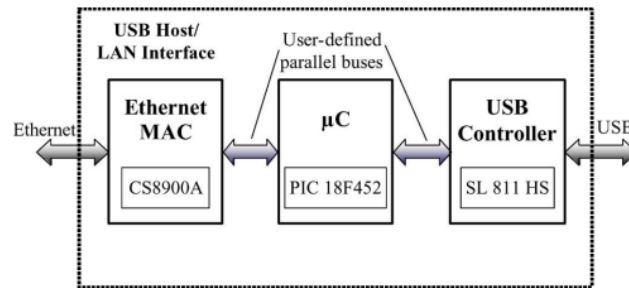


Figure 2.4: Scheme of the proposed USB host, from [9]

2.2.1 USB 2.0

USB, a specification designed for communication between devices and a host controller, is a key element in modern computer systems [32]. The USB host controller, integrated into computers, serves to enable high-speed USB functionality across ports and manages data traffic to various devices. Its crucial role is highlighted by the fact that no USB device can transfer data on the bus without explicit instructions from the host controller.

The host controller ensures efficient communication by systematically polling the bus, determining the bandwidth based on the slowest connected device. USB-compliant devices feature endpoints, acting as sources or sinks of data, as presented in Figure 2.5. These endpoints, positioned at the terminus of the communication channel, establish the interface between the device's hardware and its firmware [32].

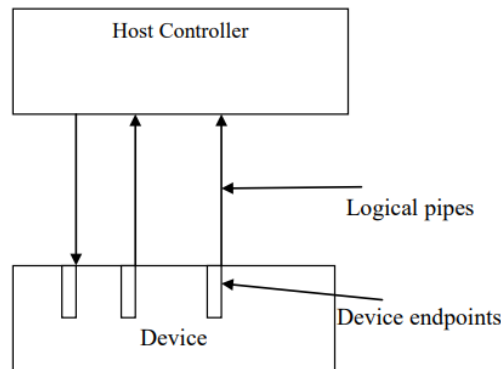


Figure 2.5: USB endpoints between USB Host and devices, from [32]

Operating on a polled bus system, the USB bus protocol involves the Host Controller initiating all data transfers. Transactions on the bus typically consist of up to three packets. The process starts with the Host Controller sending a scheduled USB packet, known as the "token packet," providing transaction details, device address, and endpoint number. The addressed USB device decodes the address fields to select itself [33].

In the course of a transaction, data is moved either from the host to a device or vice versa, as indicated in the token packet. The sender either sends a data packet or signals that there is no data to transfer, and the receiver replies with a handshake packet to communicate the success or failure of the transfer [33].

The USB data transfer model describes the communication pathway between a source or destination on the host and an endpoint on a device as a "pipe." These pipes come in two types: stream and message. Stream data does not follow a USB-defined structure, while message data adheres to one [33].

The USB system facilitates both signal and power transmission through a four-wire cable, as illustrated in Figure 2.6. Signaling activities are distributed over two wires within each point-to-point segment. This system supports three varying data rates: high-speed signaling at 480 Mb/s, full-speed signaling at 12 Mb/s, and a low-speed signaling mode with limited capabilities, operating at 1.5 Mb/s.

USB 2.0 host controllers and hubs come with functionalities that allow the transfer of full-speed and low-speed data at elevated rates between the host controller and the hub. However, when it comes to communication between the hub and the device, the data is transmitted at either full-speed or low-speed. This functionality is designed to mitigate the impact that full-speed and low-speed devices may have on the available bandwidth for high-speed devices [33].

The clock is sent alongside the differential data using Non-Return-to-Zero Inverted (NRZI) encoding, incorporating bit stuffing to guarantee sufficient transitions, Figure 2.7.

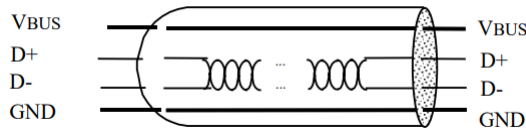


Figure 2.6: USB cable, from [33]

A SYNC field comes before each packet to enable the receiver(s) to synchronize their bit recovery clocks [33]. The USB cable consists of VBUS and GND wires for delivering power to devices. VBUS, which is nominally +5 V at the source, provides the power needed. USB allows flexibility in cable lengths by selecting an appropriate conductor gauge. This choice is based on factors like the specified Intermediate Resistance, IR , drop (voltage loss) and other considerations such as the device's power requirements and the cable's flexibility.

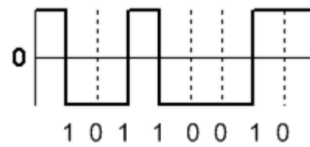


Figure 2.7: Example of NRZI encoding, from [44]

In the USB 2.0 specification, there are four specified transfer types: control, bulk, interrupt, and isochronous. Among these, interrupt and isochronous transfers are considered synchronous, while control and bulk transfers are categorized as asynchronous.

Synchronous transfers operate on a fixed service interval, being polled at a set rate, typically no more than once per microframe (125 microseconds). In contrast, asynchronous transfers lack a fixed polling rate. However, many device controllers in the market are frequently polled at a high rate, often several times per microframe. This approach aims to enhance overall device throughput and responsiveness [15].

2.2.1.1 Control Transfers

Utilized for device configuration during attachment and for other device-specific purposes, control transfers play a crucial role. These transfers are employed by the USB System Software to configure devices upon initial attachment. Additionally, other driver software has the flexibility to employ control transfers in ways specific to their implementations. It's essential to note that control transfers ensure lossless data delivery [33].

2.2.1.2 Bulk Transfers

Bulk data, often generated or consumed in substantial and bursty quantities, presents wide dynamic latitude within transmission constraints. This type of data, commonly associated with printers or scanners, is characterized by its sequential nature. The hardware-level assurance of reliable

data exchange is achieved through error detection mechanisms in hardware, complemented by a limited number of retries also managed at the hardware level. It's important to note that the bandwidth occupied by bulk data can vary based on concurrent bus activities [33].

2.2.1.3 Interrupt Transfers

Designed for the prompt and reliable delivery of data, interrupt transfers are utilized for time-sensitive applications requiring immediate feedback or echo responses, such as characters or coordinates. These transfers ensure limited-latency communication to or from a device. The USB delivers interrupt data at a rate specified by the device, allowing the device to present this data for transfer at any time. Typically, interrupt data includes event notifications, characters, or coordinates organized in one or more bytes. For instance, the coordinates from a pointing device are a common example of interrupt data. While no explicit timing rate is mandated, interactive data often has specific response time requirements supported by the USB [33].

2.2.1.4 Isochronous Transfers

Isocronous data in USB communication occupies a predetermined amount of bandwidth with a prearranged delivery latency. This type of data is characterized by continuous, real-time creation, delivery, and consumption. The consistent rate at which isochronous data is both received and transferred implies timing-related information, necessitating its delivery at the received rate to maintain synchronization. Alongside delivery rate considerations, isochronous data may also exhibit sensitivity to delivery delays. Bandwidth requirements for isochronous pipes are typically determined by the sampling characteristics associated with the specific function. The latency required is directly linked to the buffering available at each endpoint [33].

2.2.2 USB 3.0

The USB 3.0 specifications, unveiled in November 2008, introduced notable enhancements. USB 3.0 boasts an elevated data transfer rate of up to 5 Gbps, reduced power consumption, increased power output, and a crucial feature—backward compatibility with USB 2.0. Additionally, USB 3.0 incorporates a higher-speed bus called SuperSpeed, operating in parallel with the existing USB 2.0 bus [44].

This are the main differences between USB 2.0 and USB 3.0 [44]:

- Speed: The SuperSpeed USB (SS) from USB3.0 is over 10x larger than USB 2.0 High-Speed (HS) (5Gbps vs 480Mbps)
- Physical Layer: The electrical signaling in the physical layer has undergone a transformation, transitioning from the traditional two-wire system to a dual-simplex data path. This

modification enables packets to arrive and depart simultaneously. The implementation involves introducing a new set of connections alongside the existing USB 2.0 two-wire interface, which remains unchanged.

- **Polling:** a mechanism in USB 2.0 where the host consistently checks connected peripheral devices for data transfer readiness, has been replaced with asynchronous notification in USB 3.0. Unlike USB 2.0, where all devices must be continuously "on" for polling, SuperSpeed USB adopts a more efficient approach. In USB 3.0, the host remains idle until a higher-level application signals that a specific peripheral has data ready for transfer. The host then contacts the identified peripheral directly to confirm the data is readiness, facilitating the transfer once both ends of the link are prepared.
- **Power:** USB 3.0 devices benefit from increased power availability compared to USB 2.0. The power supply is elevated from 5V @500mA to 5V @900mA, providing enhanced power for USB 3.0 devices. Additionally, the unconfigured current limit is raised from 100mA to 150mA in USB 3.0.

The increased data transfer capacity offered by USB SuperSpeed (SS) transactions can be advantageous for applications such as real-time audio and video streaming, particularly those that demand higher bus bandwidth at regular intervals [5].

In the Table 2.1, it is possible to analyse the difference of transfer speeds between the three versions of USB:

Table 2.1: Transfer speeds, from [5]

	SD Movie - 6GB	USB Flash 16GB	HD Movie - 25GB
USB 1.0 (FS)	~2 hours	~6 hours	~9.25 hours
USB 2.0 (HS)	~3.25 minutes	~9 minutes	~14 minutes
USB 3.0 (SS)	~20 seconds	~54 seconds	~70 seconds

Illustrated in Figure 2.8 is a USB 3.0 topology example. Notably, this configuration facilitates the inclusion of all wired USB speeds (Low Speed, Full Speed, High Speed, and SuperSpeed), achieved through the integration of two distinct buses in USB 3.0 cables, connectors, and hubs. The diagram highlights the SuperSpeed (SS) bus depicted in red, comprising two differential signal pairs—one for transmitting packets and the other for receiving. In contrast, the standard USB 2.0 bus utilizes a single differential pair functioning in a half-duplex mode [5].

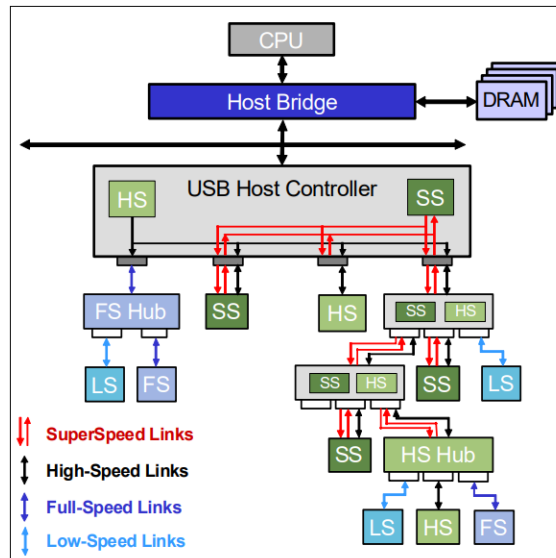


Figure 2.8: USB3.0 Topology, from [5]

In Figure 2.9, the internal structure of a USB 3.0 cable is shown, highlighting the SuperSpeed (SS) and USB 2.0 buses, as well as the VBUS power pin delivering power at 5 volts direct current (vdc) with a capacity of up to 900mA.

The SS bus adopts a dual-simplex method, enabling simultaneous transmission and reception of data packets. There are scenarios in which an SS device engages in both transmitting and receiving data concurrently. For instance, during burst transactions, a device might receive data from the host while simultaneously sending acknowledgments related to previously received data.

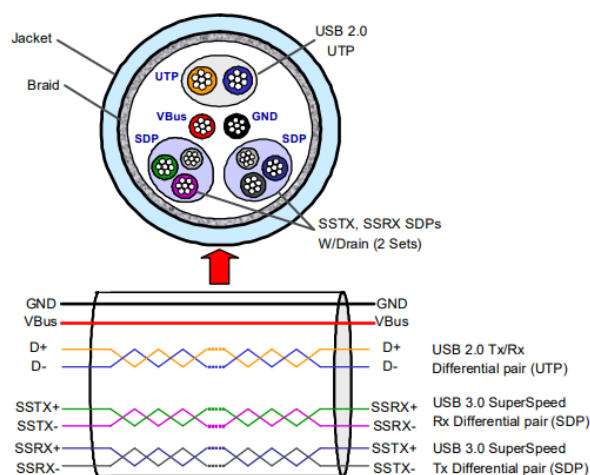


Figure 2.9: USB3.0 Cable, from [5]

The USB 3.0 communication model comprises three layers: the physical layer, the link layer, and the protocol layer. The physical layer manages low-level PCI express and signaling, including byte scrambling, encoding, and serialization for SuperSpeed (SS) transmission. The link layer facilitates successful link transfers through 12 link states, operating under the control of the protocol layer. It handles tasks like packet framing, CRC generation and verification, as well as managing Link Training, link flow control, error recovery, and other maintenance activities. The protocol layer defines communication rules between the host and device, and software organizes transactions as sequences involving token, data, and handshake packets. Communication within the USB 3.0 model is illustrated in Figure 2.9, where all interactions occur directly on the link without the implication of an interface between layers. Application and driver software communicate with device endpoints through end-to-end transfers, and the protocol layer handles transaction sequencing, while the link layer ensures the reliability of the link [23].

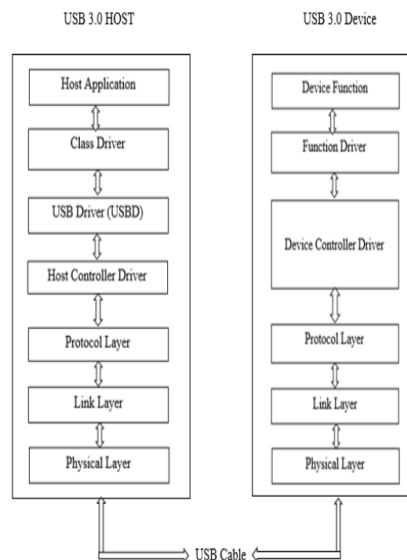


Figure 2.10: Communication layers, from [23]

2.2.3 HUB USB

According to USB specifications, USB hubs are mandated to facilitate the connection of multiple USB devices. Additionally, these hubs serve the purpose of extending the maximum link length, thereby expanding the overall size of the network. Due to the inherent limitation of a single USB cable length to 5 meters, hubs become essential in enabling distances of up to 20 meters between the host and a device. It is worth noting that USB specifications restrict the number of cascaded hubs to three [9].

A USB 2.0 hub is comprised of three main components: the Hub Controller, the Hub Repeater, and the Transaction Translator. The Hub Repeater operates as a protocol-controlled switch, directing communication between the upstream port and downstream ports. Additionally, it incorporates hardware assistance for reset and suspend/resume signaling. The Host Controller enables bidirectional communication between the host and the hub. It employs specific commands that empower the host to set up the hub, observe, and manage its ports. The Transaction Translator is responsible for enabling full-/low-speed devices connected to the hub, ensuring that all data between the host and the hub is transmitted at high-speed [33].

2.3 Related Work

In the paper [9] was designed a USB Host under microcontroller control, proficient in overseeing the interface with the sensor network and serving as a bridge to the high-level network. USB bus control is managed through a host controller (Cypress SL811HS), while the LAN interface is delegated to a 10-BaseT Ethernet controller (Crystal CS8900A). The links established between the microcontroller and these devices have been established using parallel buses through the microcontroller ports and a minimal external glue logic, as presented in Figure 2.4.

Another example of USB HUB application was developed by Yung-Hoh Sheu, Yu-Ping Dai and Duen- Shiang Fu [37]. The paper presents a generic telemedicine framework, based on an ARM-based embedded system and utilizing USB as a user-friendly interface, for cost-effective and user-friendly homecare services. The system supports multiple USB slave devices, enabling a homecare internal network, and allows remote monitoring and control via PDA or GPRS telecommunication based devices.

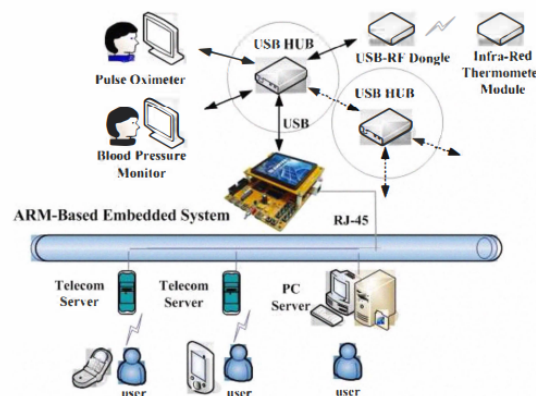


Figure 2.11: Proposed Embedded USB Homecare Internet System, from [37]

In this example, Figure 2.11, the USB hub was employed to effortlessly expand the number of USB ports, enhancing the system's connectivity.

In today's market, a broad range of controllable USB hubs are available, catering to various requirements for efficient connectivity and device management. Among these, the MCD USB-HUB from Meilhaus Electronic stands out as a notable example, [10]. This advanced USB 3.0 hub is equipped with six ports, each capable of being individually switched on or off, offering users a high degree of control and flexibility in managing their USB connections. This device offers a multitude of options for controlling the downstream ports, ensuring maximum flexibility and convenience for users.

The Meilhaus Electronic MCD USB-HUB [10] is a sophisticated USB 3.0 hub equipped with six individual downstream ports, each of which can be independently controlled and activated or deactivated through USB commands. This hub operates on a 12 to 24 VDC power supply with a minimum output power of 35 watts, providing sufficient current of up to 900 mA per port.

The design of the MCD USB-HUB incorporates semiconductor switches that isolate both the power supply (+5 V) and data lines when a port is deactivated. This functionality ensures that each port can be completely turned off, effectively cutting off both power and data transmission, which helps in preventing any inadvertent electrical or data interference when a port is not in use. This feature is particularly useful for applications requiring precise control over each USB connection, enhancing the overall efficiency and management of USB devices connected to the hub.

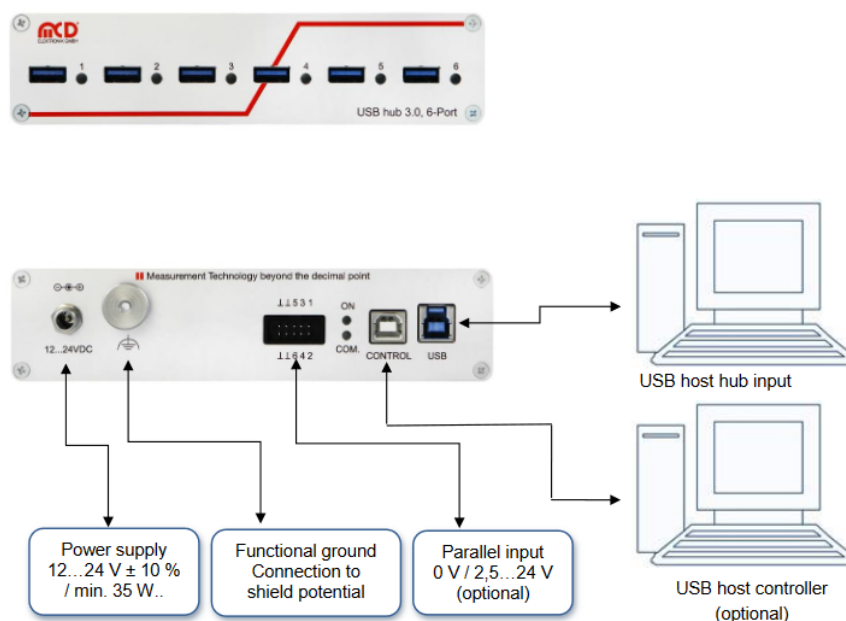


Figure 2.12: Connection between the USB HUB and PC, from [10]

The USB hub offers flexible control options, allowing it to be managed either via the hub input labeled "USB" on the right side or through an external control input labeled "ext. Control" on the left side. When the external USB control input is connected to a different host device, the external host assumes control over the hub, while the hub input retains only its basic USB hub functionality.

Furthermore, with appropriate configuration, the USB ports on the hub can also be individually activated or deactivated through the parallel input. This multi-faceted control mechanism provides versatile management capabilities, making it possible to switch the hub's ports on or off either through USB commands or through an external control interface, depending on the operational needs and the specific setup of the connected systems.

The device presented earlier, the MCD USB-HUB from Meilhaus Electronic, is a potential solution. However, its high price makes it less viable as a long-term option. Additionally, this equipment does not meet Bosch's requirement for a range of different USB ports, specifically USB 3.0, USB 2.0, and RS232 ports.

2.4 Present Contexted in the Company

The current solution for managing multiple USB ports through a computer often involves connecting multiple USB hubs. While this approach allows users to control and interact with various devices seamlessly, it comes with certain disadvantages, primarily related to the complexity of hardware management.

Connecting multiple USB hubs can lead to challenges in organizing and overseeing the connected devices. This can result in a cumbersome hardware setup, making it difficult for users to efficiently manage and keep track of each device. As the number of hubs increases, so does the potential for confusion in identifying specific devices and ensuring optimal connectivity.

Moreover, the need for multiple hubs can introduce issues related to power distribution and data transfer. Users may encounter difficulties in maintaining consistent power delivery to all connected devices, leading to potential disruptions in functionality. Additionally, the increased number of hubs can contribute to data transfer bottlenecks, affecting the overall performance of the connected devices.

In the culmination of these developments, the Yepkit USB Switchable Hub (YKUSH) board emerges as a pivotal component for efficiently toggling the power of selected devices.

YKUSH boards provide users with the ability to selectively power on and off individual USB devices connected to the downstream ports of the hub. This control is facilitated through an application on the host system, such as a connected PC. When a user toggles the power of a YKUSH downstream port with a connected device, the outcome is equivalent to physically connecting or disconnecting a device to a port on a standard USB hub.

The process of switching control involves sending commands to the on-board microcontroller, also known as the YKUSH board control unit. This microcontroller is recognized by the host as a Human Interface Device (HID) USB device, eliminating the need for specific driver installations. The communication between the host and the board control unit is established through a straightforward message protocol. In essence, the YKUSH board streamlines the power management of USB devices by providing a user-friendly interface for controlling individual ports without the hassle of additional driver installations.

The Yepkit, while offering notable advantages, comes with certain drawbacks that need consideration. One notable disadvantage is its reliance on a hub when there is a need to manage multiple USB devices. This pre-requisite introduces an additional layer in the setup, potentially complicating the overall configuration.

Another limitation is that the Yepkit operates exclusively at USB 2.0 speed. Despite USB 3.0 having the same pinout, the Yepkit does not facilitate communication at USB 3.0 full speed. This can be a constraint for users seeking the enhanced data transfer capabilities provided by USB 3.0.

The proposed solution will feature a total of seven dedicated communication ports: two for USB 3.0, four for USB 2.0, and one for RS-232. Additionally, it will include two USB Type-C ports designated exclusively for power supply.

Compared to the previous Yepkit solution, this design offers several significant advantages. First, it supports high-speed communication through both USB 3.0 and RS-232, in addition to USB 2.0, providing enhanced versatility and faster data transfer rates. Furthermore, it significantly increases the number of available ports from Yepkit's three-port limitation, allowing greater flexibility in managing multiple devices simultaneously. Another key benefit is the ability to control each port individually, offering improved control and functionality across connected devices.

This solution thus represents a more robust and scalable option, suited to environments that require high-speed communication and increased connectivity.

Chapter 3

Hub Architecture and Design

This chapter provides an in-depth explanation of the methodology employed in this work. It begins with the conceptualization of the system architecture, Section 3.1, illustrates through a block diagram and details the components used along with their competitive advantages. Section 3.3.1 provides an overview of the software used, in particular Altium Designer. In the section 3.2 of the report, a detailed explanation of the circuitry used and the components connected will be provided. This includes a comprehensive analysis of each component's role and the specific connections made within the circuit. This thorough examination aims to elucidate the design choices and functionality of the overall system, ensuring clarity and understanding of the project's technical aspects.

3.1 System architecture

3.1.1 Block diagram

The goal of this dissertation, as previously mentioned, is to create a USB hub featuring multiple types of ports. The design will include two USB 3.0 ports, four USB 2.0 ports, one RS232 port and two USB Type-C ports dedicated solely to power supply. However, to simplify the circuit in the initial stage, we will focus only on the seven ports used for data transfer and disregard the USB Type-C ports, as they will only be involved in powering the circuitry.

To achieve this, the seven ports needs to follow a certain configuration. There is two options for the block diagram: a daisy chain configuration or a cascade configuration.

In the daisy chain connection, the data from the upstream port is routed to the first block. From this block, the data is distributed to the 2 USB 3.0 ports. Additionally, the data is passed to the following block, as illustrated in Figure 3.1, which further distributes it to the USB 2.0 ports, continuing in this manner for the subsequent ports.

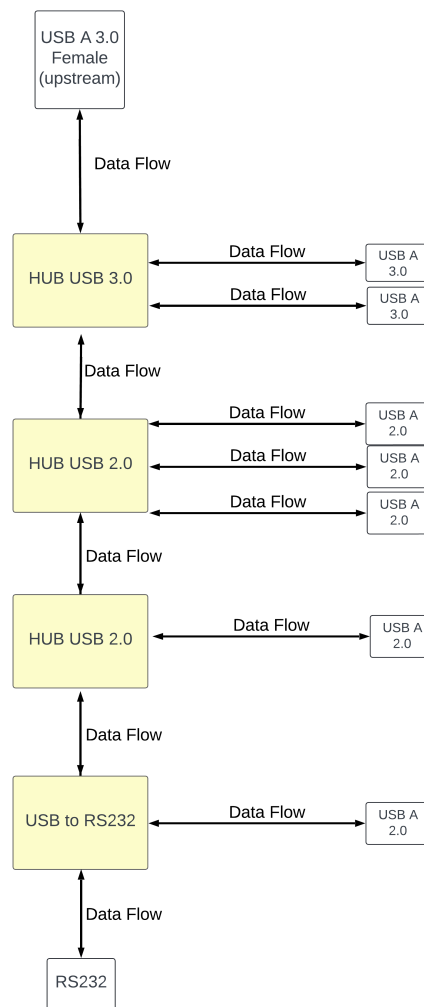


Figure 3.1: Block diagram with the daisy chain configuration

In contrast, the cascade configuration routes the data from the upstream port to the first block, as illustrated in Figure 3.2, where it is immediately separated into three distinct streams: one for USB 3.0, one for USB 2.0, and one for RS232. Each data stream is then directed to its respective block, which handles the distribution of data to the corresponding USB ports. This method ensures a more organized and efficient data flow, as each type of port receives data from a dedicated block designed specifically for its protocol.

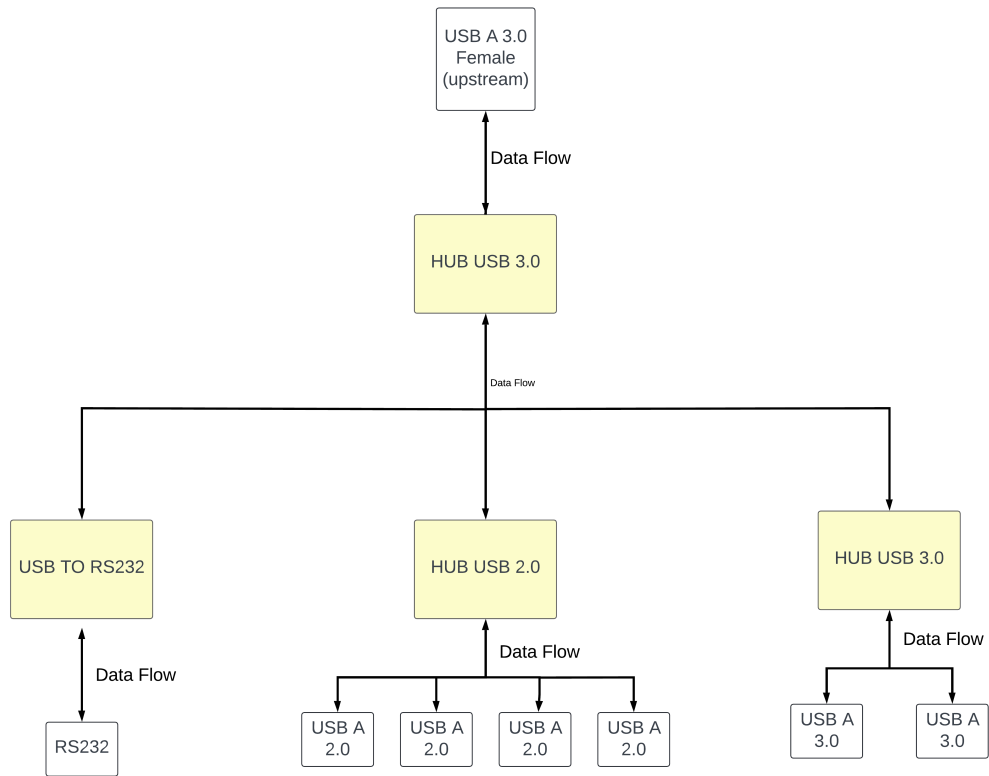
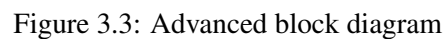


Figure 3.2: Block diagram with the cascade chain configuration

According to the USB 2.0 specifications present in [33], the maximum number of tiers permitted is seven, due to the timing constraints imposed by hub and cable propagation times. Considering this, the configuration depicted in Figure 3.2 has a distinct advantage. It experiences less propagation delay compared to the configuration presented in Figure 3.1, which consists of 4 tiers. Consequently, the daisy chain configuration suffers from increased propagation delays, making it less efficient in terms of data transfer speed and overall performance.

It is assumed that each block diagram supports a maximum of 4 downstream ports, in addition to the one responsible for connecting to the upstream port. The first configuration, daisy chain configuration, could be optimized by reducing the number of USB hubs if each block diagram could support up to 7 downstream ports. If this were feasible, all 4 USB 2.0 ports could be connected simultaneously, eliminating the need for an additional block diagram to accommodate the final USB port. This adjustment would result in fewer tiers. However, it is important to note that most USB hubs typically support a maximum of 4 downstream connections, and USB hubs with 7 downstream ports are no longer manufactured.

Based on the previously mentioned advantages, the cascade chain configuration, Figure 3.2, excels. Consequently, the advanced block diagram is presented below, incorporating power supply lines dedicated to providing power and commands issued by the microcontroller.



These switches play a pivotal role in the power supply chain to the connectors. They are programmed to terminate power transmission upon receiving a designated signal. This hierarchical control mechanism ensures precise management of power allocation throughout the USB hub, thereby optimizing operational efficiency and safeguarding connected devices against potential electrical issues.

3.1.2 Electronic Components Selection

In this section, each component necessary for the PCB will be analyzed, along with the available options and their respective advantages and disadvantages.

Beginning with the microcontroller, this component is critical as it receives commands from the computer to control the on/off state of each port. Without the microcontroller, the PCB would not function as intended.

For this function, the Raspberry Pi Pico was selected over the Arduino Nano due to several distinct advantages it offers. The Raspberry Pi Pico, with its RP2040 microcontroller, provides a range of features that enhance its suitability for controlling USB hub power management.

One significant advantage is the inclusion of onboard EEPROM, according to the Datasheet [28], in the Raspberry Pi Pico, which allows for persistent storage of configuration settings and other critical data. This capability is particularly beneficial for preserving the state of each downstream port's power settings in the event of a system shutdown due to a power cut. By utilizing the onboard EEPROM, the Raspberry Pi Pico can store the current power status of each port, ensuring that these settings are retained even when the system is powered off. In contrast, the Arduino Nano does not have built-in EEPROM, requiring external memory solutions for similar functionality.

Additionally, the Raspberry Pi Pico benefits from a more powerful processor with higher clock speeds and increased processing capabilities compared to the Arduino Nano. This enhanced performance can handle more complex tasks and manage multiple operations concurrently, which is advantageous for controlling various downstream ports and processing signals.

The Raspberry Pi Pico also offers more GPIO pins than the Arduino Nano [6] and greater flexibility in interfacing with different peripherals, making it a more versatile choice for expanding functionality and integrating additional components. Its compatibility with MicroPython and C/C++ provides a more accessible and modern programming environment, compared to the Arduino Nano's reliance on the Arduino IDE and C/C++.

For the first block diagram presented in the Figure 3.2, a USB hub capable of handling the SuperSpeed typical of USB 3.0 is required. Since it has 3 downstream ports, it is essential to consider ICs with at least 4 downstream ports. Given these specifications—USB 3.0 speed and 4 downstream ports—numerous options are available on the market.

Texas Instruments is an American company that develops semiconductors for electronic manufacturers. As one of the world's leading companies in the microcontroller market, all USB hubs from Texas Instruments that met the established requirements were analyzed [43].

Texas Instruments offers several USB hubs that meet these criteria, with only subtle variations between them, resulting in a high degree of similarity across their product line.

One notable option is the TUSB8044A [42], an IC featuring 4 downstream USB 3.2 Gen 1 ports capable of transmitting data at speeds up to 5 Gbps. This particular IC stands out due to its USB Billboard feature, which is uncommon among other ICs. The USB Billboard feature serves to communicate essential information about the alternate mode to the USB host, ensuring seamless integration and functionality within complex systems. This makes the TUSB8044A a versatile and robust choice for high-speed USB hub applications.

Another model available from Texas Instruments is the TUSB8043, [40]. Unlike the TUSB8044A, the TUSB8043 lacks the USB Billboard feature. However, it still supports communication with 4 downstream ports at speeds up to 5 Gbps. This model provides a reliable option for high-speed

data transfer while offering a cost-effective solution for applications where the USB Billboard feature is not required. With its robust performance and compatibility, the TUSB8043 is suitable for various high-speed USB hub implementations.

An enhanced version of this IC, the TUSB8043A, has been released with several bug fixes and improvements, providing a more reliable and stable operation [41]. This makes the TUSB8043A a superior choice for applications requiring robust performance and high reliability, while the TUSB8043 remains a cost-effective alternative.

Another model is the TUSB8041, which shares similar characteristics with the previously mentioned ICs according with the authors in [38]. This IC offers a straightforward solution for basic USB 3.0 hub requirements, maintaining support for 4 downstream ports with data transfer speeds up to 5 Gbps. Despite its simplicity, the TUSB8041, like the other models, can be effectively used in various applications, including computer systems, docking stations, monitors, and set-top boxes. All these ICs provide versatile solutions tailored to different levels of complexity and functionality, making them suitable for a wide range of high-speed USB hub implementations.

Now turning to another manufacturer, Microchip, several ICs are recommended for new designs, such as the USB5744 [30]. This IC targets similar applications and exhibits comparable performance to the Texas Instruments models. However, the USB5744 offers distinct advantages: it has fewer pins (56 compared to the 64 found in Texas Instruments ICs) and a more compact size (7x7mm versus 9x9mm). These features can be beneficial for PCB design, offering more flexibility and potentially reducing the overall footprint of the circuit board. The smaller size and reduced pin count can simplify the layout and integration process, making the USB5744 a compelling option for streamlined and space-constrained designs.

Another option from Microchip recommended for new designs is the USB7252C in accordance with [31]. This IC boasts a significant advantage over the previously discussed makers by supporting communication speeds of up to 10 Gbps. This is considerably higher than the 5 Gbps required for this project. While the USB7252C offers impressive performance with its superior speed capabilities, it may exceed the requirements for this particular application, making it a powerful yet potentially overqualified choice. Additionally, its advanced features come with a higher cost, which could be a consideration if budget constraints are a factor.

To provide a clearer comparison of the differences between each USB hub, the Table 3.1 highlights the key features and advantages of each model.

Table 3.1: Comparison between the different USB HUB based on [11]

USB HUB Name	Transmission Speed (Gbps)	Price (€)
TUSB8044A	5	8.25
TUSB8043	5	9.67
TUSB8043A	5	9.66
TUSB8041	5	6.52
USB5744	5	5.78
USB7252C	10	13.85

Based on the Table 3.1, the TUSB8041 and USB5744 emerge as the most advantageous USB hubs for this project, given the goal of minimizing the final cost. The decision between these two ICs hinges on the amount of available information for each. The TUSB8041 benefits from a wealth of documentation, including detailed recommendations for implementation and access to an evaluation module designed specifically for this IC. This evaluation module can significantly aid in the component's integration and implementation, providing valuable support throughout the design process. Considering all the factors discussed, the TUSB8041 emerges as an ideal choice for the first block diagram.

Moving on to the next block diagram, which connects to two downstream USB 3.0 ports, an IC is needed that supports data transmission speeds of up to 5 Gbps. However, unlike the previous requirement, this IC only needs to handle 2 downstream ports rather than 4.

Texas Instruments offers a single model suitable for this requirement: the TUSB8020. In line with the datasheet [18], this USB hub controller is designed specifically for USB 3.0, supporting data transfer speeds of up to 5 Gbps and accommodating up to 2 downstream ports. It provides high-speed data transfer capabilities along with advanced power management features to optimize power consumption and improve efficiency. Additionally, the TUSB8020 ensures backward compatibility with USB 2.0 and USB 1.1 devices, making it versatile and suitable for a broad range of applications.

Microchip offers a single option for configurations with 2 downstream ports: the USB5742B. However, this model supports only USB 2.0 data rates for its downstream ports, which does not meet the 5 Gbps requirement based on [29]. Given the lack of other available options in the market that fulfill the required specifications, the TUSB8020 from Texas Instruments emerges as the sole suitable solution for this application.

For the third block diagram, which requires a USB hub controller with 4 downstream ports, the communication rate needed is USB 2.0 rather than the faster 5 Gbps.

Texas Instruments offers two options for this configuration: the TUSB4041I and the TUSB4041I-Q1. Based on the datasheet [20], the latter is specifically designed for automotive applications, as detailed in its datasheet, which includes features tailored to automotive conditions and generally comes at a higher cost. Although it could be utilized for the project, its automotive-specific design and increased cost make it less suitable. Therefore, the TUSB4041I, which provides the necessary functionality at a more appropriate cost, is the better choice for this application, [19].

Finally, a USB-to-RS232 translation IC is required for this project. The available option for this functionality is the FT232R from FTDI. This IC provides reliable conversion between USB and RS232 interfaces, based on the datasheet [27], ensuring compatibility and efficient communication between modern USB devices and legacy RS232 equipment. With its robust performance and extensive support for various applications, the FT232R is well-suited to meet the needs of this project, facilitating seamless integration and operation.

With all the main components selected, the previously shown block diagram can now be updated to include the chosen components. The data lines have been specified to clarify communication speeds: USB 3.0 data lines are now represented by differential-line pairs with TX and RX,

while USB 2.0 lines are shown as D+ and D-. This distinction makes it easier to understand where the communication can reach 5 Gbps (USB 3.0) or 480 Mbps (USB 2.0).

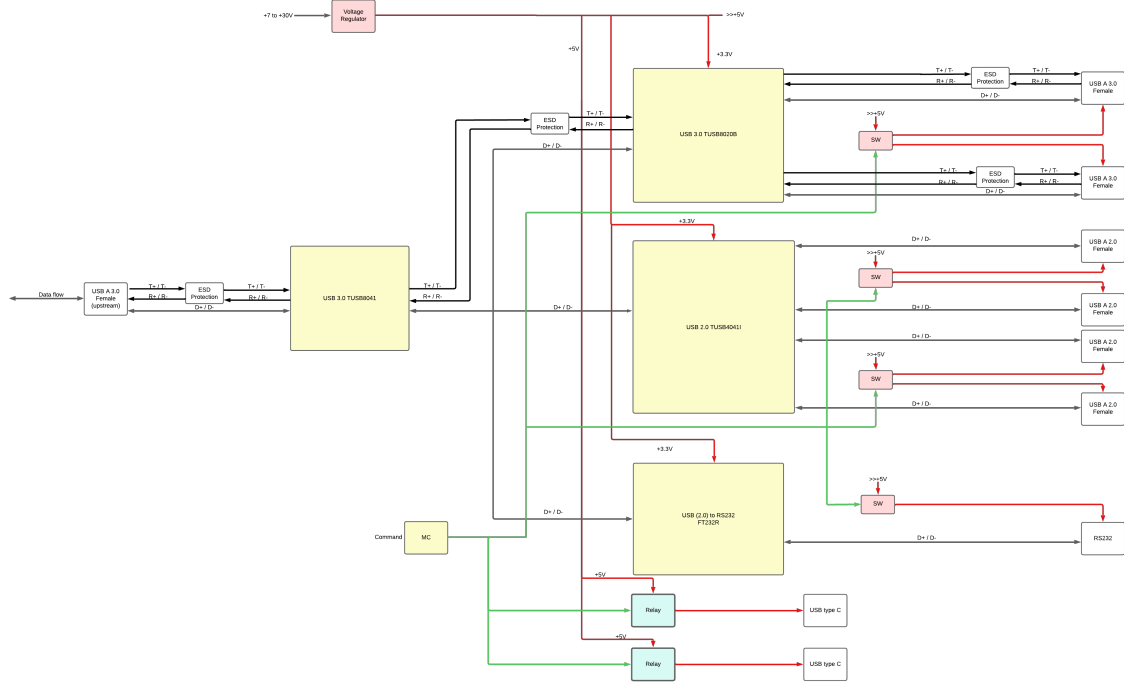


Figure 3.4: Final block diagram

By examining the updated block diagram, it can be observed the presence of a device in the differential-line pairs TX and RX. Although the datasheets for the TUSB8041 and other ICs indicate built-in ESD protection, additional ESD protection devices have been incorporated into the circuit. This extra layer of protection safeguards the board against potential electrostatic discharges that might occur during handling, enhancing the overall durability and reliability of the system [21].

3.2 System Design

3.2.1 First Layer Data Processing

Starting with the first integrated circuit, the TUSB8041 is a four-port USB 3.0 hub. According to its datasheet [38], it can provide simultaneous SuperSpeed USB (5 Gbps), high-speed (480 Mbps), full-speed (12 Mbps), and low-speed (1.5 Mbps) connections on the upstream port, while offering SuperSpeed, high-speed, full-speed, or low-speed connections on the downstream ports. If the upstream port is connected to an environment that only supports high-speed or full-speed/low-speed connections, the downstream ports' SuperSpeed USB connectivity is disabled. Similarly, if the upstream port is limited to full-speed/low-speed connections, both SuperSpeed and high-speed connectivity are disabled on the downstream ports. Essentially, the upstream connection

speed (SuperSpeed, high-speed, or full-speed) determines the maximum speed available on the downstream ports.

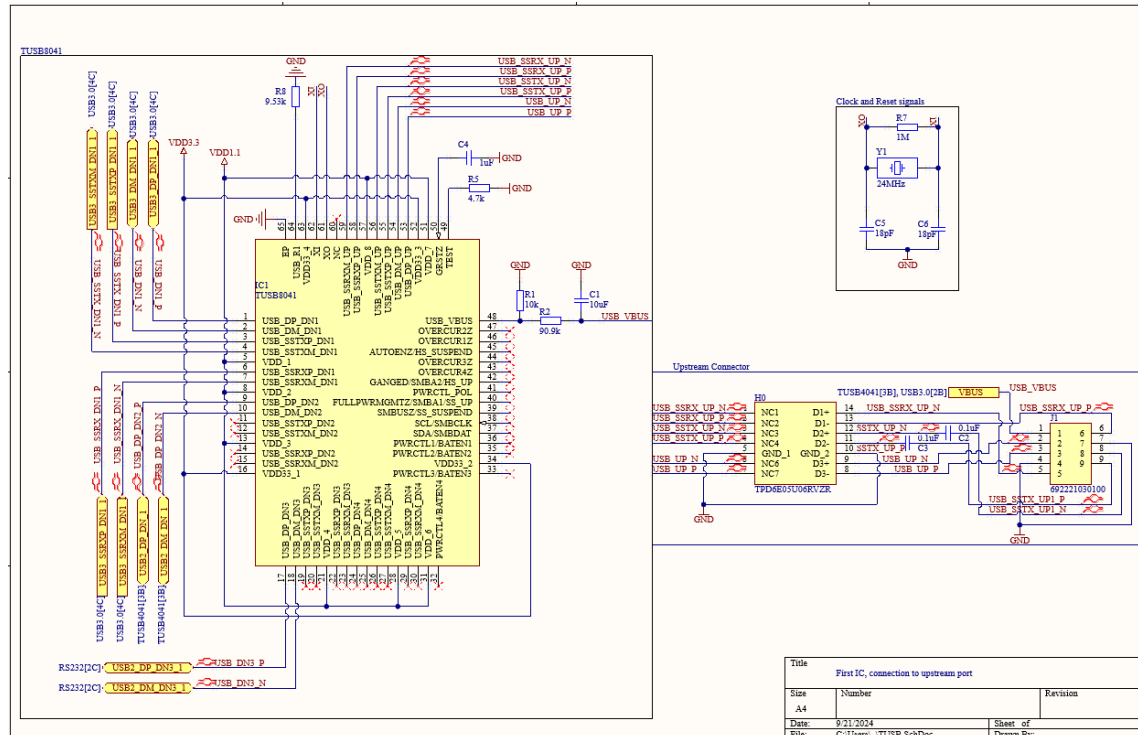


Figure 3.5: TUSB8041 Circuitry and their connections

Configuration of the TUSB8041 can be done by setting several inputs sampled at power-on reset or using an optional serial EEPROM or SMBUS host. Another type of configuration is to use the default internal pullup or pulldown resistors or they can be overridden with external pullup or pulldown resistors.

The explanation of the circuitry will follow the increasing order of the pin numbers.

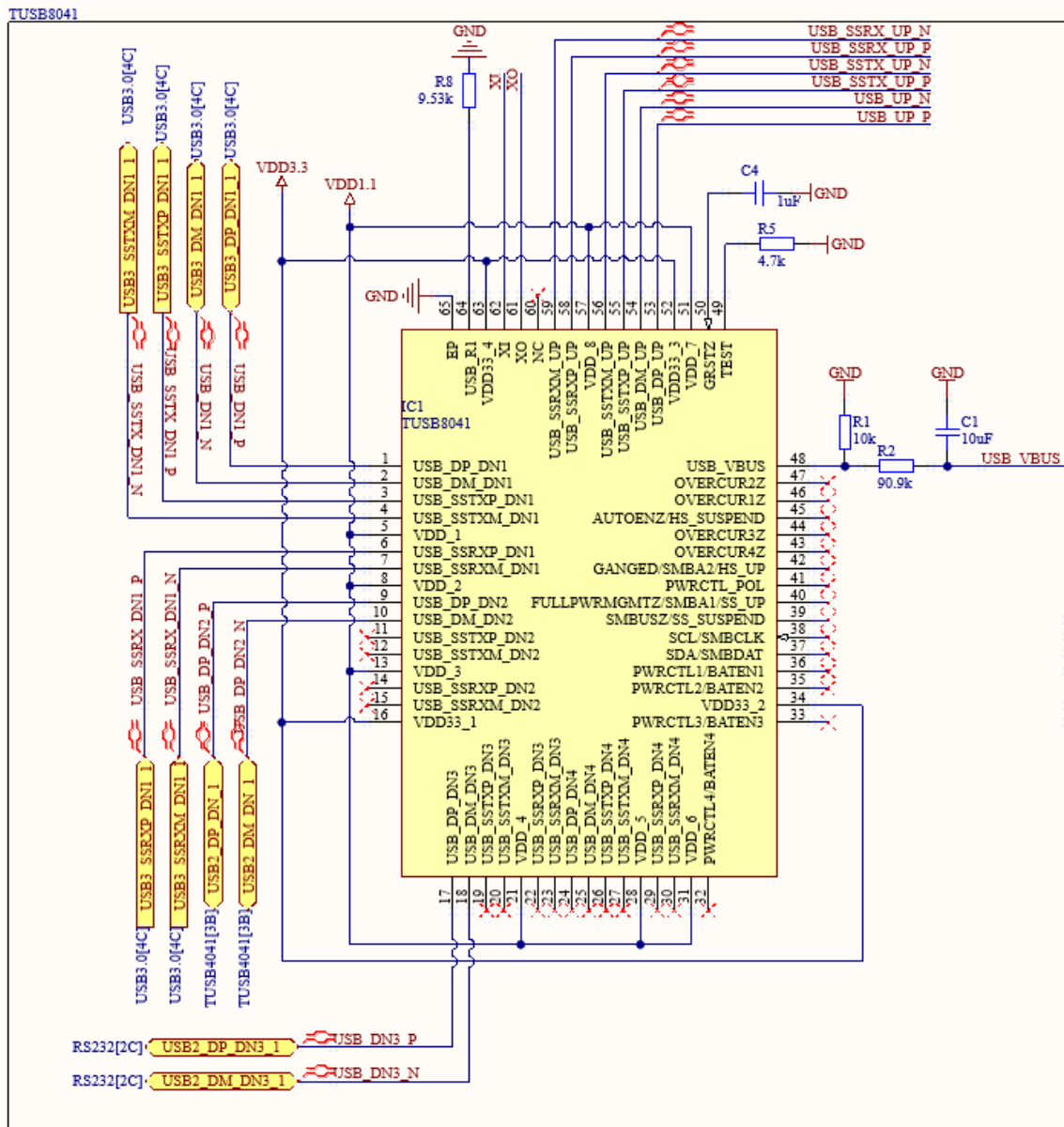


Figure 3.6: TUSB8041 Circuitry

Pins 1, 2, 3, 4, 6, and 7 are dedicated to communication with the first downstream port. Pins 1 and 2 form the differential-line pair for data minus (D-) and data plus (D+), which are essential for USB 2.0 communication. Pins 3 and 4 constitute the differential-line pair for transmission (TX), while pins 6 and 7 serve as the differential-line pair for reception (RX).

Each differential-line pair is identified using a directive (red label above each line) in Altium Designer, a useful feature for marking differential-line pairs. Additionally, each line within a differential-line pair ends with the suffix `_N` or `_P` to indicate the negative and positive lines,

respectively. This labeling system simplifies future work and design processes, and will be further elaborated in Section 3.3.5.

The yellow boxes at the end of each line indicate connections to other circuitry on different schematic pages in Altium Designer. This method helps streamline and simplify the overall circuit understanding. Adjacent to each yellow box, the corresponding page name is written in blue, indicating where the connection continues (USB3.0).

Furthermore, the use of directives and clear labeling not only aids in maintaining organization but also ensures precise and accurate routing during PCB layout. This attention to detail is crucial for achieving optimal signal integrity and performance in high-speed USB communications.

These six pins connect to the IC responsible for interfacing with the two USB 3.0 ports, as they are the only ones that provide both differential-line pairs: transmitter (TX) and receiver (RX).

Pins 9 and 10 form the differential-line pair that connects to the TUSB4041, the IC responsible for communicating with the four USB 2.0 downstream ports.

Pins 11, 12, 14, and 15 are left unconnected because they correspond to the USB 3.0 differential-line pairs (transmitter and receiver). Since the TUSB4041 only handles USB 2.0 communication and does not require USB 3.0 connections, these pins are not needed and are therefore left unconnected.

Pins 17 and 18 serve the same function as pins 9 and 10; they form a differential-line pair intended to connect with the IC responsible for converting USB 2.0 to RS232. Meanwhile, pins 19, 20, 22, and 23 are left unconnected for the same reason as pins 11, 12, 14, and 15—they correspond to the USB 3.0 differential-line pairs (transmitter and receiver), which are not required for the RS232 conversion.

Pins 24, 25, 26, 27, 29, and 30 are left unconnected because there are only three downstream ICs in this layer, as illustrated in Figure 3.3.

Pins 32, 33, 35, and 36 are used to control the downstream ports. However, in this layer, these pins will not be utilized because there is another layer where the ICs have the capability to control the downstream ports.

Pins 37 and 38 can communicate via I2C or SMBus, with the choice determined by the value of pin 39. If pin 39 is high, I2C communication is selected; if low, SMBus communication is chosen. When I2C mode is selected (pin 39 high), pin 37 functions as SDA (data line), and pin 38 as SCL (clock line) to interface with an I2C EEPROM. In SMBus mode (pin 39 low), pin 37 serves as SMBDAT, and pin 38 as SMBCLK for communication with an SMBus host.

Since it is not intend to control anything through this IC, these pins are left unconnected. However, pin 39 has an internal pullup resistor, enabling the I2C interface by default. Pins 37 and 38 have internal pulldown resistors, disabling the serial EEPROM/SMBus interface [39].

Pin 40 defines the full power management mode. Full power management refers to the ability to control power to the downstream ports of the TUSB8041 using PWRCTL[4:1]/BATEN[4:1]. If the value of this pin is low, the power switching and overcurrent inputs are supported. If the value is high, these features are not supported.

Since full power management and SMBus functionality are not implemented, this pin has been left unconnected.

The following pin, 41, is used to control the polarity of the port power control. If the value of this pin is high, the port power control polarity defaults to active high. If the value is low, the port power control polarity switches to active low. Since the full power management was not used, this pin was also left unconnected.

Pin 42 defines the hub's response to an overcurrent event. If the value of this pin is high, ganged mode is activated, meaning the power to all downstream ports will be switched off if any overcurrent event occurs. If the value of the pin is low, the power control will be individual for each downstream port. Since that there is no downstream ports to control in this layer, overcurrent events were not taken into consideration for this IC, consequently this pin was left unconnected.

This can be seen in Figure 3.6 more specifically in pins 43, 44, 46, and 47, which correspond to the overcurrent events that might be detected in the power switch and sent to these pins. These particular pins have an internal pullup resistor, meaning they are configured to interpret high signals as indicating no overcurrent event has occurred and low signals as indicating that an overcurrent event has occurred.

Pin 45 is responsible for defining the automatic charging mode, which automatically enables battery charging for downstream ports that support this feature. As this pin has an internal pull-up, automatic mode is disabled. Therefore, the pin was left unconnected to disable these features.

Pin 48, labeled as USB_VBUS, is a detection input that informs the hub when an active host is connected. This pin is connected to the upstream connector. If USB_VBUS is low, the hub will not establish a connection.

The USB_VBUS pin should be around 500 mV when power is on, according with [22]. To achieve this, a resistor divider network is used, consisting of a 90.9 k Ω resistor (R2) and a 10 k Ω resistor (R1), to step down the 5 V voltage from the upstream connector to the desired 500 mV.

According to the USB Specification [14], when a function or hub is plugged into the network, it introduces a certain amount of onboard capacitance between VBUS and ground. Additionally, the device's regulator may supply current to the output bypass capacitance and to the function as soon as power is applied. Without appropriate measures, this can result in a surge of current into the device, potentially pulling the VBUS on the hub below its minimum operating level. To address this issue, inrush current must be limited, and sufficient capacitance must be provided in each hub to maintain power tolerance across all ports.

Limiting inrush current also helps to reduce contact arcing, thereby extending the lifespan of connector contacts. Therefore, a 10 μ F capacitor (C1) is used to manage inrush current and stabilize the VBUS voltage.

Pin 49 is reserved for production test mode only. To disable this feature, a pulldown resistor was added.

Pin 50, labeled GRSTZ, serves as a global power reset. Activating this reset pin returns all internal registers of the TUSB8041 to their default state. The TUSB8041 includes an internal pullup on GRSTZ, which, combined with an external capacitor, creates an RC time constant delay.

This passive reset circuit, formed by the external capacitor and internal pullup, typically provides a sufficient reset pulse on GRSTZ for most applications.

Additionally, it's important to ensure that VDD rises before VDD33. This sequence guarantees that the output drivers receive a defined input and that the state machine is initialized properly, preventing the TUSB8041 from entering any unknown states. The internal pullup resistor will charge the capacitance present on pin 50, GRSTZ, contributing to the overall reset functionality.

Pins 53, 54, 55, 56, 58, and 59 are dedicated to connections with the upstream port.

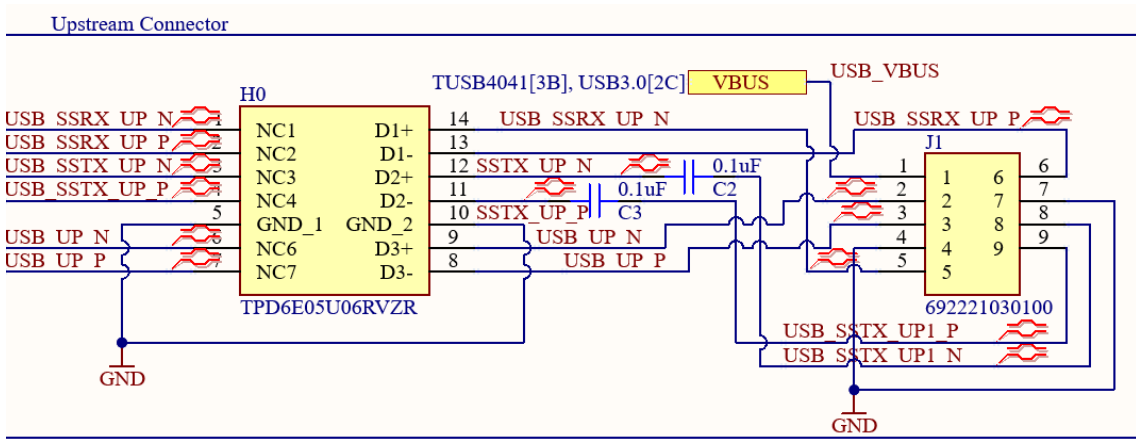


Figure 3.7: TUSB8041 Upstream Circuitry

In the figure above, the circuitry for the three differential-line pairs from the TUSB8041 to the connector can be analyzed.

Firstly, an ESD protection device was added. According to the datasheet [21], Texas Instruments recommends selecting devices that permit flow-through routing of the USB differential signal pairs, as these provide the cleanest routing. Therefore, a 6-channel ESD protection device was incorporated to safeguard against electrostatic discharge events represented in the Figure 3.7 as H0, [17].

Next, AC capacitors were added to the TX data line. These ceramic capacitors are essential for enabling USB 3.0 communication. Without these capacitors, all signaling would occur over the D+ and D- lines, limiting the communication to USB 2.0 speeds, it is also recommended by the authors of [21] to use capacitors with size 0402 in inches which represents 1mm of length and 0.5mm of width.

Following the AC capacitors, the lines are routed to the upstream connector, represented by J1, which is a USB 3.0 Type-B port. This connection ensures that the TUSB8041 hub can communicate with the host device using the high-speed differential signaling necessary for USB 3.0, thereby enabling data transfer rates up to 5 Gbps.

Referring back to Figure 3.6, pin 60 is marked as NC (No Connection), meaning it is intended to be left floating without any connections.

Pins 61 and 62 are designated as the crystal output and crystal input, respectively, for the internal oscillator. The crystal oscillator provides the necessary clock signal for the TUSB8041 to function correctly.

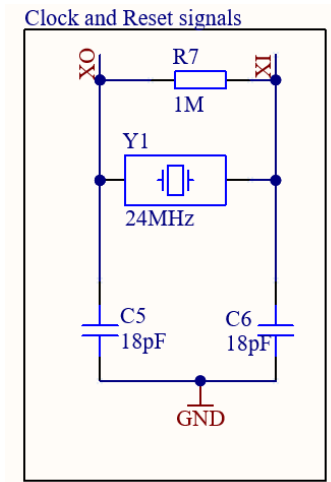


Figure 3.8: TUSB8041 Crystal Circuitry

Pins 61 and 62 are typically connected to an external crystal and associated passive components to form the oscillator circuit, which ensures stable and precise timing for data communication and other operations within the USB hub. By connecting these pins to a crystal oscillator circuit, the TUSB8041 can generate the necessary clock frequencies required for its internal operations and maintain accurate data transfer rates.

The TUSB8041 requires a 24-MHz low-ESR crystal, labeled as Y1, to provide the necessary clock signal for its operation. This crystal should be of the fundamental mode type and have a load capacitance in the range of 12 to 24 pF. For optimal performance, the crystal must also have a frequency stability rating of ± 100 ppm (parts per million) or better based on the Evaluation Board datasheet [39]. Additionally, a 1 M Ω feedback resistor should be used with the crystal to ensure proper oscillation.

Pin 64 of the TUSB8041 functions as a precision resistor reference. It is essential to connect a resistor with a value of 9.53 k Ω and a tolerance of 1% between this pin and ground. This precision resistor is crucial for maintaining accurate reference voltage levels, which ensure the proper operation and stability of the TUSB8041's internal circuits.

Pins 5, 8, 13, 21, 28, 31, 57, and 51 are all connected to a common power rail labeled VDD1.1. This rail provides a stable 1.1V power supply to the TUSB8041 integrated circuit, ensuring its proper operation.

Similarly, pins 16, 34, 63, and 52 are connected to another power rail labeled VDD3.3 which supplies 3.3V to the IC.

3.2.2 USB 3.0 Data Distribution

Moving on to the next USB hub, the TUSB8020, its primary role is to interface with the previously analyzed TUSB8041, receiving data at speeds of up to 5Gbps and communicating with two downstream USB 3.0 ports. The analysis of this IC will be similar to that of the TUSB8041, given that both ICs operate in a comparable manner and are manufactured by Texas Instruments.

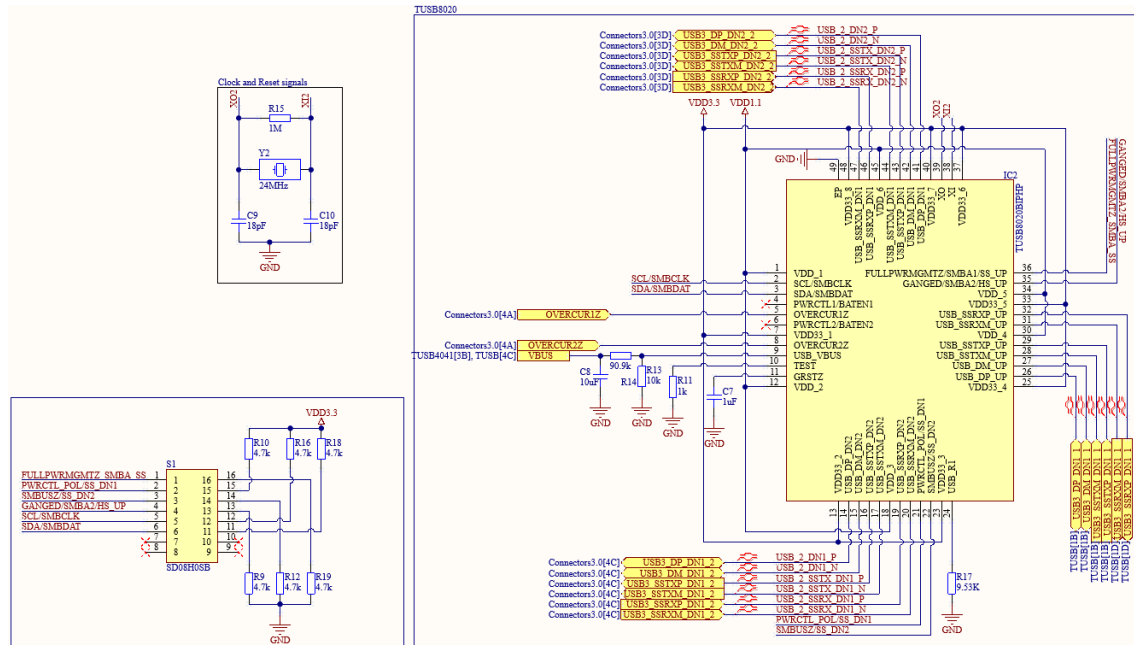


Figure 3.9: TUSB8020 Circuitry and their connections

As mentioned earlier, the TUSB8020 operates in a manner similar to the TUSB8041. The primary difference lies in their downstream communication capabilities: the TUSB8041 can interface with four downstream ports, whereas the TUSB8020 is designed to communicate with only two downstream ports. Despite this difference, the working mode and fundamental functionalities of both ICs are largely equivalent after reading the datasheet of TUSB8020, [18].

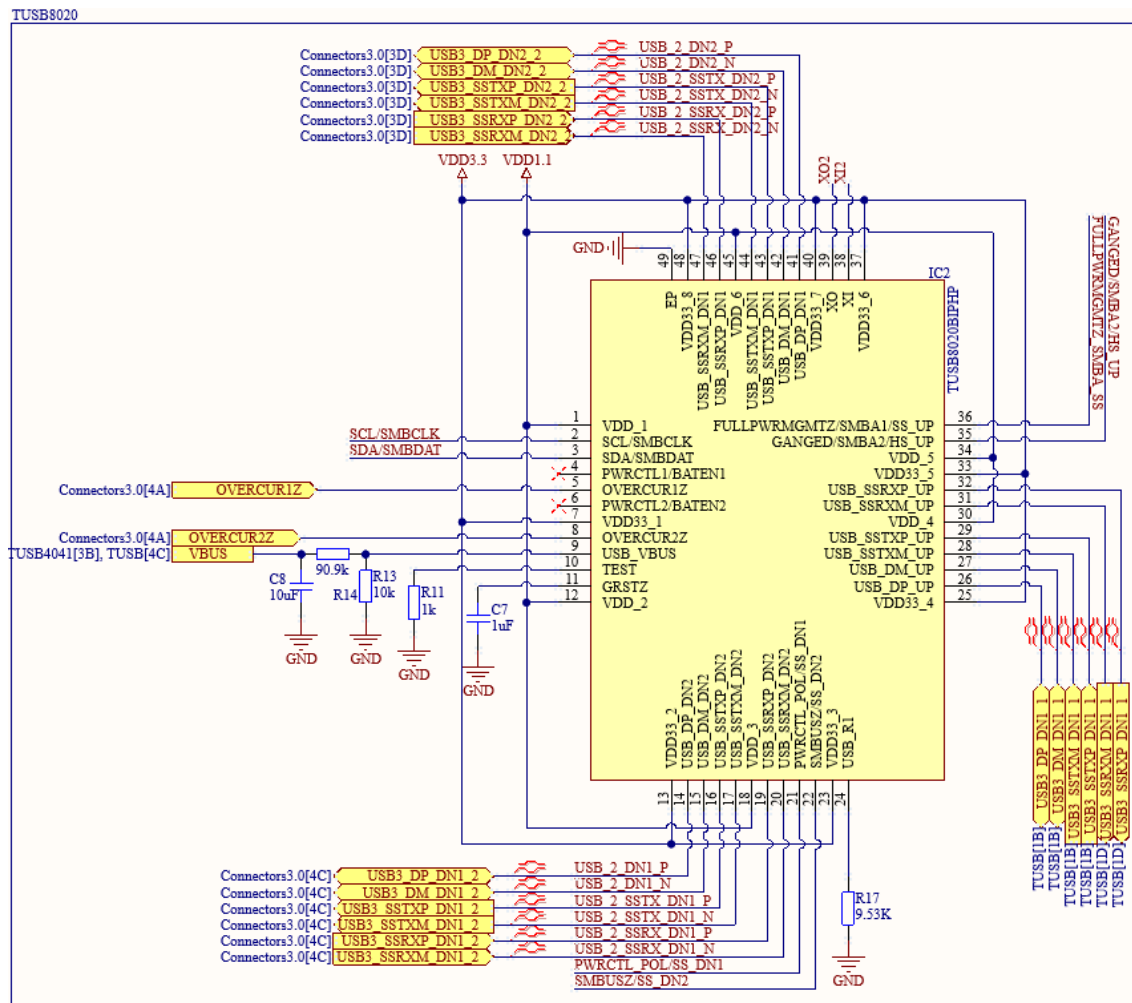


Figure 3.10: TUSB8020 Circuitry

In the circuit shown in Figure 3.9, a DIP switch is used due to the fact that the TUSB8020 has several configuration options in this specific setup, as shown in Figure 3.11. Starting with pins 2 and 3, labeled SCL/SMBCLK and SDA/SMBDAT respectively, these pins serve as communication channels for data from either an EEPROM or an SMBus host. The selection between I2C and SMBus modes is determined by pin 22, denoted as SMBUSz. When the value of pin 22 is low, the SMBus mode is activated; conversely, when the value is high, the I2C mode is selected.

If the DIP switch is in the off position for pin 22, the internal pull-up resistor will cause the value to be high, thereby enabling the I2C mode. Additionally, pins 2 and 3 have internal pull-down resistors, which means that if the DIP switch connections are off, the serial EEPROM/SMBus interface will be disabled.

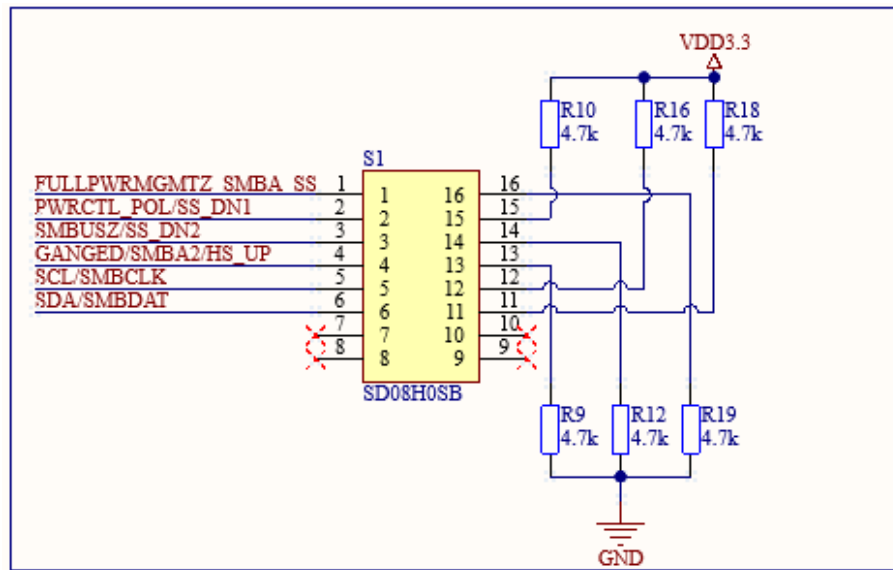


Figure 3.11: DIP Switch used in TUSB8020

The value of the resistors used in this DIP switch circuitry, 4.7 k Ω , is determined based on the output current of the pins with internal pull-up or pull-down resistors, which is approximately 250 μ A. This current corresponds to an internal resistance of about 13.2 k Ω , calculated using Ohm's law and considering the ± 250 μ A current and a typical supply voltage of 3.3 V.

To override the internal pull-up or pull-down resistor and ensure the input is firmly driven to the opposite logic level (either high or low), an external resistor with a lower resistance value must be used. This is because when the internal pull-up resistor 13.2 k Ω is attempting to pull a pin high, and a low logic level is required, the external resistor must have a smaller resistance to exert greater influence and effectively override the internal pull-up. For instance, a 4.7 k Ω resistor would pull the pin voltage more strongly than the internal 13.2 k Ω resistor, ensuring the desired logic level is achieved.

Pins 4 and 6 are designated for enabling battery charging and controlling the downstream ports. However, these pins were left unconnected because control of the downstream ports was implemented through the microcontroller on this board, allowing control via command line instead.

On the other hand, pins 5 and 8 were utilized and connected to a power switch to allow the IC to detect overcurrent events. In the event of overcurrent, this setup enables the IC to shut down communication to protect the system.

Pin 9, USB_VBUS, serves the same function as explained in Section 3.2.1, detecting when an active host is connected and ensuring the hub does not connect if USB_VBUS is low. Pin 10, TEST, is also consistent with the previous explanation, reserved for production test mode and typically pulled down to disable this feature. Similarly, pin 11, GRSTZ, function as a global power reset, bringing all internal registers of the TUSB8020 to their default state. The internal pull-up and external capacitor create an RC time constant delay, ensuring the TUSB8020 powers

up correctly and initializes the state machine to prevent entering any unknown states.

Pins 14, 15, 16, 17, 19, and 20 are designated for interfacing with the first downstream USB 3.0 port, while pins 41, 42, 43, 44, 46 and 47 are used to connect with the second downstream USB 3.0 port.

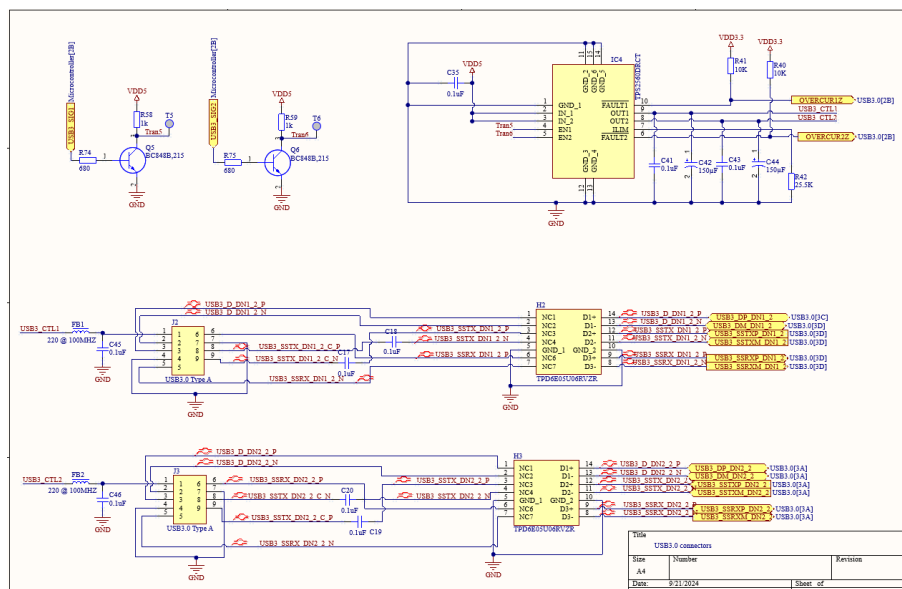
Pin 21 is linked to the DIP switch to configure different settings. Pin 21, labeled PWRCTL_POL, determines the polarity of the power control signals. If the PWRCTL_POL pin is set high, the power control signals will be active high; if set low, the signals will be active low.

Pins 26, 27, 28, 29, 31, and 32 serve to establish connections with the TUSB8041 IC, which is positioned in the upper layer of the design.

The subsequent pins mirror those described in Section 3.2.1. Pin 24 is used to connect a precision resistor, essential for accurate voltage regulation. Pin 35 determines whether the power control response to an overcurrent event will be individual or ganged, impacting how the downstream ports manage such events. Pin 36 relates to the power control capabilities, specifically whether power switching and overcurrent inputs are supported. Additionally, pins 38 and 39 are part of the crystal circuit, vital for maintaining the timing and frequency stability required for the IC's operation.

3.2.3 USB3.0 Connectors

This section provides details on the connections between the TUSB8020 IC and the two downstream USB 3.0 ports. Figure 3.12 shows the schematic illustrating how the IC is connected to the downstream ports.



Firstly, we have the TPS2560, a dual-channel power switch from Texas Instruments. This integrated circuit allows precise control of power output through its enable pins.

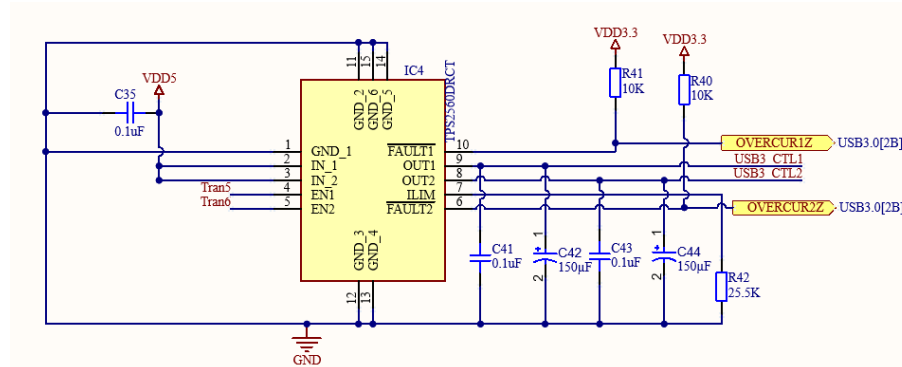


Figure 3.13: Power Switch

Based in the datasheet [16], the TPS2560 is designed to manage power distribution efficiently by providing overcurrent protection and adjustable current limits. It features two independent channels, each capable of controlling power to different parts of a system. The device incorporates built-in overcurrent protection, which helps safeguard against excessive current that could potentially damage connected components.

Each channel can be independently activated or deactivated via the enable pins, offering flexible control over power delivery.

In the TPS2560 device, pins 2 and 3 are designated for the input voltage. According to the datasheet, it is crucial to place a 100 nF capacitor close to these pins, specifically between the In and GND pins. This capacitor is essential for local noise decoupling. This measure mitigates ringing on the input caused by power-supply transients. Input capacitance may be required to limit voltage overshoot and prevent it from surpassing the device's absolute maximum voltage during severe transient conditions according with [16].

Pins 4 and 5 of the TPS2560 are the enable inputs, responsible for controlling the power in their respective channels. When these pins receive a logic low signal, the power switch is activated, allowing current to flow through the channel. These enable signals originate from the microcontroller, raspberry pi pico, presented in the board. By manipulating these signals, the user can control the logic levels sent to the TPS2560's enable pins.

Pins 6 and 10 on the TPS2560 are active-low open-drain outputs designed to signal fault conditions. These pins are used to indicate overcurrent or overtemperature events occurring on each respective channel. To ensure the FAULT pin returns to a high voltage level (logic 1) when not actively being pulled low, a 10 k Ω pull-up resistor is utilized.

Pin 7 of the TPS2560 is utilized to set the current limit threshold through the use of an external resistor. This resistor determines the maximum current that each channel can supply before the device intervenes to protect the circuit.

To set a current limit of 2 A per port, as specified for the TUSB8020, the value of the external resistor can be calculated using the formulas provided in the TPS2560 datasheet, [16]

$$I_{osmin}(mA) = \frac{61200}{R_{lim}^{1.056}} \quad (3.1)$$

$$R_{lim} = \left(\frac{61200}{I_{osmin}} \right)^{\frac{1}{1.056}} \quad (3.2)$$

The result will be 25.52kΩ. Choosing a resistor value lower than the calculated $R_{lim}=25.5k\Omega$, we use the I_{os} equations to determine the maximum current limit threshold based on this new resistor value.

$$I_{osmax} = \frac{52860}{R_{lim}^{0.957}} \quad (3.3)$$

With a 25.5 kΩ resistor, the calculated maximum current limit threshold is 2382 mA.

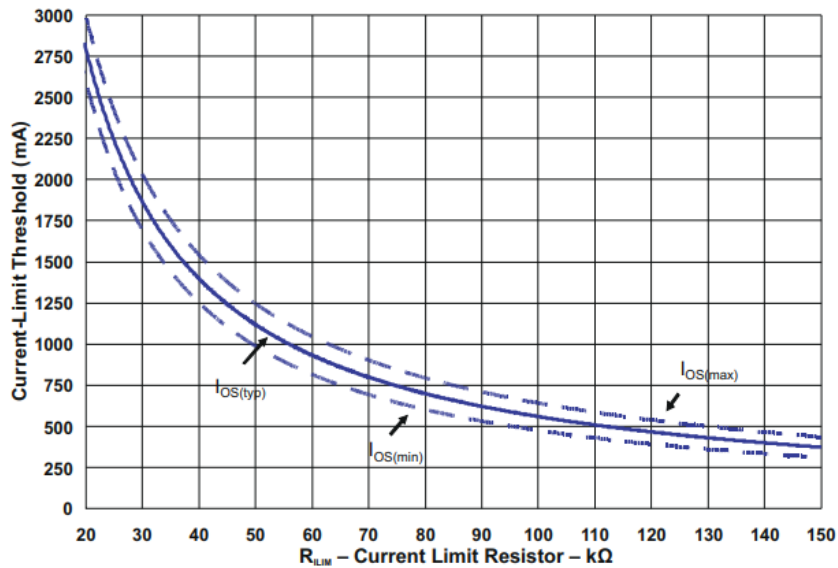


Figure 3.14: Current Limit Threshold, from [16]

Pins 8 and 9 from Figure 3.13 are designated as the power switch outputs. To manage large transient currents effectively, it is recommended to place a high-value electrolytic capacitor (150 μF) along with a 150 nF bypass capacitor on the output pin. This configuration helps in stabilizing the output and accommodating substantial transient loads.

This section of the design includes a circuit featuring BJT transistors, Q5 and Q6, as presented in Figure 3.15. These transistors are employed to step up the voltage received from the Raspberry

Pi Pico, raising it from 3.3 V to 5 V. This voltage boost is necessary to control the output of the power switch effectively.

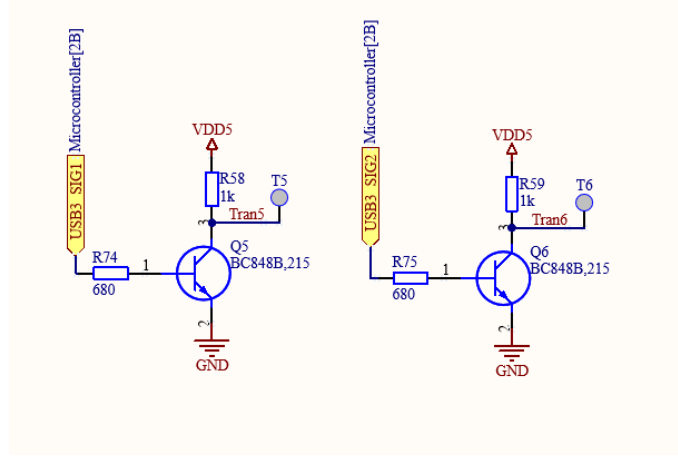


Figure 3.15: BJT transistors circuitry

The primary goal of this configuration is to control the power to the downstream ports via the microcontroller, specifically the Raspberry Pi Pico.

According to the datasheet, [7], the BJT transistor has a beta value of 290 in active zone. By powering the collector with 5 V and using a 1 k Ω resistor, a current of 5 mA is established, the base current was then chosen to set the transistor in the saturation region, being then 4 mA, with this value then the forced beta will be close to one.

The base resistor value can be determined using Ohm's Law. With 3.3 V provided by the Raspberry Pi GPIO pin and a base-emitter voltage drop of 0.7 V, the resistor value is calculated based on the current flowing through the base. The formula to find this resistor is:

$$R_b = \frac{V_{gpio} - V_{be}}{I_b} \quad (3.4)$$

Being V_{gio} 3.3 V and V_{be} 0.7 V, the calculated resistor value is approximately 650 Ω . Rounding to the nearest standard value, a resistor of 680 Ω is chosen.

As detailed in Section 3.2.1, the differential-line pairs are connected to an ESD protection device, and the transmitter lines are equipped with a capacitor.

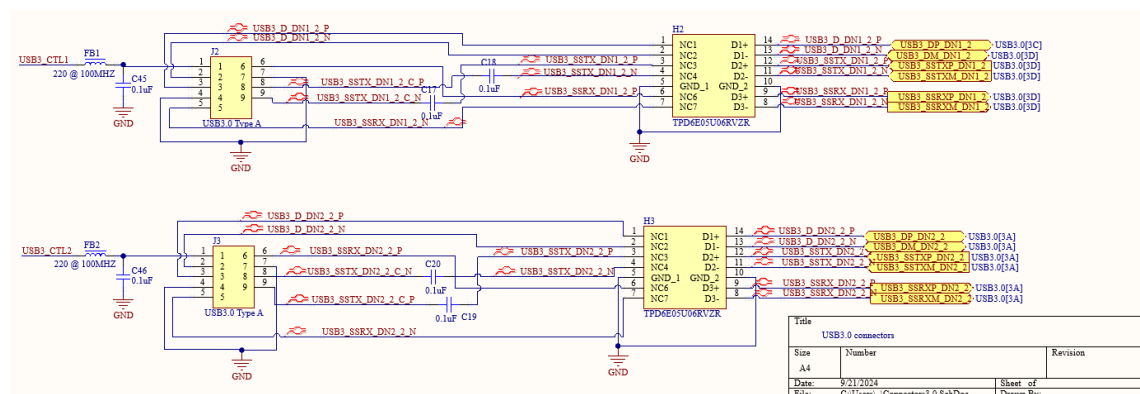


Figure 3.16: USB3.0 Ports Circuitry

Additionally, a ferrite bead is placed on the power pin (pin 1) to address both ESD (electrostatic discharge) and EMI (electromagnetic interference) concerns. A 0.1 μF capacitor is positioned on the USB connector side of the ferrite bead to offer a low-impedance path to ground for rapid rise-time ESD currents that may couple onto the VBUS trace from the cable.

3.2.4 USB 2.0 Data Distribution

The operating mode of the TUSB4041 aligns closely with the previously discussed TUSB8041 and TUSB8020 ICs. However, the key distinction lies in its communication capabilities: the TUSB4041 is limited to a maximum speed of 480 Mbps, consistent with USB 2.0 standards as shown in Figure 3.17. This IC is designed to handle USB 2.0 communication, making it suitable for applications where high-speed USB 3.0 is not required.

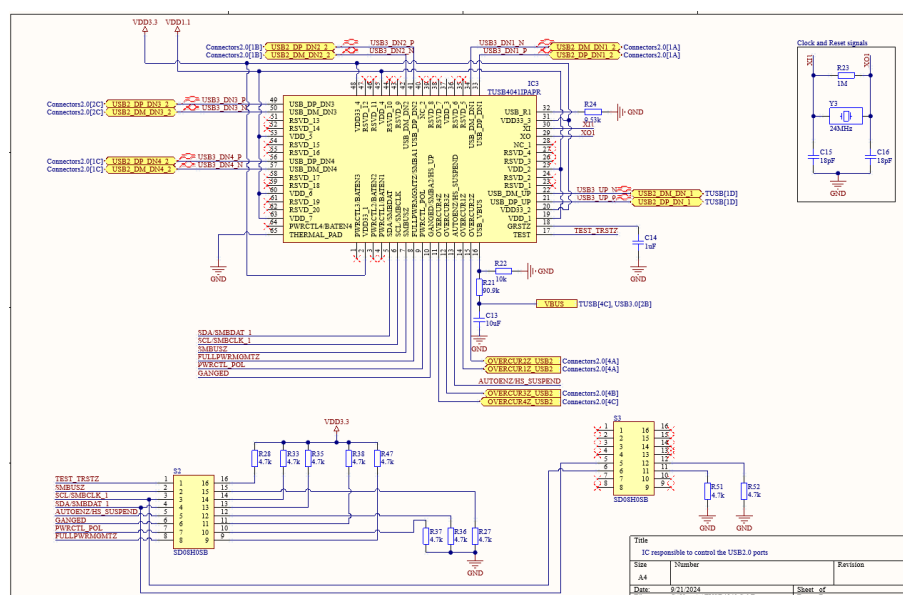


Figure 3.17: TUSB4041 Circuitry and their connections

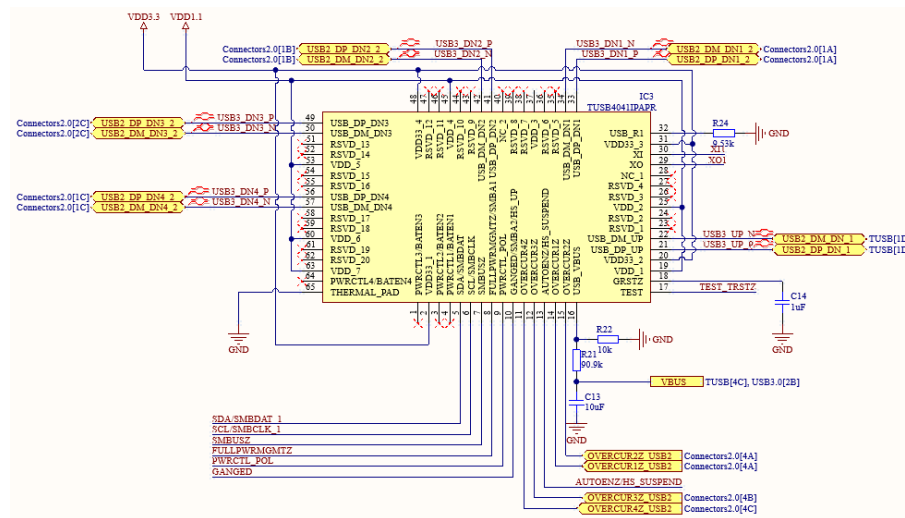


Figure 3.18: TUSB4041 Circuitry

Pins 5 and 6 from Figure 3.18 are designated for communication lines, supporting either I2C or SMBus protocols. The selection between these two communication modes depends on the value at pin 7, the SMBUSz pin. When this pin is high, the system defaults to I2C communication; when low, it switches to SMBus communication.

In this circuit, pins 5 and 6 are connected to two DIP switches, Figure 3.19. The design intention was to use one DIP switch to connect a pull-up resistor and the other to connect a pull-down resistor, enabling the selection of the desired communication protocol. However, since pin 5 and 6 internally have a pull-down resistor, connecting the second DIP switch was not strictly necessary. This minor oversight is inconsequential, as the redundant resistor has no effect on the system if the corresponding DIP switch is turned off.

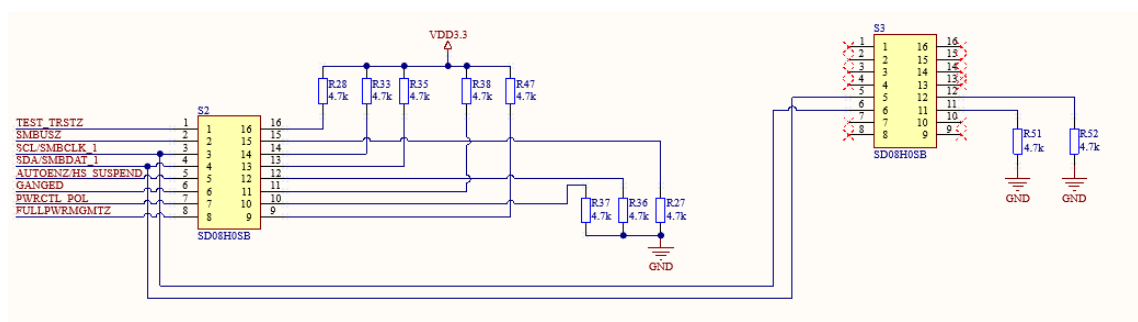


Figure 3.19: DIP Switches used in TUSB4041

Back to the Figure 3.18, the following pins, 8, 9, and 10, serve specific configuration purposes. Pin 8 determines whether power switching and overcurrent inputs are supported. Pin 9 sets the

polarity of the PWRCTL signal, defining whether it is active low or active high. Lastly, pin 10 configures the power control mode, deciding whether it operates individually or in groups.

Pins 11, 12, 14, and 15 serve as input pins that indicate whether an overcurrent event has occurred in the downstream ports.

Pin 13 is used to determine whether the automatic charge mode is enabled. This mode is applicable only to ports that are set up for battery charging when the hub is not connected.

Pin 16, USB_VBUS, functions as previously described in Section 3.2.1.

Pin 17 is reserved for factory testing and will not be utilized, thus the corresponding DIP switch will remain off. Pin 18 is designated for a global power reset, which resets the device's internal registers to their default state. Pins 21 and 22 are used for communication with the upstream port, in this case, the TUSB8041 on the preceding layer.

Many pins on the IC are labeled RSVD, such as pins 23 and 24, among others. These are reserved for internal use and should be left unconnected on the PCB. Pins 29 and 30 are part of the crystal circuit.

For the downstream USB2.0 ports, pins 33 and 34 handle communication with the first port, pins 41 and 42 with the second port, pins 49 and 50 with the third port, and pins 56 and 57 with the fourth port.

The remaining pins are dedicated to connections to the ground and the power supply, either 3.3V or 1.1V, ensuring the IC receives the necessary power for operation.

3.2.5 USB2.0 Connectors

This section outlines the connections to the four downstream USB 2.0 ports. The underlying principles and circuitry are largely consistent with those discussed in Section 3.2.3.



Figure 3.21: USB2.0 Connector Circuitry

3.2.6 RS232

The final communication involves converting USB 2.0 to RS232, which is facilitated by the FT232RL, a USB to UART interface capable of receiving data via USB 2.0 and converting it to RS232, RS422, or RS485 formats. This implementation is relatively straightforward. The following figure illustrates the circuitry used:

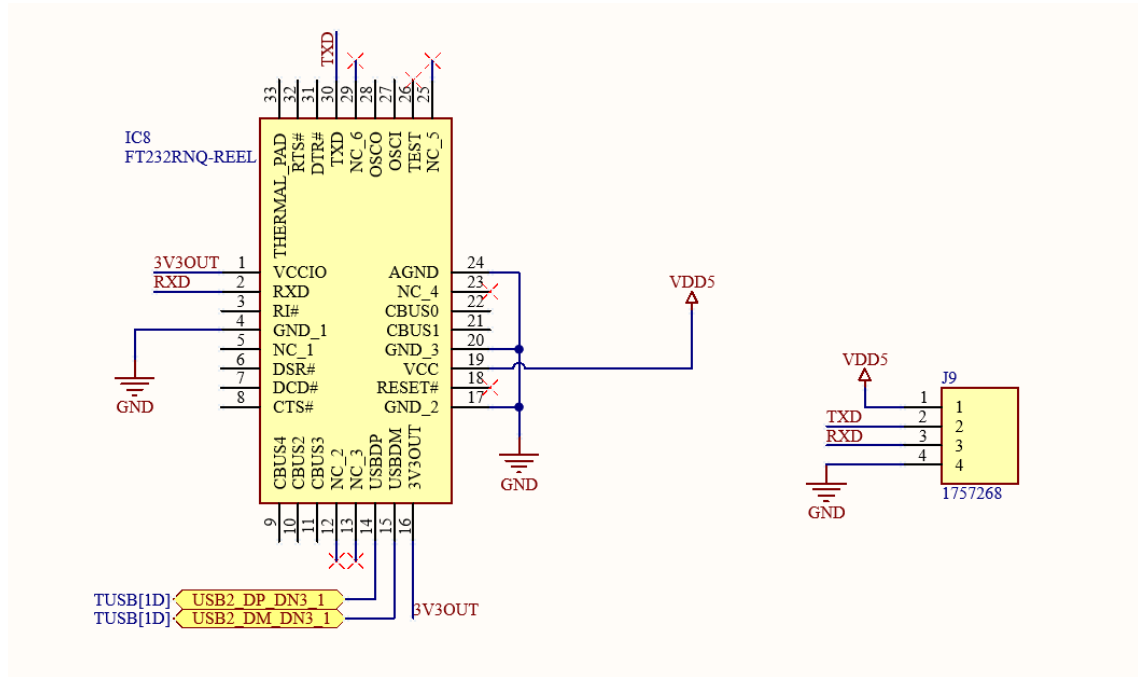


Figure 3.22: RS232 Circuitry

The goal of this communication setup is to provide a simple TX and RX interface with selectable output levels. Instead of using RS232, a TTL connection is chosen. While RS232 output levels range from -13 V to +13 V, TTL communication levels are limited to either 0 to 3.3 V or 0 to 5 V.

The first pin, VCCIO (pin 1), can accept voltages ranging from 1.8 V to 5.25 V according to the [27]. By connecting this pin to pin 16, 3V3OUT, the signal levels will be set to +3.3 V.

Pin 16 is an output pin providing 3.3 V from the integrated regulator. This output supplies the internal +3.3 V to the USB transceiver cell.

Pin 2 serves as the Receiving Asynchronous Data Input, and pin 30 is the Transmit Asynchronous Data Output. Pins 14 and 15 handle communication with the TUSB8041 via the USB data lines, USB data plus, and USB data minus.

The connector J9 is provided to facilitate connections with the respective device for communication.

3.2.7 Power Supply

This section is responsible for supplying power to all the circuitry analyzed previously.

To simplify the implementation and address the critical design considerations for USB 3.0 differential-line pairs, the power supply is designed to be provided externally.

The objective is to minimize EMI, particularly due to my inexperience in PCB design and creation. By relying on an external power supply, potential interference issues are mitigated, leading to a more reliable and efficient design.

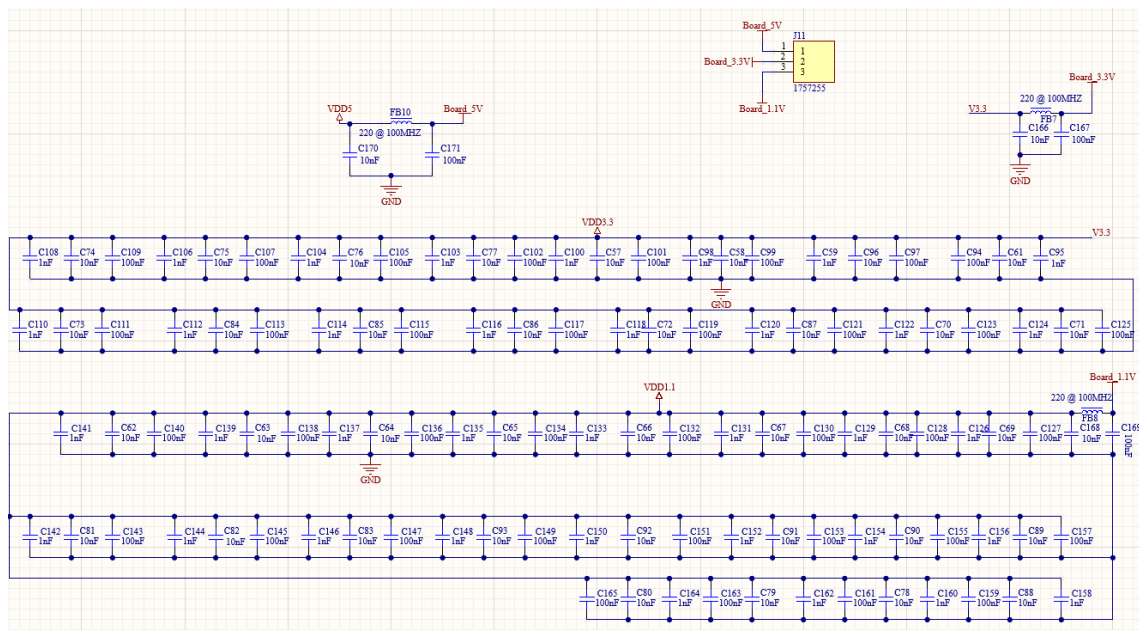


Figure 3.23: Power Supply Circuitry

The extensive use of capacitors in the design is due to the inclusion of three capacitors for each VDD1.1 or VDD3.3 pin of each IC. The choice of capacitor values - 1 nF, 10 nF and 100 nF - ensures good frequency response for the circuitry, effectively stabilising the power supply and filtering out noise over a wide frequency range. Smaller capacitors typically exhibit better frequency response, making them more effective at filtering high-frequency signals and reducing noise. In contrast, larger capacitors tend to have higher power density and are more capable of providing current during rapid changes in load conditions.

Additionally, ferrite beads are employed to reduce noise originating from the external power supply.

J11 is the connector designated to receive the power supply. However, an important oversight occurred during the design: there is a lack of an additional pin for the ground connection. As it stands, J11 has only three pins dedicated to providing 5 V, 3.3 V, and 1.1 V, respectively. This omission means there is no dedicated ground pin, which is crucial for completing the power circuit and ensuring proper operation of the device.

In the future Section 3.4, the method used to overcome this oversight will be explained in detail.

3.3 PCB Design

In this Section, the entire process of creating the printed circuit board (PCB) will be explained. Using Altium Designer and its comprehensive functionalities, the design process begins with creating the stackup layer and defining the number of layers required for the PCB.

Initially, the stackup layer is meticulously designed to ensure optimal signal integrity and power distribution. The number of layers is determined based on the complexity and requirements of the circuit, balancing factors such as signal routing, power planes, and ground planes.

Following the stackup layer configuration, the next step involves strategically positioning the components on the board. This strategic placement is crucial for simplifying connections and routing, as well as optimizing the overall performance and manufacturability of the PCB. Components are arranged to minimize trace lengths, reduce potential interference, and ensure efficient use of space.

Finally, the routing process is carried out. During this phase, electrical connections between components are established by meticulously laying out traces. Altium Designer's advanced routing tools facilitate precise control over trace width, spacing, and layer transitions, ensuring that the PCB meets electrical and mechanical design constraints.

3.3.1 Altium Designer

For this work, the PCB was fully developed using Altium Designer. Altium Designer is a product of Altium Limited [26], an Australian multinational software company specializing in electronic design automation (EDA) software for engineers who design printed circuit boards (PCBs). Altium Limited is well-known for providing powerful tools that streamline the PCB design process, enabling engineers to create sophisticated electronic systems with greater efficiency and accuracy.

This software offers an integrated and user-friendly platform that encompasses everything from schematics and layout to simulation and analysis. Widely regarded as the best PCB design software, Altium Designer proved to be an invaluable tool for this project, facilitating efficient and effective PCB development.

Although Altium Designer is not a free tool, it is highly utilized within Bosch, making it more convenient to use this software for the project. Its widespread adoption at Bosch ensures familiarity and ease of access, facilitating smoother and more efficient PCB design and development processes.

3.3.2 Stackup Layer

As discussed in the paper by Kevin Cai, Wei Jiang, Bidyut Sen, and Guoan Wang titled "Practical High-Speed PCB Stackup Tool Generation and Validation" [24], the technology of multilayer printed circuit boards (PCBs) has advanced rapidly. This progress is attributed to multilayer PCBs' ability to reduce circuit size, enhance design flexibility, and mitigate electromagnetic interference

(EMI) through the strategic arrangement of power and ground layers. Concurrently, the complexity and density of multilayer PCBs used in modern commercial electronic devices such as cellphones, laptops, and servers have steadily increased over the past few decades.

To achieve effective routing capabilities and fulfill system requirements related to signal transmission, as well as mechanical and thermal considerations, it is essential to specify PCB thickness and layer thickness, including both substrate and metal layers, in a precise and clear manner.

Starting with the definition of the number of layers, as outlined in the paper "High-Speed Layout Guidelines for Signal Conditioners and USB Hubs" by Texas Instruments, [21], it is recommended that high-speed signals be routed on the top or bottom layer with an adjacent ground (GND) layer. Additionally, high-speed traces should not be routed under or near crystals, oscillators, clock signal generators, and switching power regulators to avoid interference. Therefore, the design strategy involves placing all high-speed traces (critical signals) on the top layer and allocating non-critical signals, such as those from crystals and low-speed signals, to the bottom layer.

To prevent interference in the signals, a ground layer should be used to shield the signals in the adjacent layers, [21], resulting in a total of four layers thus far. However, another layer is needed for the power plane.

Despite this need, it is not advisable to use a total of five layers. There are several reasons for this according with [25]:

- **Cost Considerations:** Odd-layer PCBs are generally more expensive due to the asymmetry, which requires additional processing. The raw materials for odd-numbered PCB boards are slightly less expensive than those for even-numbered boards, but the overall manufacturing cost is higher.
- **Mechanical Stability:** Odd-layer circuit boards are prone to bending. When the PCB cools after the multi-layer circuit bonding process, the differing lamination tension between the nuclear structure and the foil-coated structure can cause the PCB to bend. The risk of bending increases with the thickness of the circuit board. To mitigate this risk, it is essential to use balanced lamination, which is easier to achieve with an even number of layers.

In the Figure 3.24, the detailed schematic layout of a multilayer printed circuit board (PCB) is depicted. It is evident that the upper and lower parts of the PCB are entirely symmetrical with respect to the central symmetry line. This symmetrical design helps to ensure balanced lamination, which is crucial for maintaining the mechanical stability and minimizing the risk of bending in the PCB [24].

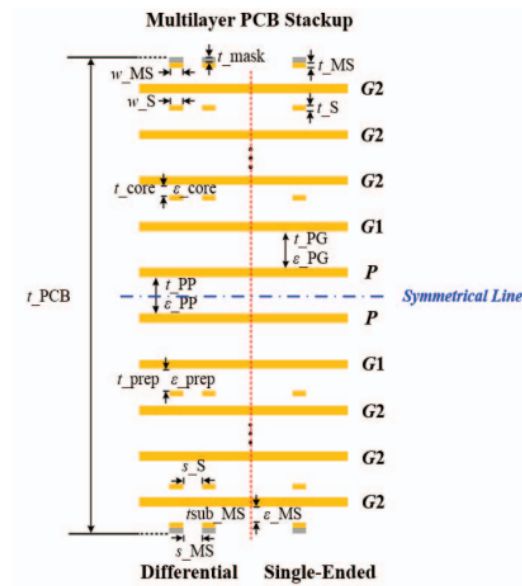


Figure 3.24: Multilayer Symmetry, from [24]

With this symmetrical design, the total number of layers will be six. The two central layers are designated for power distribution, accommodating the board's requirement for three different voltage levels. This arrangement simplifies power management and ensures that the PCB maintains a balanced structure, providing effective power distribution across the board.

Based on the table provided in the paper [21], we can analyze and determine the optimal stackup for a six-layer PCB. The goal is to achieve a balance between signal integrity, Electro-magnetic compatibility and decoupling.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
Layer 1	SIGNAL	SIGNAL	SIGNAL	GROUND	SIGNAL	SIGNAL	SIGNAL
Layer 2	GROUND	SIGNAL	GROUND	POWER	GROUND	GROUND	GROUND
Layer 3	SIGNAL	POWER	POWER	SIGNAL	POWER	POWER	POWER
Layer 4	SIGNAL	GROUND	POWER	SIGNAL	SIGNAL	GROUND	GROUND
Layer 5	POWER	SIGNAL	GROUND	POWER	GROUND	NOT USED	SIGNAL
Layer 6	SIGNAL	SIGNAL	SIGNAL	GROUND	SIGNAL	SIGNAL	SIGNAL
Decoupling	Good	Good	Good	Good	Good	Good	Good
EMC	satisfactory	Bad	Good	satisfactory	satisfactory	Good	Good
Signal Integrity		Bad	Good	Bad	Good	Good	Bad

Figure 3.25: Possible Board Stack-up on a Six-Layer PCB, from [21]

Given that high-speed signals should be routed with an adjacent ground layer, and referencing the information presented in Figure 3.25, the chosen stack-up for this PCB is as follows: signal, ground, power, power, ground, and signal. This configuration corresponds to model 3 from Figure 3.25.

Bosch Security Systems has established a partnership with EuroCircuits, a PCB manufacturer. EuroCircuits specializes in producing prototype and small batch boards at its facilities in Germany and Hungary.

Given that this manufacturer will be responsible for producing the PCB, it is valuable to review their website to gain insights into the guidelines and processes involved in PCB creation.

EuroCircuits offers a webpage that allows users to create simulations for PCBs, including aspects such as layers, material, and thickness. This webpage features a tool called the "Buildup Editor," which enables users to visualize and configure the stackup of a PCB with 6 layers [12].

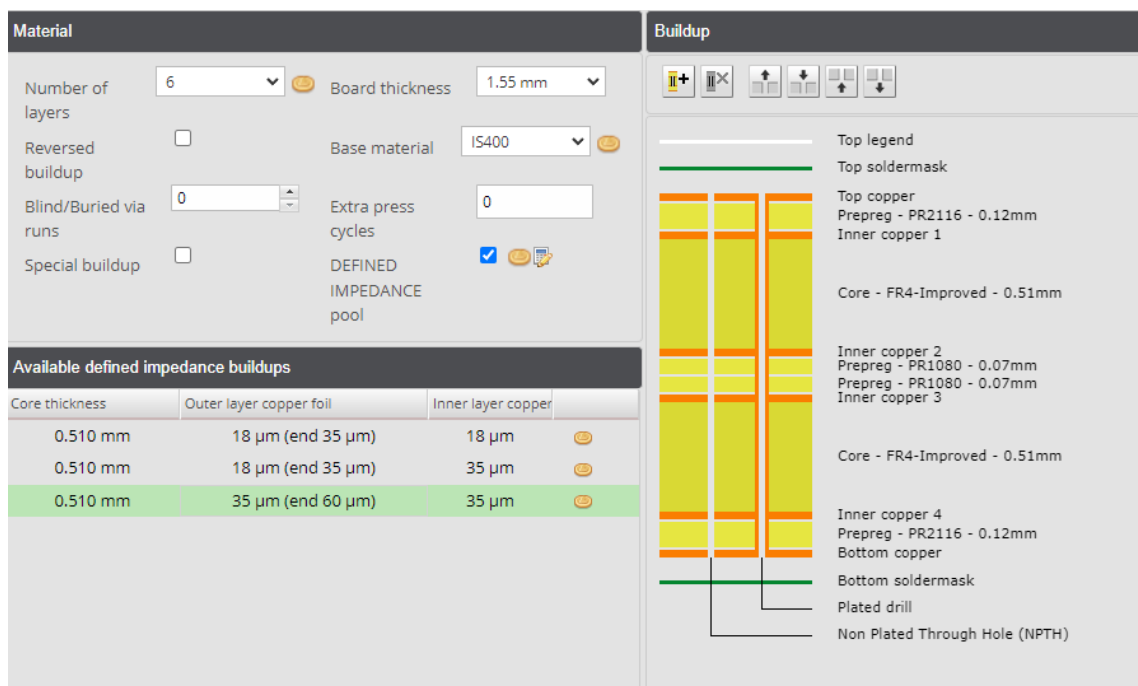


Figure 3.26: EuroCircuits Buildup Editor, from [12]

In the "Buildup Editor" tool, it is possible to select the thickness of the outer and inner layer copper foil, with options of 35 µm and 18 µm. The decision was made to use 35 µm thickness for both inner and outer layer copper foils. This choice is due to the fact that thicker copper layers reduce DC resistance and voltage drop, making the PCB more suitable for high-current or high-temperature applications, according with [3].

With this information, it was possible to understand how EuroCircuits creates the PCB stackup and the materials they use for it. Between layers 1 and 2, layers 3 and 4, and layers 5 and 6, prepreg material is used, while between layers 2 and 3 and layers 4 and 5, a core material is employed.

Based in the information available in Altium forum [35], the core is essentially one or more prepreg laminates that are pressed, hardened, and cured with heat. This core is plated with copper foil on each side. The prepreg material itself is impregnated with a resin that is hardened but left uncured. There can be differences in the dielectric constants between PCB core and prepreg

materials, depending on the resin content, type of resin, and glass weave used. This distinction is important for the electrical performance and overall stability of the PCB.

With knowledge of the materials used for the PCB, it is now feasible to create the stackup in Altium Designer by specifying the parameters for material type, thickness, dissipation factor (Df), and dielectric constant (Dk).

+ Add ✎ Modify 🗑 Delete							
#	Name	Material	Type	Thickness	Dk	Df	Weight
	Top Overlay		Overlay				
	Top Solder	Solder Resist	Solder Mask	0.01016mm	3.5		
1	Top Layer 1	CF-004	Signal	0.035mm			1oz
	Dielectric 4	PR2116	Prepreg	0.11999mm	4.13	0.0186	
2	GND	CF-004	Plane	0.035mm			1oz
	Dielectric 2	FR4	Core	0.51001mm	3.32	0.004	
3	VCC		Signal	0.035mm			1oz
	Dielectric 1	PP-017	Prepreg	0.12954mm	4.3	0.02	
4	VCC_1		Signal	0.035mm			1oz
	Dielectric 3	FR4	Core	0.51001mm	3.32	0.004	
5	GND_1	CF-004	Plane	0.035mm			1oz
	Dielectric 5	PR2116	Prepreg	0.11999mm	4.13	0.0186	
6	Bottom Layer 1	CF-004	Signal	0.035mm			1oz
	Bottom Solder	Solder Resist	Solder Mask	0.01016mm	3.5		
	Bottom Overlay		Overlay				

Figure 3.27: Stackup defined in Altium Designer

It is possible to obtain the stackup of the PCB with a detailed description of each layer as follows:

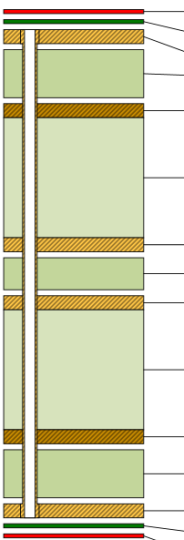
Layer Stack Legend						
	Material	Layer	Thickness	Dielectric Material	Type	Gerber
		Top Overlay			Legend	GTO
	Surface Material	Top Solder	0.01mm	Solder Resist	Solder Mask	GTS
	CF-004	Top Layer 1	0.04mm		Signal	GTL
	Prepreg		0.12mm	PR2116	Dielectric	
	CF-004	GND	0.04mm		Internal Plane	GP1
	Core		0.51mm	FR4	Dielectric	
	Copper	VCC	0.04mm		Signal	G1
	Prepreg		0.08mm	PR-1080	Dielectric	
	Copper	VCC_1	0.04mm		Signal	G2
	Core		0.51mm	FR4	Dielectric	
	CF-004	GND_1	0.04mm		Internal Plane	GP2
	Prepreg		0.12mm	PR2116	Dielectric	
	CF-004	Bottom Layer 1	0.04mm		Signal	GBL
	Surface Material	Bottom Solder	0.01mm	Solder Resist	Solder Mask	GBS
		Bottom Overlay			Legend	GBO
Total thickness: 1.57mm						

Figure 3.28: Stackup defined in Altium Designer with description

3.3.3 Defined Differential Impedance

To understand the defined impedance, it is necessary to grasp the concept of a differential-line pair and its operation.

In its most basic form, a differential-line pair consists of two transmission lines that carry signals with equal magnitude and opposite polarity. The key characteristic of these two signals is that they are equal in magnitude but opposite in phase, and they are tightly synchronized with each other [36].

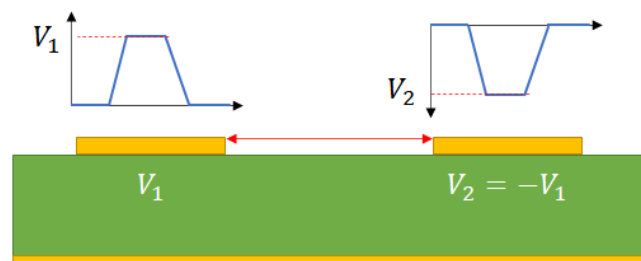


Figure 3.29: Differential signaling on a differential-line pair, from [2]

A differential signal offers optimal noise immunity. This is because any noise that couples with a pair of closely spaced parallel conductors typically appears equally on both conductors, manifesting as common-mode noise. Since the receiver only responds to the voltage difference between the lines, it can effectively ignore crosstalk noise in a twisted-pair cable, as the noise is picked up equally by both lines [8].

When multiple signals are transmitted within the same cable bundle over any distance, mutual coupling between the wires can cause portions of one signal to be transferred to another, and vice versa. This phenomenon is known as crosstalk. Crosstalk is not limited to cables but can occur in any situation where there is unwanted interaction between channels that are supposed to be uncoupled. The coupling responsible for crosstalk can be primarily capacitive, inductive, or due to transmission-line effects [45].

Single-ended impedance refers to the impedance of a trace relative to the ground plane while differential impedance is the impedance between the two signals of a differential-line pair [21].

It is influenced by factors such as the geometric dimensions of the conductors, the gap between them, the thickness of the dielectric, and the dielectric constant of the material. Altium Designer can automatically calculate these values and then define the differential impedance based on the specified parameters [4].

According to the information available in the datasheet from Texas Instruments [21], the differential impedance for USB 3.0 and USB 2.0 is 90 ohms $\pm$ 15%, and the single-ended impedance is 45 ohms $\pm$ 15%.

Before defining the differential impedance, and consequently, the trace width and the gap between them, it is necessary to review the manufacturing capabilities available on the EuroCircuits website to determine the permissible trace thickness and spacing that can be achieved.

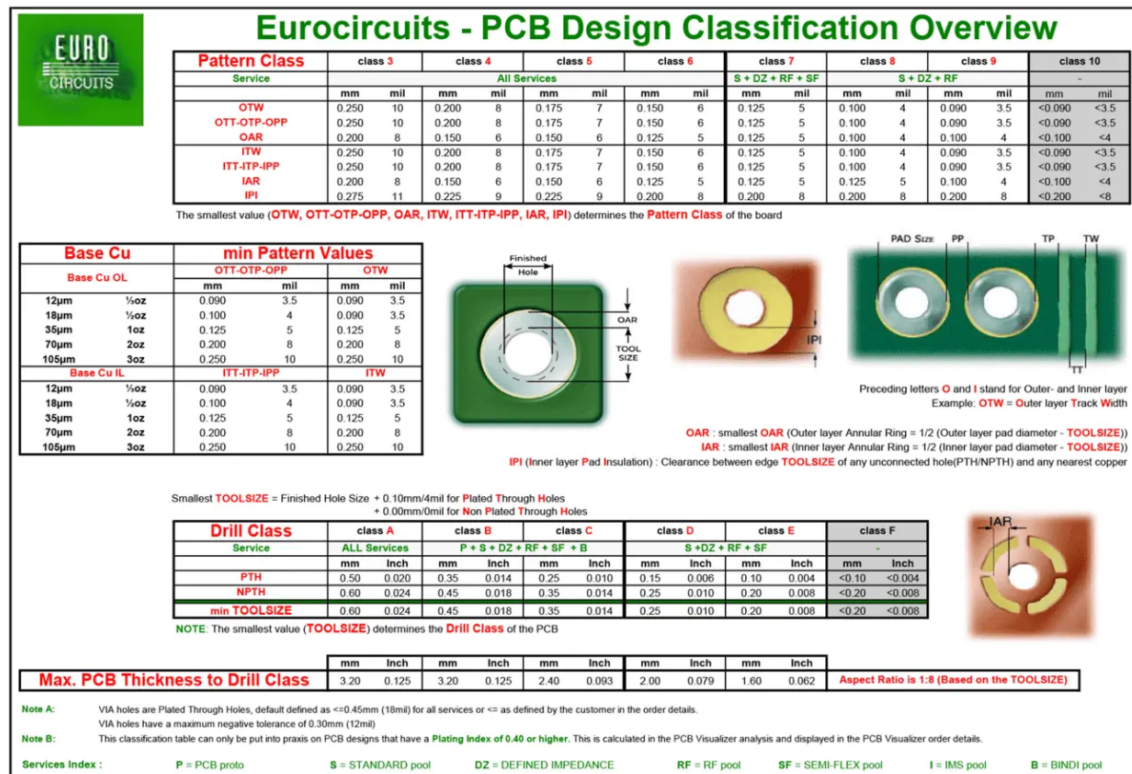


Figure 3.30: PCB Design Classification Overview, from [13]

In the first table of the Figure 3.30, different sizes of trace width (OTW) and gap between traces (OTT) are presented, categorized into classes. As the class number increases, the trace width and gap size decrease, resulting in a higher cost and a more expensive PCB.

Various combinations of trace width and gap can be utilized as long as the differential impedance is maintained. To achieve the desired 90 ohms, a trace width of 8 mil and a gap of 10 mil were selected.

In Altium Designer, these values can be entered using the Layer Stack Manager functionality. This includes inputting the target impedance and the target tolerance.

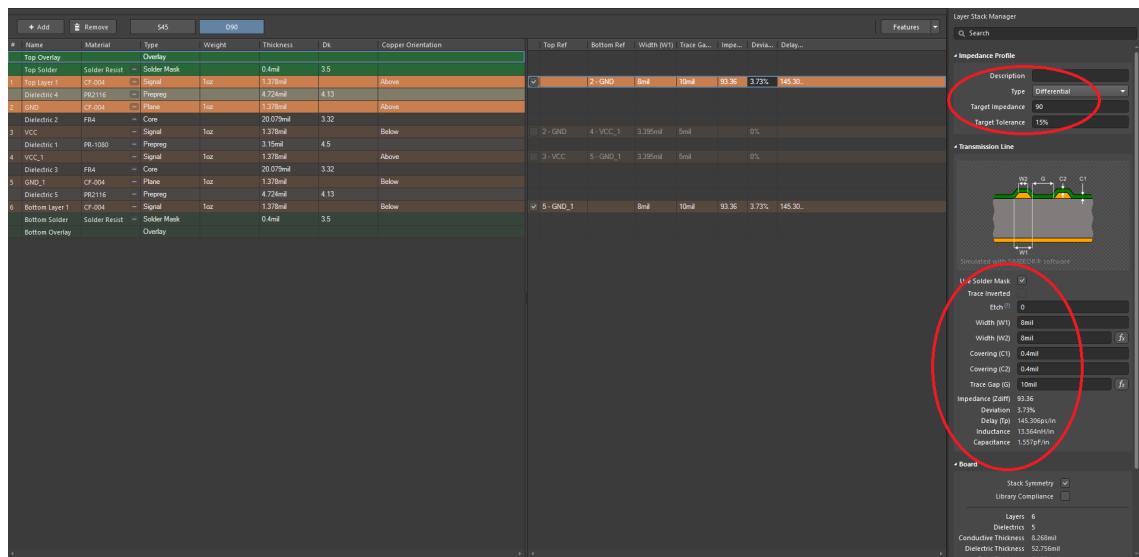


Figure 3.31: Differential Impedance parameters in Altium Designer

If the impedance requirements are not met, Altium will display a notification.

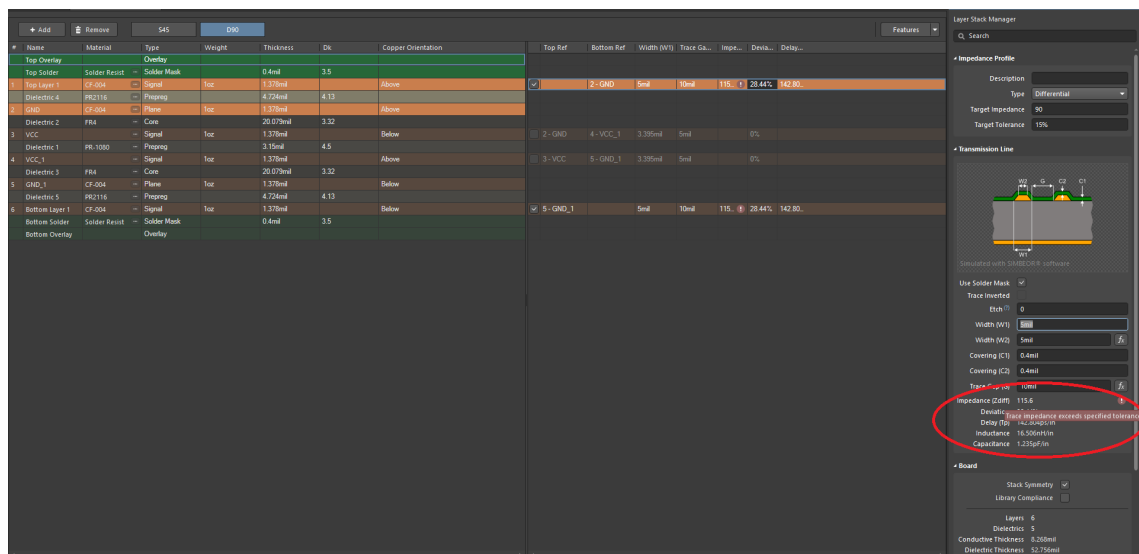


Figure 3.32: Message when the target differential impedance is not met

The same process was followed for the single-ended impedance, where the target impedance is 45 ohms. In this case, there is no parameter for the gap between traces since this impedance pertains to a single trace.

3.3.4 Components Placement

The placement of the components required considerable attention because it significantly impacts the difficulty of the routing process. A well-planned placement can make the routing job easier,

while a poor placement can make it much more challenging.

In the following picture 3.33, the arrangement of the most crucial elements (ICs, USB connectors, Raspberry Pi Pico, and DIP switches) on the top layer is shown. The strategy was to place the upstream port on one side of the board and the downstream ports on the opposite side. Specifically, the four USB 2.0 downstream ports were positioned on the left side of the board, while the two USB 3.0 downstream ports were placed at the bottom of the board next to the RS232 connector. The primary IC, TUSB8041, was located near the upstream connector, and the remaining ICs were positioned close to their respective downstream ports.

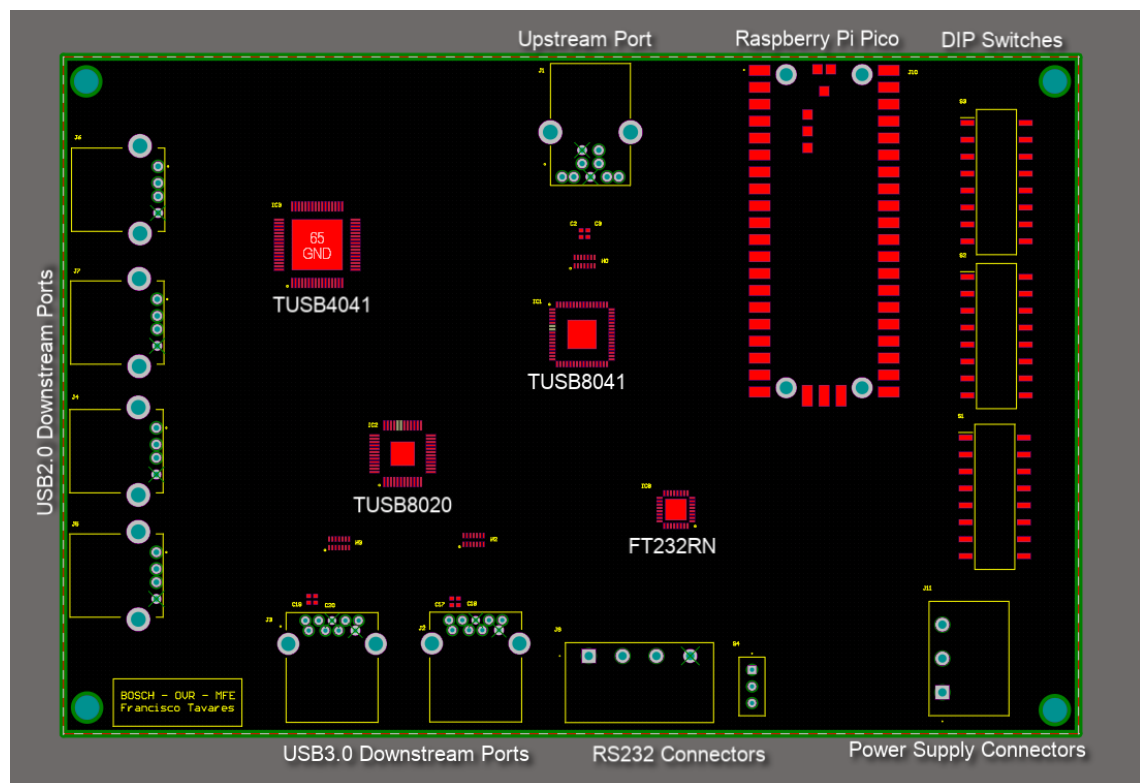


Figure 3.33: Components Placement on the top layer

On the bottom layer, the capacitors were strategically placed near the power supply pins of each IC, as well as near the crystal circuitry. This careful placement ensures optimal performance and stability by minimizing noise and providing adequate decoupling for the power supply.

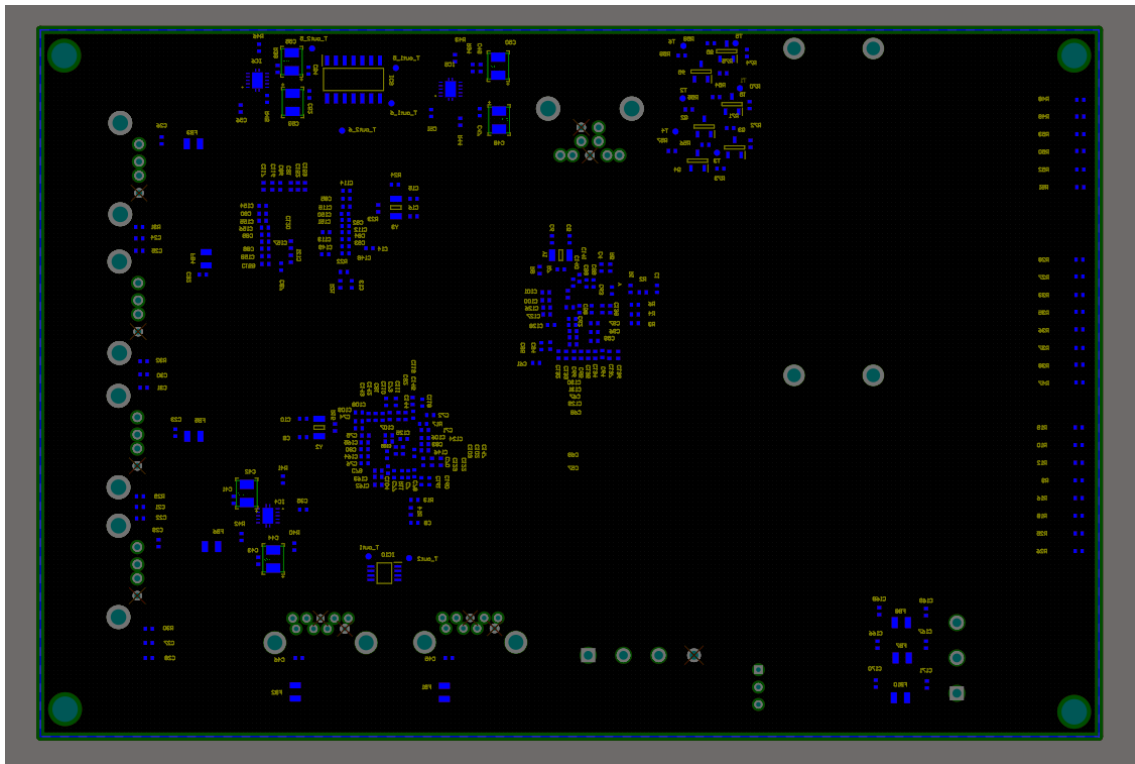


Figure 3.34: Components Placement on the bottom layer

3.3.5 Routing

Routing is a critical aspect of PCB design, and several important details were taken into consideration during this process. According to the "High-Speed Layout Guidelines for Signal Conditioners and USB Hubs" [21], high-speed differential signals must not have any probes or test points placed on them, as this can introduce noise and degrade signal integrity. Additionally, high-speed traces should not be routed under or near crystals, oscillators, switching power regulators, or magnetic devices to avoid interference and potential signal degradation.

Furthermore, traces should not form 90-degree angles, as such sharp bends can cause signal reflection and impedance discontinuities. Instead, any bends in high-speed differential signals should maintain an angle greater than 135 degrees. This approach ensures that bends are as gradual as possible, minimizing length mismatches caused by the bends and thereby reducing the impact on electromagnetic interference (EMI).

It is also crucial to minimize the trace lengths of differential-line pairs to reduce signal delay and potential crosstalk. By adhering to these guidelines, the design ensures optimal signal integrity and overall performance of the high-speed circuits.

The etch lengths of differential-line pairs within a given group, such as SSRX and SSTX, do not need to be identical. However, it is essential to minimize the trace lengths overall. For individual differential-line pairs, it is crucial to match the etch lengths of the positive and negative

lanes within each pair, such as DP and DM or SSRXP and SSRXM. This alignment helps ensure signal integrity by reducing intra-pair skew, which refers to the difference in length between the + and - traces of a differential-line pair [21].

To address intra-pair skew, serpentine routing techniques can be employed to match the lengths of the traces as closely as possible at their mismatched ends. It is advisable to maintain a difference of less than 5 mils between a single-ended differential-line pair signal and its complement to ensure signal coherence and optimal performance [21].

In the Figure 3.35, it is evident that serpentine routing was utilized to equalize the etch lengths of both traces within the differential-line pair. This method successfully achieved a length discrepancy of only 1 mil, which is well within the recommended tolerance of 5 mils as advised by Texas Instruments.

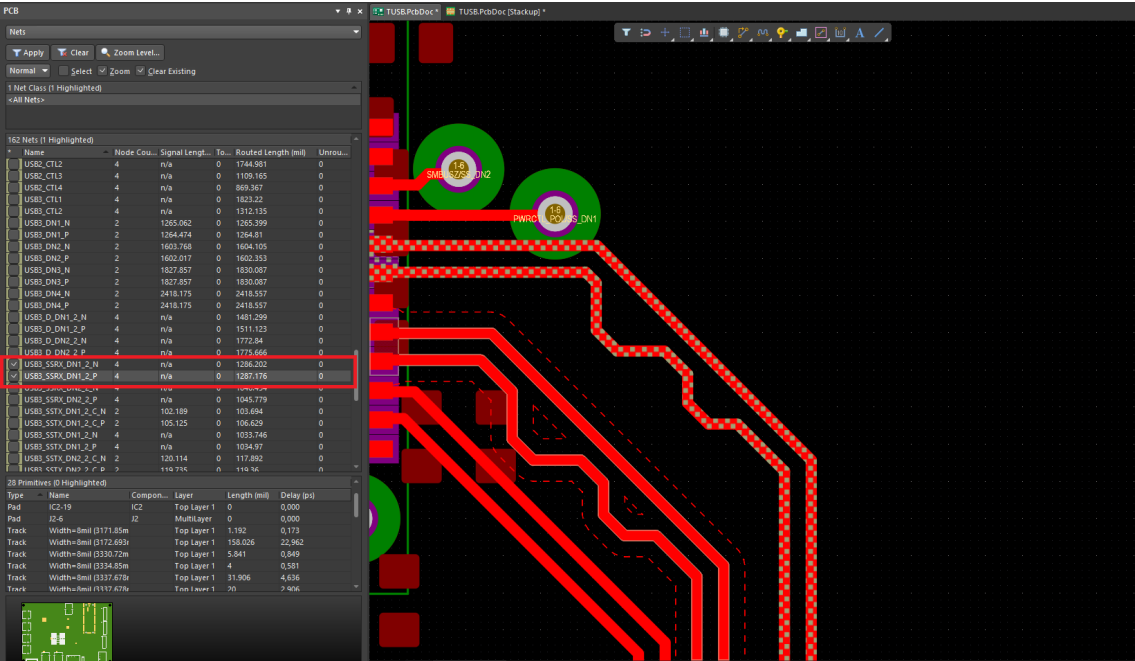


Figure 3.35: Serpentine Routing

According to guidelines for high-speed signal routing [21], it is ideal to maintain all traces within the same plane to ensure a consistent ground reference from the signal’s origin to its termination. However, as depicted in Figure 3.37, it was not feasible to adhere to this guideline completely.

To address this, vias were employed to connect the ground planes, thereby ensuring continuous grounding and uniform impedance throughout the signal path. These vias were strategically placed in a manner that is symmetrical relative to the signal transition vias, thereby maintaining signal integrity and minimizing potential disruptions.

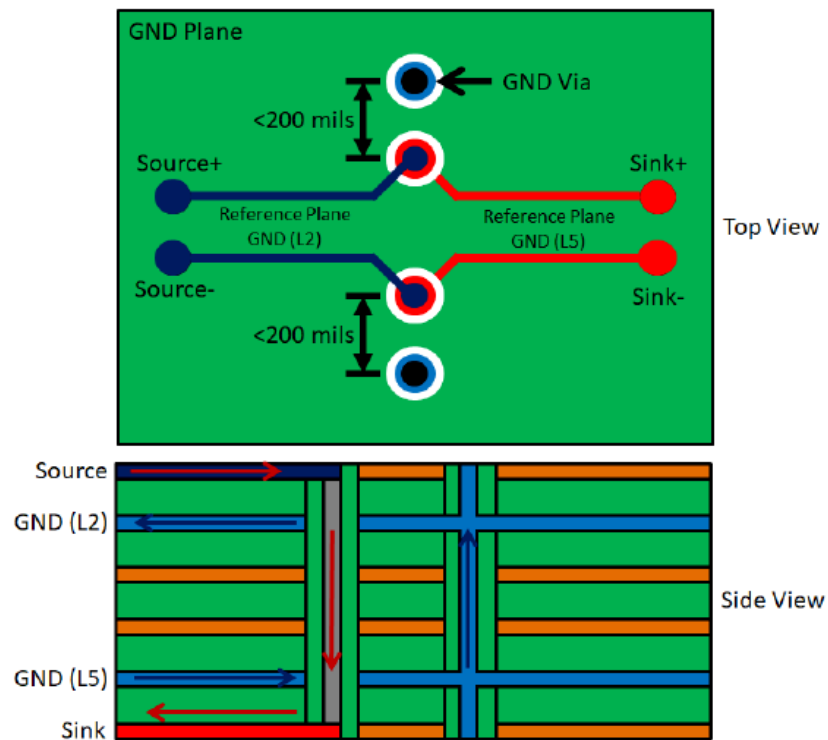


Figure 3.36: Return Path for differential-line pair Vias with Ground Vias, from [21]

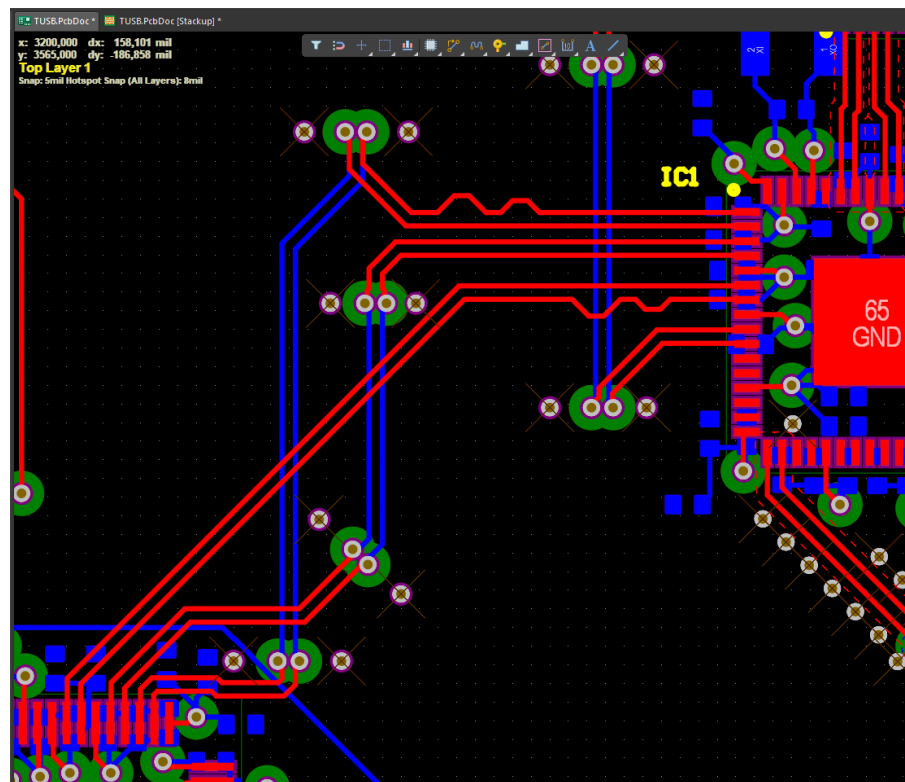


Figure 3.37: Ground Vias in differential-line pairs

At higher frequencies, return currents travel along the path of least impedance, which is typically the reference plane closest to the signal layer. Consequently, it is advisable to place a ground or power plane directly above or below the signal layer. This arrangement provides a stable return path, minimizing impedance variations and reducing electromagnetic interference (EMI) problems [21]. This is way ground vias were used when differential-line pair traces changed the signal plane.

3.3.6 Polygon Pours

A polygon pour is a complex design element composed of simpler, primitive objects.

These polygon pours are utilized to create either solid or patterned (lattice) areas on a PCB layer, employing Region objects or a combination of Track and Arc objects. Also known as copper pours, polygon pours are similar to regions but offer additional flexibility. They can cover irregularly shaped areas on a board and automatically route around existing components, connecting only to elements within the same net as the pour.

On signal layers, a solid polygon pour can be employed to define areas that handle substantial power supply currents or to establish ground connections that provide electromagnetic shielding. This approach helps in managing high current densities and mitigating electromagnetic interference by ensuring continuous and effective grounding across the board [1].

In the PCB design, polygon pours were employed in multiple ways: on layers 1 and 6 to provide electromagnetic shielding, and on layers 3 and 4 to manage substantial power supply currents.

Since layers 2 and 5 serve as dedicated ground planes, the use of polygon pours was not necessary for these layers.

Given that the circuits operate with three different voltage levels and only two PCB layers are dedicated to power supply, polygon pours proved to be a valuable tool. They were used to divide layer 4 into distinct sections, Figure 3.38, allowing for the effective management of two voltage levels (5V and 1.1V). This segmentation ensures that each voltage level is properly handled and isolated within the designated power supply layers.

In Figure 3.38, two distinct polygon pours are illustrated on layer 4: the inner polygon is dedicated to distributing 1.1V, while the outer polygon is allocated for the 5V supply.

This configuration effectively segregates the power distribution, enabling each voltage level to be delivered to its specific destinations without the necessity for additional traces. Instead, power is routed efficiently through vias, which directly connect to the designated layers. This approach not only simplifies the routing process but also enhances the reliability and performance of power delivery across the PCB.

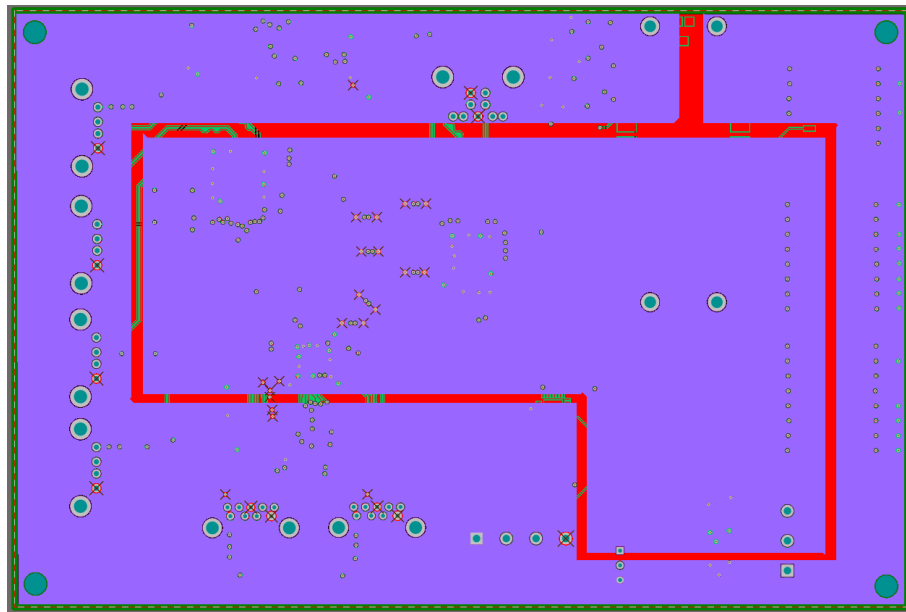


Figure 3.38: Polygon Pour in Layer 4

Figure 3.39 illustrates the polygon pour implemented on the first layer, designed specifically to mitigate electromagnetic interference (EMI). This polygon pour establishes extensive ground connections across the layer, which play a crucial role in shielding sensitive circuits from external noise and ensuring stable signal integrity.

A more detailed explanation of these ground connections and their impact on electromagnetic shielding will be provided in Chapter 3.3.7.

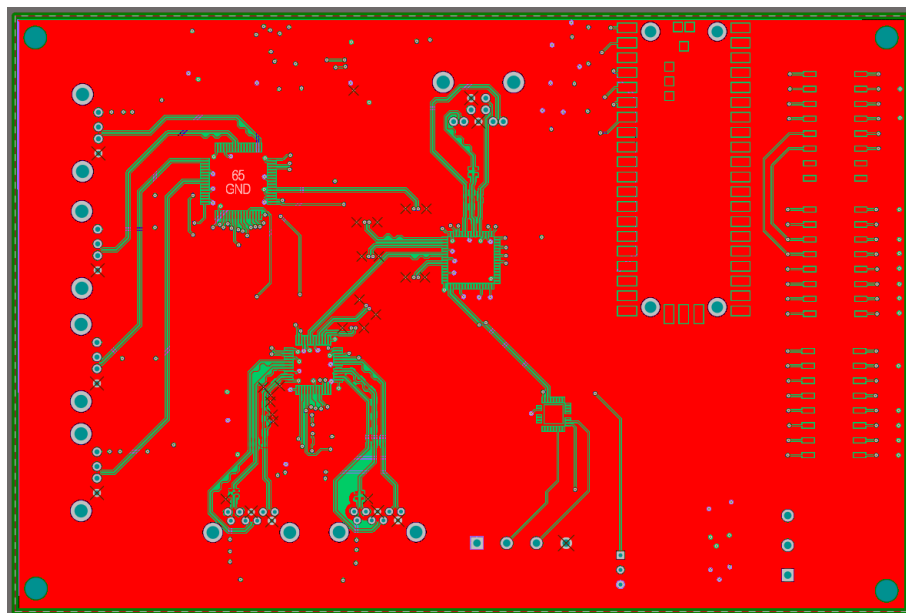


Figure 3.39: Polygon Pour in Layer 1

3.3.7 Stitching and Shielding Vias

Stitching vias are fundamental structures in PCB design, consisting of a periodic array of vias that are typically grounded across the PCB stackup. These vias serve to connect ground nets across multiple layers, thereby enhancing the electromagnetic shielding provided by polygon pours. By establishing these connections, stitching vias play a crucial role in mitigating electromagnetic interference (EMI) and ensuring the stability of the ground plane.

In radio frequency (RF) designs, stitching vias are often used in conjunction with guard rings to form a via wall. This configuration helps create an electromagnetically 'quiet' PCB environment by preventing the ingress of unwanted noise and interference. Via stitching is also employed to connect isolated copper areas to their respective nets, ensuring a continuous and effective ground connection.

In the following Figure 3.40, the stitching vias can be identified in the open areas of the PCB.

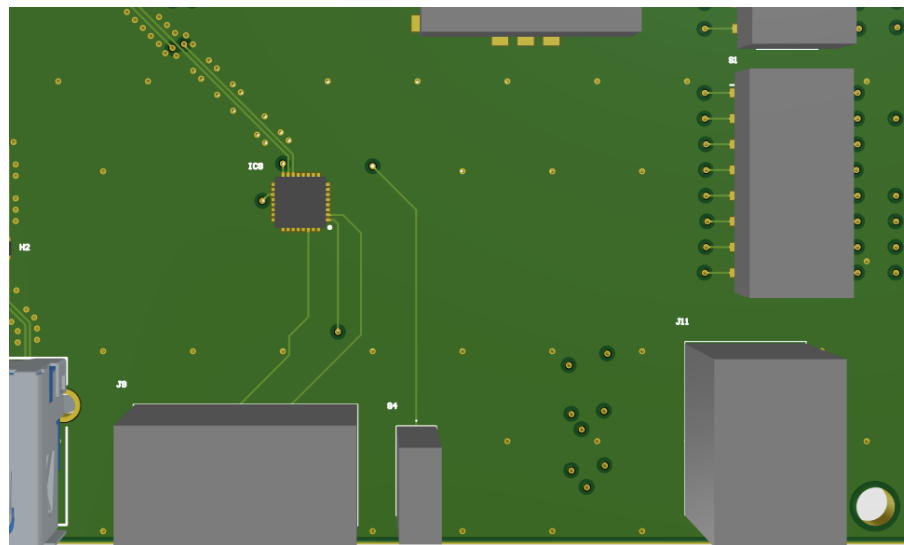


Figure 3.40: Stitching Vias

Additionally, via shielding, which is different from via stitching, is used to reduce crosstalk and electromagnetic interference specifically in routes carrying RF signals. This method involves creating a via shield, also known as a via fence or picket fence, by placing one or more rows of vias alongside the signal's route path. This arrangement provides a physical barrier that helps to contain the high speed signal and prevent it from coupling with adjacent traces. In Altium Designer, this practice is referred to as via shielding and is essential for maintaining signal integrity in high-frequency designs.

Incorporating both stitching vias and via shielding into a PCB design ensures robust EMI protection and enhances the overall performance of high speed circuits.

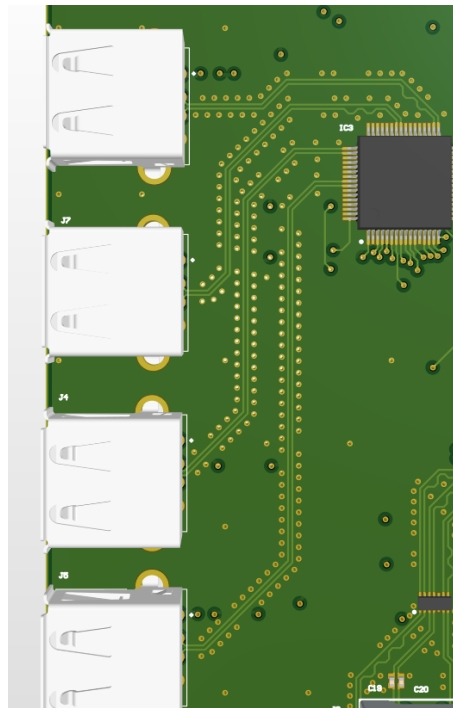


Figure 3.41: Shielding Vias

In certain instances, stitching vias were strategically placed within the footprint of the integrated circuit (IC). This was necessary because some ICs lack dedicated ground pins. These stitching vias are essential for establishing a robust connection between the IC and the ground plane of the PCB.

The process works as follows: the vias connect the polygon pour situated beneath the IC directly to the ground plane. During the soldering process, these vias facilitate the electrical connection between the IC's ground and the PCB's ground plane.

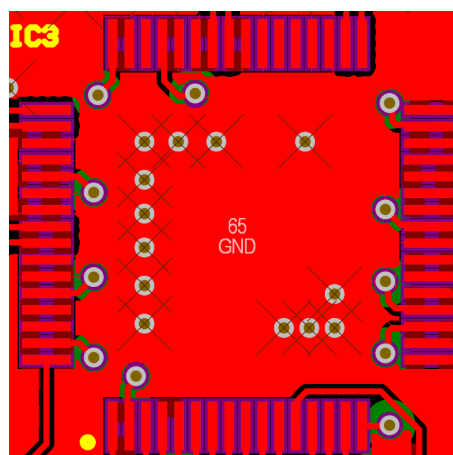


Figure 3.42: Shitching Vias in IC's grounds

To clearly identify these special vias, they are marked with a cross symbol.

3.3.8 Order PCB and Components

After finishing the design process, the gerber files were created and sent to the manufacturer.

Using Altium Designer, it was possible to visualize the final aspect of the PCB, providing a realistic preview of its appearance and layout in its completed form.

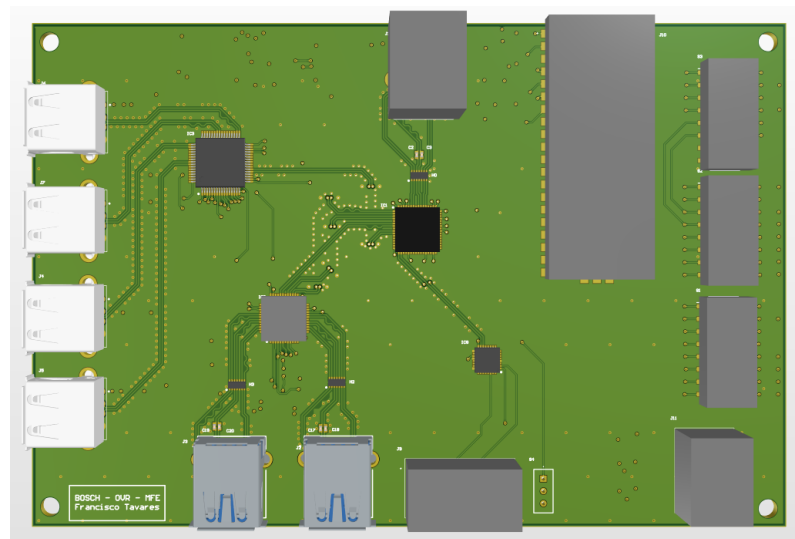


Figure 3.43: 3D View of the PCB

In addition to the PCB, components were sourced from various manufacturers based on availability. The prices of these components, along with the total cost, are detailed in the table present in Section A.

The component sizes were selected based on the specifications provided in the datasheet of each integrated circuit (IC).

Due to Bosch's partnerships with multiple manufacturers, the component prices were significantly reduced compared to the prices available to the general public. It is also important to note that ordering in bulk quantities would further reduce the cost per component, making the PCB production more financially sustainable.

The cost of the PCB itself amounted to €187.63. This elevated price can be attributed to the complex stackup design featuring multiple layers, as well as the requirement for controlled impedance. When this cost is combined with the total expenses for the components, the overall production cost of the PCB reached €237.53.

The Figure 3.44 shows the final aspect of the PCB.

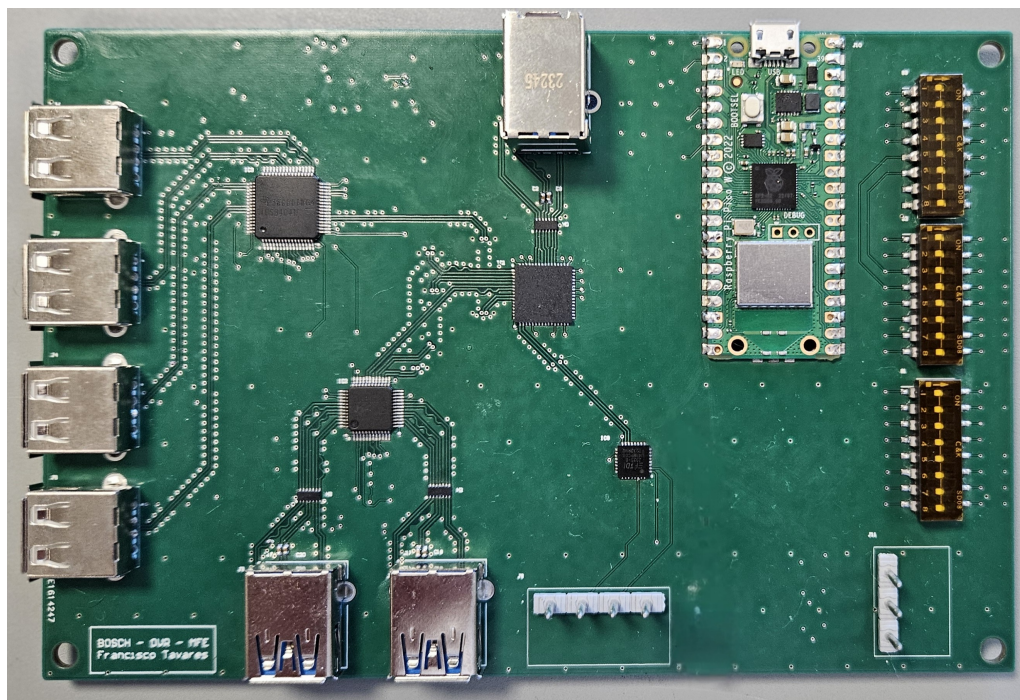


Figure 3.44: Final PCB

3.4 Reworking to Overcome Difficulties

Throughout the course of this project, several challenges were encountered, and it was essential to address and overcome them in order to achieve the project's objective: developing a USB hub capable of both communicating with connected devices and controlling the power to each downstream port.

The first difficulty found in the footprint design, a component footprint did not match the actual size of the component, leading to discrepancies. Specifically, the footprint for the ground area under the TUSB8041 IC was larger than the corresponding ground area on the board. This misalignment posed a risk of short circuits, as some vias could inadvertently connect to the ground.

Given the limited time available for this project, it was not feasible to redesign and reorder the PCB. As an alternative, a solution was devised to address the issue. The approach involved using heat-resistant tape, specifically designed to withstand the high temperatures of the soldering process. This tape was applied to the area around the ground on the PCB to prevent unintended contacts with the IC's ground connections, as shown in Figure 3.45.

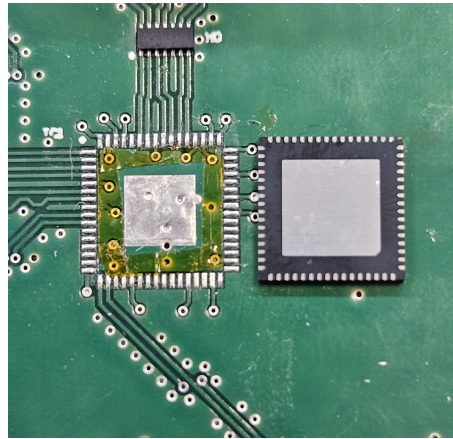


Figure 3.45: Heat-resistant tape covering the vias

After applying the heat-resistant tape, precise soldering of the IC became unfeasible due to the tape's height. Although the height was minimal, it was still sufficient to interfere with the delicate soldering process. As a result, this solution was abandoned.

Another solution involved applying a type of varnish, specifically a solder mask. This solder mask is already present on the PCB, serving to prevent short circuits, protect the copper from oxidation, and ensure solder is applied only to the intended areas. The solder mask was applied over the area to cover all the vias and was then cured under UV light to ensure durability and protection, particularly as this area would be exposed to high temperatures.

After soldering the IC, some continuity tests were conducted to ensure that all connections remained intact and that no inadvertent connections had occurred. This method effectively mitigated the risk of short circuits and ensured the PCB's functionality despite the initial design flaw.

Another issue, as detailed in Section 3.2.7, was the omission of a ground pin in the power supply input. Fortunately, this problem was relatively straightforward to resolve. Given the abundance of ground pins distributed across the PCB, establishing a ground connection was simple and effective. By leveraging the existing ground points on the board, a proper ground connection was quickly established, ensuring the power supply functioned correctly despite the initial oversight.

During some of the continuity tests, a short circuit was detected between the ground and the 1.1V power rail. Identifying this short circuit was critical, as powering the circuit without addressing it could have caused significant damage. Finding the source of the issue took considerable time due to the number of components that could potentially cause the problem. Ultimately, it was found that a capacitor had not been properly soldered, resulting in a direct connection between the 1.1V rail and ground.

After, a mistake was made in the power supply connections, resulting in an incorrect connection that caused one of the ICs to be damaged, leading to a short circuit between the 3.3V rail and ground. To expedite the identification of the faulty component, a thermal camera was used to observe which component heated up faster than the others. A small voltage with a high current was applied to create conditions that would heat the faulty component without causing further

damage to the circuit. Once the problematic component was identified, it was replaced, and testing continued.

Another issue that could potentially limit the functionality of the board was discovered in the connectors of the USB 2.0 ports. In Figure 3.20 it is possible to see that the pin 2 of each connector is directly connected with the Positive signal of the differential-line pair, and consequently, pin 3 was connected to the Negative signal of the same pair. The connections for the USB 2.0 ports were incorrectly made, preventing the IC from detecting any connected devices. Due to time constraints, redesigning and reordering the PCB was not possible. Instead, a workaround was implemented by using a USB extension cable, manually switching the data signals inside the extension to reverse the incorrect connections made on the PCB, thus restoring functionality, Figure 3.46.

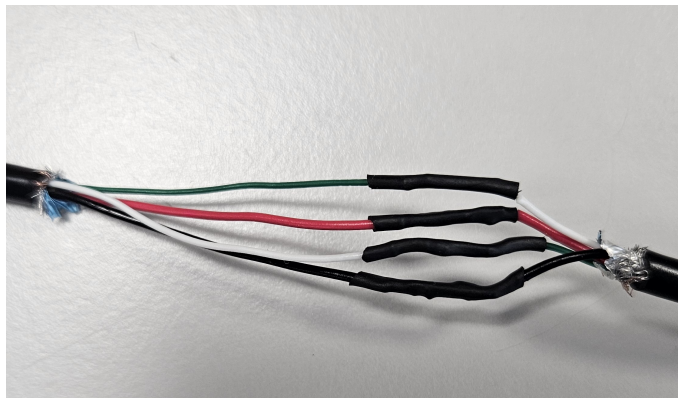


Figure 3.46: USB extension with data signals exchanged

Another challenging task encountered during the assembly process was the soldering of each capacitor. This task proved to be particularly difficult due to the small size of the capacitors, which are classified as 0402. These dimensions correspond to 1.0 mm in length and 0.5 mm in width. The manual soldering of each capacitor, one by one, required a high level of precision and demanded considerable time and patience to complete successfully.

The capacitors were chosen in this size due to the recommendations provided in the datasheet for each integrated circuit (IC). This selection is critical for ensuring optimal performance and functionality of the circuitry, as the specified capacitor dimensions directly align with the design parameters outlined by the manufacturer.

Chapter 4

Firmware

In this chapter, the code developed for controlling the downstream ports will be analyzed and explained in detail.

It will be examined the structure and functionality of the code, highlighting key components and their roles in managing communication between the Raspberry Pi Pico and the downstream connectors.

By providing a comprehensive breakdown of the code, this chapter aims to offer insights into the programming strategies employed to achieve effective control over the downstream ports.

4.1 Thonny

The firmware for this project was developed entirely in Python, chosen for its widespread use, ease of programming, and simplicity of code. Additionally, Python is fully compatible with the Raspberry Pi Pico, making it an ideal choice for this application.

Given the relatively low complexity of the code, the Thonny IDE was used for development. Thonny is an open-source integrated development environment specifically designed for Python programming. It offers an intuitive interface, making it well-suited for both beginners and experienced developers. Thonny also provides features such as code debugging, automatic installation of dependencies, and seamless interaction with microcontroller platforms like the Raspberry Pi Pico, which made it a practical tool for this project.

4.2 Code Explanation

The software concept involve running a program on the Raspberry Pi Pico, where commands are sent from a computer via the command line. These signals are interpreted by the Raspberry Pi Pico, allowing it to execute the requested actions for each downstream port on the board. Figure 4.1 represents the concept of the software's logic.

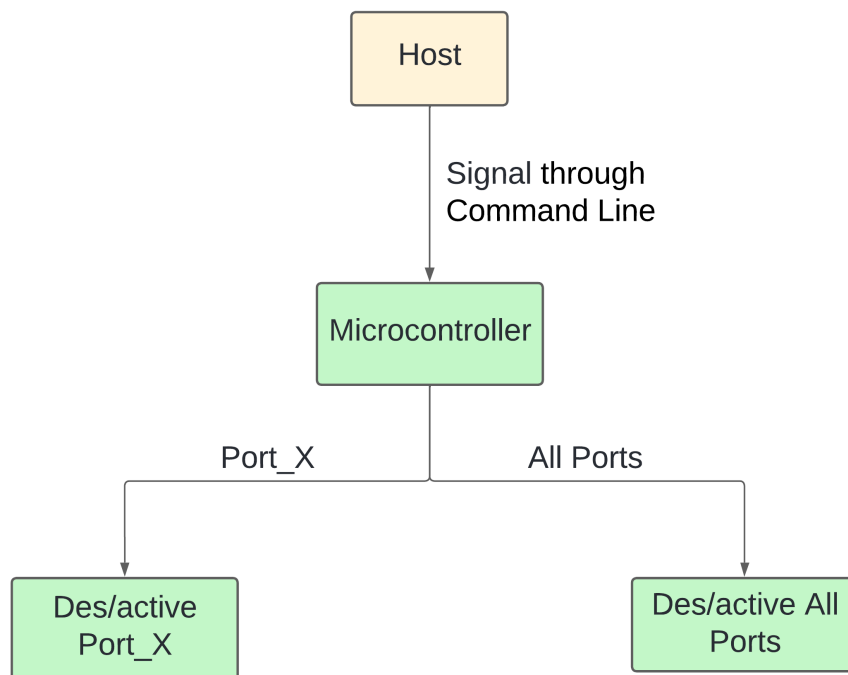


Figure 4.1: Firmware Logic

The first code to be reviewed is the one uploaded to the Raspberry Pi Pico:

```

4  # Definition of the pins based on GPIO numbering
5  PIN_GP0 = Pin(0, Pin.OUT) # GPIO 0
6  PIN_GP1 = Pin(1, Pin.OUT) # GPIO 1
7  PIN_GP2 = Pin(2, Pin.OUT) # GPIO 2
8  PIN_GP3 = Pin(3, Pin.OUT) # GPIO 3
9  PIN_GP4 = Pin(4, Pin.OUT) # GPIO 4
10 PIN_GP5 = Pin(5, Pin.OUT) # GPIO 5
11
12 # Initialize pins to LOW
13 PIN_GP0.value(0)
14 PIN_GP1.value(0)
15 PIN_GP2.value(0)
16 PIN_GP3.value(0)
17 PIN_GP4.value(0)
18 PIN_GP5.value(0)
19
20 # Pin mapping
21 pins = {
22     '1': PIN_GP0,
23     '2': PIN_GP1,
24     '3': PIN_GP2,
25     '4': PIN_GP3,
26     '5': PIN_GP4,
27     '6': PIN_GP5,
28 }

```

Figure 4.2: Pin Definition Code

In the Figure 4.2, the GPIO pins of the microcontroller are defined based on their GPIO numbers. Using the `Pin()` function, six GPIO pins are assigned to control each of the six ports and are configured as output pins. Once defined, the code sets the output value of each pin to '0', which represents a LOW state (turning off the corresponding output).

A dictionary named 'pins' is then created, mapping strings ('1' to '6') to their corresponding GPIO pins, making it easier to reference the pins by these numbers later in the code. This allows for simpler and more intuitive pin control throughout the program.

The following code snippet begins with the definition of a function called `toggle_pin`. This function takes two arguments: a pin name and a state. It retrieves the corresponding pin from the `pins` dictionary and sets its output value to the specified state. Additionally, the function checks whether the pin corresponds to a USB 2.0 or USB 3.0 port and prints a message indicating whether the port was set to HIGH or LOW, providing feedback on the port's status.

```

30 def toggle_pin(pin_name, state):
31     pin = pins[pin_name]
32     pin.value(state)
33
34     # Determine if the pin belongs to USB3.0 or USB2.0
35     if pin_name in ['1', '2']:
36         print(f"Set USB3.0 Port{pin_name} to {'HIGH' if state else 'LOW'}")
37     else:
38         print(f"Set USB2.0 Port{pin_name} to {'HIGH' if state else 'LOW'}")
39
40 print("Ready to receive commands...")
41
42 while True:
43     command = sys.stdin.readline().strip()
44
45     if command:
46         if len(command) == 3 and command[1] == '-':
47             segments = command.split('-')
48             if len(segments) == 2 and segments[1] in ['0', '1']:
49                 pin_number = segments[0]
50                 state = int(segments[1])
51
52                 if pin_number in pins:
53                     toggle_pin(pin_number, state)
54                 else:
55                     print("Invalid pin number")
56             else:
57                 print("Invalid command format or state value")
58
59         elif len(command) == 6:
60             for i, pin_name in enumerate(['1', '2', '3', '4', '5', '6']):
61                 pins[pin_name].value(1 if command[i] == '1' else 0)
62
63             print(f"Set pins to: {command}")
64         else:
65             print("Invalid command format")
66     else:
67         print("No command received")
68

```

Figure 4.3: Pin Control and Command Processing

After defining the function, the code enters a continuous loop, waiting for user input. The input is processed in two ways:

- **Single Pin Command:**

If the input command consists of three characters in the format `pinNumber-state` (e.g., "1-1"), the code extracts the pin number and desired state (either `HIGH` or `LOW`). It then calls the `toggle_pin` function to update the pin's state. If the command is invalid or the pin number does not exist, an error message is displayed.

- **All Pins Command:**

If the input is six characters long (e.g., "101010"), the code interprets the input as the desired states for all six pins. For each pin, '1' represents `HIGH` and '0' represents `LOW`. The code then updates each pin's state accordingly and prints a message confirming the new states of the pins.

If the command does not match either expected format, an error message is shown. If no input is received, it prints "No command received," indicating that the system is waiting for user interaction.

The Figure 4.4 present the code that is utilized on the computer for sending commands via the command line. This code defines a function named `send_command`, which facilitates communication with a device over a specified serial port.

```

4  def send_command(port, command):
5      try:
6          with serial.Serial(port, 115200, timeout=1) as ser:
7              ser.write((command + '\n').encode())
8              if command:
9                  if len(command) == 3:
10                     if command[0] == '1':
11                         print("USB3.0 Port_1")
12                     elif command[0] == '2':
13                         print("USB3.0 Port_2")
14                     elif command[0] == '3':
15                         print("USB2.0 Port_1")
16                     elif command[0] == '4':
17                         print("USB2.0 Port_2")
18                     elif command[0] == '5':
19                         print("USB2.0 Port_3")
20                     elif command[0] == '6':
21                         print("USB2.0 Port_4")
22                 elif len(command) == 6:
23                     for i in range(6):
24                         port_state = 'active' if command[i] == '1' else 'inactive'
25                         print(f"Port {i+1} will be {port_state}")
26                 elif command == '-help':
27                     print("To (de)activate only one port, type the number"
28                           "of the port you want to activate (X) and the logic"
29                           "value (1 for high, 0 for low): e.g. X-1")
30                     print("To (de)activate multiple ports, type the value"
31                           "of each one in ascending order (e.g. 111000)")
32
33             response = ser.readline().decode().strip()
34             if response:
35                 print(f"Response from Pico W: {response}")
36             else:
37                 print("No response from Pico W.")
38         except serial.SerialException as e:
39             print(f"Error: {e}")

```

Figure 4.4: Serial Command Sender for USB Port Control

The function accepts two parameters: `port`, representing the name of the serial port, and `command`, which is the command string intended for transmission to the device. Within the function, a `try` block is employed to handle potential errors associated with the serial connection. The serial port is opened using the `serial.Serial()` manager, configured with a baud rate of 115200 and a timeout of 1 second. During this context, the command is sent to the serial port by appending a newline character and encoding it to bytes before writing it to the port.

The function subsequently checks if the command is non-empty. If the length of the command is 3 characters, it inspects the first character to determine which USB port is referenced, printing a corresponding message such as "USB3.0 Port_1" or "USB2.0 Port_3". If the command length is 6, it iterates through each character of the command, determining the active or inactive state of the port based on whether the character is '1' or '0', respectively, and prints the status of each port.

If the command matches `-help`, the function provides usage instructions for the user, detailing how to activate or deactivate individual or multiple ports. After processing the command, the function waits for a response from the serial device, reading a line from the port, decoding it, and stripping any extraneous whitespace. If a response is received, it is printed; if not, a message indicating that no response was received is displayed.

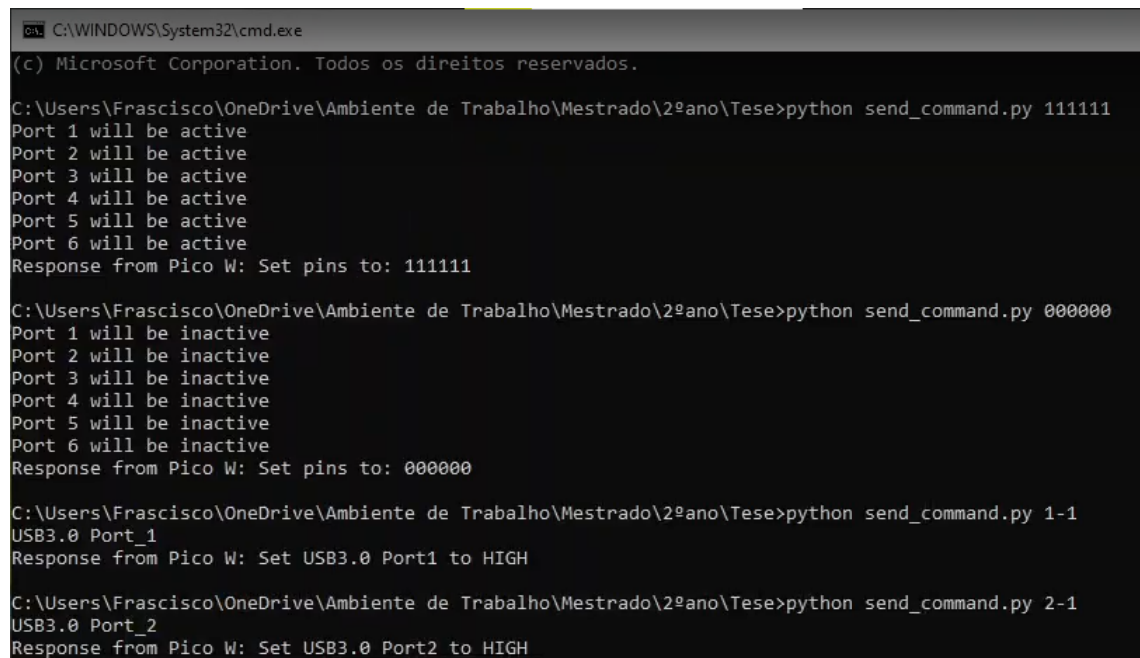
In the event of any issues arising while attempting to open the serial port or during communication, the `except` block catches `serial.SerialException` errors and prints an error message containing the exception details.

```
38 if __name__ == "__main__":
39     port = 'COM17' # Replace with the correct COM port
40
41     if len(sys.argv) > 1:
42         command = sys.argv[1] # Get the argument (e.g., '1-1')
43         send_command(port, command)
44     else:
45         print("Usage: python send_command.py <command>")
46         print("Example: python send_command.py 1-1")
47
```

Figure 4.5: Serial Port Command Interface

This code above checks whether the script is being executed as the main program. If so, it sets a variable named `port` to a specified COM port, in this case, 'COM17'. The code then verifies if any command-line arguments were provided during the script execution. If there is more than one argument (the first argument is always the script name), it assigns the second argument to the variable `command`, which is expected to follow a format such as '1-1'. Subsequently, the `send_command` function is invoked with the `port` and `command` as arguments, enabling the user to send commands to the specified serial port.

If no command-line argument is provided, the script prints a usage message to guide the user on how to run the script correctly. This message includes an example, indicating that the user should run the script with a command like '1-1' to activate or deactivate a specific port.



```
C:\WINDOWS\System32\cmd.exe
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Frascisco\OneDrive\Ambiente de Trabalho\Mestrado\2ºano\Tese>python send_command.py 111111
Port 1 will be active
Port 2 will be active
Port 3 will be active
Port 4 will be active
Port 5 will be active
Port 6 will be active
Response from Pico W: Set pins to: 111111

C:\Users\Frascisco\OneDrive\Ambiente de Trabalho\Mestrado\2ºano\Tese>python send_command.py 000000
Port 1 will be inactive
Port 2 will be inactive
Port 3 will be inactive
Port 4 will be inactive
Port 5 will be inactive
Port 6 will be inactive
Response from Pico W: Set pins to: 000000

C:\Users\Frascisco\OneDrive\Ambiente de Trabalho\Mestrado\2ºano\Tese>python send_command.py 1-1
USB3.0 Port_1
Response from Pico W: Set USB3.0 Port1 to HIGH

C:\Users\Frascisco\OneDrive\Ambiente de Trabalho\Mestrado\2ºano\Tese>python send_command.py 2-1
USB3.0 Port_2
Response from Pico W: Set USB3.0 Port2 to HIGH
```

Figure 4.6: Example of Commands

In Figure 4.6, the usage of a command line interface is illustrated, along with the corresponding responses from the Raspberry Pi Pico, demonstrating the activation and deactivation of each port as per user requests.

Initially, the command issued was to activate all ports, followed by a command to deactivate them. Subsequently, ports 1 and 2 were manually activated. The system consistently responded to the user's commands, reflecting the desired actions for each port.

Chapter 5

Results and Performance Analysis

This chapter provides an in-depth analysis of the project results, discussing key aspects and their significance. It reviews the outcomes achieved during the project and suggests possible improvements.

To measure the communication speed of each type of port, CrystalDiskMark was employed, an open-source disk drive benchmarking tool for Windows that is commonly used to evaluate the performance of solid-state storage devices. Initial measurements involved connecting a single device directly to the computer to determine its maximum speed. Following this, the communication speed was measured with the device connected to board developed in this project. This approach facilitated a comprehensive comparison of performance across both configurations.

5.1 USB2.0

To make the communication with these types of ports, a USB extension was utilized, as noted in Chapter 3.4, due to an error in the connections made during the board’s design. To test the board and the USB2.0 connectors, it was used a USB2.0 Pen drive, firstly connected to the computer.

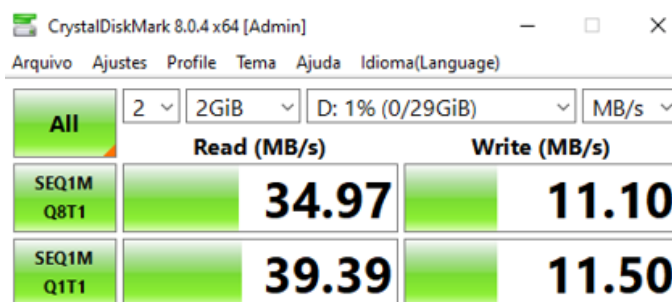


Figure 5.1: Pen 2.0 Connected to Computer

In Figure 5.1, the number 5 indicates the number of tests conducted, and the read and write values shown are the averages of these tests. A 128MiB file was used for each test. The appropriate disk was selected for analysis, and the results are displayed in megabytes per second (MB/s).

The first row shows tests using a sequential 1 MB operation with a queue depth of 8 and 1 thread. Sequential operations involve simple read and write actions performed in a continuous series of blocks. The queue depth refers to the number of input/output requests the storage device can handle simultaneously, while the thread count represents how many processes are accessing the drive.

In the second row, the tests are identical except that the queue depth is reduced to 1 instead of 8, as seen in the first row.

Making the comparison between the Figure 5.1 with the Figure 5.2 it is possible to conclude that the speeds are basically the same.

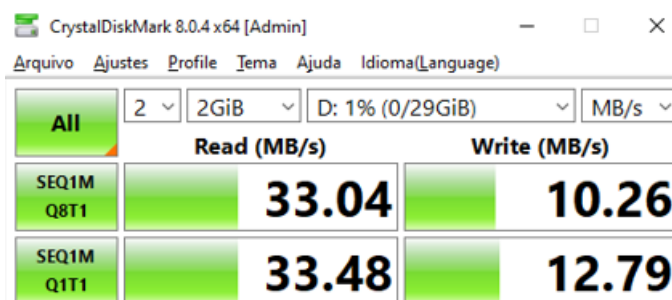


Figure 5.2: Pen 2.0 Connected to the Board

5.2 USB3.0

The first performance test was conducted using a Samsung T7 SSD, which, according to the manufacturer's specifications, is capable of achieving data transfer speeds of up to 1050 MB/s. Initially, the test was performed by connecting the SSD directly to the computer, yielding the following results:

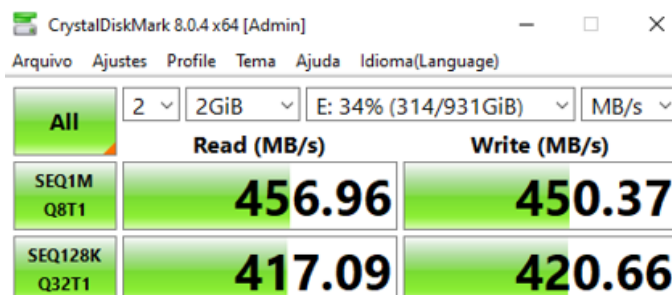


Figure 5.3: SSD Disk Connected to the Computer

Subsequently, the test was conducted using the board as the intermediary connection for the SSD.

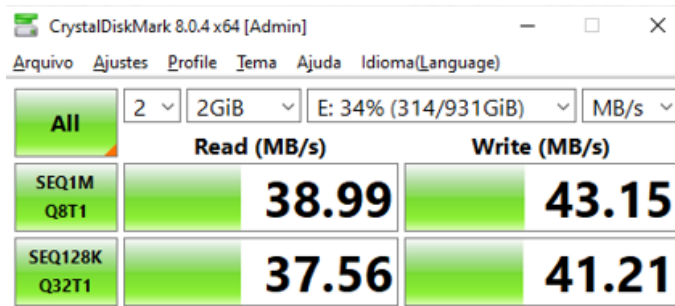


Figure 5.4: SSD Disk Connected to the Board

Upon analyzing the results, it became evident that the maximum speed achieved by the board using USB 3.0 connectors was identical to that achieved with USB 2.0 connectors. This unexpected outcome necessitated further investigation into why the board could not obtain higher speeds with the USB 3.0 connectors.

The issue was traced to the Superspeed differential pairs and their corresponding connections between the integrated circuits (ICs). Specifically, the TX pairs from the first IC, TUSB8041, were incorrectly connected to the TX pins of the subsequent IC, TUSB8020. Proper functionality requires connecting the TX pin of the TUSB8041 to the RX pin of the TUSB8020, as illustrated in Figure 2.9, Section 2.2.2. This misconnection explains the observed limitations in data transfer speed.

To address this mistake, the tracks for the differential pairs were cut, and external connections were made using wires to link the TX pins of the TUSB8041 to the RX pins of the TUSB8020, and vice versa. While this solution is not ideal for ensuring maximum speed—due to factors such as the introduction of noise, varying wire lengths, and significant changes in differential impedance—it was implemented as a temporary fix. The goal was to determine whether the issue indeed stemmed from the incorrect connections, despite the limitations this method introduces in maintaining optimal signal integrity.

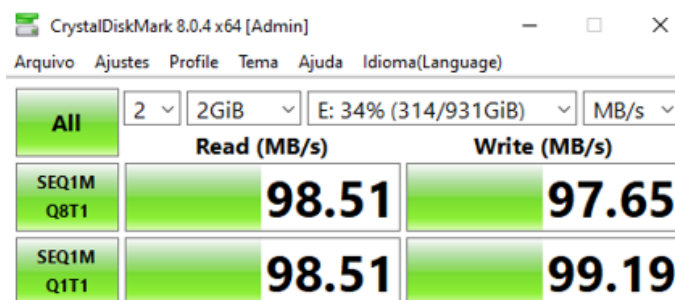


Figure 5.5: SSD Disk Connected to the Board

The results obtained from this modification showed slight improvements; however, they remained significantly below the speeds achieved when the SSD was directly connected to the computer, as anticipated. As demonstrated, the error in connecting the TX to TX and RX to RX pins between different ICs caused the USB 3.0 ports to operate at only USB 2.0 speeds. A temporary

fix, using external wires to make the correct connections, resolved the issue and highlighted an improvement needed for the next version of the electronic system. With this correction, it is expected that these ports will achieve significantly higher speeds, closer to those obtained when devices are connected directly to the computer.

5.3 RS232

This section explains the process used to test the RS-232 communication.

First, an FTDI cable, which converts USB to TTL, was used. From this cable, the communication wires—RX (Receive) and TX (Transmit)—were utilized. These wires were connected to the corresponding connectors on the board designated for RS-232 communication. The USB end of the FTDI cable was then plugged into a USB 2.0 port on the board.

To test the communication between the two ports, a software called PuTTY was used. PuTTY is a terminal emulator and serial console, particularly useful for serial communication, as it allows data to be sent between two points.

In the software, the serial connection type was selected, and the appropriate COM ports were configured—one corresponding to the USB connection and the other to the RS-232 port. The baud rate was set to 19200, as shown in Figure 5.6, and it is crucial that both terminals are configured with the same baud rate to ensure proper communication.

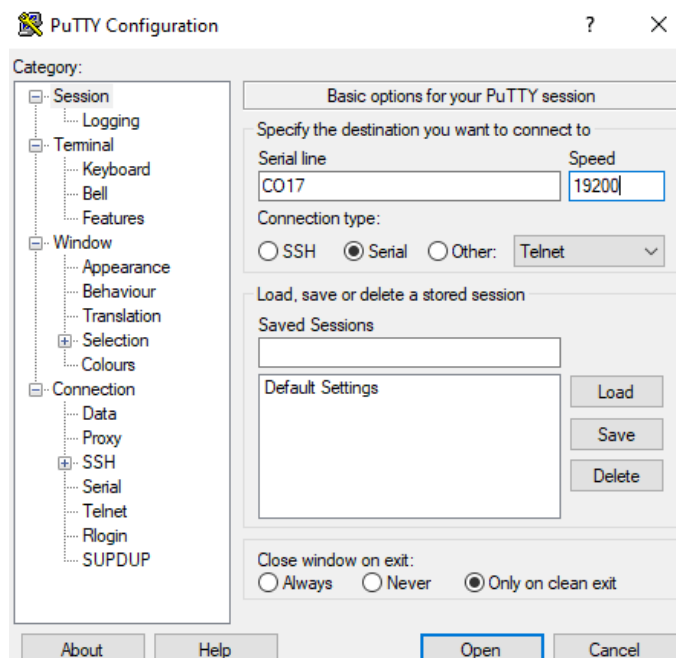


Figure 5.6: Putty Settings

Using PuTTY, it was possible to send commands from one terminal and receive them in the other. Additionally, an oscilloscope was used to analyze and decode the signal transmitted between

the two points. The oscilloscope was connected to the RX wire of the FTDI cable and configured with the same baud rate as in PuTTY, set to 19200. The oscilloscope was also set to trigger on the falling edge of the signal, as the line remains at 3.3V when idle and drops to 0V when communication begins.

In Figure 5.7, a command was sent, and the signal was decoded from hexadecimal into ASCII format. This illustrates how the transmitted data was interpreted.

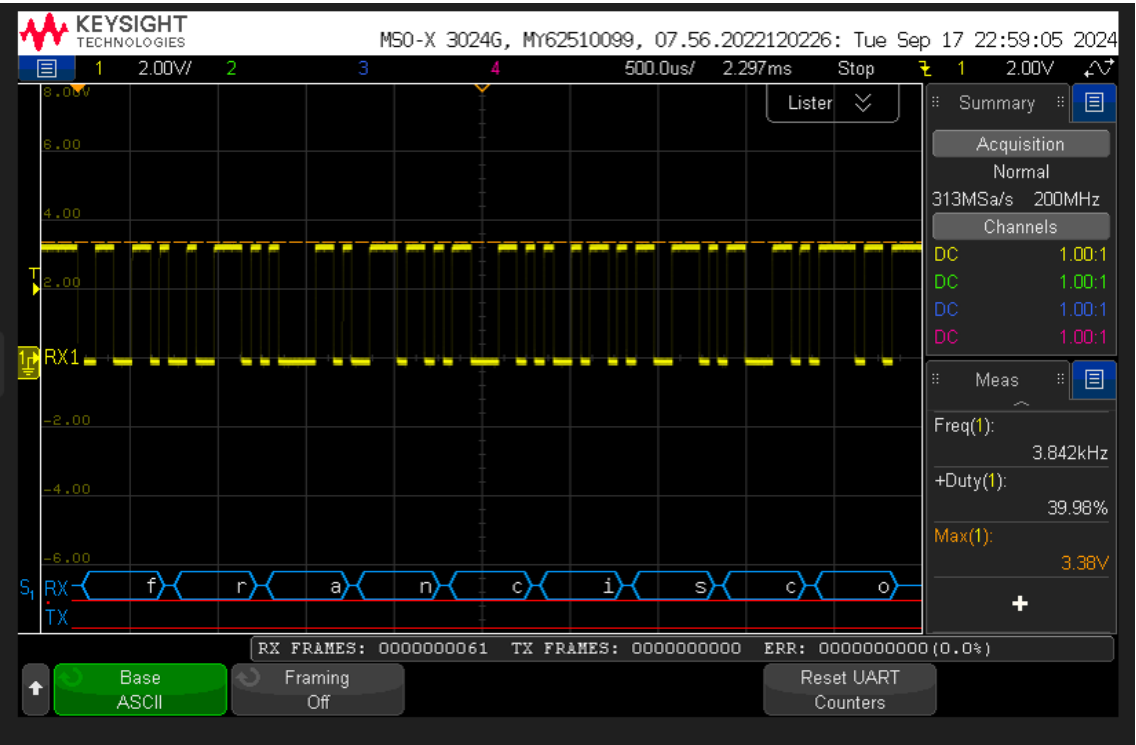


Figure 5.7: Signal Analysis and ASCII Decoding of Transmitted Commands

Chapter 6

Conclusions

This dissertation presents the design and development of a controllable USB hub tailored to meet the specific industrial requirements of a company. The hub features four USB 2.0 ports, two USB 3.0 ports, and an RS232 interface, with the ability to individually control the power supply to each downstream port. This capability addresses a critical need for the company, enhancing the flexibility and efficiency of their operations.

The project journey began with a thorough investigation of existing solutions in the market, leading to the realization that a custom-designed hub would be more cost-effective and tailored to the company's precise needs. The decision to use the Raspberry Pi Pico as the microcontroller was driven by its ease of use, onboard EEPROM, and cost-effectiveness compared to alternatives like the Arduino Nano. This choice ensured reliable storage of port power states even in the event of a power cut, which is crucial for maintaining operational consistency.

The design process involved meticulous planning and strategic placement of components to facilitate efficient routing. Adhering to high-speed signal layout guidelines and ensuring proper grounding and impedance control were paramount to achieving a robust and reliable PCB design. The use of polygon pours for electromagnetic shielding and power distribution, combined with careful consideration of via placement, ensured optimal performance and minimized interference.

Despite encountering challenges, such as mismatched footprints and the omission of a ground pin, the project demonstrated the ability to adapt and implement effective solutions.

6.1 Future Work

Looking ahead, the next phase of this project involves integrating the power supply directly into the PCB. Currently, the hub relies on an external power source, which, while functional, is not as streamlined or efficient as it could be. By incorporating the power supply onto the PCB itself, we can create a more compact, self-contained, and reliable device.

This future work will focus on selecting and integrating appropriate voltage regulators to ensure stable and adequate power delivery to each downstream port. Voltage regulators are critical

components that will provide the necessary voltage levels required by the USB 2.0 and USB 3.0 ports, as well as the RS232 interface.

Connecting the Raspberry Pi Pico directly to the first integrated circuit (IC), the TUSB8041, would represent a significant improvement in the design. This direct connection would reduce the number of cables linking the board to the computer, thereby streamlining the setup. Additionally, it would enable commands to control the downstream ports to be issued directly through the upstream connection, enhancing the overall efficiency of communication and control.

Furthermore, during the redesign of the board, the removal of the DIP switches is planned, as they are no longer necessary for programming the IC. Eliminating these components will contribute to a more compact design, resulting in a smaller overall footprint for the board. This reduction in size will not only make the board easier to integrate into the test fixtures used in production lines, but it will also simplify assembly and maintenance processes. A more compact design can lead to improved reliability and performance, as it minimizes potential points of failure and reduces the complexity of wiring.

In the upcoming iteration of the board, which will incorporate an integrated power regulator, future work will focus on enhancing its functionality by adding USB Type-C ports specifically designated for power supply. This addition aligns with the objectives outlined in the initial phase of the project, where the integration of advanced connectivity options was a requirement.

References

- [1] Altium. Polygon Pour. https://www.altium.com/documentation/altium-designer/pcb-polygon-pour?version=18.1&srsltid=AfmBOopaQ3WgLVxcweljZwNLplh8dEjkmt7uoyh_9HFphgLDxnkEfCwr, December 2017. Last accessed: 2024-6-5.
- [2] Altium - Zachariah Peterson. What Are Differential Pairs and Differential Signals? <https://resources.altium.com/p/what-are-differential-pairs-and-differential-signals>, October 2021. Last accessed: 2024-6-21.
- [3] Altium - Zachariah Peterson. What PCB Copper Thickness Should You Use? <https://resources.altium.com/p/what-pcb-copper-thickness-should-you-use>, July 2021. Last accessed: 2024-6-12.
- [4] Altium - Zachariah Peterson. Differential Pair Impedance: Using a Calculator to Design Your PCB. <https://resources.altium.com/p/differential-pair-impedance-using-calculator-design-your-pcb>, January 2022. Last accessed: 2024-4-29.
- [5] Donovan Anderson and Jay Trodden. *USB 3.0 Technology*. MindShare Press, First edition, 10 2013.
- [6] Arduino. *Arduino® Nano*, April 2022. [Revised 25 April 2024].
- [7] Nexperia BV. *General description NPN general-purpose transistors in Surface Mounted Device (SMD)*, November 2009.
- [8] Spartaco Caniggia and Francescaromana Maradei. *SIGNAL INTEGRITY AND RADIATED EMISSION OF HIGH-SPEED DIGITAL SYSTEMS*. Wiley, 2008.
- [9] Alessandro Depari, Alessandra Flammini, Daniele Marioli, and Andrea Taroni. Usb sensor network for industrial applications. *IEEE Transactions on Instrumentation and Measurement*, 57:1344–1349, 7 2008.
- [10] Meilhaus Electronic. *USB hub 3.0 6-Port, Switchable, 2 Control Inputs*, 2018. [Product Datasheet - Technical Specifications].
- [11] Mouser Eletronics. Mouser Eletronics. https://pt.mouser.com/c/?q=usb%20hub&srsltid=AfmBOoqelP_9ueomuqV-MUjhzHJiQDbLHniREN0T8uHyztTSTo_HBccA&sort=manufacturerpartnumber&pg=8. Accessed: 2024-6-15.

- [12] EuroCircuits. Buildup Editor. <https://be.eurocircuits.com/shop/assembly/configurator.aspx?sessionid=3f7f3743-765-42ed-9a3b-663b606b7278&lang=en&quantity=2&service=pcbproto&deliveryTerm=3&layers=2&dimY=80&dimX=100>. Last accessed: 2024-6-12.
- [13] EuroCircuits. Classification. <https://www.eurocircuits.com/pcb-design-guidelines/classification/>, January 2020. Last accessed: 2024-7-21.
- [14] Compaq Hewlett-Packard. Universal Serial Bus Specification, 2000.
- [15] Teoh Tiong Hong, Sarwar Zeeshan, and Mohd Zaid Abdullah. A power efficient usb 2.0 device controller architecture and its implementation. pages 388–392. IEEE, 6 2011.
- [16] Texas Instruments. *TPS256x Dual Channel Precision Adjustable Current-limited Power Switches*, December 2009. [Revised October 2020].
- [17] Texas Instruments. *TPDxE05U06 1, 4, 6 Channel ESD Protection Device for Super-Speed (Up to 6 Gbps) Interface 1 Features*, December 2012. [Revised February 2022].
- [18] Texas Instruments. *TUSB8020B Two-Port USB 3.0 Hub*, July 2014. [Revised June 2017].
- [19] Texas Instruments. *TUSB4041i*, July 2015. [Revised September 2017].
- [20] Texas Instruments. *TUSB4041I-Q1 Four-Port USB 2.0 Hub*, July 2015. [Revised July 2024].
- [21] Texas Instruments. High-speed layout guidelines for signal conditioners and usb hubs application report high-speed layout guidelines for signal conditioners and usb hubs, August 2018.
- [22] Texas Instruments. TUSB4020BI: USB_VBUS connection for usb power delivery application, 5 2021.
- [23] Kamini Jha, Ajit. B. Patil, and Deepti. S. Khurge. Design and verification of Itssm in usb 3.0. pages 1–4. IEEE, 8 2018.
- [24] Wei Jiang, Kevin Cai, Bidyut Sen, and Guoan Wang. Practical high speed pcb stackup tool - generation and validation. In *Proceedings - Electronic Components and Technology Conference*, volume 2018-May, pages 2288–2294. Institute of Electrical and Electronics Engineers Inc., 8 2018.
- [25] Felix Lee. Why are Multilayer PCBs Mostly Even-Numbered Layers?, March 2022. Accessed: 2024-06-14.
- [26] Altium Limited. Altium Designer. <https://www.altium.com/altium-designer?srsltid=AfmBOopBU4MltEPM9XszxFLs4wOnIJSJSTw4GkUlo4mP0FNsyG2uA2fO>. Accessed: 2024-6-21.
- [27] Future Technology Devices International Ltd. *FT232RN Datasheet*.
- [28] Raspberry Pi Ltd. *Raspberry Pi Pico Datasheet*, January 2021. [Revised 2 May 2024].
- [29] Microchip. *USB5742B 2-Port SS/HS USB Controller Hub*.
- [30] Microchip. *USB5744 4-Port SS/HS USB Controller Hub*.

- [31] Microchip. *USB7252C 4-Port USB 3.2 Gen 2 Type-C® Controller Hub*.
- [32] Nishith R Naik. "USB based data communication system". 2010. <https://scholar.google.com/scholar?q=USB%20based%20data%20communication%20system> (accessed Jan. 3, 2023).
- [33] Compaq Hewlett packard Intel, Lucent Microsoft, and N E C Philips. Universal Serial Bus Specification. *Group*, April 2000.
- [34] Rayming PCB and Assembly. Smt vs tht vs smt: What are the differences. <https://www.raypcb.com/smd-vs-tht-vs-smt/> (accessed Dec. 27, 2023).
- [35] Altium Zachariah Peterson. Differences Between Prepreg Material vs. PCB Core: What Designers Need to Know. <https://resources.altium.com/p/what-pcb-copper-thickness-should-you-use>, January 2020. Last accessed: 2024-6-12.
- [36] Lee Ritchey. A treatment of differential signaling and its design requirements prepared by speeding edge, May 2008.
- [37] Yung-Hoh Sheu, Yu-Ping Dai, and Duen-Shiang Fu. Embedded usb homecare internet system. pages 442–445. IEEE, 12 2010.
- [38] Texas Instruments. *TUSB8041*, June 2014. [Revised June 2016].
- [39] Texas Instruments. *User's Guide TUSB8041RGC REVD Evaluation Module*, May 2014.
- [40] Texas Instruments. *TUSB8043 datasheet*, June 2017. [Revised August 2017].
- [41] Texas Instruments. *TUSB8043A datasheet*, June 2019.
- [42] Texas Instruments. *TUSB8044A datasheet*, 2019. [Revised March 2019].
- [43] Texas Instruments. Texas Instruments Website, July 2024.
- [44] Ankur Tomar and Edmund Lim. Introduction to superspeed usb 3.0 protocol, 4 2011.
- [45] Tim Williams. *The Circuit Designer's Companion*. Elsevier, second edition, 2005.
- [46] Ihsan Onur Yilmaz. *Development and Evaluation of Setup Strategies in Printed Circuit Board Assembly*. Gabler, 2008.

Appendix A

Appendix

Comment	Description	Designator	Quant	Unit Price	Total
1757255	Connector	J11	1	0,06	0,06
1757268	Connector	J9	1	0,06	0,06
USB2.0 Type A	Connector	J4, J5, J6, J7	4	0,19	0,76
USB3.0 Type A	Connector	J2, J3	2	0,881	1,762
0.1uF	Capacitor SMD 0402	C2, C3, C17, C18, C19, C20, C21, C22, C23, C24, C25, C26, C27, C28, C29, C30, C31, C32, C35, C41, C43, C45, C46, C47, C49, C51, C52, C54, C56	29	0,0487	1,4123
1nF	Capacitor SMD 0402	C59, C95, C98, C100, C103, C104, C106, C108, C110, C112, C114, C116, C118, C120, C122, C124, C126, C129, C131, C133, C135, C137, C139, C141, C142, C144, C146, C148, C150, C152, C154, C156, C158, C160, C162, C164	36	0,0186	0,6696
1uF	Capacitor SMD 0402	C4, C7, C14	3	0,0282	0,0846
10nF	Capacitor SMD 0402	C57, C58, C61, C62, C63, C64, C65, C66, C67, C68, C69, C70, C71, C72, C73, C74, C75, C76, C77, C78, C79, C80, C81, C82, C83, C84, C85, C86, C87, C88, C89, C90, C91, C92, C93, C96, C166, C168, C170	39	0,03	1,17
10uF	Capacitor SMD 0402	C1, C8, C13	3	0,0125	0,0375
18pF	Capacitor SMD 0402	C5, C6, C9, C10, C15, C16	6	0,01	0,06
100nF	Capacitor SMD 0402	C94, C97, C101, C102, C105, C107, C109, C111, C113, C115, C117, C119, C121, C123, C125, C127, C128, C130, C132, C134, C136, C140, C143, C145, C147, C149, C151, C153, C155, C157, C159, C161, C163, C165, C167, C169, C171	38	0,0487	1,8506
100uF	Capacitor SMD 0402	C99	1	0,0325	0,0325
220 @ 100MHZ	Ferrite Bead SMD 0805	FB1, FB2, FB3, FB4, FB5, FB6, FB7, FB8, FB10	9	0,06	0,54
PICO	Connector	J10	1	5,42	5,42
Capacitor 150μF	Tantalum Capacitor	C42, C44, C48, C50, C53, C55	6	0,39	2,34

Comment	Description	Designator	Quant	Unit Price	Total
FT232RNQ-REEL	Integrated Circuit	IC8	1	3,32	3,32
TUSB8041	Integrated Circuit	IC1	1	6,52	6,52
TUSB8020BIPHP	Integrated Circuit	IC2	1	6,51	6,51
TUSB4041IPAPR	Integrated Circuit	IC3	1	5,3	5,3
1k	Resistor SMD 0402	R9, R10, R11, R12, R16, R18, R19, R25, R26, R28, R33, R35, R48, R49, R50, R51, R52, R53, R54, R55, R56, R57, R58, R59	24	0,0042	0,1008
1M	Resistor SMD 0402	R7, R15, R23, R29, R30, R31, R32	7	0,0129	0,0903
2k7	Resistor SMD 0402	R74, R75	2	0,0072	0,0144
4.7k	Resistor SMD 0402	R3, R4, R5, R6, R27, R36, R37, R38, R47	9	0,0106	0,0954
9.09K	Resistor SMD 0402	R17	1	0,0099	0,0099
9.53k	Resistor SMD 0402	R8, R24	2	0,006	0,012
10k	Resistor SMD 0402	R1, R13, R22, R40, R41, R43, R44, R45, R46	9	0,0063	0,0567
25.5K	Resistor SMD 0402	R34, R39, R42	3	0,0099	0,0297
90.9k	Resistor SMD 0402	R2, R14, R21	3	0,0914	0,2742
680	Resistor SMD 0402	R70, R71, R72, R73	4	0,006	0,024
SLW-883935-2A-D	Switch	S4	1	1,18	1,18
74HC08D,653	Integrated Circuit	IC9	1	0,372	0,372
TPD6E05U06RVZR	Hardware	H0, H2, H3	3	0,698	2,094
SN74LVC2G08DCTR	Integrated Circuit	IC10	1	0,573	0,573
SD08H0SB	Switch	S1, S2, S3	3	0,62	1,86
BC848B,215	Transistor BJT NPN	Q1, Q2, Q3, Q4, Q5, Q6	6	0,016	0,096
TPS2560DRCT	Integrated Circuit	IC4, IC5, IC6	3	1,26	3,78
692221030100	Connector USB Type B	J1	1	0,776	0,776
24MHz	Crystal SMD 3215	Y1, Y2, Y3	3	0,1853	0,5559
					49,9034