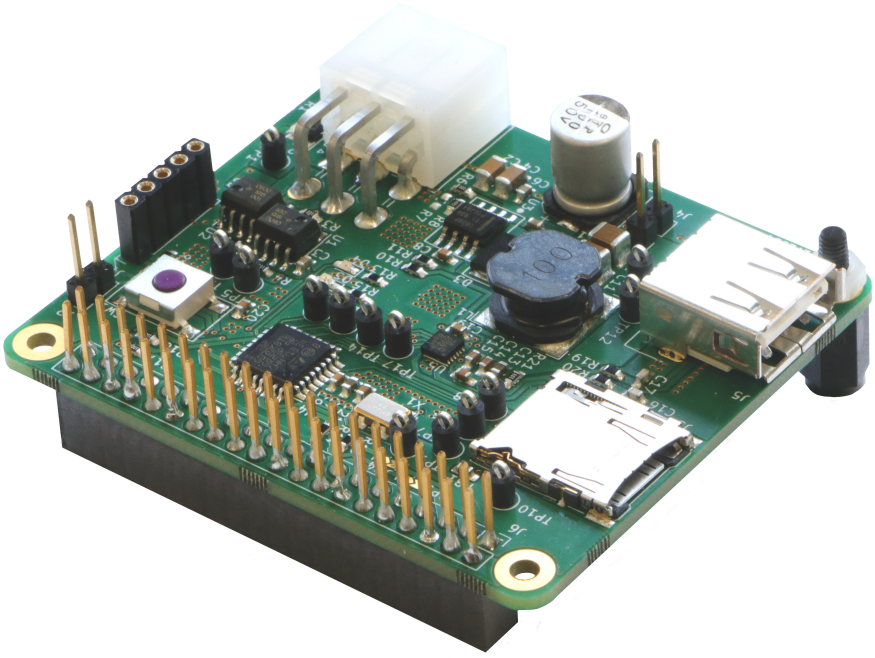
Bern University
of Applied Sciences

Embedded system for GNU/Taler

Designing an embedded system for cashless payment on a vending machine

Bachelor's Thesis



Field of Studies:	Mikro- und Medizintechnik
Course:	Embedded systems
Author:	Dominik Wenger
Supervisors:	Prof. Andreas Habegger, Prof. Dr. Christian Grothoff
Expert:	Rico Zoss
Date:	July 28, 2020

ABSTRACT

Digital wallet payment services are expected to account for $47\,\%$ of all E-Commerce and $28\,\%$ of all point of sale payment methods by 2022 [1]. Unfortunately, some people have concerns about their privacy by using such digital wallet applications, since most of the providers are big tech companies with unclear intentions.

A new participant in this global market will be GNU/Taler. It is a digital wallet service, that wants to provide a fast and easy payment system which ensures the user's privacy.

During this thesis, an embedded system shall be developed to allow the use of GNU/Taler for mobile payments in vending machines.

The fundamental knowledge about vending machines has been gathered and a hardware specification book was written. After the elaboration of different concepts, it was decided to develop a vending machine interface gateway for a Raspberry Pi and as a further step an embedded platform, where the system shall become more tailored and thus more compact. The gateway board was designed, manufactured and tested. Up to a few design issues, the gateway fulfills the desired requirements. The further developed embedded platform is not manufactured yet. The schematics have been developed and the components are selected. An outstanding process is the final board design.

After this thesis, the development of the platform will be continued. During the next year, all coffee-vending machines at the Berner Fachhochschule shall be equipped with the embedded platform to enable GNU/Taler payments.

TABLE OF CONTENTS

Part I.

Preliminary Study

1. Introduction

In an increasingly digitized society, money in form of notes and coins is becoming less and less important. The role of cash got degraded by credit- and debit cards in the last few decades. But these cards were only forerunners to prepare the society for the future. A completely cashless world.

In the past decade, smartphones became a very central device in most peoples lifes. Mobile banking and mobile payment applications have become very popular and payment systems like Apple pay, Android pay, Samsung pay, Amazon pay, Paypal, Alipay, Libra have generated a lot of attention.

The advantages of such mobile payment applications seem obvious. In modern civilizations with comprehensive mobile internet there is simply no need for cash anymore.

But there are concerns about mobile payment applications. If all my transactions are handled by an application on my smartphone, what will happen with the gathered information about my consumer behavior? If you look closely at the above listed payment applications, you can see that they are all founded by tech giants. Cashless payment generate a lot of detailed data about a users' consumer behavior what can be of high interest for some parties. By allowing big tech companies to act as payment providers, the economic sovereignty is in danger and every users privacy too [2].

Unfortunately, there is currently no payment system on the market which guarantees privacy to its users. But there is one to come and it is named Taler. Taler was founded in 2014 and is part of the GNU-Project, which is engaged to provide free software[1] to society.

Taler is currently only available for demo usage in online transactions. But this will change. As part of this thesis, an embedded system for a vending machine shall be designed which enables the use of Taler in such devices.

In advance to this thesis, a prototype based on a Raspberry Pi was developed by Dominik Hofer and Marco Boss. This prototype was presented at the famous 36C3 [5] and was even shown at the WEF in 2020.

The now further developed system shall fulfill industry standards and should be able to be produced in higher quantities. Additionally, it is a personal goal to use mostly free software (or at least open-source) tools for the development of this system to mesh with the GNU philosophy.

[1] Software that respects the users' freedom. Official definition

1.1. About GNU/Taler

Taler stands for Taxable Anonymous Libre Economic Reserves and is a free software based microtransaction payment software. The project is led by Christian Grothoff and Florian Dold [6].

The goal of the software is to provide a payment system that makes privacy-friendly online transactions fast and easy [3]. It is set up in such way that everybody can make a Taler account by simply registering himself with a username and a password. There is no proof of identity necessary. This means that literally everybody is able to participate the monetary system.

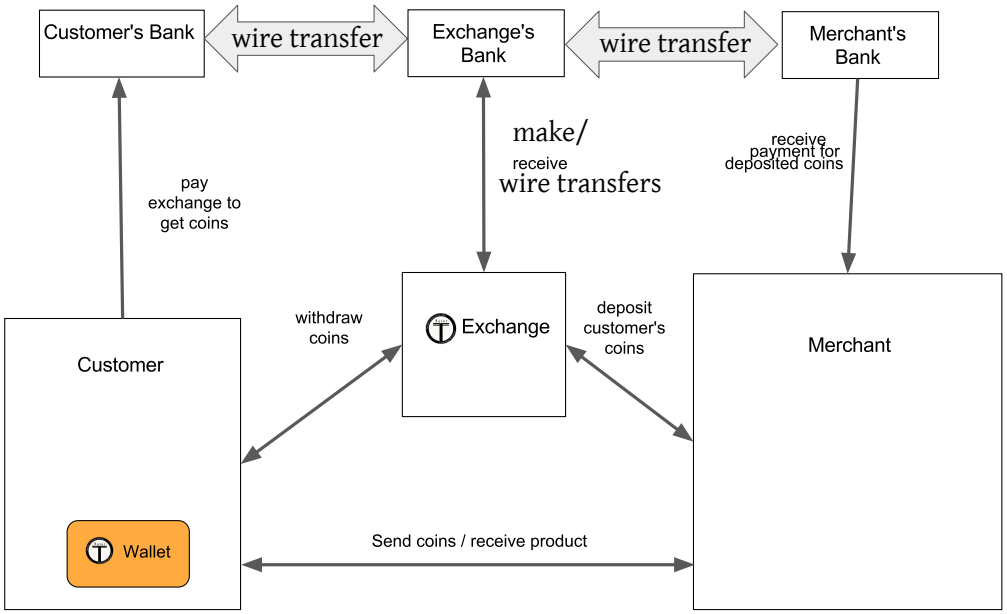


Figure 1.1.: Simplified Taler system architecture [7]

After the registration, the Taler user is able to advise his bank to send money to his Taler wallet. This transfer is made trough the Taler exchange, which anonymizes the cash flow. The money that the user receives can be found in the wallet. It has the same currency as specified in the bank transfer.

From now on, the user can perform payments and acts thereby fully anonymous to its merchants or to its bank. Since the amount of money in this wallet is anonymized, there is consequently also no way of refunding the money in case of theft or loss of user data.

If a payment is made, the merchant receives the money coming directly from the user's wallet. An automatic system becomes active and informs the government, that a transaction has been observed. There is a minimal amount of information gathered about this transaction to ensure that the merchant cannot evade taxes.

Through the Taler exchange, the merchant can transfer its earned money back to its preferred bank.

This example is very simplified. For a more detailed insight into the Taler project, it is recommended to visit the website [2].

[2] www.taler.net

1.2. About vending machines

In order to design an embedded system, which shall interact with a vending machine, it is very important to get familiar with the vending machine's structure and communication interface.

Fortunately, every vending machine's communication protocol is standardized by a consortium of the NAMA (National Automatic Merchandising Association), the EVA (European Vending Association) and the EVMMA (European Vending Machine Manufacturers Association). This protocol standard is called MDB/ICP, which stands for Multi Drop Bus / Internal Communication Protocol [8].

NOTE: *The following remarks about the MDB/ICP protocol shall be interpreted as an introduction into the thematic to the reader. The most important conceptual compositions about the interface are explained in order to get the necessary understanding about cashless devices in vending machines. However, some details are not (or not fully) covered.*

A vending machine across this standard consists of a Vending Machine Controller (VMC) which is connected to up to 32 slaves. Every slave represents a physical device such as coin changers, coin validators and of course cashless devices. Each peripheral type has its defined address, its own instruction set and a specified state machine. The peripherals are connected in serial with the VMC and are using a multidrop-bus interface.

Generally, the master (VMC) checks every slave periodically for activity. This periodical check is called polling and will happen every $25 - 200\,\mathrm{ms}$. A peripheral has three different possibilities about how to answer to a master poll. They can either answer with an acknowledge signal (`ACK`) or they can answer with specific process data in order to initiate an interaction with the VMC. If the master's message integrity is not validated, the slave can also answer with an non-acknowledge signal (`NAK`). When the peripheral does not respond in a particular time, the VMC will automatically interpret this silence as a `NAK`. Every communication will be initiated by the VMC. The peripheral is not allowed to start an interaction by itself. This must always be done subsequently to a poll command.

If the peripheral sends data to the VMC, it will receive an `ACK/NAK` in return. The master also has a third option, the retransmit signal (`RET`). This option will force an immediate retransmission of the slave's previous sent data.

1.2.1. Communication

On the lowest layer, the communication is done by using a $9\,\mathrm{bit}$ UART interface with eight data bits, one mode bit and one stop bit:

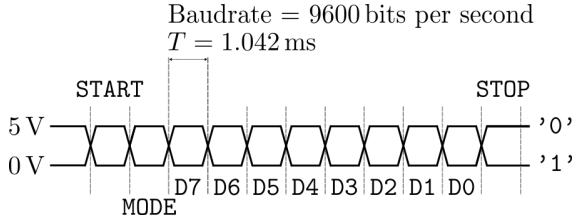


Figure 1.2.: UART-Frame of an MDB/ICP message byte

The maximum message length in MDB/ICP amounts 36 of such frames (including a checksum- and address byte). This means that there can be up to $34\,\mathrm{bytes}$ of effective data in one message block. The `MODE` bit is introduced to distinguish between different types of data. This is urgently necessary since all bus participants are connected in serial with one single transmit- and receive-line. Hence, this special bit is used for two different purposes.

- **Master to slave transmission:** The master has the ability to send data to a specific slave. This is done by sending the address of this slave in the first message frame. By setting the `MODE` bit to `'1'`, he empowers

the slaves to interpret the data in this specific frame as an address. From this point on, only the addressed slave has to listen actively to the following data.

- **Slave to master transmission:** Slaves do not need to send the address-byte, since they are only able to talk to the master. Even if all other slaves could technically read this information too, it is not intended to allow such communications [3]. In this case, the `MODE` bit is used to signalize the last message frame of an outgoing communication. This end-token is necessary because there can be a lot of traffic on the master-receive-line, since all slaves are communicating through this single wire.

At the end of each message block, a checksum-byte is added to protect against transmission-errors. However, there are three cases where no checksum-byte has to be calculated:

- The master/slave sends an `ACK` signal
- The master/slave sends a `NAK` signal
- The master sends a `RET` signal

The data for the checksum-byte is obtained by a simple addition of all message bytes (except of the checksum itself, naturally). If the calculated checksum is too big to get stored in one byte, it is truncated. The device, which receives the message, will do the same calculations. If the checksum differs from the received one, there is probably a transmission error. The device should signalize this error by answering with `NAK`.

NOTE: *This is only a fictitious example. There is no such slave and consequently no such slave data specified in the standard.*

The following example shows a minimal master to slave message block:

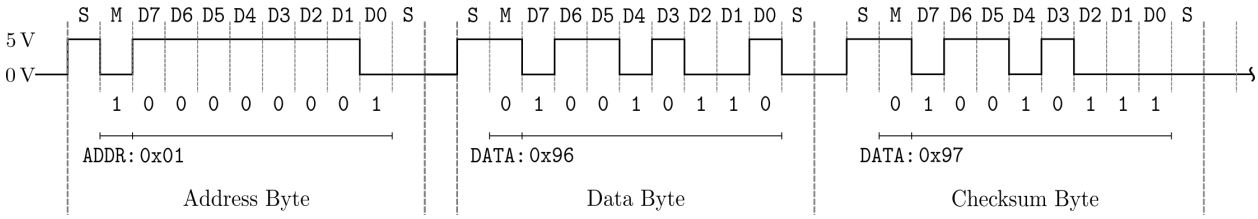


Figure 1.3.: Minimal master to slave MDB/ICP message block

In the figure above, it should be clearly visible how the `MODE` bit is used to address the slave, which listens to `0x01`. The following data byte contains specific instructions to this slave which are specified in the standard. There could, as previously mentioned, even be more data bytes in the message block. At the end of the transmit, the master calculates the checksum by adding both data values (`0x01 + 0x96 = 0x97`). If the checksum increases to a number bigger than representable in one byte, it gets truncated. As an example, a checksum of `0x2F6` gets truncated to `0xF6`.

Since there is no unique end-token specified to signalize the end of a master message, the slave has to detect it through a timeout measurement. Actually, there are a few time constraints specified to ensure safe communication. This will be discussed later.

[3] There is a File Transport Layer (FTL), which allows communications between peripherals. The VMC will act as a network manager to enable the communication between the two participants. However, this feature is intended for i.e. Loading system parameters and not for standard communication. The FTL will therefore not be covered furthermore in this document.

A possible answer of the slave could look like this:

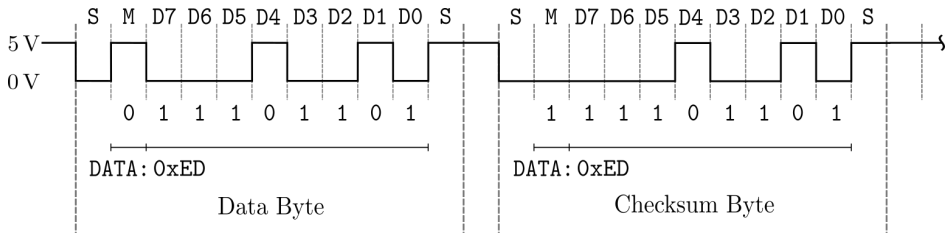


Figure 1.4.: Minimal slave to master MDB/ICP message block

Note that the slave's idle voltage level is at $5\,\mathrm{V}$, whereas the master's is at $0\,\mathrm{V}$. In contrast to the master, the slave does not have to set the VMC's address in order to send a message, since the VMC is the only participant which actually listens on this wire. The `MODE` bit is set by the slave at the last transmitted message frame, in this case the checksum byte. There is also the potential to send more data in one message block as specified by the maximal block length. If the slave has no data to send, it could answer the master's request with the `ACK` signal. The checksum byte could therefore be omitted. When the checksum validation of the master request fails, the slave will answer with a `NAK` signal.

As already mentioned, there are a few time constraints specified by the standard. Such time specifications are very important and must be followed strictly by every developer. The specified constraints are presented graphically to get a clear view.

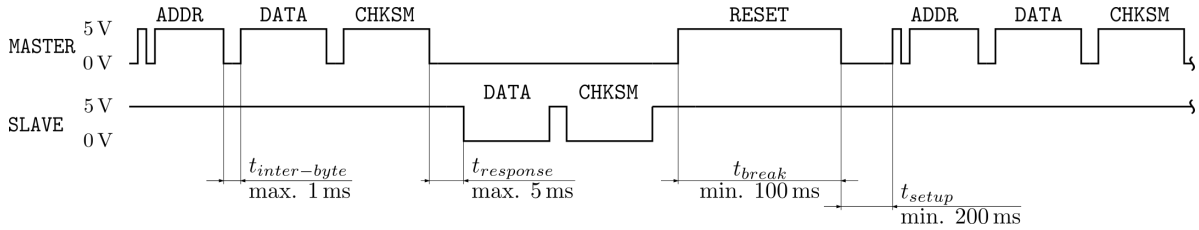


Figure 1.5.: MDB/ICP timing constraints

Such timing properties have to be specified in every bus protocol, in order to allow a proper communication. Especially for larger scale projects where devices from multiple manufacturers should be able to intercommunicate. The various constraints are explained a bit closer:

- $t_{inter-byte}$
  Specifies the maximal allowed time between two message frames inside a message block. Slaves use this value as timeout to detect message ends on the master line.

- $t_{response}$
  One of the most important constrains for peripheral developers. If the slave is not able to answer to a master request in this specific amount of time, communication will be impossible. After this amount of time, the VMC automatically interprets the absence of an answer as `NAK`.

- t_{break}
  The master can force all slaves to reset themselves by pulling the line up for this amount of time.

- t_{setup}
  After a reset, the master has to wait for at least this long before sending new requests. This allows the devices to return to their power on reset (POR) state and getting ready for new instructions.

The now gathered information about the bus communication level explains how the data is transferred in this environment.

1.2.2. Cashless Devices

In this project, the Taler application will be implemented as a cashless device. The cashless device is a standard vending machine peripheral specified by MDB/ICP. There are two addresses reserved for vending machines to provide i.e. simultaneous mobile and credit card payment.

The functionality of most vending machine peripherals is specified by MDB/ICP. This functionality was changed/extended over different protocol versions. To distinguish between the different functionalities of the peripherals, one introduced levels and options. Levels were established to indicate major changes on the instruction set, whereas options are used to go further than the standard.

The master has to ensure that he only sends commands to it's peripherals that are supported by them. In order to do this, he has to determine each peripheral's level before starting any interaction. The same applies for a cashless device. It has to gather the information about the VMC's level before any payment interaction. Generally, it is recommended that new products are supporting all device-levels.

Cashless devices are specified for the following levels and options:

NOTE: This information is based on the MDB/ICP protocol standard 4.2.

Level	Options	Description
1	below	Supports standard commands and Expansion ID command. Readers do not have revaluation capability
	b0*	Reader is capable of restoring funds to card
	b1*	Reader is multivend capable
	b2*	Reader has a display available
	b3*	Reader supports VEND-CASH SALE command
		*bits in the SETUP-Config command
2	above	Supports Revalue, Time /Date, Read User File (obsolete), and Write User File (obsolete) commands
3	above & below	Supports expansion ID command with options and optionally supports commands for features below (bits in the Level 3 Expansion ID command)
	b0**	File Transport Layer (FTL)
	b1**	16 or 32 Bit Monetary Format
	b2**	Multi Currency / Multi Lingual
	b3**	Negative Vend
	b4**	Data Entry
	b5**	Always Idle Session
		**bits in the Level 3 Expansion ID command

Table 1.1.: Levels and options of a cashless device [8]

The cashless device for this project will be a level 3 peripheral. It's state machine is structured like this:

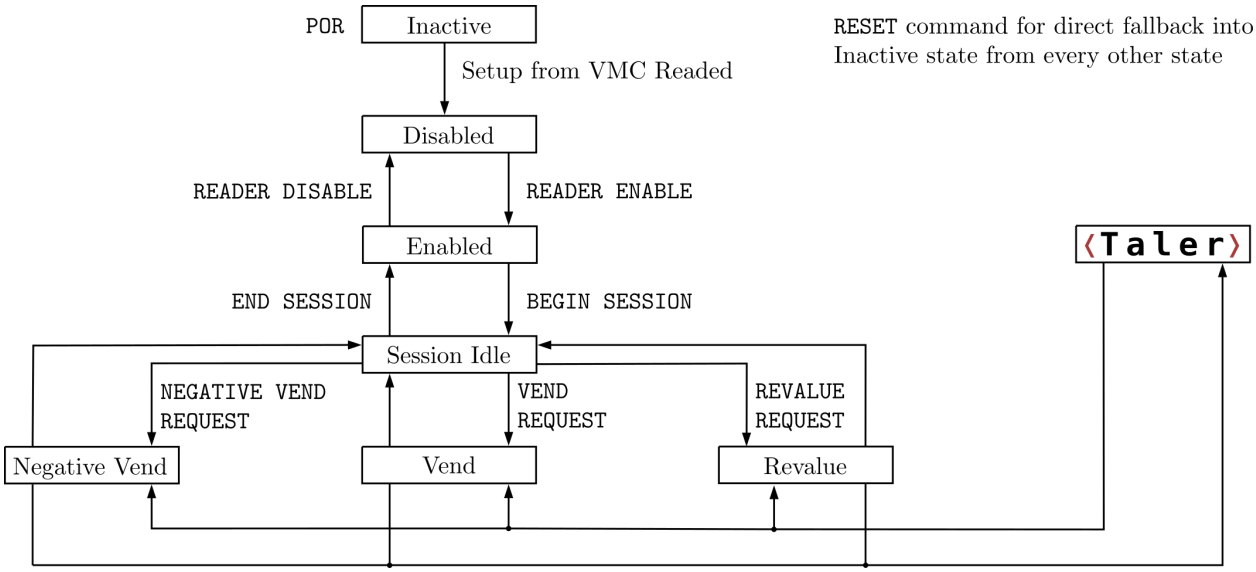


Figure 1.6.: FSM of a cashless device according to MDB/ICP

- **Inactive:**
  This is the cashless device's initial state after a power-up or a reset. The device will answer with the `JUST RESET` command to a master poll in order to start a configuration exchange. During this exchange, the master sends it's configuration data (i.e. It's level) to the peripheral. The peripheral will answer with his own configuration data. After a few data exchanges, the VMC will enable the cashless devices options (if possible). The reader is now initialized and will automatically enter the Disabled state.

- **Disabled:**
  While the Disabled state is active, the device is not available for payments. It will proceed to the next state after receiving the `READER ENABLE` command.

- **Enabled:**
  The device is now ready for transactions. It proceeds to the next state if a valid payment media is read. The cashless device will then inform the VMC with a `BEGIN SESSION` command at the next poll and go further to the next state. However, not every payment method can read a payment media (i.e. payments per QR-Code for the Taler project). For such reasons, there is also an optional feature (Level 3) to be in *Always idle* mode. This optional mode allows the cashless device to start a vending process directly from the Enabled state.

- **Session Idle:**
  A session with a valid payment media has started. If the cashless device is multi currency capable, the currency type will be set at the begin of this state. The VMC can initiate a payment process with `VEND REQUEST`, `REVALUE REQUEST`, `NEGATIVE VEND REQUEST`. Each of these processes will lead automatically back into this state. If the VMC sends the `SESSION COMPLETE` command, the cashless device will respond with `END SESSION` and returns to the *Enabled* state.

- **Vend:**
  The customer has entered a valid item number and now wants to pay. The item number and price is transmitted with the `VEND REQUEST` command. It is now up to the cashless device to handle the purchase. The VMC will poll the device until it has received a `VEND APPROVED` / `VEND DENIED` signal. During this vending process, the cashless device is allowed to give no answer to the VMC polls at all for a non-response time, which is defined by the cashless device itself. Everytime the device will send an `ACK` signal to a master poll, the non-response timer is restarted. If the cashless device has approved the vend, the machine will try to hand over the desired item. After a successful product delivery, the VMC will inform the device with a `VEND SUCCESS` which leads to a fallback into the Always Idle state. If the VMC responds with `VEND FAILURE` (i.e. The product could not be released), it is up to the cashless device to handle the

refund session. The customer can also abort a vend session by pressing the coin mechanism escrow button. The VMC will then send a `VEND CANCEL` request, which will lead to a `VEND DENIED` response from the cashless device.

- **Revalue:**
  This session is entered because some coins or bills were accepted by the machine or a balance is left after a vend. The revalue amount is also appended to the `REVALUE REQUEST` command. The cashless device can define a revalue amount limit, which will not be exceeded by the VMC. This limit amount shall be requested by the VMC at the beginning of the Session Idle state. The device will respond with a `REVALUE APPROVED` / `REVALUE DENIED` and fall back into the Session Idle state.

- **Negative Vend:**
  This vend session is similar to the normal vend state except that the in the `NEGATIVE VEND REQUEST` requested amount of money will be transacted to the customer. Such transactions are used for can- or bottle-return machines. However, the VMC has to ensure that the items are checked and not longer accessible to the user in order to prevent fraud. This vending session is also an optional level 3 device feature and does not have to be available.

In general, the cashless device does not need to immediately send process data to a corresponding VMC request. If the data cannot be gathered during the specified response time of $5\,\mathrm{ms}$, the cashless device can also respond with an `ACK` signal or even with no signal at all until the application's maximum response time has passed over. This response time can be defined by the cashless device developer itself. A default response time is specified with $5\,\mathrm{s}$.

If the VMC sends a command, that does not correspond to the current state of the state machine, it will answer with `COMMAND OUT OF SEQUENCE`. The VMC will then issue a reset of the cashless device.

If the VMC has a display, the cashless device can optionally request it for displaying data. Information about whether a display is available and about the display size is exchanged in the Inactive state.

2. Conceptioning

Since the basic knowledge about the system's environment was aquired in the previous part, one is now able to specify the required system characteristics.

This first conceptioning part is primarily focussed on the definition of the embedded system's hardware. The embedded system will therefore be divided into two parts; A MDB/ICP converter module and a Taler application module. This division is made, since both modules have different functionalities and should work independently from each other.

A specification book for each module will support a systematic elaboration of different concepts. The comparison of each concept shall lead to a comprehensible decision.

2.1. MDB/ICP converter module

The converter module will provide the communication interface to the vending machine controller. A vending machine peripheral is connected to the circuit through a standardized 6-Pin connector. There are two pins for the data transmission and reception ($5\,\mathrm{V}/0\,\mathrm{V}$). A third pin provides the data line's reference ground. Additionally, there are two pins available, which provide a $34\,\mathrm{V}/0\,\mathrm{V}$ rated power supply from the VMC. The remaining pin is not connected.

The module must at least provide a voltage converter and a computing unit, which provides the necessary communication interfaces. As additional desire, there could be a serial port interface (USB) which allows a computer to sniff the MDB/ICP traffic and a SD-Card interface where the communication history can be logged and reviewed.

Probably the most crucial issue will be the implementation of the $9\,\mathrm{bit}$ UART interface, which is not very common nowadays.

2.1.1. Specification book

The MDB/ICP converter module's hardware requirements are listed in a specification book. The specifications are separated into three categories:

- **F**: Fixed requirement
- **M**: Minimal requirement
- **D**: Desired requirement

Every elaborated concept must cover at least the fixed and minimal requirements.

Specification book: MDB/ICP converter module			
Nr.	Requirements	F/M/D	Changes
1	**PCB I/O Ports**		
1.1	Standard MDB connector as input	F	
1.2	Standard MDB connector as output (device in the middle for sniffing)	D	
1.3	Connector to communicate with the Taler module	F	
1.4	Power output connector to drive external devices	D	
1.5	USB output for sniffing / debugging	M	
1.6	SD-Card slot for logging	D	
1.7	Power supply input for external supply	D	
2	**Computing unit connectivity**		
2.1	9-Bit Full Duplex UART for MDB/ICP communication	F	
2.2	8-Bit UART/USART/USB for sniffing	F	
2.3	UART/USART/SPI/I2C for communication with the Taler module	F	
2.4	SDMMC/SPI for SD-Card logging	D	
2.5	Total response time of max. $5\,\mathrm{ms}$ (VMC timeout)	M	
3	**Hardware**		
3.1	Optical decoupling of MDB/ICP data lines	F	
3.2	Step-down converter for power supply conversion from vending machine	F	
3.3	Status indication LED's	D	
3.4	Minimized hardware costs	D	
4	**Maintenance**		
4.1	Hardware testpoints for measurements	F	
4.2	Firmware-update interface for computing unit	F	
4.3	Part availability of min. $10\,\mathrm{years}$	M	

Table 2.1.: Specification book for MDB/ICP converter module

2.1.2. Concept A: Direct communication on embedded processing board

Even though modern computers, such as an embedded processing-board, are not directly supporting a $9\,\mathrm{bit}$ UART communication anymore, there still is a workaround available. Technically, it is possible to make a $8\,\mathrm{bit}$ UART interface plus its parity bit acting as a $9\,\mathrm{bit}$ UART interface.

Sending data
The 9th bit's state can be controlled by switching between the `EVEN` and `ODD` parity mode depending on the message content. Therefore, it would be relatively simple to control the parity bit as the MDB/ICP's 9th bit.

Receiving data
An implementation of this part is way more complicated to achieve. The problem occurs at the reception of the MDB/ICP commands. A parity mode (`EVEN` / `ODD`) has to be chosen for the message receiving. But the incoming message does not correspond to this pattern and therefore, there will be a lot of parity errors. The 9th bit's state can then be determined by a recalculation of the $8\,\mathrm{bit}$ data and the knowledge about if there was a parity error or not.

A more convenient method arises with the availability of two additional parity modes. The `MARK` (always `'1'`) and the `SPACE` (always `'0'`) mode. It would be much more pleasant to have the opportunity to activate a `MARK/SPACE` parity mode because it allows a very simple distinction between address- and data bytes. As an example: If the mark parity mode is activated, every address byte passes the parity check and every data byte will fail. This makes a differentiation of the 9th bit very simple to handle. The undocumented `CMSPAR` flag [9] apparently implements the `MARK/SPACE` parities for some GNU/Linux distributions.

Even though this kind of MDB/ICP handling seems to be fairly complicated, there are still some advantages.

Advantages

- **No hardware overhead**
  The converting module and the Taler application could operate on the same processing board. This means that there is no need for an additional computing unit, which reduces the hardware overhead.
- **Reduced part costs**
  By dispensing with an additional computing unit, the total part costs can be reduced.
- **Signal integrity**
  Since the communication is only processed by one controller, there is a smaller risk of getting communication errors. This also has a positive effect on the systems response time.

Disadvantages

- **No dispense for additional hardware**
  Even though this solution comes without an additional computing unit, there is still the need to design a PCB, which handles the power supply and the optical decoupling of the signals.
- **Complicated UART-handling**
  The short feasibility study shows, that the handling of the $9\,\mathrm{bit}$ UART bus can be a real pain and maybe a little experimental. There is the risk of software overhead which reduces the system's speed. This can have a negative impact on the development time, especially for developers which are not familiar with kernel code and conventions.
- **MDB/ICP traffic interrupts**
  Since the MDB/ICP communicates trough a single wire, a lot of traffic is expected which has to be processed very frequently by the embedded system. Even if only a few messages are effectively relevant for the payment system.

2.1.3. Concept B: Indirect communication via MCU

The problem with the uncommon $9\,\mathrm{bit}$ protocol could be ideally handled with a microcontroller (MCU). Fortunately, there are multiple devices on the market, which are able to handle such a communication. As an example, most mainstream ST Electronic's $32\,\mathrm{bit}$ microcontrollers are supporting $9\,\mathrm{bit}$ UART communication.

The message reception and transmission therefore can be handled with the MCU's UART driver. A simple method to convert the vending machine's commands into a more convenient format could be done by truncating the `MODE` bit from the rest of the data. If the MDB/ICP converter knows the address of it's peripheral, it could act as a filter who only transmits peripheral-relevant data. The slave itself thus, does not need the `MODE` bit information, because every data which is transmitted is dedicated data. On the other hand, the converter appends the `MODE` bit to the messages, which are sent by the peripheral to the master controller. Another possibility is to set the `MODE` bit directly inside the embedded board, since it should be relatively simple.

Advantages

- **Useful hardware abstraction layer**
  The additional microcontroller can handle the whole MDB/ICP traffic and only passes necessary data to the payment system. Due to this, a very meaningful abstraction layer is introduced into the system.
- **Generic MDB/ICP converter**
  There is the big opportunity of designing a fully independent MDB/ICP converter which can be used for any MDB/ICP peripheral. The converter could also be used as sniffing tool or as a crash box when a SD-card interface is implemented. Such MDB/ICP converters already exist on the market, however, they are relatively expensive and not open-source. A well-developed system could have a lot of potential on the vending machine market.

Disadvantages

- **Additional controller raises cost**
  The additional microcontroller raises the total parts costs. As an example, an appropriate mainstream STM32 MCU costs about $3\,\mathrm{CHF}$ (single-unit price).
- **More complex hardware-design**
  In addition, the interfacing of the microcontroller requires a more complex hardware design which has a negative impact on the development time.
- **Critical timing**
  This concept's main weakness is probably a slow total response time, since the communication is not done directly. However, the VMC's timeout of $5\,\mathrm{ms}$ should be feasible.

2.1.4. Concept C: Indirect communication via FPGA

As an alternative to the microcontroller solution, one could also use a FPGA (Fixed Programmable Gate Array) to get the job done. However, the program procedure is very similar to the microcontroller concept ones. The FPGA has a much higher flexibility and is generally much faster in signal processing than a microcontroller.

Advantages

- **Flexibility and speed**
  The main advantage of a FPGA is it's very high processing speed. Highly parallel operation could be beneficial for sending data trough different interfaces, even if it is not necessary. However, the fast processing speeds are interesting because the total response time has to be shorter than the VMC's timeout of $5\,\mathrm{ms}$.
- **Proper testing method**
  The FPGA code can be tested (simulated) with a testbench, which checks the behavior of the logic for any possible input. This decreases the chance of having unexpected bugs on the field.

Disadvantages

- **Complicated interfacing and programming**
  A FPGA is a relatively complex module which takes much more effort in developing. The hardware description (*coding language*) is not very intuitive and can be tricky. It is also more difficult to place it on a PCB because a FPGA often has many pins and fairly complex datasheets with many aspects, which have to be respected.
- **Increased cost**
  With the given complexity, the cost can be expected to be higher than with the microcontroller solution.
- **Low acceptance in communities**
  Because FPGA's are much more complicated than microcontrollers, there is a low chance that people will participate to this converter due to missing knowledge.

2.1.5. Evaluation

The three elaborated concepts all have their advantages and disadvantages.

The first evaluation takes place between the direct and indirect communication concepts. The main disadvantage of the direct communication is, that the necessity of an additional PCB in order to handle the power supply and signal decoupling still remains. Additionally, the high bus traffic leads to frequent interrupts which could disturb the payment system during the purchase. On the other hand, the indirect communication leads to a more complex hardware, thus to an increase in cost.

But the indirect communication concepts have huge advantages as well. With the development of a multifunctional converter, there would be a pretty usable testing and debugging tool. With an open source schematic and MCU source code, the converter could get a lot of acceptance in the community, since existing converters are expensive and do not offer a very high flexibility. An external converter also leads to a very useful abstraction to the MDB/ICP bus so that the developer of a vending machine device does not need to bother too much with the bus protocol's communication layer.

In comparison, the direct communication concept does not have such great advantages except of the dispense of an additional control unit. The extra cost of the indirect communication concept should not be too massive in respect to the other hardware parts which are needed for this project.

Therefore the indirect communication concepts (B, C) are preferred over the direct communication concept (A).

For indirect communication there are the possibilities to handle the data conversion either with a microcontroller or with a FPGA.

The main advantages of the FPGA are the very high flexibility and fast signal processing capabilities, which contributes to a short system response time. But a very important drawback is the higher complexity of such FPGA's. The work is not done by simply selecting a chip. There is also the need for additional memory and external clocks. A microcontroller, in comparison, is much simpler when compared to a FPGA, because it usually has everything integrated in one chip.

In consideration of the additional complexity, there is no comparable advantage by using a FPGA. Therefore, the decision falls on concept B which uses an indirect conversion with a microcontroller.

2.2. Taler application module

The Taler application will run on a microprocessor, which uses a GNU/Linux based operating system. Since integration of a modern microprocessor on a PCB from scratch is a complex task, it is recommended to use an existing product from a third party provider. Fortunately, there are a lot of open source single board computer (SBC) projects around the world. Such SBC's offer a ready-to-use fully integrated microprocessor with a variety of standard peripheral connectors.

The most critical issues in the selection of a SBC will be the long-term support of the hardware and the connectivity. It is assumed that the Taler application does not have any special system requirements.

2.2.1. Specification Book

The requirements for the single board computer are listed in the specification book below. The specifications are separated into three categories:

- **F**: Fixed requirement
- **M**: Minimal requirement
- **D**: Desired requirement

Every elaborated concept must at least cover the fixed and minimal requirements.

Specification book: embedded single board computer			
Nr.	Requirements	F/M/D	Changes
1	**I/O Ports**		
1.1	USART/UART/I2C/SPI for MDB converter module data exchange	F	
1.2	USB for NFC-reader	F	
1.3	HDMI / MIPI-DSI for QR-code display	F	
1.4	Ethernet port for Taler exchange	F	
1.5	Power supply input $5\,\mathrm{V}$ or $3{,}3\,\mathrm{V}$	F	
2	**Processing unit**		
2.1	$32\,\mathrm{bit}$ architecture	M	
2.2	$256\,\mathrm{MB}$ RAM	M	
2.3	$2\,\mathrm{GB}$ onboard memory	M	
2.4	Open source design	D	
2.5	Supports GNU/Linux operation system	F	
3	**Maintenance**		
3.1	Cross compile toolchain available	F	
3.2	Part availability of min. $10\,\mathrm{years}$	M	

Table 2.2.: Specification book for embedded SBC

2.2.2. Concepts: List of single board computers

As previously said, there are a lot of providers that offer single board computers. A very pleasant fact is that many products have an open source design. The few most interesting boards are listed below.

Name	CPU	Clock	Memory	I/O Ports	Interfaces	Price
Beagleboard Black	Texas Instruments AM3358 ARM Coretex-A8	1 GHz	0,5 GB DDR3 4 GB eMMC	2x USB 2.0 1x microHDMI 1x Ethernet 1x microSD 2x 46 PinHDR	6x UART 2x SPI 3x I2C 2x McASP 2x CAN	61 CHF
Banana Pi M1	Allwinner A20 ARM Coretex-A7	1 GHz	1 GB DDR3	2x USB 2.0 1x USB OTG 1x HDMI 1x Ethrenet 1x AUX(3.5) 1x SD-Card 1x SATA 1x DSI 1x AV Viedo 1x MIC 1x CSI 1x IR 1x 26 PinHDR	1x UART 1x SPI 1x I2C	41 CHF
Olinuxino Lime2-n8GB	Allwinner A20 ARM Coretex-A7	1 GHz	1 GB DDR3 8 GB FLASH	2x USB 2.0 1x USB OTG 1x HDMI 1x SATA 1x microSD 1x Lipo 160x PinHDR 1xLCD	1x UART 1x SPI 1x I2C	48 EUR
STM32MP157A-DK1	STM32MP157 ARM Cortex-A7 + Cortex-M4	650 MHz	0,5 GB DDR3	4x USB 2.0 1x HDMI 1x Ethernet 1x MIPI DSI 1x AUX(3.5) 1x microSD 2x 48 PinHDR 1x USB-C	4x UART 4x USART 6x I2C 6x SPI	71 CHF
ODYSSEY STM32MP157C	STM32MP157 ARM Cortex-A7 + Cortex-M4	650 MHz	0,5 GB DDR3	2x USB 2.0 1x USB-C 1x Ethernet 1x MIPI DSI 1x AUX(3.5) 1x microSD 1x 40 PinHDR	4x UART 4x USART 6x I2C 6x SPI	56 CHF

Table 2.3.: Specification overview of different SBC's

2.2.3. Evaluation

Unfortunately, a big problem arose during the research about the long-term support. None of the SBC's listed above has a guaranteed part availability of more than $4\,\mathrm{years}$. There are also other proprietary industrial grade SBC providers, but their products are mostly overdimensioned in terms of computing power and generally multiple times more expensive than the ones above. Another downside is that all boards are coming with a lot of peripherals and features that are not really needed in this project's use case.

This difficulty leads to the conclusion, that it may be still a good trade off to develop an own single board computer, which is tailored to the application's requirements. This is possible because the microprocessors themselves usually have a guaranteed lifetime. Only the SBC's as entities themselves do not offer such availability.

To shorten the development time, one can reuse some parts of one of the open source designs listed above. The STM32MP1 microprocessor family seems like an ideal CPU, since it consists of a $32\,\mathrm{bit}$ ARM cortex-A7 dual core processor, which can operate on ST's GNU/Linux distribution, and an ARM cortex-M4 core where the MDB/CP converter functionality can be placed. This combination of computing cores in one chip is ideal for this project's requirements. The data exchange between both modules could then happen over shared memory segments and would therefore be very fast and safe.

Especially the Seeed Studio's ODYSSEY board offers an interesting feature, because it has an interchangeable computing module [10]. This computing module includes the microprocessor with it's RAM and an embedded eMMC memory chip. The computing module STM32MP157C SoM (System on Module) can be bought separately and also is specified as open hardware.

The Seeed SoM would offer the project the possibility to design a tailored embedded platform, where the most critical parts (the microprocessor and RAM) are outsourced on an exchangeable module. A drawback of this decision is, that the SoM neither offers the guaranteed part availability of $10\,\mathrm{years}$. Nevermind, the problem is defused since it's design is available as open hardware and could therefore be manufactured at own in the worst case. Fortunately, it also offers the possibility to updgrade the whole system by designing an own, more powerful module with the same connections to the SoM's carrier board.

For small part quantities, the product cost will probably be higher with a self developed board than with a commercial industry graded SBC, but the price difference should fade in larger scales. One has thereby to consider, that with a proprietary SBC product, there would be still the need for an additional MDB/ICP converter module which raises the costs.

The decision is therefore to use the Seeed Studio's SoM as cumputing module and to design an own carrier board to it, which has it's interfaces tailored to the actual requirements.

3. Prototyping

The elaborated concepts are now to be put into practice. In a first step, the goal is to develop an independently operating MDB/ICP converter module. This module will be configured to be compatible to a Raspberry Pi. The Raspberry was chosen because it is probably the most widespread computing module in the world. Additionally, the converter module will use a mainstream STM32 microcontroller. The developed source code of this microcontroller will be reused for the end product later.

This stepwise approach to the final product is introduced to allow a better focus on the specific problems which leads to a more regulated development progress.

3.1. STM32 software validation

Before any hardware is designed, it is to be verified that the STM32 microcontroller is capable of fulfilling the desired task. A possible critical issue of the converter module is the specified maximal response time of $5\,\mathrm{ms}$. The goal of this prototype is to specify how long it will take the microcontroller to receive, process and resend MDB/ICP data. Although, one wants to find out how the controller peripherals have to be configured to allow such operations.

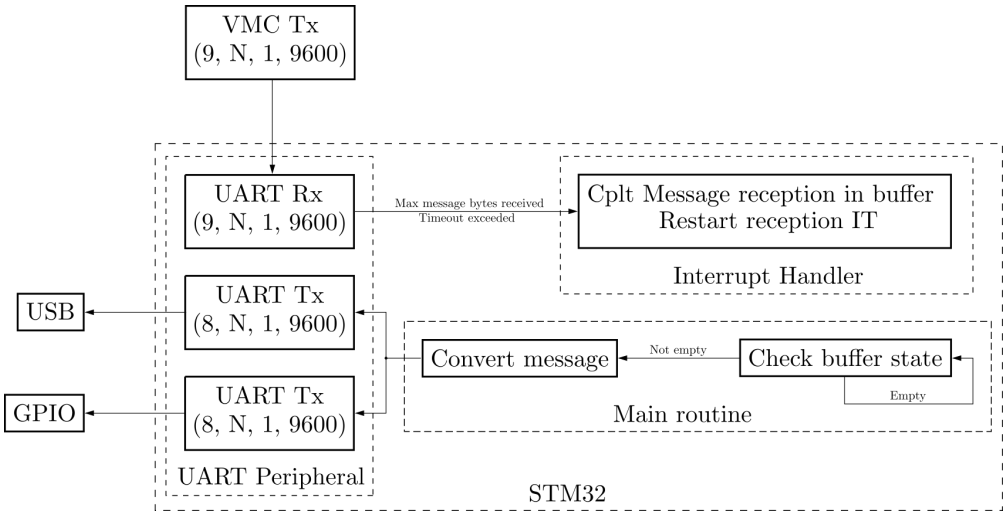


Figure 3.1.: MDB/ICP prototype software structure

The idea is to receive a MDB/ICP master command with the $9-$bit UART peripheral of a STM32 microcontroller. An interrupt gets triggered each time the maximal data length of $36\,\mathrm{bytes}$ was received or the inter byte timeout of $1\,\mathrm{ms}$ is exceeded.

Inside the interrupt handler, the gathered data block gets copied by the DMA controller to a certain place inside a message buffer and the data reception is restarted.

In the mean time, the main application is constantly checking the buffer's state. As long as the buffer is not empty, each message block is converted into a $8\,\mathrm{bit}$ format by truncation of the `MODE` bit. Before the conversion is completed, the checksum- and the address-byte will be checked. If one of them is invalid, the message will be marked accordingly.

The resending of the converted signal will be handled by two $8\,\mathrm{bit}$ UART peripherals. One controller will send the signal over the USB-interface to a connected computer and the other controller's output will be used for measuring the transition time. Therefore, the DMA will be advised to copy the converted data from the message buffer to the UART peripheral. After completion, the buffer data gets marked as consumed and can now be overwritten.

The microcontroller peripherals are initialized with ST Electronic's configuration tool CubeMX [11]. The software is written in C++.

3.1.1. Application test setup

The previously described application is implemented on a STM32 Nucleo board [12]. Nucleos are development boards, that are provided by ST. These boards are available for almost any $32-$bit processor in their product range. The used Nucleo board in this prototype is equipped with a L452RE microcontroller. The L452RE is aimed

at ultra low-power applications and therefore has some special features. It should be pointed out that there is no need for such a low-power controller in this application. The only reason why the L452RE was selected is because it was currently available in the lab. Since all ST microcontrollers are providing more or less the same interface to their peripherals by the hardware abstraction layer (HAL) library, it does not really matter which controller family is used for such a simple test application.

A proprietary MDB/ICP controller is used to emulate the VMC commands. It was manufactured by the company Qibixx and is an extension board to the Raspberry Pi. The communication between the Raspberry Pi and it's extension board is managed via a $8\,\mathrm{bit}$ UART bus with a baud rate of $115\,200\,\mathrm{bit\,s^{-1}}$.

The Raspberry Pi itself is a model 3+. It's operating system, called Raspian, is based on debian, thus fairly familiar out of the box and can easily be accessed over SSH[1].

Additionally, a saleae logic analyzer is used to measure the signals on the bus layer. This device will be able to measure the signal transition time trough the ST microcontroller.

The following picture shows the final test setup.

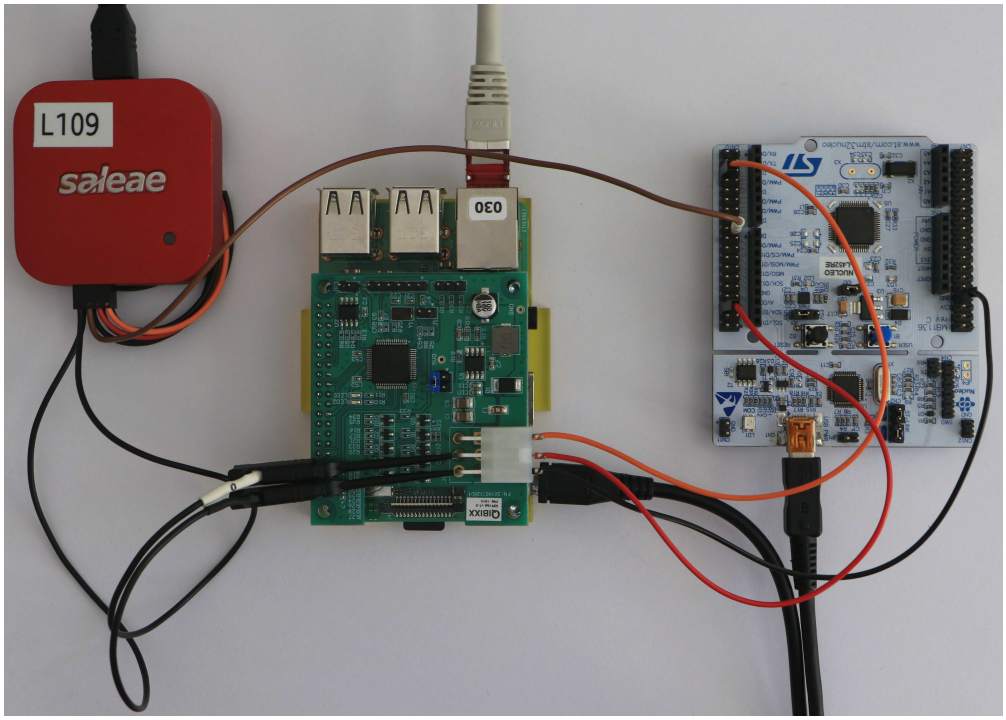


Figure 3.2.: Prototype composition with logic analyzer, Raspberry Pi and Nucleo

Emulated VMC (center) to saleae logic analyzer (left)

- Black cable with white label:
  MDB/ICP master-to-peripheral line.
- Black cable:
  Ground reference.

Emulated VMC (center) to STM32 Nucleo (right)

- Red cable:
  MDB/ICP master-to-peripheral data line.

[1]The operating system installation and SSH set-up procedure is documented in the git repository of the project.

- Orange cable:
  MDB/ICP peripheral-to-master line. Currently not in use.
- Black cable:
  Ground reference.

STM32 Nucleo (right) to saleae logic analyzer (left)

- Brown cable:
  Converted message UART output.

3.1.2. Measurements

The logic analyzer finally delivers the measurement results.

Figure 3.3.: Measurement 1: Minimum of a VMC message

The minimum message length (figure 3.3) has a transition time of $1{,}2001\,\mathrm{ms}$. One has thereby to consider, that the UART controller has to wait until the inter block timeout has exceeded. This timeout value has been specified as $1{,}042\,\mathrm{ms}$. Actually, the value specified in the MDB/ICP standard would be $1\,\mathrm{ms}$. However, in ST's UART interface, one can only specify values as multiples of the baud rate period time.

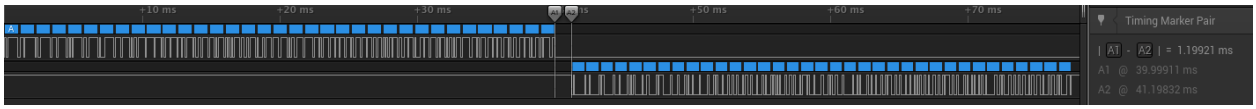


Figure 3.4.: Measurement 2: $35\,\mathrm{byte}$ VMC message

It is very pleasant to observe that the transition time does not rise with a higher message length (figure 3.4). The reason why it is even shorter ($1{,}199\,21\,\mathrm{ms}$) than in the previous message could be because the resending of the converted signal is currently handled in the main routine of the STM32 application and not via interrupts.

Figure 3.5.: Measurement 3: Maximum of a VMC message

When the maximum message length is received (figure 3.5), the transition time is even shorter ($0{,}154\,78\,\mathrm{ms}$). Since the message length is defined with $36\,\mathrm{bytes}$ one does not have to wait for the inter block timeout, but can immediately start with the conversion.

Overall, one can say that the measurement results predict that the STM32 microcontroller will be able to handle MDB/ICP conversion. An open question is which bus should be used to intercommunicate between the Raspberry Pi and the extension board. In order to guarantee the total response time of $5\,\mathrm{ms}$, one should theoretically select an interface which is able to transfer data at a speed of at least $400\,000\,\mathrm{bit\,s^{-1}}$.

This intercommunication transfer rate is chosen with the following calculation:

- Transition time STM32 message reception: $1{,}2\,\mathrm{ms}$
- Intercommunication STM32-Raspberry ($35\,\mathrm{B}$ at $400\,000\,\mathrm{bit\,s^{-1}}$): $0{,}7\,\mathrm{ms}$
- Raspberry Pi transition time (estimated): $2\,\mathrm{ms}$
- Intercommunication Raspberry-STM32 ($35\,\mathrm{B}$ at $400\,000\,\mathrm{bit\,s^{-1}}$): $0{,}7\,\mathrm{ms}$
- Transition time STM32 message sending (estimated): $0{,}2\,\mathrm{ms}$
- Total response time (estimated): $4{,}8\,\mathrm{ms}$

Within this intercommunication time, the bus communication overhead is not respected. However, this worst case scenario is fairly unrealistic.

In case of a cashless device, one can argue that there is no need for a direct response with process data. The cashless device on the Raspberry Pi could simply decide to respond with a shorter `ACK` signal to the direct request and will then send the requested data as a response of the next master poll. This fact allows, that the transfer rate could even be slower.

Another method to avoid this timing problem could be implemented by using the STM32 as buffer device where standard polls are answered automatically. If the Raspberry Pi sends data to the microcontroller, it will be stored and used for the next poll. This leads to an asynchronous communication which enables a higher flexibility for the MDB/ICP peripheral on the Raspberry Pi.

However, the exact behaviour of the Raspberry Pi has to be measured in order to determine which approach is the most appropriate. Since the experiment could confirm that the setup is qualified for the expected tasks, one can now start with the development of the associated hardware. To keep the bus type for intercommunication between both boards flexible, one will implement the signal paths for I2C, UART and SPI on the PCB. Since only one bus type will be needed in the final module, the paths will be equipped with solder bridges which enables to connect and disconnect them.

Part II.

Thesis

4. Introduction

During the preliminary study, the Multi Drop Bus / Internal Communication Protocol (MDB/ICP), which specifies the internal processes of a vending machine has been studied and was explained to the reader in section 1.2. The Taler payment platform was separated into two independent modules, whereof each one is defined by a specification book in chapter 2. The modules are called:

- MDB/ICP converter module
- Taler application module

For each module, different concepts were elaborated and compared to each other based on their specification book's requirements. This decision-making process led to the development of two Printed Circuit Boards (PCB).

The MDB/ICP converter module's function is to provide an interface to the vending machine, which translates the MDB/ICP bus messages and sends them in a readable format to the connected peripheral. The hardware interface includes the standard bus connector, signal decoupling optocouplers, and a step down voltage converter for bus power supply. Message conversion is handled by a $32\,\mathrm{bit}$ microcontroller, which supports the required $9\,\mathrm{bit}$ UART communication. Previous tests have shown, that a microcontroller's capabilities fulfill the requirements (see chapter 3).

During the bachelor's thesis, a converter module was developed and manufactured. It is further called MDB/ICP gateway and is an extension board to the Raspberry Pi. This gateway provides the above listed hardware interfaces and also has additional features, which will be discussed during the next chapter.

Later, an embededded Taler platform was developed. It is based on an interchangeable, open-hardware computing module from the manufacturer Seeed Studios [10]. This module (also called System on Module *SoM*) consists of a microprocessor, its RAM and an additional eMMC memory block. The platform provides the vending machine interface along with other connectors that will be used to connect a NFC-Reader and a display. Unfortunately, this platform could not be finished during the thesis. The missing part is the design of the PCB. It will take approximately a week of work until the platform is ready for manufacturing.

5. MDB/ICP Gateway

In this chapter, the development steps of the MDB/ICP gateway will be discussed. The hardware tests will give an insight about the capabilities of the designed gateway and possible improvements. There was no software developed for this board. Netherless, some basic ideas about a possible workflow are presented.

The following specification presents a quick overwiew.

5.1. Specification Sheet

MDB/ICP VENDING MACHINE PERIPHERAL GATEWAY FOR RASPBERRY PI

FEATURES

- Standard MDB/ICP interface connector
- Optical decoupled signal lines
- Interface abstraction with microcontroller
- MDB/ICP bus supply
- Up to three independent communication interfaces to Raspberry Pi
- USB service port
- SD-Card data log interface
- open Hard- and Software
- SWD interface

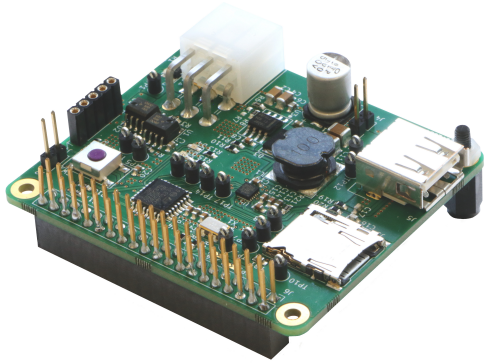


Figure 5.1.: MDB/ICP Gateway

APPLICATIONS

- Vending machine peripheral development
- MDB/ICP bus debugging

GENERAL DESCRIPTION

The MDB/ICP gateway is an ideal development tool for everyone who is interested in vending machine peripheral applications. The gateway follows the MDB/ICP version 4.2 specifications and allows a reliable interface abstraction. Open soft- and hardware offer a high degree of customizability.

The Raspberry Pi extension board is connected to a vending machine trough the provided connector. The $32\,\mathrm{bit}$ ST Electronics STM32G070 microcontroller handles the bus data conversion and implements three independent communication interfaces (UART, I2C, SPI) to the Raspberry Pi.

A Micro SD-Card connector allows to externally store process-data. The USB receptacle is meant as a service port to live monitor bus-data on an external device.

The gateway is powered by the Raspberry Pi's $3{,}3\,\mathrm{V}$ GPIO header pins. The Power supply from the vending machine interface serves the Raspberry Pi's 5V supply pin. The voltage conversion has an efficiency of up to $85\,\%$ and can optionally be disabled.

The Serial Wire Debug (SWD) interface allows programming and debugging of the microcontroller. Alternatively, the microcontroller can be programmed trough the Raspberry Pi I2C or UART.

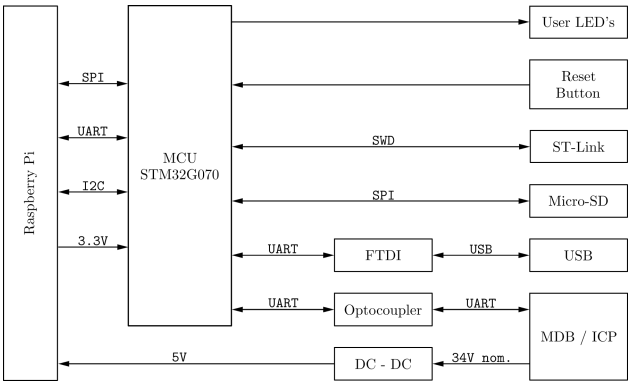


Figure 5.2.: MDB / ICP Gateway block diagram

5.2. Hardware Development

The conceptual work for the planned MDB/ICP converter module is over and the goal is now to design the appropriate hardware. All functional requirements are derived from the MDB/ICP specification and were considered during the system conception.

The hardware development consists of the design of a Printed Circuit Board (PCB) and the selection of it's components. In a first step, the electrical schematic is to be developed. An electrical schematic is represented by a logical and easy readable drawing about how the different electrical components are connected against each other. The PCB design itself will define how these components are physically aligned and connected on the board.

The MDB/ICP gateway hardware development process is discussed in the following sections. Each section will cover a specific module.

5.2.1. Microcontroller

The microcontroller (MCU) is clearly the core of the gateway, as it will handle the entire signal processing. Every communication to and from the Raspberry Pi, the Vending Machine Controller (VMC), the USB and the SD-Card will be controlled by this component. These communication interfaces represent the most important requirements for the MCU selection. As the first prototype measurement results have suggested (see chapter 3), one wants to implement different communication interfaces to the Raspberry Pi.

This decision has the following reasons. First of all, the measurements and calculations have shown, that the communication speed between both devices should be around $400\,000\,\mathrm{bit\,s^{-1}}$ or faster. Probably not every standard bus type will probably offer the same stability at such communication speeds and maybe one wants to go even faster. Another reason is, that the handling of certain bus types could be more complicated than others. Lastly, one does not want to restrict the communication to one single interface, since there could be environments where the Raspberry Pi is already connected to other devices and thus has not its full availability. The implementation of different interfaces thus gives the developer a higher flexibility.

The required interfaces and their purpose are listed below:

- **3x UART**
  MDB/ICP, Raspberry Pi, FTDI-USB (alternatively USB direct on chip)
- **2x SPI**
  Raspberry Pi, Micro SD (alternatively SDMMC)
- **1x I2C**
  Raspberry Pi

Additionally, a product longevity of min. $10\,\mathrm{years}$ is required by the specification book.

The microcontroller selection was limited to ST Electronics products, since the developer is familiar to their $32\,\mathrm{bit}$ microcontroller family and their framework. A parametric search on the ST-MCU-Finder [13] shows, that there are plenty of controllers, which meet these requirements. The final decision was made on the cheapest controller, the STM32G070KB.

The STM32G070KB is a comparatively small microcontroller with only $32\,\mathrm{pins}$ and thus ensures minimal space requirements. Its ARM cortex-M0 core can run on up to $64\,\mathrm{MHz}$. With $128\,\mathrm{kB}$ of Flash memory and $36\,\mathrm{kB}$ RAM, the MCU has relatively big storage capacities compared to its size [14]. It is estimated, that this amount of Flash- and RAM memory will be sufficient. The estimation is based on the stack analyzer data of the source-code, which was written for the prototype measurement application in chapter 3. This code occupies $2{,}61\,\mathrm{kB}$ ($7{,}25\,\%$) of RAM and $17{,}78\,\mathrm{kB}$ ($13{,}9\,\%$) of the G070's Flash memory.

Because of the STM32G070KB's lower CPU clock frequency, the MDB/ICP data conversion will perform slower than the reference in the prototype application. The STM32L452RE microcontroller, which was used for these measurements had a clock frequency of $80\,\mathrm{MHz}$. Even though, the evaluation has shown that the purely data conversion will not be the system's bottleneck.

Before the selected MCU can be used inside the schematic, it is necessary to initialize it's pins. This process is done with ST Electronics' microcontroller initialization tool CubeMX [11]. The tool offers an overview of all available peripherals on the controller and enables easy initialization of them. It is the task of the developer to assign the MCU pins in a reasonable order.

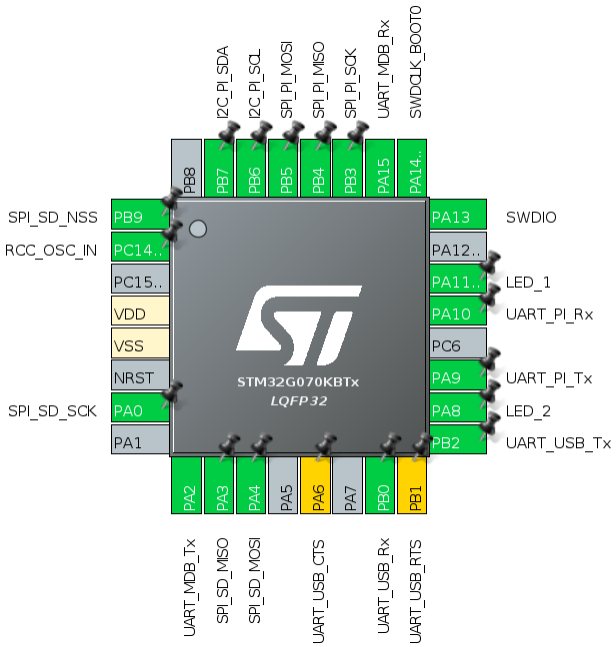


Figure 5.3.: CubeMX view of STM32G070KB's pinout

The pins of a particular peripheral should be as close to each other as possible to allow a more comfortable signal routing on the PCB. Unfortunately, this is not always possible, since not every pin offers the same functionality. The trade-offs made during this design are mostly concerning the `SPI_SD_xx` and the `UART_MDB_xx` pins. However, this pinout will not be a big issue during the routing process, since these interfaces are not very critical in terms of signal speed.

It is always very important to double check the initialization to ensure that no peripheral has been forgotten. The extension board is equipped with two LED's, which can be controller by the MCU over the timer peripheral or a simple General Purpose Input/Output (GPIO) pin.

Programming and debugging of this microcontroller will be done trough the Serial Wire Debug (SWD) interface. There will be also the possibility to program it from the Raspberry Pi through the I2C or UART interface.

Additionally, the controller is be equipped with a $12\,\mathrm{MHz}$ external oscillator as bypass clock source. External clock generators promise to be more accurate and less temperature sensitive than the internal ones. They are generally used, when dealing with high speed bus signals that need accurate timing. It is hard to determine whether this external clock is actually required in this application. The decision was to implement it as a precaution.

Now, the microcontroller is ready for the insertion into the schematic.

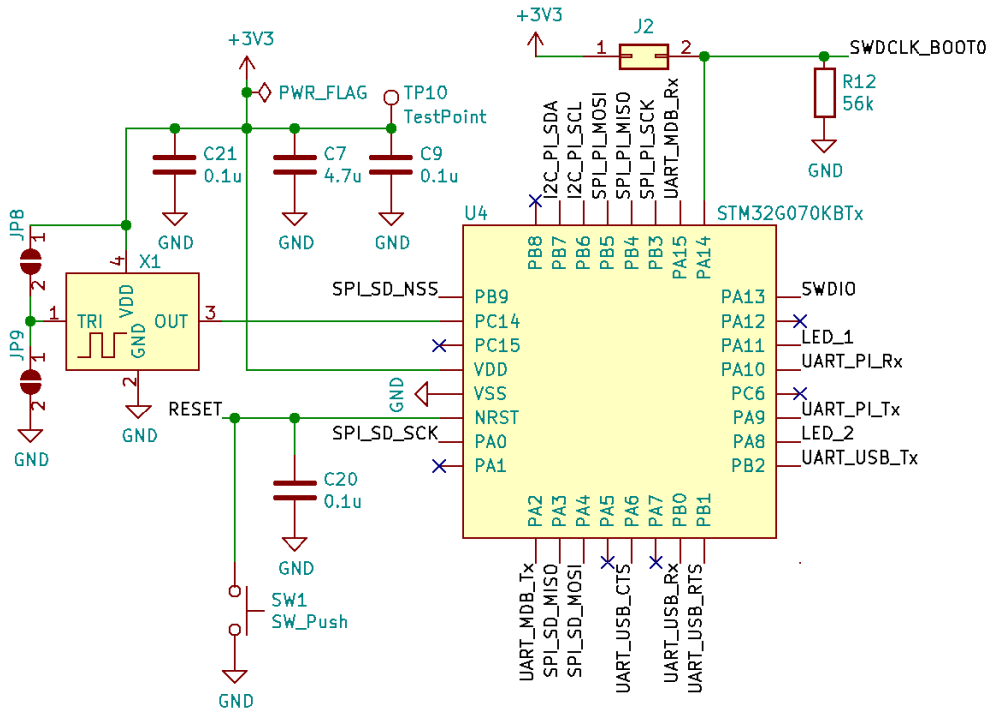


Figure 5.4.: STM32G070KB as schematic symbol

During the schematic design, it is always recommended to read the manufacturer's design guides and datasheets. In this case, the decoupling capacitor values of `C7` and `C9` (figure 5.4) were selected based on the MCU's datasheet [14]. For the PCB design, it will be important to place these capacitors as close to the microcontroller's `VDD` pin as possible to ensure a good decoupling. The design of the reset button `NRST` is also based on the MCU's datasheet. It is low active and uses a capacitor `C20` as low pass filter.

The external $12\,\mathrm{MHz}$ bypass oscillator must be placed as close to the clock input pin `PC14` as possible to ensure a minimal signal distortion by outer effects. The oscillator itself is decoupled with the capacitor `C21`. Solder jumpers `JP8` and `JP9` are implemented to enable or disable the oscillator. If no solder connection is made, the component will be enabled.

The SWD is the main interface to program and debug the MCU. This ARM controller specific protocol uses two wires for communication. The `SWDCLK` on (`PA14`) as clock line and `SWDIO` on (`PA13`) as I/O communication line. With ST Electronics' programmer ST-Link [15], the developer will be able to program and debug the MCU from a computer. There will be a dedicated pin header to connect the ST-Link as shown in the figure below.

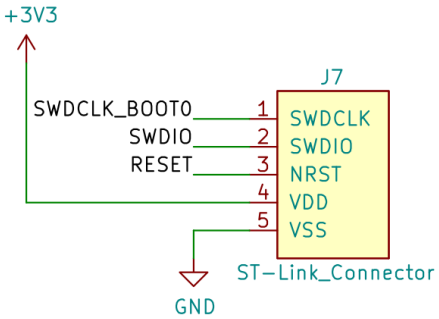


Figure 5.5.: ST-Link programming interface

Alternatively, the STM32G070 can be programmed directly from the Raspberry Pi over the I2C or the UART interface. The ST Electronics application note AN2606 [16] introduces the microcontroller's different memory boot options. After a system reset, the MCU will activate it's bootloader, when the `BOOT` pin (`PA14`) is pulled high. The user can force this pull-up by closing the bridge `J2` (figure 5.4). The flashing of the microcontroller is then performed with separate software.

According to the application note AN2606, the bootloader interface wires must be pulled high as shown in the figure below. The resistor values correspond to the application notes' recommendations.

It turns out, that this interface could be improved by enabling the possibility to control the microcontroller's `BOOT` and `NRST` pins (figure 5.4) with the Raspberry Pi GPIO pins. This would allow the developer to perform remote MCU programming and thus had a lot of potential. In a second version of this gateway, one should definitely implement this feature.

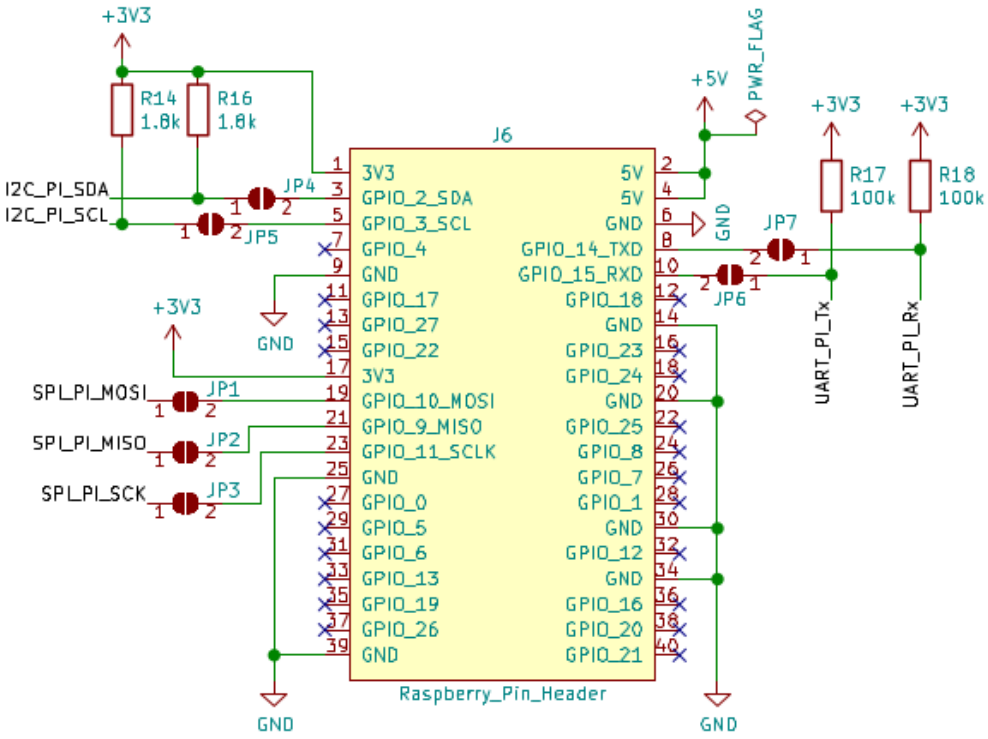


Figure 5.6.: Raspberry Pi pin header interface

The communication interfaces (UART, I2C, SPI) to the Raspberry Pi have to be physically enabled by connecting the solder bridges `JP1–JP7` (figure 5.6).

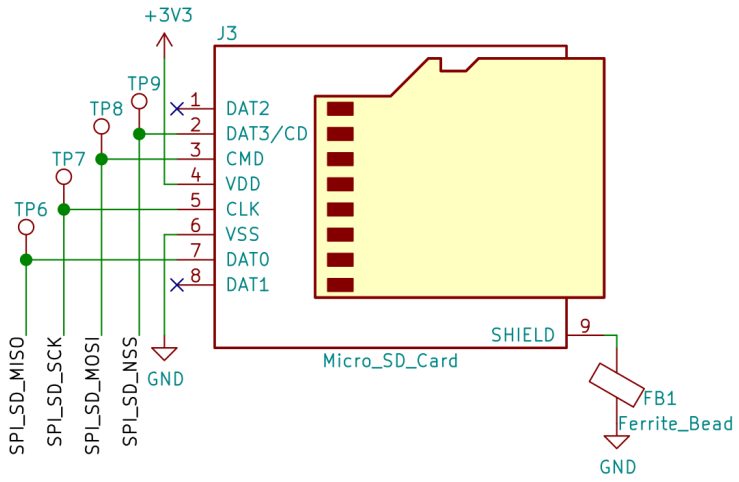


Figure 5.7.: Micro SD-Card interface

The SD-Card is accessed trough a SPI interface. It is intended to provide an external storage, which can be used to log the MDB/ICP bus traffic.

Unfortunately, one did not think about the mechanical SD-Card detection interface, which is offered by various receptacle manufacturers. This feature will be definitely implemented in a next version of the MDB/ICP gateway.

5.2.2. Bus Decoupling

A special functionality of the vending machine interface is represented by the optical signal decoupling, which is prescribed by the MDB/ICP standard. Optocouplers offer a simple and reliable way to decouple communication signals. Generally, these devices use the data signal power to enlighten a LED, which is encapsulated in a chip. A photodiode detects the emitted light and switches a transistor to pull down the signal receiving data line. Important specifications of such optocouplers are the rated current of the signal to empower the LED and the rise- and fall-times of the photodiode, which mainly limits the data-throughput. Additionally, the voltage insulation is surely an important parameter when working in environments, where these values are specified. In case of the MDB/ICP protocol, no insulation voltage is prescribed. Hence, this parameter was not specially considered during the component research.

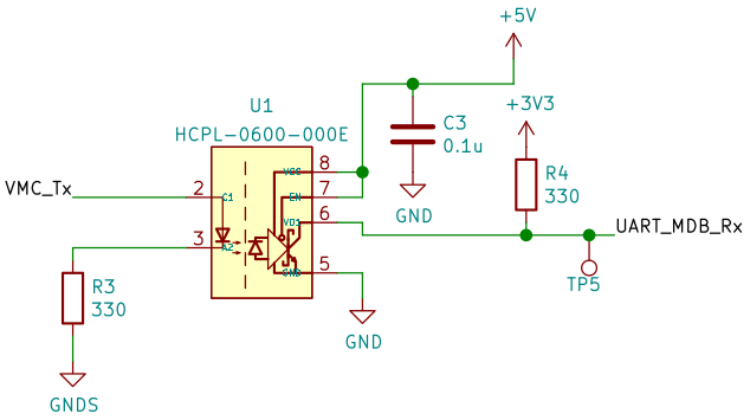


Figure 5.8.: VMC signal reception

According to MDB/ICP, the data transmission line is able to deliver up to $15\,\mathrm{mA}$ at the rated bus voltage of $5\,\mathrm{V}$ per device [8]. The chosen optocoupler from Broadcom (figure 5.8) specifies a LED forward voltage of $1{,}5\,\mathrm{V}$ at $10\,\mathrm{mA}$. The LED threshold current to trigger the photodiode is rated at $5\,\mathrm{mA}$. Propagation delay time from high to low output and inverse is rated at $50\,\mathrm{ns}$ [17].

The vending machine's signal lines are referring to their own decoupled ground signal `GNDS`. For the signal reception, the `VMC_TX` line is connected to the optocoupler's LED anode. The cathode side of the LED is connected over a resistor to the signal ground `GNDS`. This resistor `R3` limits the current to $10{,}6\,\mathrm{mA}$.

$$I_{LED} = \frac{V_{bus} - V_f}{R} = \frac{5\,\mathrm{V} - 1{,}5\,\mathrm{V}}{330\,\Omega} = 10{,}6\,\mathrm{mA} \tag{5.1}$$

The MCU signal reception side (figure 5.8) is connected to the transistors collector and can be pulled low trough the transistor switching.

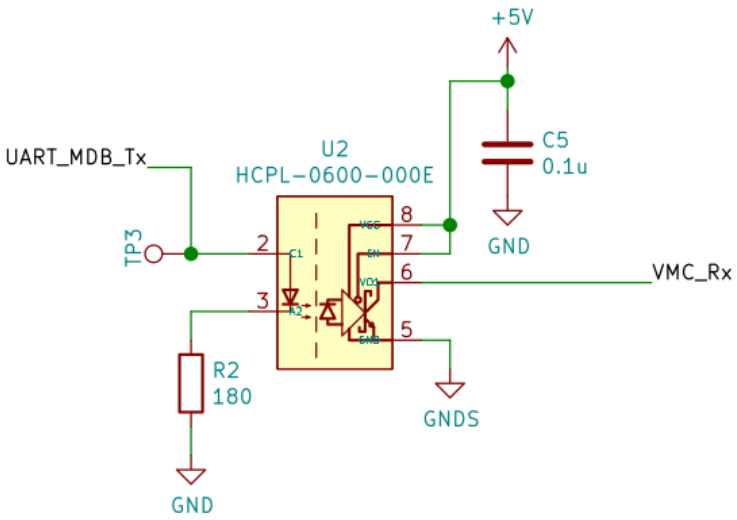


Figure 5.9.: VMC signal transmission

The data transmission to the vending machine works similar as the signal reception logic. The `UART_TX` signal from the MCU is connected to the optocouplers LED. Since the microcontroller only has a pin voltage of $3{,}3\,\mathrm{V}$ one has to select a different resistor value to keep the same current on the LED.

$$I_{LED} = \frac{V_{bus} - V_f}{R} = \frac{3{,}3\,\mathrm{V} - 1{,}5\,\mathrm{V}}{180\,\Omega} = 10\,\mathrm{mA} \tag{5.2}$$

Unfortunately, the transistor side of the optocoupler in figure 5.9 was implemented incorrectly. The problem is, that the 5V supply voltage of the component is generated from the vending machine's bus supply voltage (or delivered from the Raspberry Pi) and is referring to the main ground `GND`. Since one has to switch the decoupled VMC bus signal, the signal ground `GNDS` was connected to the transistor's collector pin, which is also the optocoupler's reference ground. This connection unfortunately makes a current flow trough the optocoupler's internal logic impossible, since the signal ground is decoupled from the rest of the circuitry. Therefore, the component is disabled all the time. On the manufactured PCB, this error was fixed by connecting both grounds `GND` and `GNDS` together trough a wire. Unfortunately, this connection disables the signal decoupling. To correct this bug, an extra $5\,\mathrm{V}$ voltage source, which refers to `GNDS` must be implemented to supply this optocoupler.

5.2.3. USB Device

To offer the developer a service port for live data monitoring, a USB device port is intended to be implemented. Since the selected microcontroller doesn't include a USB PHY, one has to use a dedicated converter chip. A famous manufacturer of such converter chips is called Future Technology Devices International (FTDI). They offer a whole product family of different IC's to handle the USB protocol. The selected component is called FT234XD. It supports USB 2.0 full-speed device standards and has an internal FIFO buffer of $512\,\mathrm{B}$. The chip supports baudrates of up to $3\,\mathrm{MHz}$.

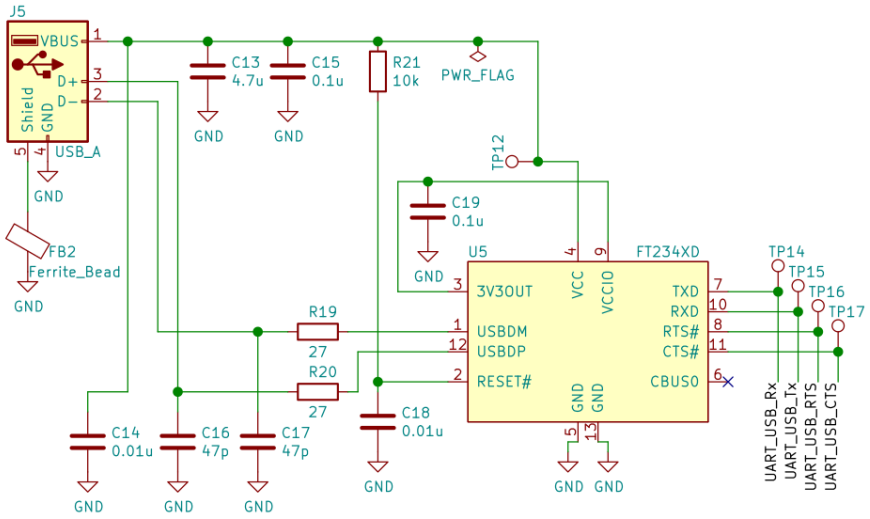


Figure 5.10.: USB to UART converter chip

As the baud-rate of the USB port will not exceed $3\,\mathrm{MHz}$, one will not consider special roules for routing the `USBDM` and `USBDP` traces on the PCB. In this simple case one just followed the design guides inside FTDI's datasheet [18], which recommend the use of termination resistors (`R19,R20`) and load capacitors (`C16,C17`), which can be found in figure 5.10.

The FTDI chip is directly powered by the USB host. During testing, it turned out that the UART's RTS/CTS signals are not necessary for operation.

An unlucky mistake was made by the selection of the USB receptacle type. Later researches showed that the USB Type-A is reserved for USB hosts only [19]. USB devices should use the Type-B or Type-C port. Therefore, the Type-A port must be replaced by an appropriate Type-B or Type-C connector to fulfill the USB device standards.

5.2.4. Power Supply

The MDB/ICP gateway shall be able to power the Raspberry Pi. It is intended to implement a DC-DC step down converter, which converts the VMC's incoming nominal $34\,\mathrm{V}$ supply to $5\,\mathrm{V}$. The converted $5\,\mathrm{V}$ signal shall then power the Raspberry Pi trough the designated pin on it's GPIO header. The Raspberry Pi has internal voltage regulations, which will be used to power the microcontroller circuitry from the $3{,}3\,\mathrm{V}$ net.

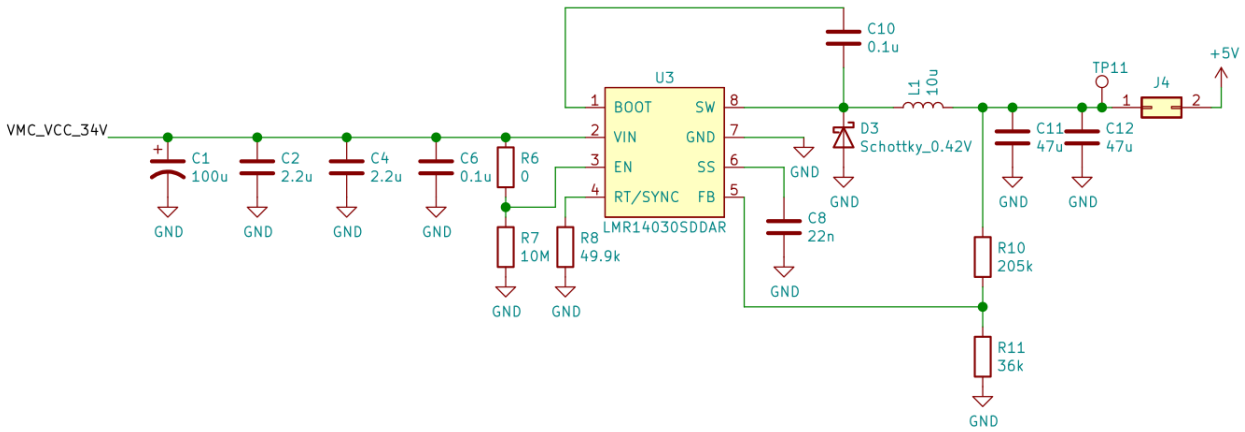


Figure 5.11.: Vending machine interface power supply

DC-DC converters are often used in any kinds of circuits. They offer a way to change power voltage levels with high efficiency. The principe behind it is to switch the input load to an inductor. This inductor will store energy as a magnetic field while the switch is on and will degrade it when the switch is off. A free-wheeling diode is implemented in front of the inductor. It becomes conductive during the switch's off-state, since the degrading inductor draws current trough it. A parallel capacitor on the output will then smoothen the switching noise to a DC-voltage. Most DC-DC converter chips use a fixed PWM frequency to drive a mosfet switch. The duty cycle can be adapted to handle various loads.

When designing a DC-DC converter it is important to keep in mind, that the switching frequency of the mosfet influences the values needed for the inductor and capacitor at the output stage. A higher switching frequency will lead to smaller values and therefore smaller chip sizes but higher radiations and a lower switching frequency to bigger components but less radiation.

The selected chip from Texas Instruments (`U3` in figure 5.11) is capable to switch up to $40\,\mathrm{V}$ with it's integrated mosfet. The switching frequency can be selected from $200\,\mathrm{kHz}$ up to $2{,}2\,\mathrm{MHz}$ trough an external resistor on the `RT/SYNC` pin (see figure 5.11). A switching frequency of $500\,\mathrm{kHz}$ was chosen, since it seems to be a common value.

The input and output filter values are selected or calculated according to the converter's datasheet [20]. The most critical part during the converter design is the selection of the output filter components. The inductor needs a low dc-resistance and the capacitor a low ESR in the range of the controllers switching frequency to avoid voltage ripples as good as possible. Another important part on the output stage is the schottky diode `D3`. This component is responsible to allow a current path during the magnetic field degrading process of the inductor when the switch is off. The diode has to provide a fast response time and a low forward voltage to maximize the converter's efficiency. The whole output stage of the converter has to be aligned very compact to allow short current paths and low impedance traces.

Through the feedback pin `FB` in `U3`, the converter is able to control the output voltage, which is adjusted by the resistor divider `R10` and `R11`. The voltage supply from the vending machine has to be enabled/disabled with the jumper `J4`.

The MDB/ICP standard claims, that the nominal input voltage range will be $34\,\mathrm{V}$ (rectified and filtered) or $24\,\mathrm{V}$ RMS (rectified only). Lower end specifiaction is at $20\,\mathrm{V}$ RMS (rectified only) and upper limitation is at $42{,}5\,\mathrm{V}$ ripple voltage upper limit.

During the DC-DC converter design, it was permanently thought about the $34\,\mathrm{V}$ filtered DC input. A bad designed vending machine controller supply, which delivers permanently more than $40\,\mathrm{V}$ could possibly damage the converter chip, since it is only designed for $40\,\mathrm{V}$.

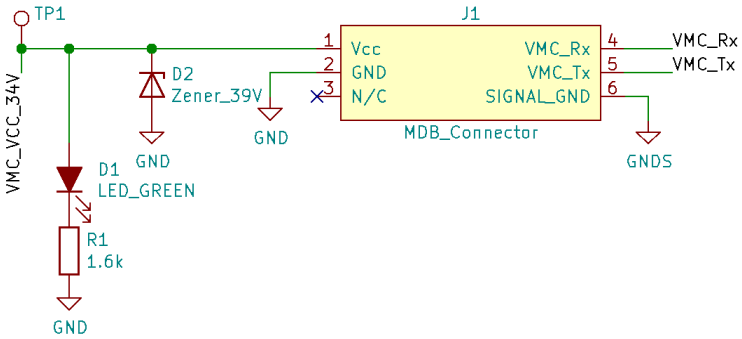


Figure 5.12.: Vending machine interface connector

The converter is therefore protected with a $39\,\mathrm{V}$ zener diode `D2` (figure 5.12).

Retrospectively, it would have been a better idea to chose a higher rated component. A pin compatible, higher rated chip is available and should be used for a second board version.

The power indication LED `D1` could alternatively be placed onto the $5\,\mathrm{V}$ net, since it will produce less power dissipation.

5.2.5. PCB Design

The explained schematic was realized as a two layer PCB. The design itself will not be discussed in any more detail, since it had no special hurdle.

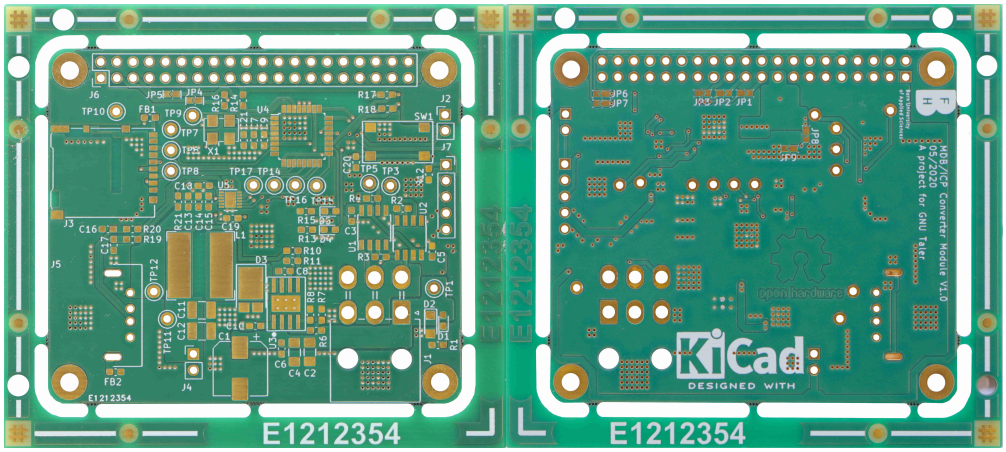


Figure 5.13.: The finished, unsoldered MDB/ICP Gateway PCB

The board is manufactured by Eurocircuits and was designed with the open-source tool KiCad. All surface mount components were placed manually on the PCB and reflow soldered at BFH's HuCE electronics lab. The solder paste was applied to the board with the use of a stencil. Remaining through hole parts were soldered manually.

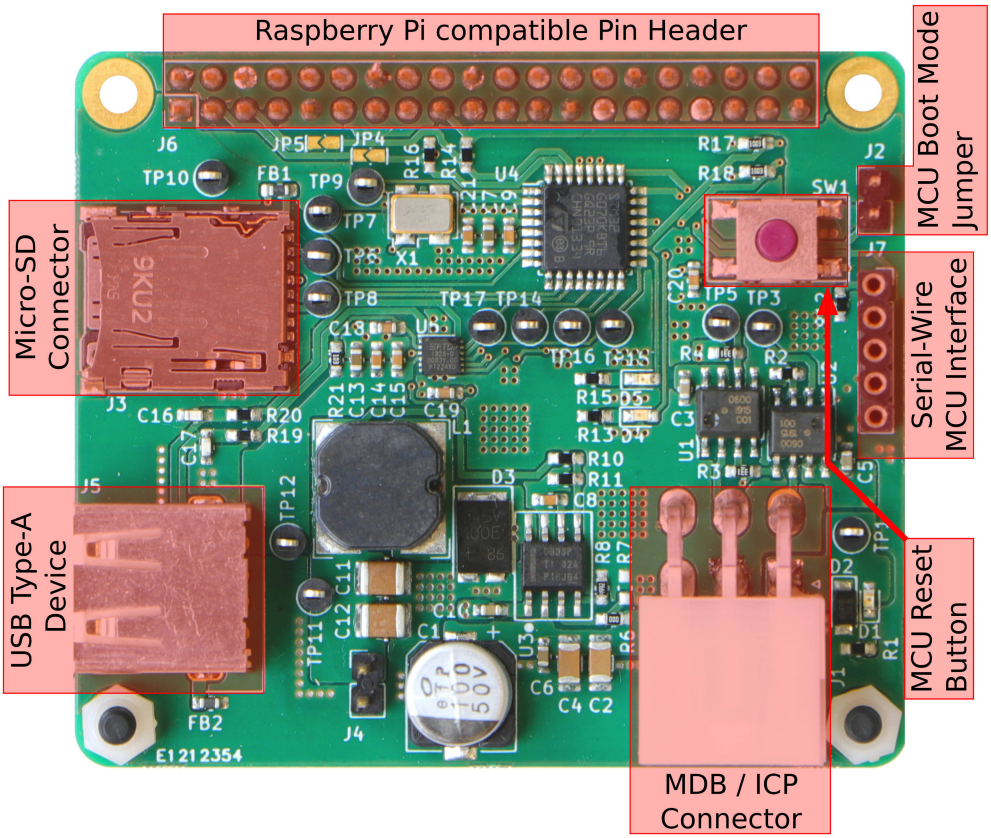


Figure 5.14.: The finished MDB/ICP Gateway PCB with it's different interfaces

The physical interfaces on the MDB/ICP Gateway are shown in the illustration above.

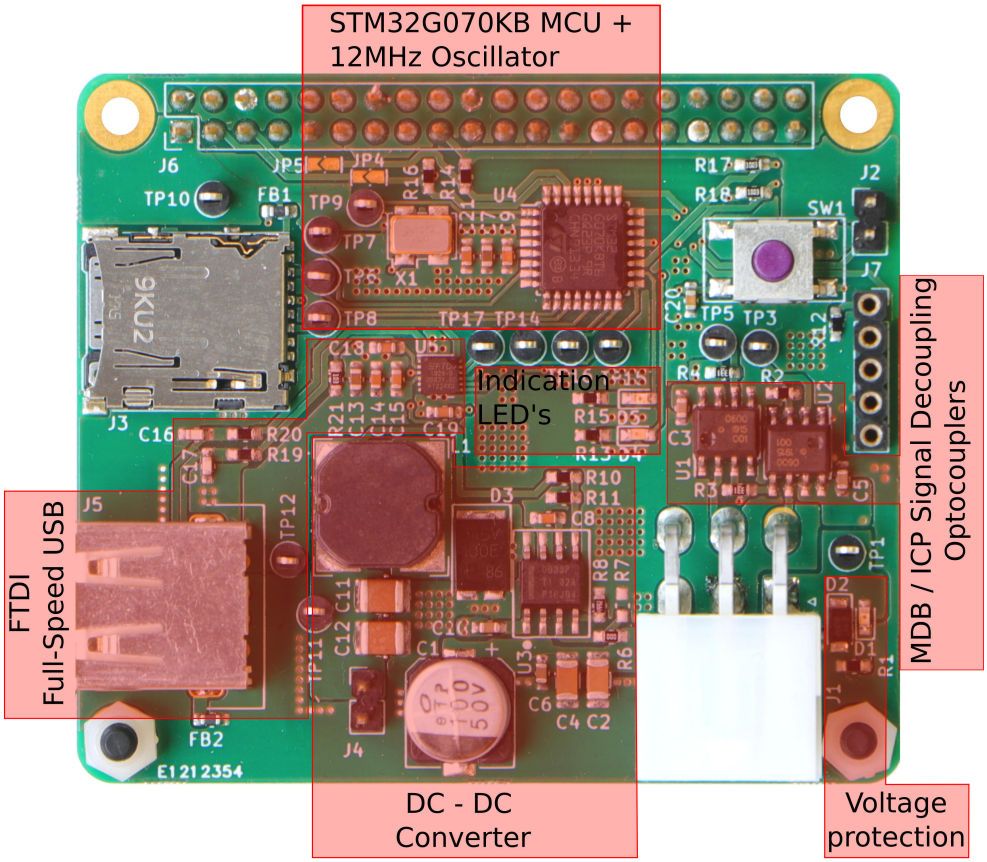


Figure 5.15.: The finished MDB/ICP Gateway PCB with it's different modules

The different modules are grouped together in a compact form.

5.3. Hardware Testing

The developed MDB/ICP gateway must now be tested to ensure, that every module and every communication interface is behaving as intended. A hardware test protocol was elaborated to systematically test every important design for functionality and limitations.

This protocol includes the following testpoints:

- **DC-DC converter**
  Check output voltage, ripple voltage and efficiency factor for loads between $0{,}1\,\mathrm{A}$-$3{,}0\,\mathrm{A}$ in $0{,}2\,\mathrm{A}$ steps.
- **ST-Link**
  Check the functionality of the SWD interface by programming the microcontroller.
- **MDB/ICP**
  Check the bidirectional MDB/ICP signal transmission. Determine the optocoupler's transition time.
- **USB**
  Check the bidirectional USB signal transmission. Determine the speed limitations.
- **Raspberry Pi**
  Check bidirectional signal transmission for SPI, UART and I2C bus. Determine speed limitations. Check MCU programming capabilities over I2C and UART.
- **SD-Card**
  Check bidirectional signal transmission on the SD-Card and determine speed limitations.

The measurements will be discussed in the subsequent sections.

5.3.1. DC-DC Converter Analysis

The MDB/ICP gateway offers the possibility to power the Raspberry Pi with $5\,\mathrm{V}$. Specifications about the Raspberry Pi's power supply requirements can be found on the official website [21].

All Raspberry Pi's must be powered with $5\,\mathrm{V}$ $\pm 5\,\%$ ($4{,}75\,\mathrm{V}$-$5{,}25\,\mathrm{V}$). For the Raspberry Pi Model 3B+, a power supply with capabilities to deliver up $2{,}5\,\mathrm{A}$ is recommended and the model 4 even recommends a $3\,\mathrm{A}$ capable supply. The typical current consumption is rated at $0{,}5\,\mathrm{A}$.

For a proper functionality of the whole circuitry, it is therefore very important to guarantee these standards. A common problem of DC-DC converters is their ripple voltage. Voltage ripples are caused by the switching of the output stage and can be the source of various problems in analog and digital circuits.

For the output voltage tests, the DC-DC converter is powered with $34\,\mathrm{V}$ DC trough the MDB/ICP connector. The voltage is probed on tespoint `TP11`. A variable load resistor is connected to the jumper `J4` through a multimeter to measure the load current.

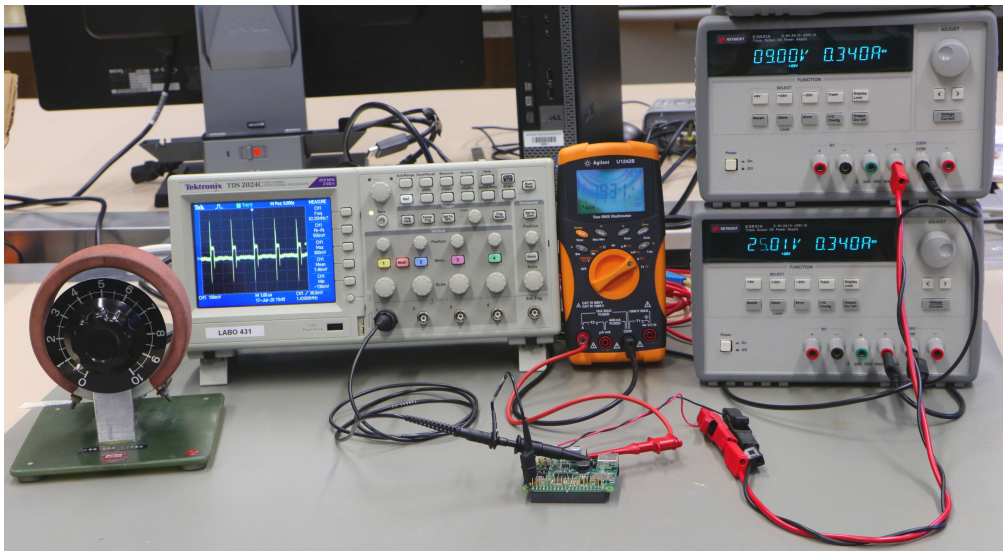


Figure 5.16.: The DC-DC converter test setup

The test setup is shown in the illustration above. Two power supplies (right) are required to generate the desired voltage of $34\,\mathrm{V}$. Their output is connected to the MDB/ICP connector's pins. The load resistor (left) is connected to the converter's output at the jumper `J4`. Between the load resistor and the converter board, a current measurement is done by the multimeter. Finally, the oscilloscope probes the output voltage at the designated testpoint `TP11`.

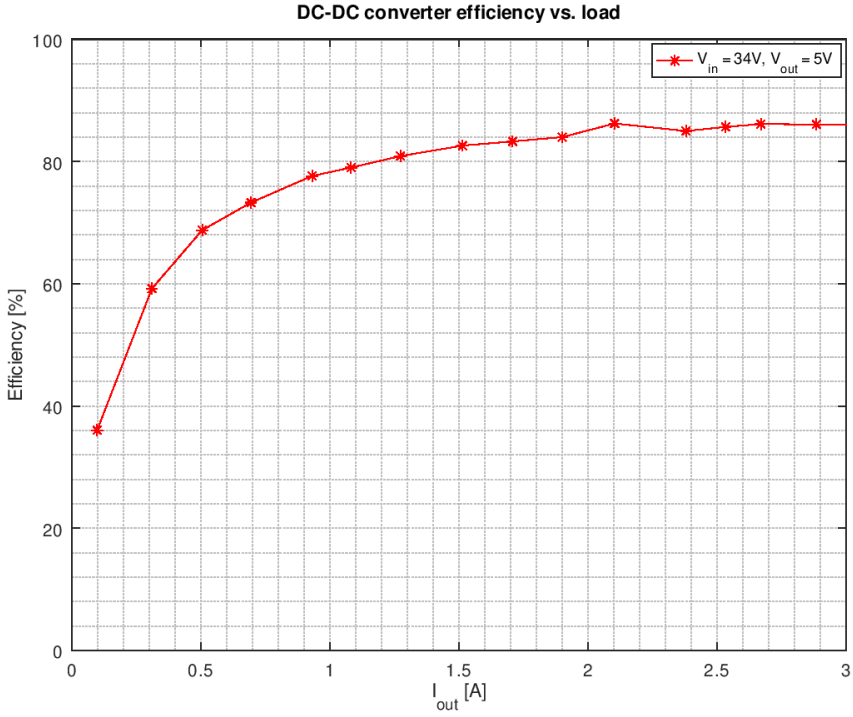


Figure 5.17.: The DC-DC converter's measured efficiency

Figure 5.17 illustrates the efficiency measurement's outcome. The efficiency is calculated with the following equation.

$$\eta = \frac{P_{out}}{P_{in}} = \frac{U_{out} \cdot I_{out}}{U_{in} \cdot I_{in}} \tag{5.3}$$

The step down converter does not to appear to operate efficient at low output loads. A special sleep mode is activated by the dc-dc controller for output loads below $300\,\mathrm{mA}$, which could be the reason for this observation. Fortunately, the efficiency at higher loads behaves as expected. Power conversion losses are mainly caused by the converter's switching mosfet resistance $R_{DS,on}$ $(90\,\mathrm{m\Omega})$, the series resistance of the inductor $(30\,\mathrm{m\Omega})$ and the shottky diode's forward voltage $(420\,\mathrm{mV})$. These losses are converted into heat and will be dissipated trough the PCB's copper planes and the components surfaces. However, the temperature of the PCB remains in an acceptable range. The output voltage stays at $5\,\mathrm{V}$ during the entire load range.

Another very interesting parameter to quantify the buck converter's quality is the output ripple voltage. It is measured with the oscilloscope and uses the same test setup as shown in figure 5.16. The probe's alligator clip was replaced by a pigtail to enable proper ground probing [22].

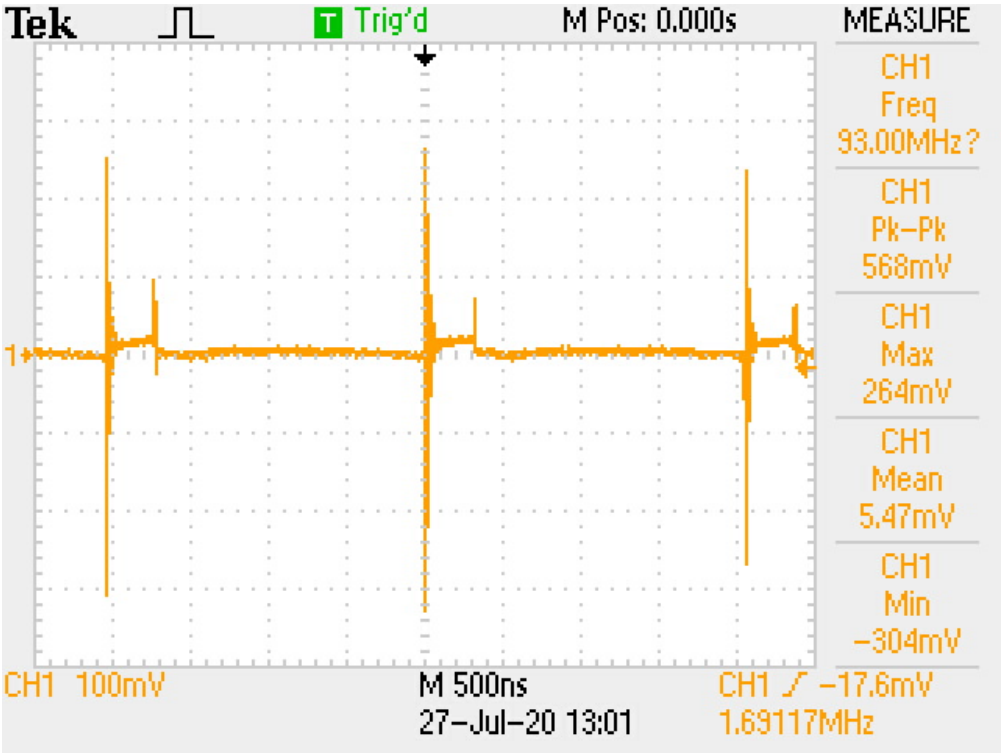


Figure 5.18.: The DC-DC converter's output ripple at $5\,\mathrm{V}/1\,\mathrm{A}$

The test result is not as expected. In figure 5.18, one can see the $568\,\mathrm{mV}$ output ripple voltage, which has a very high frequency noise (more than $20\,\mathrm{MHz}$). This noise produces radiations that could cause distortions in other signals. Therefore it must be damped under any circumstances.

A Texas Instruments DC-DC noise filtering application report claims that the high frequency (HF) ripple can be caused by a big ground loop between the converter's input capacitor and the inductor [22].

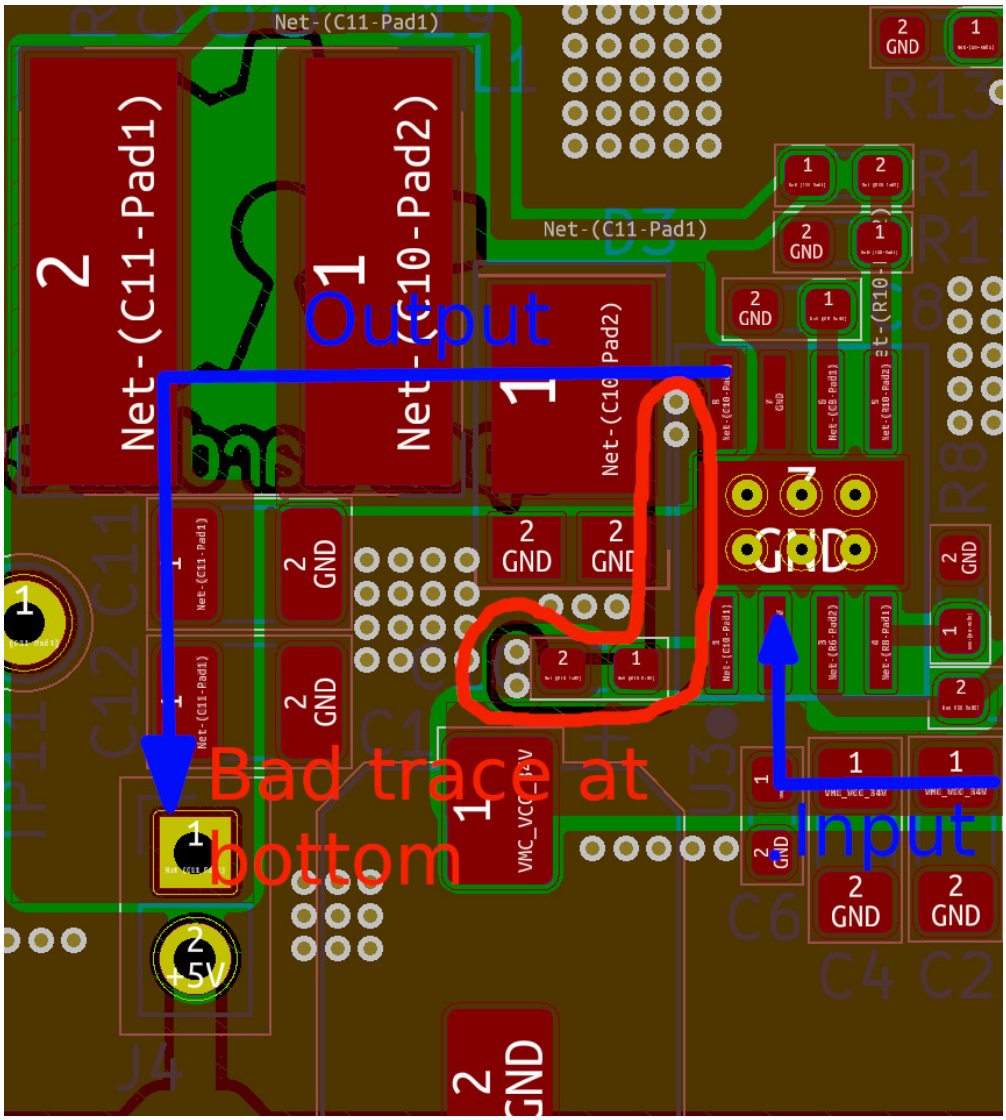


Figure 5.19.: The DC-DC converter PCB layout

Indeed, it was not specially considered to keep this ground loop as small as possible. It could be, that the improper routing of a trace (the converter's bootstrap capacitor) at the bottom layer is the reason for the distortions. In figure 5.19 it is visible, that the bottom ground plane between the input and output stage is cut-off by the bootstrap capacitor trace. It was tried to overcome this cut-off by stitching the top and bottom ground plane but apparently this does not help.

Another possibility to remove HF ripples is the attachment of a LC-lowpass filter at the converter's output [22]. Fortunately, the existing design offers ideal planes, where such a filter can be attached manually.

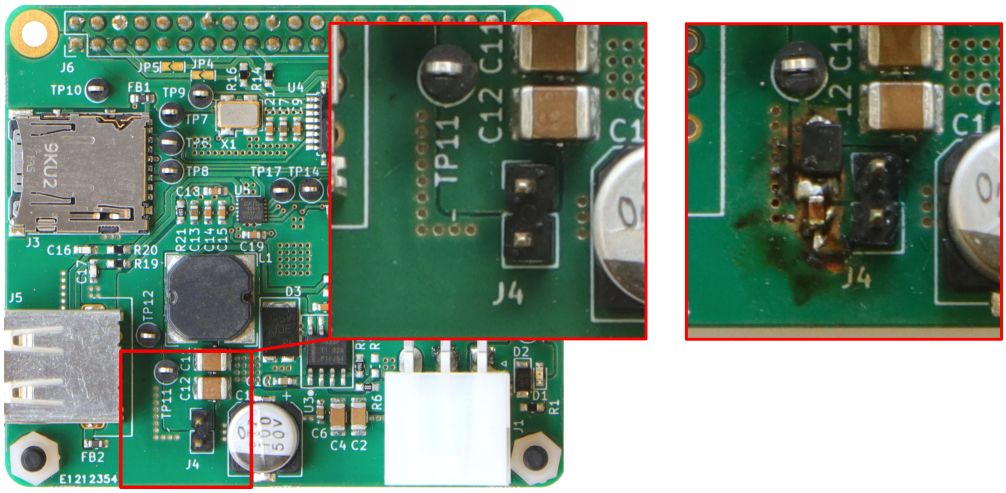


Figure 5.20.: The additional output filter on the PCB

Figure 5.20 shows how the LC lowpass filter was attached. The $5\,\mathrm{V}$ plane is horizontally cut between the output capacitor `C12` and the jumper `J4` and reconnected trough the inductor. A capacitor was added in parallel to the jumper `J4`. The corner frequency of this second order filter amounts:

$$f_c = \frac{1}{2 \cdot \pi \cdot \sqrt{L \cdot C}} = \frac{1}{2 \cdot \pi \sqrt{1\,\mu\mathrm{H} \cdot 680\,\mathrm{nF}}} = 193\,\mathrm{kHz} \tag{5.4}$$

Since the frequency of the HF noise is greater than $20\,\mathrm{MHz}$ it should be damped by more than $-80\,\mathrm{dB}$. But surprisingly, the ripple is not damped at all. A possible reason for this behavior could be the usage of bad components and the fact, that this filter attachment is far from perfect. It will be a good idea to simulate the converter with the attached components to get a more reliable result.

Another idea to damp this high frequency ripple could be the implementation of a ferrite bead. Ferrite beads are basically inductors that are designed to filter high frequency ripples generally in the range of $10\,\mathrm{MHz}$-$100\,\mathrm{MHz}$. The voltage ripple of the converter is rated at $80\,\mathrm{mV}$ when the oscilloscope's $20\,\mathrm{MHz}$ bandwidth limitation is enabled. Thus, the ferrite bead could be the ideal solution to this problem [23]. The use of such components was not tested due to the lack of time.

In conclusion it can be said, that the HF ripple problem is not solved at the moment. The PCB Layout must be adapted to remove the improper trace, which was shown in figure 5.19. Additionally, a LC-lowpass filter and a ferrite bead should be added to the converter's output stage. A simulation could help to select the right components.

5.3.2. Communication Interface Analysis

The MDB/ICP gateway implements multiple communication interfaces, which have to be tested to verify their functionality and to get a basic knowledge about the speed limitations.

In the first place, the functionality of the serial wire debug (SWD) interface was tested, since it was used as main programming and debugging interface for further tests. The SWD is an ARM specific derivative of the famous JTAG (Jointed Test Action Group). JTAG is a popular interface for microcontroller and FPGA programming, debugging and production testing with boundary scanning [24]. It uses a 5-Pin communication. SWD, in comparison, has similar capabilities as JTAG but only uses two wires [25]. The main drawbacks of SWD are the limitation to ARM controllers and the inability for boundary scans.

Nevermind, the SWD interface was tested by flashing a simple program to the microcontroller, which toggles one of the available user LED's. It worked as expected.

As a next step, every interface was tested successively by writing the corresponding source code and checking the signal on the recipient side combined with the logic analyzer. This sequential proceeding trough every bus was a good starting point but is not a confident method to test multiple boards. Therefore, a simple testbench application was written, which is able to test the bidirectional communication of every available on the board. The SD-Card interface was excluded from this application. The reason for this will be discussed later.

The application uses a STM32 Nucleo board to simulate the vending machine communication and the Raspberry Pi. It basically implements a very simple communication chain, which is illustrated below.

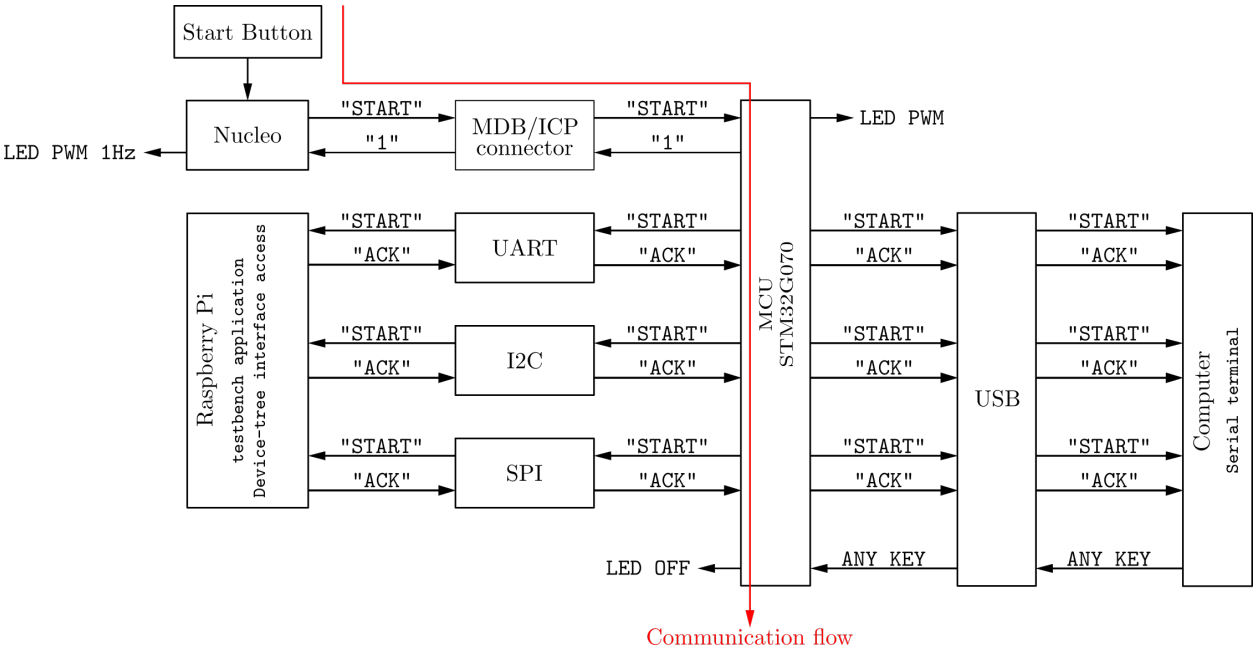


Figure 5.21.: The testbench's message chain diagram

As start condition, the user needs to press a button on the Nucleo board. The Nucleo will then send a start message to the MCU on the gateway by using the UART communication interface. As a reaction, the MCU will enable its LED's and answer with a number, which will be used on the Nucleo to toggle a LED with the corresponding frequency. Furthermore, the microcontroller will start a communication loop trough every interface, which is connected to the Raspberry Pi. Each transmitted and received message will be sent to the developer's computer trough the USB interface. To finish the test, the developer will be prompted to send any message to the gateway. When the microcontroller has received a message, it will turn of its LEDs as a sign of acknowledgment.

Now the application has terminated and every interface is guaranteed to work bidirectional. Further information about the configurations are available in the MDB/ICP gateway's gitlab repository under the `testbench` branch.

The speed limitations of each communication interface was found by continuously increasing it's baudrate. Since the MDB/ICP interface is guaranteed to stay at $9600\,\mathrm{bit\,s^{-1}}$ it was excluded from this tests. The found limitations are listed below:

- Raspberry Pi UART: $3\,\mathrm{Mbit\,s^{-1}}$ (Limitation by Raspberry Pi)
- Raspberry Pi I2C: $400\,\mathrm{kHz}$ SCL (Limitation by Raspberry Pi)
- Raspberry Pi SPI: $10\,\mathrm{Mbit\,s^{-1}}$
  The Raspberry Pi seems to have problems by generating some clock speeds. There is maybe a higher speed

available.

- FTDI USB: $3\,\mathrm{Mbit\,s^{-1}}$ (Limitation by FTDI-Chip)

Fortunately, the limitation is always set by the capability of a particular component and not by the signal distortion of the PCB traces.

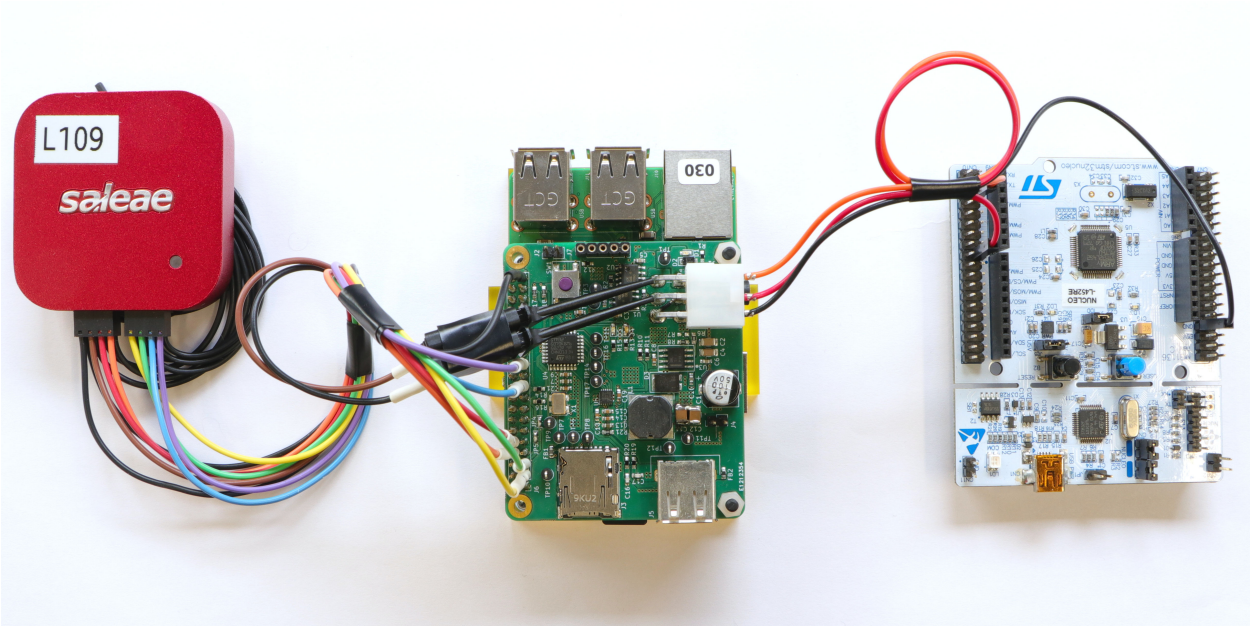


Figure 5.22.: Communication interface testbench composition

The testbench's setup composition is as showed in the figure above. Unfortunately, the logic analyzer does not offer enough channels to observe all signals at once. A typical logic analyzer output is showed in the image below.

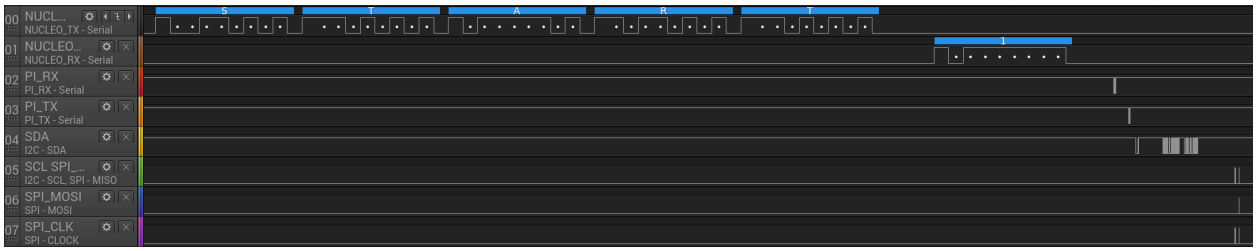


Figure 5.23.: Testbench signals observed with logic analyzer

It is impressive to see the huge differences between each peripherals signal speed. The testbench application is not optimized for a fast signal response time since the source code is not optimized.

```
[INFO]:    Open UART device /dev/ttyAMA0
[INFO]:    Acces to UART device /dev/ttyAMA0 successful
[INFO]:    Configuration of UART device /dev/ttyAMA0 successful

[INFO]:    Open I2C device /dev/i2c-1
[INFO]:    Acces to I2C device /dev/i2c-1 successful
[INFO]:    Setting up slave adres 0x06 on /dev/i2c-1

[INFO]:    Open SPI device /dev/spidev0.0
[INFO]:    SPI Initialized successful

[INFO]:    Test UART connection...
[INFO]:    Waiting for data reception..
[INFO]:    Received: "START" in 5b
[INFO]:    Transmitted: "ACK" in 3b
[INFO]:    UART communication successful

[INFO]:    Test I2C connection...
[INFO]:    Waiting for data reception..
[INFO]:    Received: "START" in 5b
[INFO]:    Transmitted: "ACK" in 3b
[INFO]:    I2C communication successful

[INFO]:    Test SPI connection...
[INFO]:    Waiting for data reception..
[INFO]:    Received: "START" in 5b
[INFO]:    Transmitted: "ACK" in 3b
[INFO]:    SPI communication successful

[INFO]:    Closing UART controller /dev/ttyAMA0 filestream
[INFO]:    Closing I2C controller /dev/i2c-1 filestream
[INFO]:    Closing SPI controller /dev/spidev0.0 filestream
```

Figure 5.24.: Raspberry Pi testbench output

A program was written in order to execute the testbench application on the Raspberry Pi. It accesses each peripheral trough the `/dev` interface, configures it and handles the data I/O [26] [27] [28]. The actual program status is printed to the terminal as illustrated in figure 5.24.

```
Received: START from NUCLEO (MDB)
Set LED PWM frequency on NUCLEO (MDB) to 1 Hz
Enable Amber LED on converter board
Send: START to Pi UART
Received: ACKRT from Pi UART
Send: START to PI I2C
Received: ACKRT from Pi I2C
Send: START to PI SPI
Received: ACKRT from Pi SPI
Communication Test completed. Eyery interface is working
Send any key to disable the LED on the expansion board
```

Figure 5.25.: USB serial terminal output

The output on the USB port can be displayed with the usage of a serial-interface terminal. A successful testbench cycle will look like the output in figure 5.25.

5.4. Conclusion

Overall, the first version of the MDB/ICP gateway was successful. The most important interfaces could be tested have shown the capabilities of this board. Weaknesses were recognized during the tests, which enables further improvement in a next implementation.

For further design steps it is very important to revise the dc-dc converter's PCB layout and to add the recommended LC filter and ferrite bead to the output. The SC-Card interface has not been tested, since it's interfacing trough SPI is more complicated and requires more time, which was not available during this thesis. But since all other bus connections are working without any problems, it can be assumed that this interface will work too with the corresponding software.

6. Embedded Platform

The MDB/ICP gateway is now manufactured and tested. As a follow-up, the goal is to process one step further to an embedded platform. The platform promises a more integrated and thus more compact hardware, which is tailored to the Taler application's requirements.

Unfortunately, it's development was not finished during the thesis, because there was not enough time. The remaining part is the routing of the PCB, which would take approximately a week of full-time work to finish.

This chapter will discuss the developed hardware's schematic. The following specification sheet shall give a quick overview to the platform.

6.1. Specification Sheet

EMBEDDED MDB/ICP TALER PLATFORM

FEATURES

- Standard MDB/ICP Interface Connector
- 10/100/1000 Ethernet Connector
- 2x USB Type-A Connector
- USB Type-C Connector
- MIPI-DSI Connector
- Raspberry Pi GPIO Header
- 2x Grove Pi Connector
- Three options for power supply
- Interchangeable computing module
- Open Hard- and Software
- JTAG Interface

APPLICATIONS

- Vending machine peripherals
- GNU/Taler cashless device on vending machines

GENERAL DESCRIPTION

The embedded MDB/ICP platform is based on a computing module from the manufacturer Seeed Studios. It includes a ST Electronics STM32MP157C microprocessor with two cortex-A7 cores and a cortex-M4 core, $500\,\mathrm{MB}$ of RAM and $4\,\mathrm{GB}$ eMMC storage. It is mounted on the carrier board trough three connector headers and thus is interchangeable. On the carrier board, an Ethernet connector, 2x USB Type-A and 1x USB Type-C connector, and a MIPI DSI connector are offering standard higher level computer interfaces. Additionally, the board provides the necessary MDB/ICP connector to enable communication with a vending machine. Lower level interfaces, are provided by a Raspberry Pi compatible GPIO header and 2x Grove Pi connectors.

The platform can be supplied either trough the vending machine bus supply, a $12\,\mathrm{V}$-$24\,\mathrm{V}$ DC-Jack or trough the USB Type-C port. An additional connector is applied to backup the module with a battery.

FUNCTIONAL BLOCK DIAGRAM

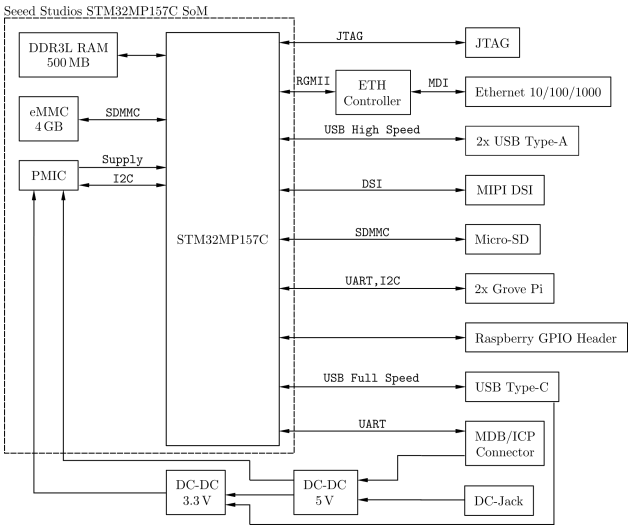


Figure 6.1.: Embedded MDB/ICP platform

6.2. Hardware development

The Seeed Studio's Odyssey System-on-Module is the core of the platform. It uses the microprocessor, which was selected during the preliminary study (see chapter 2) and is specified as open-hardware. The module is connected to the main PCB, the carrier board, trough three connectors with a total pin count of $210\,\text{pins}$ [10]. It was chosen, since it provides all essential aspects, which are needed for the system.

Advanced microprocessors are highly integrated components, which are mostly packaged in Ball-Grid-Array (BGA) packages. This packaging type has the advantage that it is very compact in size. A main drawback of this feature is that the connector pins, which have to be soldered onto the PCB, have a small pitch and are located underneath the chip. This increases the complexity during the PCB soldering because of the higher risk of generating electrical shortages. Another drawback comes in place during the routing of the traces of a BGA packaged chip, because all connections are very concentrated and have to be distributed to several places on the PCB. The first step of distribution is called fan-out and is engaged to decentralyze the various traces around the chip to enable a more confident routing. Such processes require a lot of time and experience. Fortunately, the most integrated components, the microprocessor, the RAM, the eMMC and the Power Management IC (PMIC) are placed on Seeed Studio's module. This simplifies the embedded platform design a lot.

As a first part of the schematic development, one has to draw a custom symbol for the STM32MP157C SoM. The pins of the microprocessor are directly routed to the connector headers. A pinout of these connectors can be found inside Seeed Studio's wiki page [29].

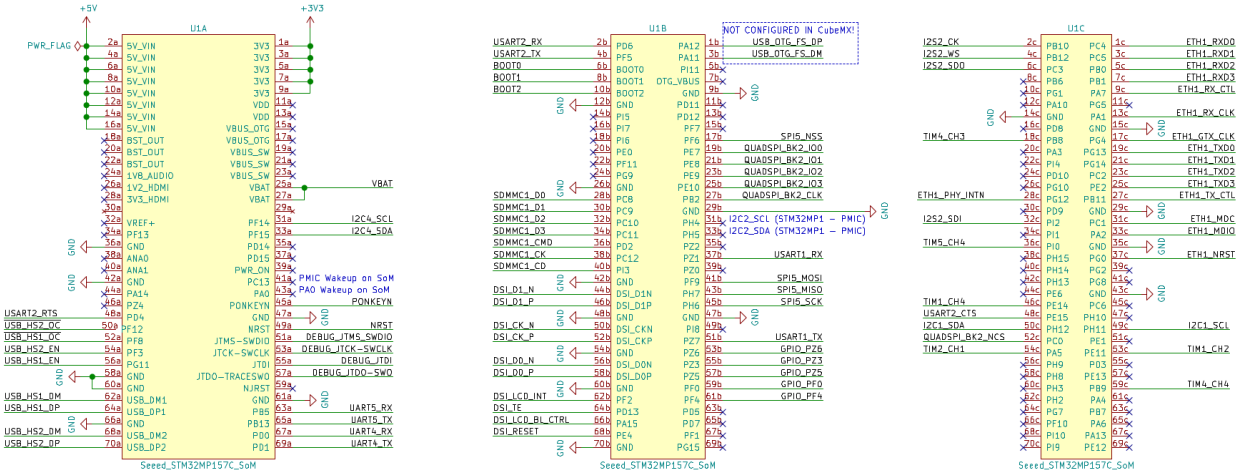


Figure 6.2.: Seeed Studio's SoM schematic

In the SoM's schematic symbol above, one can regard the three connector headers as independent objects. The first step after the generation of the symbol is to check which pins are already occupied on the SoM internally to avoid overriding of them. During the whole development process, ST Electronic's initialization tool CubeMX [11] was used to perform the pin assignments.

The schematic is divided into five different parts:

- Power Supply
- USB interfaces
- MDB/ICP interface
- Ethernet interface

- SD-Card interface and boot mode selection

6.2.1. Power Supply

The embedded platform has three power supply options:

- $22\,\mathrm{V}$-$42{,}5\,\mathrm{V}$ supply trough vending machine interface over MDB/ICP connector
- $12\,\mathrm{V}$ or $24\,\mathrm{V}$ supply from $5{,}5\,\mathrm{mm}/2{,}5\,\mathrm{mm}$ DC-Jack input
- $5\,\mathrm{V}$ supply trough USB Type-C

The PCB is mainly supplied trough a $3{,}3\,\mathrm{V}$ and a $5\,\mathrm{V}$ net. The $5\,\mathrm{V}$ supply net is necessary for the SoM's power management IC and for the USB Type-A hosts, which must have the ability to power their connected devices. The Power Management IC on the Seeed module takes responsibility to supply the MPU with it's required core voltages.

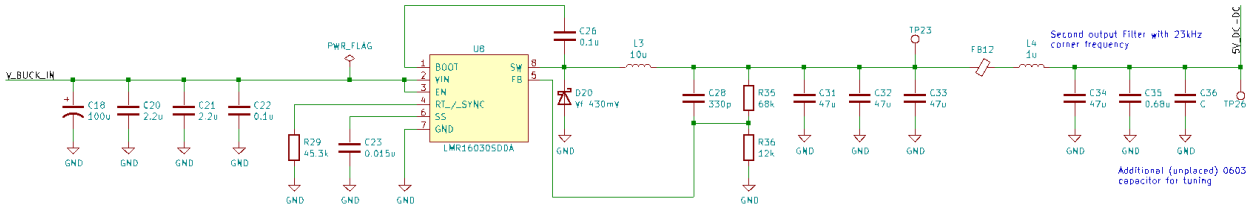


Figure 6.3.: 5V DC-DC converter

Both, the MDB/ICP and the DC-Jack power input are converted to $5\,\mathrm{V}$ by a DC-DC converter, which is similar to the circuit used in the MDB/ICP gateway design. The major change to the older gateway is the replacement of the converter (`U8` in figure 6.3) with a pin compatible version that offers higher voltage rates of up to $60\,\mathrm{V}$ instead of $40\,\mathrm{V}$, as it was recommended in the gateway's review. Thus it is also necessary to upgrade the Schottky diode `D20` to a higher voltage rated model. Additionally, a third $47\,\mu\mathrm{F}$ output capacitor (`C31`) is attached based on the higher voltage rating. The ferrite bead `FB12` is implemented to remove high frequency ripples, as it was discussed in the MDB/ICP gateway's measurement part 5.3.1. The secondary output filter with it's cutoff frequency at $23\,\mathrm{kHz}$ offers the ability to be tuned with an additional capacitor, which will not be placed by default. It is currently unclear how the output ripple voltage will behave under the new layout. A general caution must be kept during the routing of this converter module to ensure short a ground connection between the input capacitors and the converter output ground.

An additional power distribution logic ensures that two sources can be applied to the board without any damage. In case of the $5\,\mathrm{V}$ buck converter, the logic automatically prefers the DC-Jack's input over the MDB/ICP bus supply and protects both supplies from reverse currents.

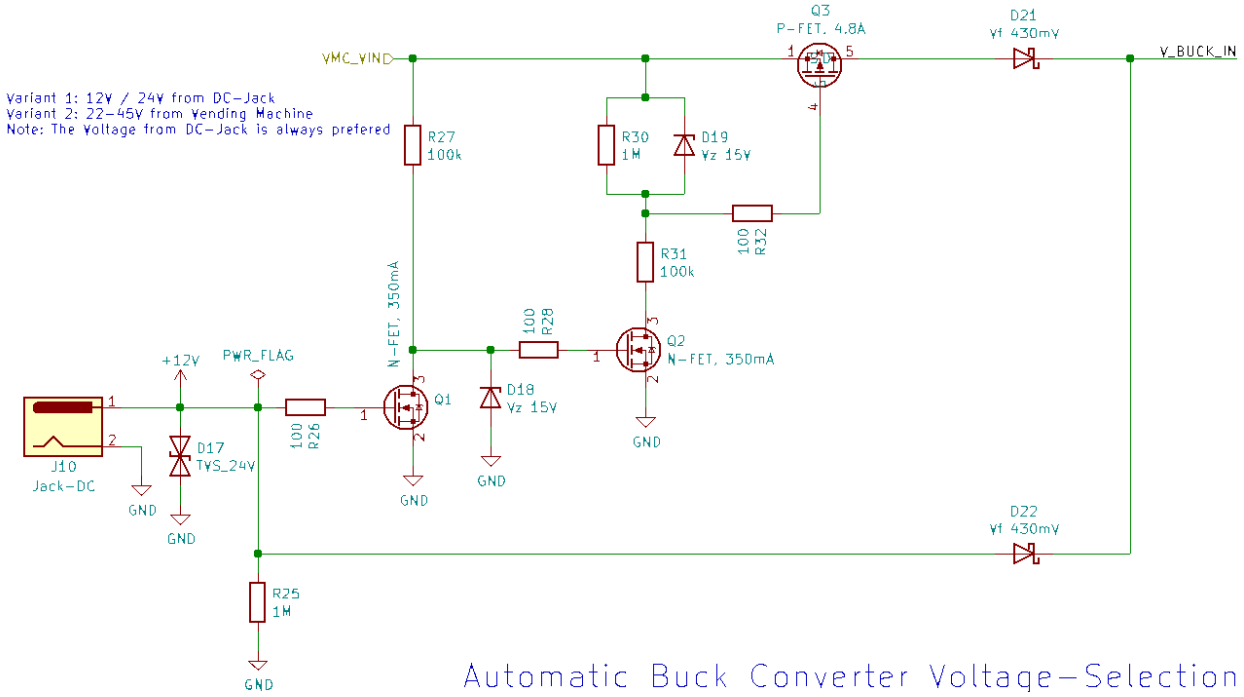


Figure 6.4.: Power Mux to automatically select voltage source

The logic in figure 6.4, mainly consists of a high side switch `Q2,Q3`, a signal inverter `Q1` and two diodes `D21,D22` to prevent reverse current flow. It shall ensure that the DC-Jack input is always preferred over the vending machine's supply. An additional DC-Jack could provide a more reliable power supply and thus should be preferred over the vending machine's supply. The four different cases during the operation are now shortly explained:

Case 1: MDB/ICP supply inactive, DC-Jack supply inactive

This case is the most trivial. Since no voltage is applied to any of both inputs, there will be no output.

Case 2: MDB/ICP supply active, DC-Jack supply inactive

- Q1: $V_{GS} = 0\,\mathrm{V}$
- Q2: $V_{GS} = 15\,\mathrm{V}$
- Q3: $V_{GS} = -15\,\mathrm{V}$

The pull-down resistor `R25` will ensure a stable ground connection on the gate of `Q1`. Since the N-Channel mosfet has a gate-source voltage of $0\,\mathrm{V}$, it won't conduct. On `Q2`, the zener diode limits the gate voltage to $15\,\mathrm{V}$ along with the resistor `R27`, which limits the current. The gate-source voltage on the N-Channel fet is therefore $15\,\mathrm{V}$, which means that it will conduct. This conduction pulls the P-Channel gate circuit down. The zener diode `D19` becomes active and limits the gate voltage on `Q3` to $V_g = \mathtt{VMC\_VIN} - 15\,\mathrm{V}$. The parallel $1\,\mathrm{M\Omega}$ resistor `R30` can be neglected, since it's resistance is many times higher than the equivalent resistance of the zener diode. Zener current is limited by `R31`. The P-Channel mosfet becomes conductive since its gate-source voltage is $-15\,\mathrm{V}$. The Schottky diode `D22` protects the circuit from reverse current on the DC-Jack line.

Case 3: MDB/ICP supply active, DC-Jack supply active

- Q1: $V_{GS} = 12\,\mathrm{V}$
- Q2: $V_{GS} = 0\,\mathrm{V}$

- Q3: $V_{GS} = 0\,\mathrm{V}$

Since a voltage is now applied on the DC-Jack input, `Q1` has a positive gate-source voltage and thus conducts. Due to this conduction, the gate voltage of `Q2` is pulled down to $0\,\mathrm{V}$. On `Q3`, the gate voltage will be the same voltage as `VMC_VIN`, since it can pass trough the resistor `R30`. The MDB/ICP bus voltage will be blocked by `Q3` and the DC-Jack voltage is able to power the circutit.

Case 4: MDB/ICP supply inactive, DC-Jack supply active

- Q1: $V_{GS} = 12\,\mathrm{V}$
- Q2: $V_{GS} = 0\,\mathrm{V}$
- Q3: $V_{GS} = 0\,\mathrm{V}$

The same pattern as in case 3 applies. Since `Q1` is conductive, it will pull `VMC_VIN` to ground level. The Schottky diode `D21` protects the MDB/ICP line against reverse current.

The two N-Channel mosfets are selected to whitstand a V_{DS} voltage of more than $45\,\mathrm{V}$ and a V_{GS} voltage of more than $24\,\mathrm{V}$. Since these fet's don't have to switch a lot of power, one does not have to be very careful during the component selection, except of the breakdown voltage ratings above.

The selection of the P-Chanel fet is more cirtical, since it will switch the power line and therefore needs a low $R_{DS,on}$ resistance to reduce the voltage drop across the fet during it's on-state. A voltage drop across the mosfet implies a power loss, which heats up the component. This voltage drop can be determined with the graphic provided in most manufacturers datasheets. Another requirement on the P-Channel mosfet is the V_{GS} breakdown voltage of less than $-15\,\mathrm{V}$ and the V_{DS} breakdown voltage of less than $45\,\mathrm{V}$.

The Schottky diodes are the same type as used for the DC-DC converter.

To get more confident with the circuit, a simulation was built up with the famous tool LT Spice [30]. LT Spice is a free, proprietary simulation toolbox, which is very popular to simulate electrical circuits realistically. Many component manufacturers offer accurate spice models to their components.

The simulation output below shows the expected circuit's behavior.

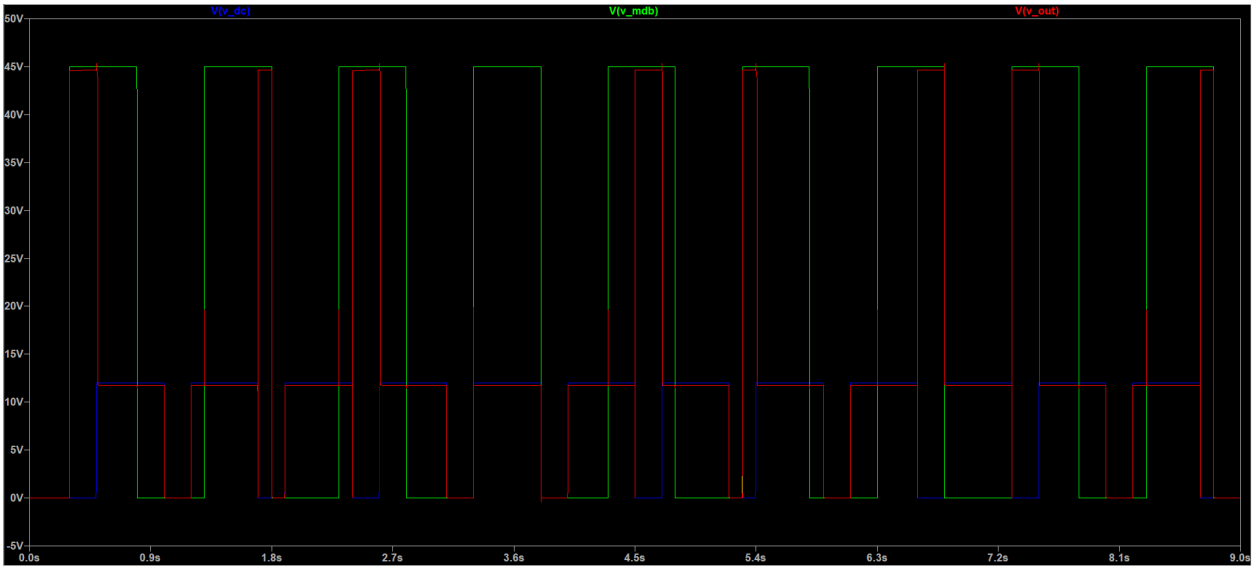


Figure 6.5.: LT-Spice simulation of power mux for $5\,\mathrm{V}$ converter input

Figure 6.5 shows that the circuit's output should behave as expected. The square wave signals were applied to provide the visualization of all cases inside one single graphic. In the real world, a hot-plug will not occur very frequent. The used mosfet's are exact models, provided by the component's manufacturers.

The other available voltage level on the board will be the $3{,}3\,\mathrm{V}$ net. It uses a buck converter to convert the $5\,\mathrm{V}$ input to $3{,}3\,\mathrm{V}$ output, which will then supply the rest of the board.

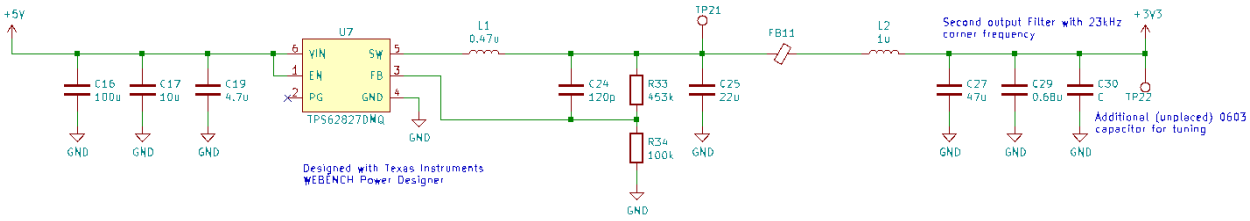


Figure 6.6.: $3{,}3\,\mathrm{V}$ DC-DC converter

This circuit in figure 6.6 was entirely designed by the use of Texas Instrument's WEBENCH Power Designer [31], which allows a quick converter design where all recommended components are directly listed. The selected DC-DC controller uses a rather different topology, where the input is switched with a half bridge. Thus, the Schottky diode is not needed anymore, since it is replaced by the half bride's lower switching mosfet. Additionally to the provided design, one decided to add a ferrit bead and a second order lowpass filter to the converter as it is done on the other converter circuit. These additional components do not have to be placed if they are not needed.

The input voltage of this DC-DC converter will source from either the $5\,\mathrm{V}$ DC-DC converter output or as additional feature from the USB Type-C port. A power mux is designed to automatically select the right source.

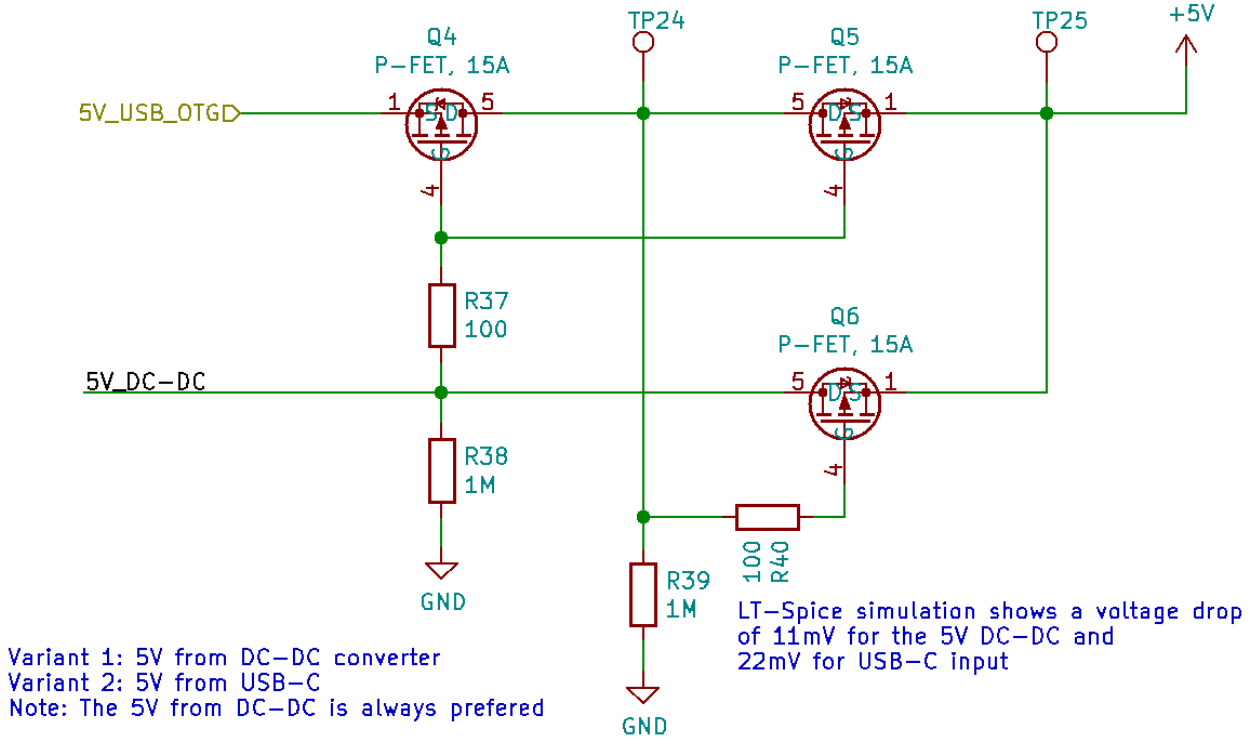


Figure 6.7.: $3{,}3\,\mathrm{V}$ DC-DC converter switching logic

The circuit above will always prefer the DC-DC converter as voltage source over the USB Type-C. The reason for this behavior is, that the USB Type-C may be used as debugging interface or something else and will therefore be

connected to a computer. Unfortunately, a computer will not be able to power the whole platform, since they are limited to $500\,\mathrm{mA}$ output current. This logic will prevent this unwanted behavior by automatically select another source. The reverse conclusion of this statement is, that the platform has to be powered from a different source when the USB Type-C port is used for connection.

The designed power switch uses three P-Channel mosfets. `Q5` and `Q6` are mainly used as diodes with very small voltage drop. Only `Q4` is used to actually switch the source. The four different input voltage combination cases are now shortly explained

Case 1: USB Type-C supply inactive, DC-DC supply inactive

- Q4: $V_{GS} = 0\,\mathrm{V}$
- Q5: $V_{GS} = 0\,\mathrm{V}$
- Q6: $V_{GS} = 0\,\mathrm{V}$

This case is the most trivial. Since no voltage is applied to any of both inputs, there will be no output.

Case 2: USB Type-C supply active, DC-DC supply inactive

- Q4: $V_{GS} = -5\,\mathrm{V}$
- Q5: $V_{GS} = -5\,\mathrm{V}$
- Q6: $V_{GS} = 0\,\mathrm{V}$

The DC-DC converter line is pulled to ground trough `R38`. On `Q4` a gate-source voltage of $-5\,\mathrm{V}$ applies, which makes it conductive. The freewheeling diode on `Q5` will pass these $5\,\mathrm{V}$ and thus, the gate-source voltage will settle at $-5\,\mathrm{V}$ too, which makes it conductive and thus ensures a low voltage drop `Q6` will block the DC-DC circuit from reverse current, since it's gate-source voltage will settle at $0\,\mathrm{V}$.

Case 3: USB Type-C supply active, DC-DC supply active

- Q4: $V_{GS} = 0\,\mathrm{V}$
- Q5: $V_{GS} = 0\,\mathrm{V}$
- Q6: $V_{GS} = -5\,\mathrm{V}$

The USB Type-C input gets blocked at `Q4` since the voltage level at the gate is now the same as at the source. `Q5` is also non conductive and blocks reverse currents. On the DC-DC line, `Q6` will initially pass the power trough the freewheeling diode. The gate-source voltage settles at $-5\,\mathrm{V}$, which makes the mosfet conductive and ensures a low voltage drop.

Case 4: USB Type-C supply inactive, DC-DC supply active

- Q4: $V_{GS} = 5\,\mathrm{V}$
- Q5: $V_{GS} = 0\,\mathrm{V}$
- Q6: $V_{GS} = -5\,\mathrm{V}$

The same pattern as in case 4 applies.

For this power mux, a ultra low voltage drop on `Q5` and `Q6` is required to meet the USB supply and other circuit's voltage levels. Therefore, one has to be very carefully during the component selection to get a minimal V_{DS} voltage drop during the on-state.

Again, the logic has to be verified by a simulation with the exact mosfet models in LT-Spice.

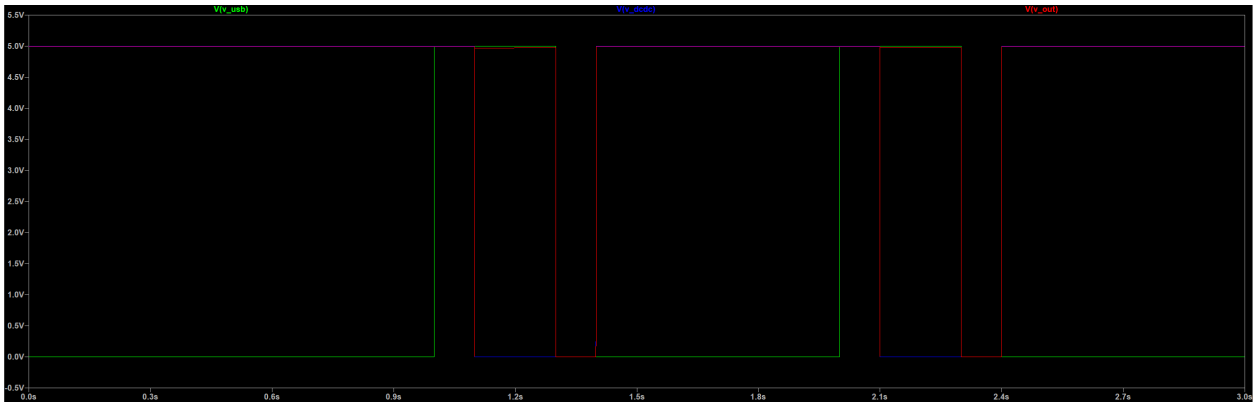


Figure 6.8.: LT-Spice simulation of power mux for $3{,}3\,\mathrm{V}$ converter input

The simulation output in figure 6.8 shows, that the calculations and component selection was correct. One can expect a voltage drop of only $11\,\mathrm{mV}$ when powering from the DC-DC converter and $22\,\mathrm{mV}$ (because of two mosfet stages) when powering from USB Type-C.

6.2.2. USB interfaces

From the USB Type-C connector as power supply, it is now time to discuss the implementation of the USB Type-C in terms of data exchange, or in more general all available USB ports on the platform.

USB Type-C

The USB Type-C is the newest in a series of USB connector types. It offers a symmetrical connector shape with a total pin count of 24, from where only $12\,\mathrm{a}$re maximal used at once, since the connector can be mounted bidirectional and is only connected to one side at a time. USB Type-C supports all specified USB protocols up to $3{,}1$ Additionally, there are more features such as USB-PD (Power Delivery for up to $100\,\mathrm{W}$), an alternate function, which enables Display Port functionality and the so called USB-OTG (on the go) functionality, which can operate as USB-host and -slave [32].

However, the USB Type-C connector on this platform only supports basic USB2.0 full-speed ($12\,\mathrm{MB\,s^{-1}}$) communication as an USB-device. The main difference between an USB-host and -device is the power supply. A USB-host has to provide the power on the bus, while the device consumes it. The USB full-speed standard is chosen, since it was the only available option on the microprocessor.

The USB Type-C connector is able to deliver power up to $3\,\mathrm{A}$ ($15\,\mathrm{W}$) in standard mode. The USB-PD feature is not needed, since the standard mode specification will suffice the requirements. The desired power consumption on USB Type-C cables is negotiated trough the CC-Pin (Configuration Channel), where the power consumption is determined trough a resistor ladder. On the host side, the CC Pin is pulled up to $5\,\mathrm{V}$ trough a resistor. The device pulls the CC Pin to ground trough another resistor. The host senses the voltage, which settles between both resistors and thus is able to determine the device's power consume capabilities.

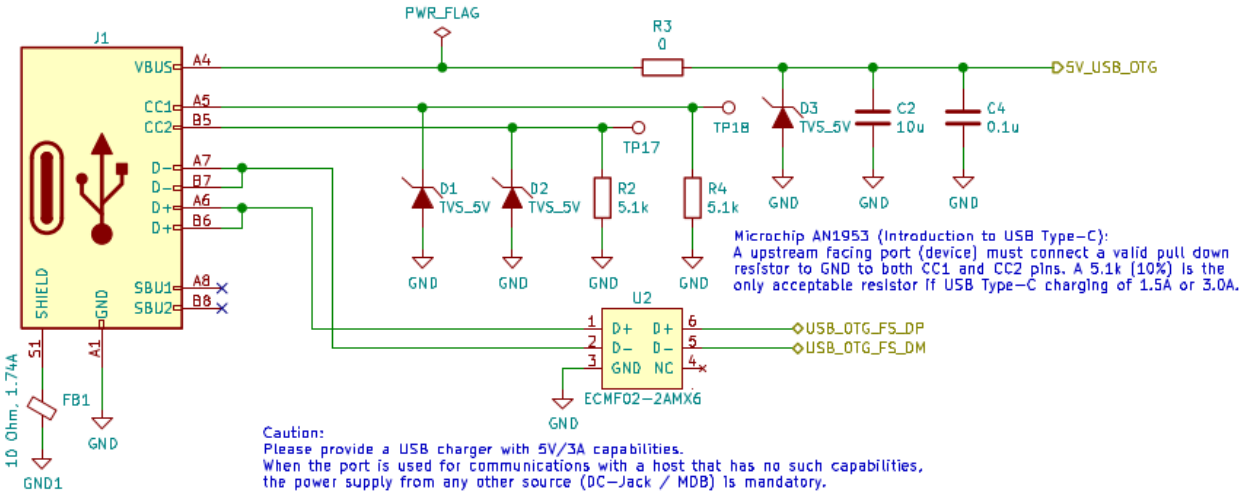


Figure 6.9.: USB Type-C connector

The standard specifies a $5{,}1\,\mathrm{k\Omega}\ \pm 10\,\%$ resistor to ground (`R2`, `R4` in figure 6.9) on the device to allow up to $3\,\mathrm{A}$ supply current. It is intended, that the device constantly senses the voltage on the CC pin, since the host can communicate its current delivery capabilities by changing the voltage level. The device should then adapt its consumption according to this information from the host. Such capability could be implemented with a simple USB Type-C power control IC, which is offered by multiple manufacturers. Since the platform will not be able to change it's power consumption, the use of such chips will not bring any fortune. It was rather decided to request a $3\,\mathrm{A}$ capable charging adapter for power supply applications. Here it will be the responsibility of the user to connect a compatible device.

The USB standard implements a differential data line, which is directly connected to the microprocessor. Unfortunately, this peripheral could not be initialized on CubeMX, since they have not implemented this feature yet, thought it can be initialized on the linux device-tree [33].

Several ESD-protection diodes are added to protect the platform against high voltage bursts. The data lines `D+` and `D-` are protected and common-mode filtered with `U2`, an extra USB rated chip.

USB Type-A

The USB Type-A connectors are both specified as USB-2.0 high speed ($480\,\mathrm{MB\,s^{-1}}$) host ports.

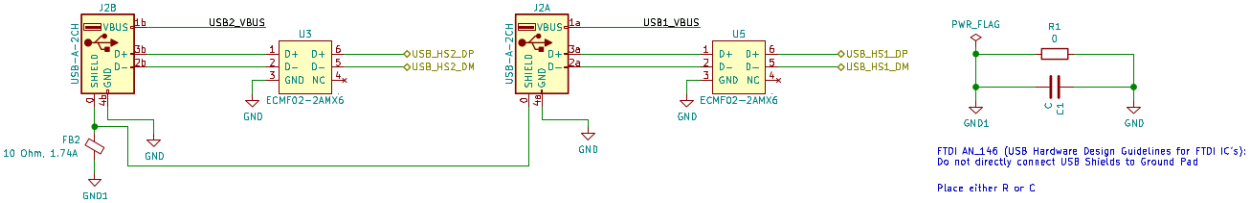


Figure 6.10.: USB Type-A connector

The physical connector is a dual port housing `J2x` (figure 6.10). Therefore, the USB-Shield pins are connected together. The USB data lines are ESD protected and common mode filtered with the same chip as the USB Type-C port uses. They are directly connected to the microprocessor, which handles the whole protocol. For EMI purposes, several design guides recommend to connect the shield of the USB connector to a separate ground plane, which is connected to the main ground on only one physical point [34].

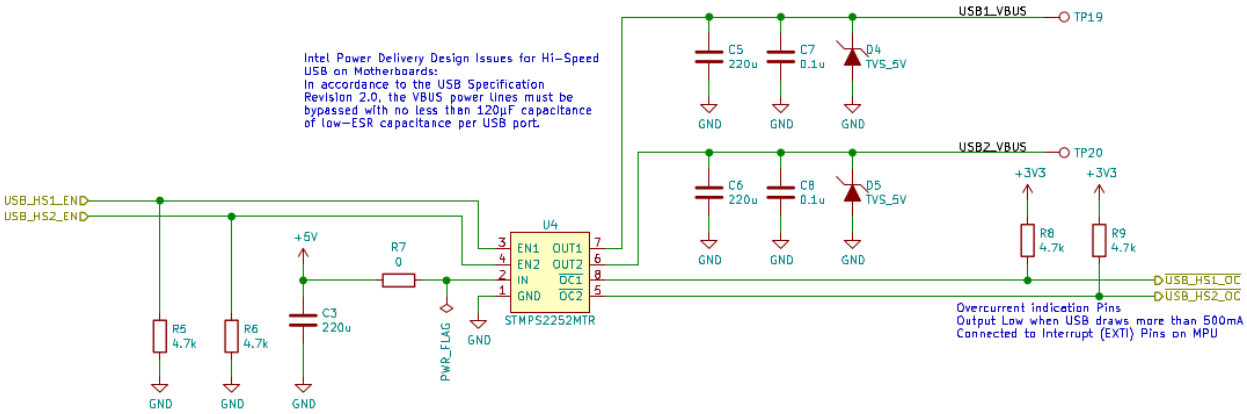


Figure 6.11.: USB Type-A host power distribution

The USB hosts need to power the connected device trough the `VBUS` pin. A high-side power switch IC (`U4`) (figure 6.11) is used for power distribution. The ports can be enabled and disabled from the microprocessor trough the standard GPIO pins `USB_HSx_EN`. The power switch has the ability to detect overcurrents and will signalize it trough the low-active pins `USB_HSx_OC`, which are connected to the microprocessor. On the microprocessor, the overcurrent pins are assigned to an external interrupt service routine (EXTI). This interrupt routine shall immediately disable the related USB port when an overcurrent case occurs.

Across the USB standard specification and several design guides, the USB host supply shall be bypassed with a bulk capacitor of at least $120\,\mu F$ to protect the bus supply net from undervoltage during a hot-plug [35].

6.2.3. MDB/ICP interface

The MDB/ICP interface implements the improvements, which were recommended during the MDB/ICP gateway hardware implementation.

To ensure the decoupling of the signal lines, a linear voltage regulator was added to the design to provide a $5\,V$ supply voltage, which refers to the signal reference ground `GNDS`.

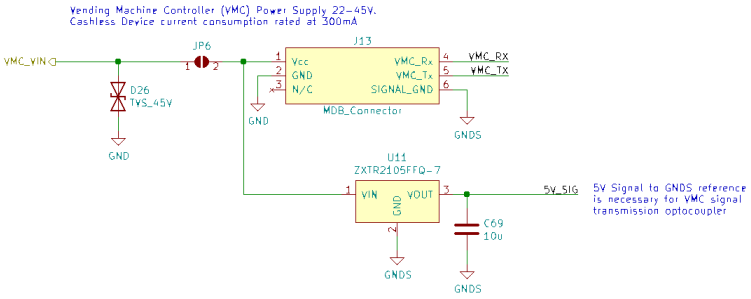


Figure 6.12.: Improved MDB/ICP implementation, by adding a linear voltage regulator to provide the optocoupler's power signal

Linear voltage regulators are simple circuits, that convert higher signal voltages into lower ones by dissipating power trough a switching fet type transistor. Thus, they are not very efficient and not suited for high power applications. In the circuit as illustrated in figure 6.12, the linear voltage reagulator `U11` is only used to power the signal transmission optocuopler, which has a low power consumption of less than $50\,mA$.

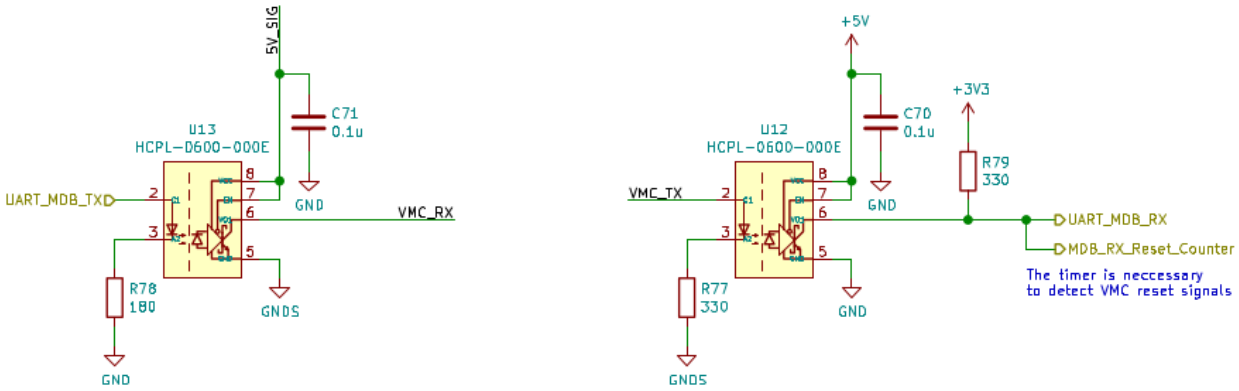


Figure 6.13.: Improved MDB/ICP implementation with separately powered optocoupler

The improvement of this additional voltage source should now vanish the problems during the previous implementation on the MDB/ICP gateway. In figure 6.13, the optocoupler `U13` is now supplied from it's own voltage supply.

6.2.4. Ethernet interface

The Ethernet interface is the most complex component on this board and will constitute several demands on the PCB design.

Ethernet is a family of computer networking technologies and covers cable and connector standards, signal transmission forms and data packaging [36]. It implements the first layer (physical layer) and second layer (data-link layer) of the OSI (Open Systems Interconnection) model. The OSI model is a standard model, which describes how different hard- and software components have to interact together to enable a network communication [38].

Ethernet defines multiple physical communication interfaces, which differ in data transfer rate, cable type and line encoding. The today's most common interfaces are presented shortly:

- 10 BASE-T:
  Has a data transfer rate of $10\,\mathrm{Mbit\,s^{-1}}$. It uses two twisted pair cables for signal transmission on a CAT-3 or CAT-5 cable. The interface uses Manchester-code [39] as line encoding standard, which has the advantage to have on DC-component and thus can be used for magnetic coupling.

- 100 BASE-TX:
  Has a data transfer rate of $100\,\mathrm{Mbit\,s^{-1}}$. This interface also uses two twisted pair cables for signal transmission. The higher transfer rate demands at least an unshielded CAT-5 cable. To enable higher speed transmissions, the 4B5B-code [40] is used together with MTL-3 [41] as line encoding standard. MTL-3 is the encoding standard, which is applied on the signal cable. It uses three voltage levels (0, $+$, -) for data transmission and reduces the signal bandwith by only changing the signal voltage level when a logical `'1'` is transmitted. Unfortunately, it does not offer direct timing recovery, which is necessary for communication. Here the 4B5B-code is used to enable the timing recovery. It covers a unique mapping of four data bits onto 5 signal bits. The standard procedure on 100 BASE-TX signal transmission is to firstly convert the data into 4B5B-code and then sending it with MTL-3 encoding. On the reception side, the signal is reconverted to 4B5B-code again and furthermore to the original data. Another drawback of the MTL-3 encoding is, that the signal transmission has a DC-component. Signal coupling trough magnetics will therefore cause a so-called baseline wandering, which has to be compensated.

- 1000 BASE-T:
  Has a data transfer rate of $1\,\mathrm{Gbit\,s^{-1}}$ and is often called Gigabit-Ethernet (GbE or GigE). In contrary to the previous interfaces, 1000 BASE-T uses four twistet pair cables for signal transmission and demands for

an at least CAT-5 UTP cable. The wire transmission uses the PAM-5 [42], a pulse-amplitude modulation based protocol, which uses five different amplitudes ($1\,\mathrm{V}$, $0{,}5\,\mathrm{V}$, $-0{,}5\,\mathrm{V}$, $-1\,\mathrm{V}$). All four twisted pair wires are used for parallel signal transmission with a clock rate of $125\,\mathrm{MHz}$. PAM-5 enables a transmission rate of two bit per clock-tick, which scales the transfer rate up to $1\,\mathrm{B}$ per clock-tick. With five amplitude voltage levels and four parallel wires, there are $5^4 = 625$ different tokens available. A single byte only uses $2^8 = 256$ token combinations. The remaining $369\,\mathrm{token}$ combinations can be used as forward error correction (FEC) symbols. Before a signal is transmitted trough the PAM-5-coding, the original data-byte is divided up into four pairs of two bits each. This two-bit-pair is extended to three bits with the Trellis-Code-Modulation [43] (TCM). The TCM convolves the two message bits and adds a third redundant bit, which is part of the FEC symbols. These redundant bits is packaged with the information of the two data bits. The actual transmission rate of the 1000 BASE-T interface is $125\,\mathrm{MSymbol\,s^{-1}} \cdot 4\,\mathrm{Wire} \cdot 3\,\mathrm{bit\,Symbol^{-1}} = 1500\,\mathrm{bit\,s^{-1}}$ when redundant information is counted.

An Ethernet controller, or Ethernet PHY (Physical Interface) is able to handle these different interfaces. In more general, these different communication standards are called Media Dependent Interfaces (MDI). The Ethernet PHY has the main burden to detect and handle these MDI's. The communication with the microprocessor (the MAC) is done trough a so-called Media Independent Interface (MII), which uses, as implied by the name, a protocol independent data encoding. There are several MII's defined for different transfer rates such as GMII (Gigabit), SGMII (Serial Gigabit) and XGMII (10 Gigabit) [44]. Additionally, there are also reduced versions available such as RMII and RGMII, where mainly the number of data lines is halved and the data transmission on each line is doubled by clocking data on both clock edges. The selected Ethernet controller will use a RGMII interface for communication with the MPU.

Now lets go further to the implementation. The Ethernet PHY was selected based on the chip, that is used on the Seeed Odyssey STM32MP157C carrier board. This component selection was made, since the developer has no experience with Ethernet at all and wanted to have the ability to backup on an existing design.

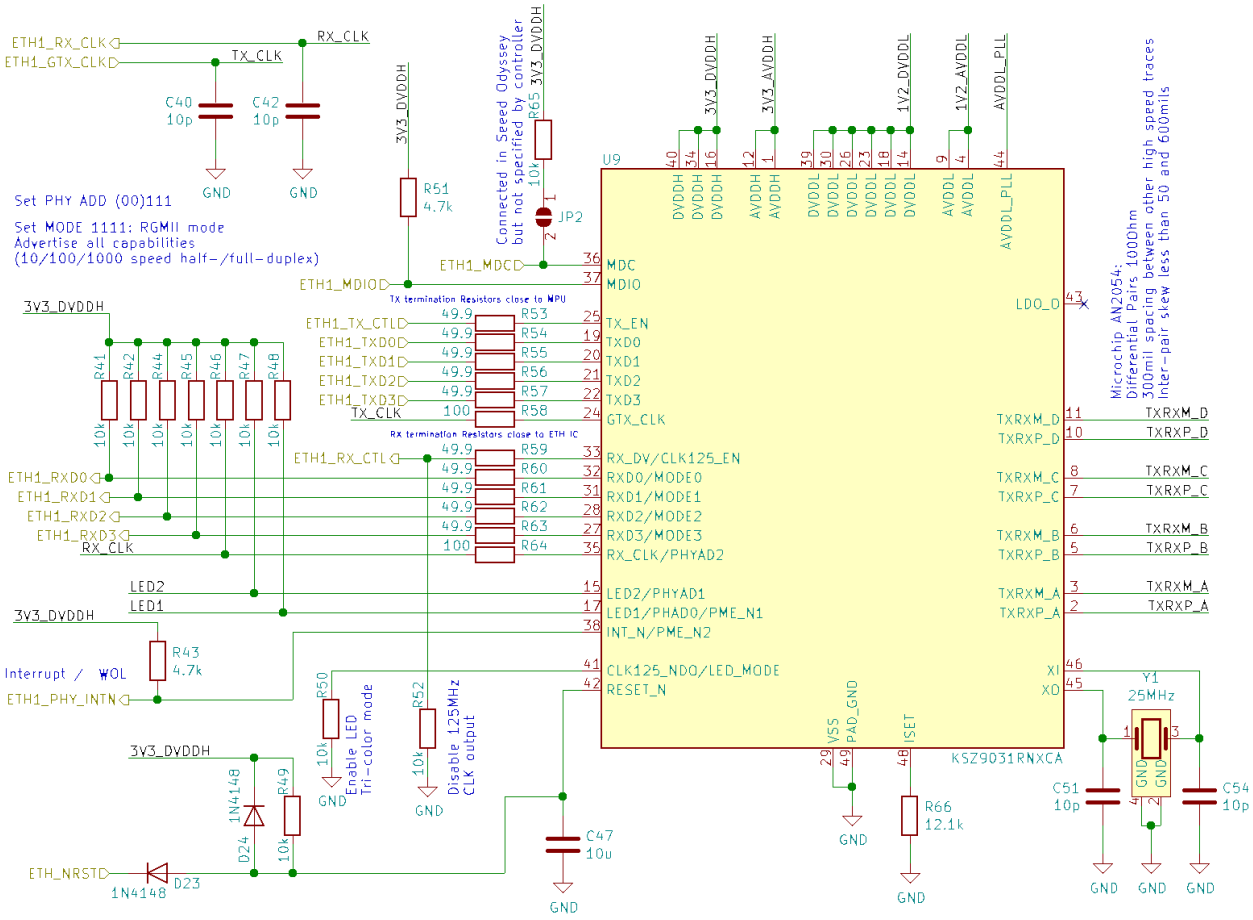


Figure 6.14.: The implemented Ethernet transceiver

The Ethernet PHY is very complicated in comparison with the other components used for this platform. The Pins on the components right side $2, 3, 5, 6, 7, 8, 10, 11$ represent the MDI signal lines in figure 6.14 and are connected to the RJ-45 receptacle. Depending on the connected interface, these four differential pair lines will be used for parallel data transmission at a clock rate of $125\,\mathrm{MHz}$. During the PCB design, the developer must be very careful to ensure a lenght matching throughout all data lines to minimize the pad-skew (timing mismatches because of different signal transfer distances). Another important issue is to ensure a defined impedance between the differential pairs to avoid signal reflections. An impedance of $100\,\Omega$ is specified in the PHY's application note [45].

An external $25\,\mathrm{MHz}$ crystal oscillator is used as reference to generate a $125\,\mathrm{MHz}$ clock signal with an internal Phase Locked Loop (PLL), which is required for the RGMII data transmission. The RGMII interface is situated on the component's left hand side. The full duplex transmission capable protocol implements four parallel data lines a reference clock signal and an enable signal for each direction. Since both communication directions use their own reference clock, the developer only has to match the trace length and impedance of one direction. Termination resistors are needed to prevent eventual signal reflections on the high speed interface. Unfortunately, one can not directly determine the required resistance value, since they are dependent on the PCB layout and other environmental impacts. The manufacturer's application note [45] recommends an initial value of $30\,\Omega$. However, the Seeed Studio's carrier board implementation uses $49{,}9\,\Omega$ resistors for the data and enable pins and $100\,\Omega$ on the clock pins along with a $10\,\mathrm{pF}$ capacitor in parallel. It was decided to use the same values as Seeed Studios does because their Ethernet design will probably very similar and thus will need fewer changes.

A separate communication channel, the Mode Independent Interface Management (MIIM), is available to offer a possibility to the microprocessor to change the Ethernet PHY's settings. It consists of a data and a clock line and is similar to the known I2C bus. The interface on the left hand side (pins $36, 37$) are equipped with external

pull-up resistors. It is specified in the controller's datasheet to use a pull-up resistor on the MDIO pin. The Seeed Studio's implements also a pull-up on MDC, which should not be necessary. Therefore, one decided to implement the additional resistor as an optional part.

During startup, the PHY will set it's configuration based on the pin states on the different configuration channels. The initial configuration can be changed afterwards through the MIIM interface. With the MODE-pins 32, 31, 28, 27, the RGMII operation mode is selected. Since all pins are pulled-up, the controller mode 1111 will be loaded, which advertises all capabilities (10/100/1000 speed half-/full-duplex) to the PHY.

Through the ADDRESS-pins 17, 15, 35, the controller's MIIM address is determined, which has to be used in order to initiate any MIIM interaction. Pins 15 and 17 are also used to drive the LED's on the RJ-45 connectors. The LED operation mode is selected by the LED_MODE pin 41. This implementation uses the tri-state LED mode, where both LED's on the RJ-45 are in use and indicate bus activity and the current transmission speed mode (10/100/1000).

A reset pin will provide the microprocessor a possibility to restart the PHY if an error occurs. It implements the recommended reset pin design.

Another feature of the selected Ethernet conroller is the Wake-up on LAN (WOL) capability. A special message frame is specified in the Ethernet standard to remotely trigger devices to wake up from stand by mode. The WOL 38 is handled on the microprocessor by an external interrupt controller.

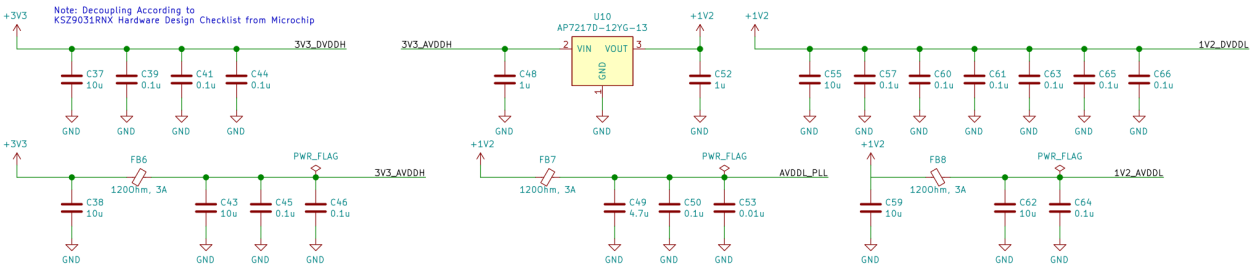


Figure 6.15.: Ethernet PHY power supply

Special demands are desired to the PHY's power supply. The component requires two $3{,}3\,\mathrm{V}$ and two $1{,}2\,\mathrm{V}$ sources for digital and analog signals. Additionally, there is a separate source for the PLL-controller. The application note [46] advises the developer to use a decoupling capacitor on each input pin and recommends the capacitance value. Ferrite beads are implemented to suppress high frequency ripples and back-up capacitors will ensure the decoupling between other planes. The $1{,}2\,\mathrm{V}$ reference is generated with a Low Dropout Regulator (LDO) U10 and is able to provide the recommended $500\,\mathrm{mA}$ of maximal current capability

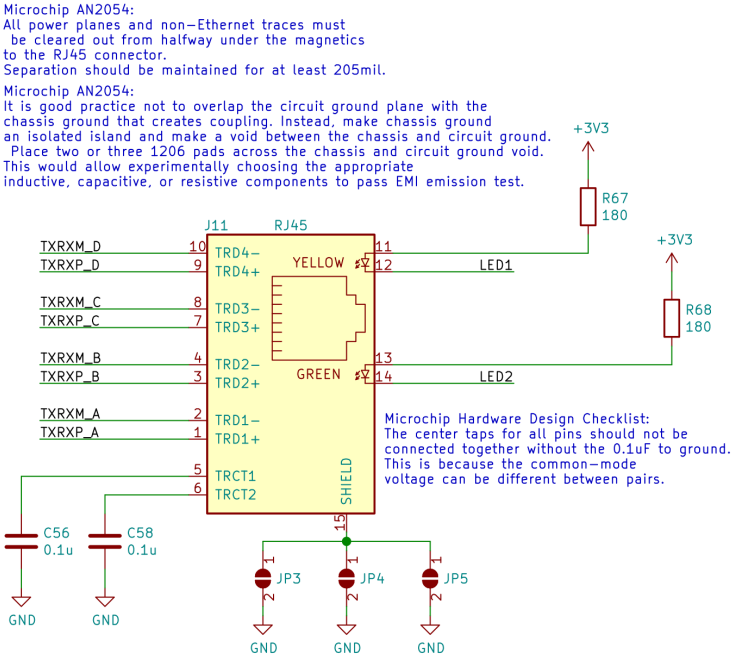


Figure 6.16.: RJ-45 Receptacle

The MDI twisted pairs are connected to the RJ-45 receptacle in figure 6.16, which internally uses magnetics for decoupled signal transmission. The Center Tap (CT) are connected as recommended by application note [45]. To meet EMV standards, three `1206` jumper connectors are connected beween the receptacle's shield and the actual platform's ground plane. They can be bridged with ferrite beads and capacitors. The definitive setup can only be determined based on measurements.

6.2.5. SD-Card interface and boot mode selection

The SD-Card interface is intended to have two different functionalities. One of them will use the SD-Card as boot medium for the embedded operating system. However, it is not the goal to use the SD-Card as boot medium for the operating system during production, since SD-Cards only have a limited lifetime and additionally are not well protected against vibrations. Hereby, the SoM's on board eMMC storage will be a more reliable solution. The SD-Card is moreover intended for testing purposes and for operating system image transfers to the eMMC storage.

Another feature would use the card's storage to log MDB/ICP bus data as it was intended in the MDB/ICP gateway.

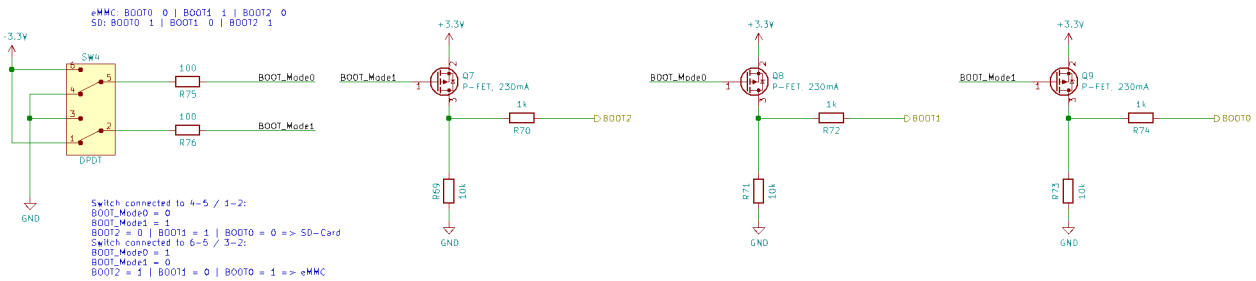


Figure 6.17.: Boot mode interface

The boot configuration is controlled by a switch `SW4` in figure 6.17. It's idea and implementation origins from Seeed Studio's module [47].

6.3. Conclusion

The embedded Taler platform is way more complicated in terms of the schematic design than the MDB/ICP gateway module. Retrospective, it was a good idea to start with the smaller gateway board, since it offered the possibility to test the MDB/ICP interface. A lot of components could be reused during the design of the more advanced board, which gives the developer a lot more confidence.

The outstanding PCB design will be a difficult job, since the platform has a few high speed data signals, which need to be treated specially (impedance control, differential pairs, ...).

7. Conclusion

With the MDB/ICP gateway, a Raspberry Pi extension board with a lot of additional features was developed. The PCB manufacturing- and component costs of a single board amount approximately $65\,\mathrm{CHF}$. Thus, it is comparable with existing products on the marked such as Qibixx's MDB Pi Hat [48]($100\,\mathrm{EUR}$) or Abrantix's MDB2Pi [49] ($120\,\mathrm{USD}$). However, the developed gateway provides more features than the mentioned products.

Before the MDB/ICP gateway is ready to use in development applications, it is recommended to perform a redesign of the board. The weaknesses were described during the schematic explanation as also in the hardware testing part. With the further improvements, this board could be a very interesting product for everyone who is interested in vending machine preipheral development.

The further integration of the gateway to an embedded platform was a good step into the direction of a module, which is ready for commercial applications in vending machines. Even though the development of the board could not be finished during this thesis, there was made a big progress. The final schematic has been developed and the components are selected. It is expected, that the PCB manufacturing- and the component costs will amount $160\,\mathrm{CHF}$.

However, there is still a lot of work before the platform is ready for it's commercial application. In a first step, the PCB will be designed, tested and eventually further improved. Another big part of the project will be the development of the associated software.

Fortunately, the embedded platform is being further developed by an assistant of the Berner Fachhochschule. It is expected that every coffee vending machine at the Berner Fachhochschule will be equipped with this platform during the next year.

Bibliography

[1] worldpay, "Global payments report", *worldpay*, 2018.

[2] Youtube. (2017). Gnu taler (sha2017), [Online]. Available: https://www.youtube.com/watch?v=fY-DWLHVLk4 (visited on 07/25/2020).

[3] C. Grothoff and F. Dold. (2020). Gnu taler, [Online]. Available: https://taler.net/de/index.html (visited on 03/16/2020).

[4] GNU. (2020). Gnu-project, [Online]. Available: www.gnu.org (visited on 04/16/2020).

[5] D. Hofer and M. Boss. (2020). 36c3 taler presentation, [Online]. Available: https://media.ccc.de/v/36c3-oio-186-buying-snacks-via-nfc-with-gnu-taler (visited on 04/16/2020).

[6] Wikipedia. (2020). Gnu taler, [Online]. Available: https://en.wikipedia.org/wiki/GNU_Taler (visited on 03/16/2020).

[7] T. SA. (2020). Talers git repository, [Online]. Available: https://git.taler.net/ (visited on 05/04/2020).

[8] *Multi-drop bus / internal communication protocol*, 4.2, National Automatic Merchandising Association, 20 N. Wacker Drive, Suite 3500, Chicago, Illinois 60606-3120 USA, Feb. 2011.

[9] T. Lochmatter. (2000). Linux and mark/space parity, [Online]. Available: https://viereck.ch/linux-mark-space-parity/ (visited on 03/02/2020).

[10] S. Studios. (2020). Seeed studios stm32mp157c som, [Online]. Available: https://www.seeedstudio.com/Seeed-SoM-STM32MP157C-p-4500.html (visited on 07/25/2020).

[11] S. Electronics. (2020). Stm32cube initialization code generator, [Online]. Available: https://www.st.com/en/development-tools/stm32cubemx.html (visited on 07/25/2020).

[12] ——, (2020). Stm32 nucleo boards, [Online]. Available: https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html (visited on 07/25/2020).

[13] ——, (2020). Stm32-mcu-finder, [Online]. Available: https://www.st.com/en/development-tools/st-mcu-finder.html (visited on 07/26/2020).

[14] ——, (2020). Stm32g070kb, [Online]. Available: https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32g0-series/stm32g0x0-value-line/stm32g070kb.html (visited on 07/26/2020).

[15] ——, (2020). St-link/v2, [Online]. Available: https://www.st.com/en/development-tools/st-link-v2.html (visited on 07/26/2020).

[16] *An2606: Stm32 microcontroller system memory boot mode*, 43rd ed., ST Electronics, Jun. 2020.

[17] *Broadcom hcpl-0600-000e optocoupler datasheet*, 2nd ed., Broadcom, Apr. 2019.

[18] *Ftdi234xd datasheet*, 1.2, Future Technolology Devices International, Aug. 2015.

[19] Wikipedia. (2020). Usb, [Online]. Available: https://en.wikipedia.org/wiki/USB (visited on 07/26/2020).

[20] *Texas instruments lmr14030sddar*, A, Texas Instruments, Feb. 2015.

[21] R. P. Foundation. (2020). Power supply, [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md (visited on 07/27/2020).

[22] T. Instruments. (2019). Output noise filtering for dc/dc power modules, [Online]. Available: https://www.ti.com/lit/an/snva871/snva871.pdf?ts=1595851336780&ref_url=https%253A%252F%252Fwww.google.com%252F (visited on 07/27/2020).

[23] J. E. Aldrick Limjoco. (2016). Ferrite beads demystified, [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/ferrite-beads-demystified.html (visited on 07/27/2020).

[24] J. P. Nicolle. (2016). Jtag4 - run a boundary-scan, [Online]. Available: https://www.fpga4fun.com/JTAG4.html (visited on 07/27/2020).

[25] e. Ei. (2020). Swd vs jtag: Similarities & differences explained..!!, [Online]. Available: https://embeddedinventor.com/swd-vs-jtag-differences-explained/ (visited on 07/27/2020).

[26] kernel.org. (2017). Spidev, [Online]. Available: https://www.kernel.org/doc/Documentation/spi/spidev (visited on 07/27/2020).

[27] ——, (2018). I2cdev, [Online]. Available: https://www.kernel.org/doc/Documentation/i2c/dev-interface (visited on 07/27/2020).

[28] ——, (2018). Serial driver, [Online]. Available: https://www.kernel.org/doc/Documentation/serial/driver (visited on 07/27/2020).

[29] S. Studios. (2020). Stm32mp157c som wiki, [Online]. Available: https://wiki.seeedstudio.com/SEEED-SOM-STM32MP157C/ (visited on 07/27/2020).

[30] A. Devices. (2020). Ltspice, [Online]. Available: https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html (visited on 07/28/2020).

[31] T. Instruments. (2020). Webench power designer, [Online]. Available: https://www.ti.com/design-resources/design-tools-simulation/webench-power-designer.html (visited on 07/28/2020).

[32] Microchip. (2015). Introduction to usb type-c, [Online]. Available: http://ww1.microchip.com/downloads/en/appnotes/00001953a.pdf (visited on 07/28/2020).

[33] S.-E. (Milan). (2020). Usb_otg initialization, [Online]. Available: https://community.st.com/s/question/0D53W00000CPyYxSAL/how-to-initialize-usbotgfs-in-cubemx (visited on 07/28/2020).

[34] Intel. (2020). Emi design guidelines for usb components, [Online]. Available: https://www.ti.com/sc/docs/apps/msp/intrface/usb/emitest.pdf (visited on 07/28/2020).

[35] ——, (2002). Power delivery design issues for hi-speed usb on motherboards, [Online]. Available: https://www.usb.org/sites/default/files/power_delivery_motherboards.pdf (visited on 07/28/2020).

[36] Wikipedia. (2006). Ethernet, [Online]. Available: https://de.wikipedia.org/wiki/Ethernet (visited on 07/28/2020).

[37] J. Burke. (2020). Tcp/ip- versus osi-modell: Was sind die unterschiede?, [Online]. Available: https://www.computerweekly.com/de/antwort/Was-ist-der-Unterschied-zwischen-dem-TCP-IP-und-dem-OSI-Modell (visited on 07/28/2020).

[38] Wikipedia. (2007). Osi-modell, [Online]. Available: https://de.wikipedia.org/wiki/OSI-Modell (visited on 07/28/2020).

[39] ——, (2020). Manchester-code, [Online]. Available: https://de.wikipedia.org/wiki/Manchester-Code (visited on 07/28/2020).

[40] ——, (2018). 4b5b-code, [Online]. Available: https://de.wikipedia.org/wiki/4B5B-Code (visited on 07/28/2020).

[41] ——, (2020). Mtl-3-code, [Online]. Available: https://de.wikipedia.org/wiki/MTL-3-Code (visited on 07/28/2020).

[42] ——, (2020). 5-pam, [Online]. Available: https://de.wikipedia.org/wiki/5-PAM (visited on 07/28/2020).

[43] ——, (2018). Trellis-code, [Online]. Available: https://de.wikipedia.org/wiki/Trellis-Code (visited on 07/28/2020).

[44] H. Electronics. (2020). Media independent interfaces, [Online]. Available: http://www.horizonelectronics.com/knowledgebase/media-independent-interface.php (visited on 07/28/2020).

[45] Microchip. (2016). An2054: Gigabit ethernet design guide, [Online]. Available: http://ww1.microchip.com/downloads/en/Appnotes/00002054A.pdf (visited on 07/28/2020).

[46] ——, (2019). Hardware design checklist, [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/KSZ9031RNX-HW-Design-Checklist-00003391.pdf (visited on 07/28/2020).

[47] S. Studios. (2020). Odyssey-stm32mp157c evaluation board, [Online]. Available: https://www.seeedstudio.com/ODYSSEY-STM32MP157C-p-4464.html (visited on 07/28/2020).

[48] Qibix. (2020). Qibixx mdb pi hat, [Online]. Available: https://www.qiba.pt/products/mdb-pi-hat/ (visited on 07/28/2020).

[49] Abrantix. (2020). Abrantix mdb2pi, [Online]. Available: https://blog.abrantix.com/webshop/product/mdb-to-raspberry-pi/ (visited on 07/28/2020).

A. Declaration of Autorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indi-rect sources used are acknowledged as references.

I am aware that the thesis in digital form can be examined for the use of unauthor-ized aid and in order to determine whether the thesis as a whole or parts incorpo-rated in it may be deemed as plagiarism. For the comparison of my work with exist-ing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted. Further rights of reproduction and usage, however, are not granted here.

This paper was not previously presented to another examination board and has not been published.

Dominik Wenger

B. Working journal

Sheet1

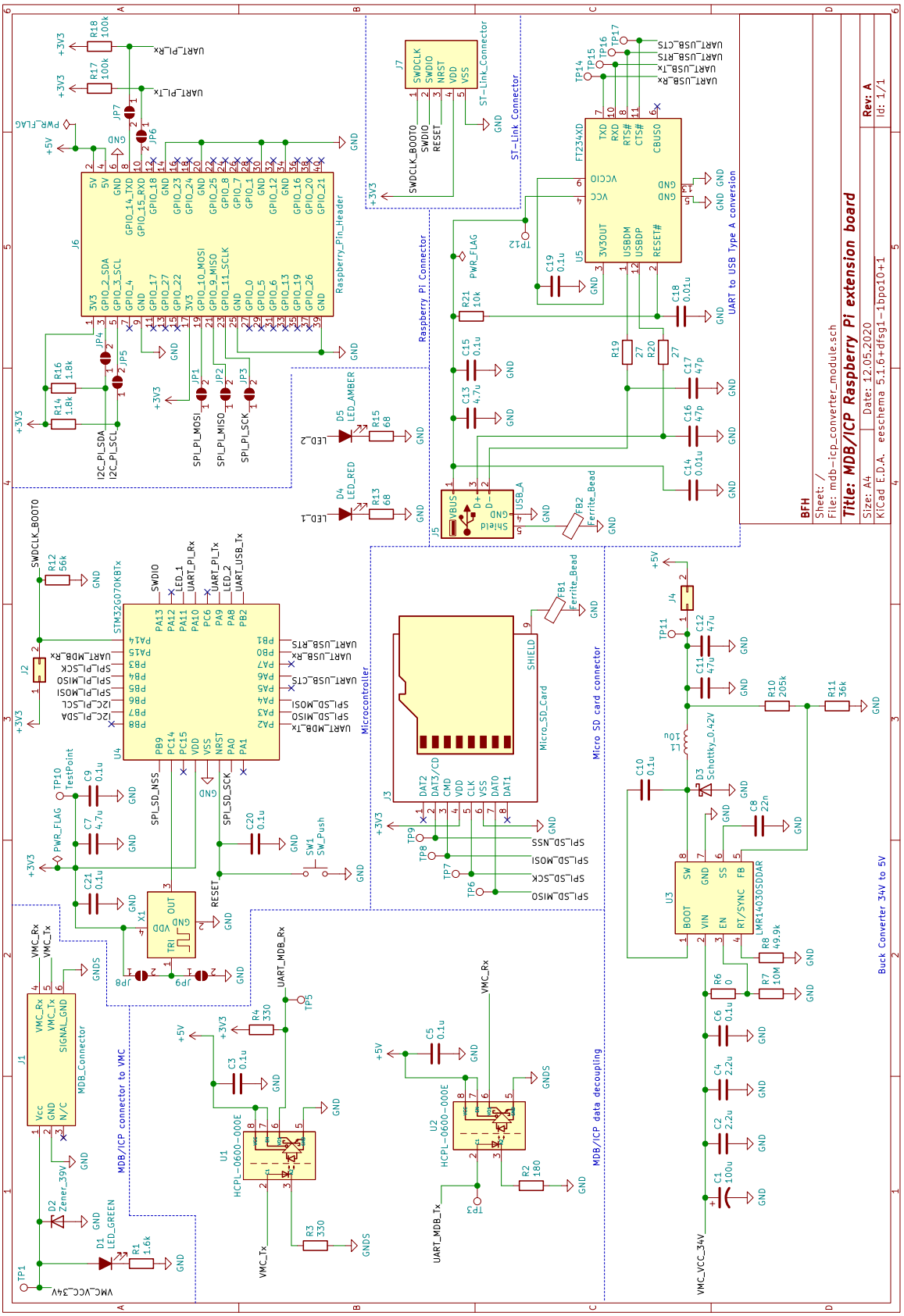
Working Journal Dominik Wenger

This Journal lists the different tasks that were done during the Bachelor thesis

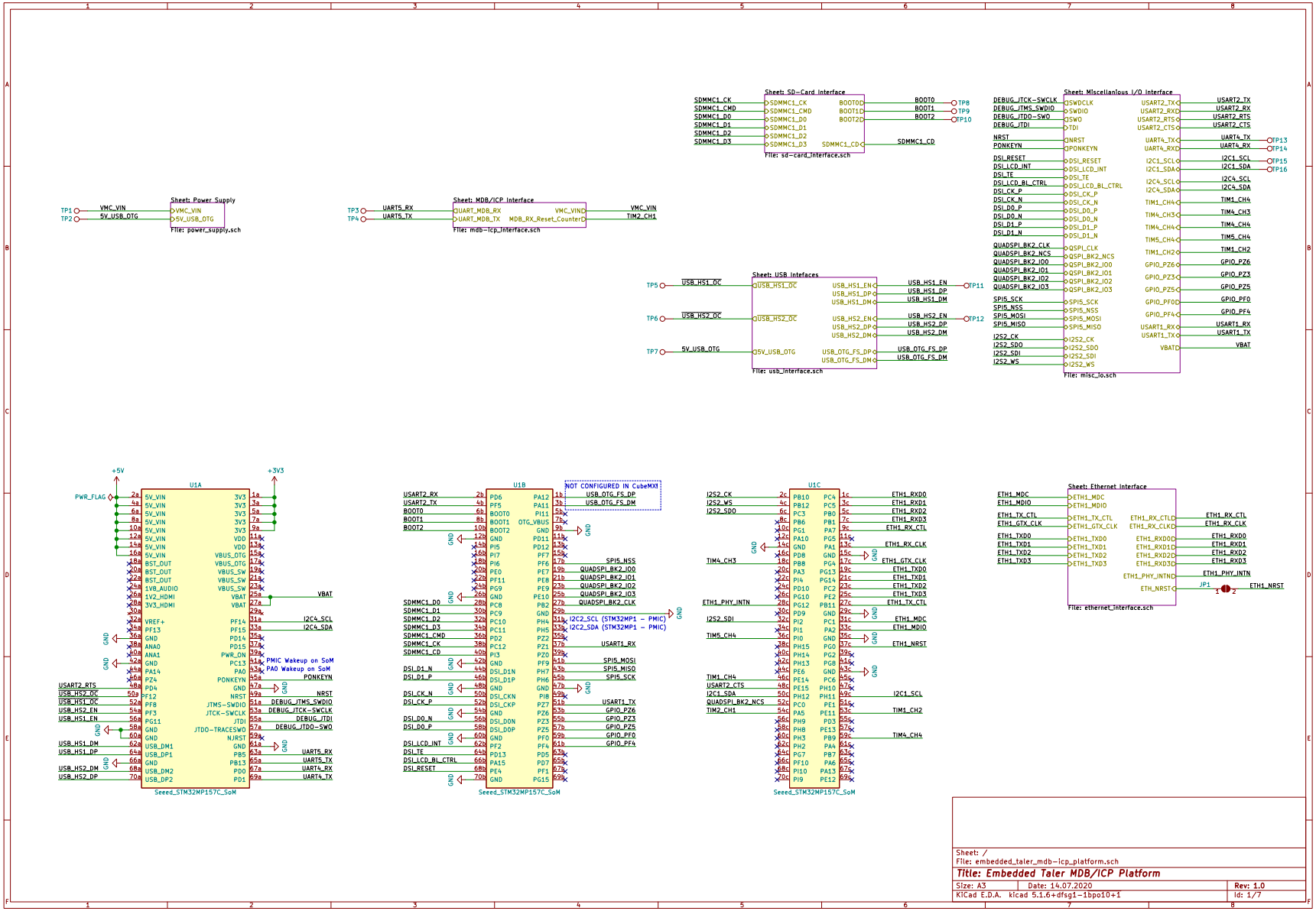
Date:	Workload [h]:	Description:
20.02.2020	2	Introduction to the thesis procedure with other students. Getting own workplace in Sensor Lab.
20.02.2020	2	First introduction to thesis by Dominik Hofer. **Meeting Protocol 1**
20.02.2020	6	Research: Getting familiar with MDB/ICP bus by reading the official protocol by NAMA
24.02.2020	7	Finish MDB/ICP research. Starting with first ideas. Create milestones and Issues for determining the most suitable solution. Setting up second meeting with Andreas Habegger.
26.02.2020	3	Documentation: Setting up the table layout for the specification book with changeable column widths
27.02.2020	9	Documentation / Research: Setting up an specification book for the MDB/ICP converter. Getting stuck with setting adequate specifications. Changing latexmk settings for bibtex_use
02.03.2020	9	Documentation / Research: Develop embedded and microcontroller MDB/ICP concept
05.03.2020	7	Documentation / Research: Finish MDB/ICP conceptioning and writing ideas/concerns in pros and cons.
07.03.2020	7	Documentation / Research: Market analysis of open-source SBC. Writing down the few most interesting products and start comparing them
09.03.2020	5	Completing comparison and generating a table. Maybe its a good idea to adapt an existing design to desired needs.
09.03.2020	1.5	Meeting with Andreas Habegger to present actual status. Discussing the various concepts determine further steps. **Meeting Protocol 2**
12.03.2020	7	Documentation:Adapting the Latex Template and proceed writing.
19.03.2020	3	Meeting with Andreas Habegger to talk about Home-Office. Preparing Raspberry Pi for first Prototype. **Meeting Protocol 3**
26.03.2020	8	Documentation: Writing an Cheatsheet for the Raspberry SSH-Configs. Getting started with Qibixx controller
02.04.2020	8	Starting with Nucleo Software prototype. UART receive bytewise IT. **Meeting Protocol 4**
06.04.2020	6	Software Prototype: receiving VMC signal blockwise with DMA
07.04.2020	10	Software Prototype: Adding Buffer Class
08.04.2020	6	Software Prototype: Rewriting Buffer Class. Measurements done. Little Documentation
16.04.2020	10	Documentation: Rewriting Intro
17.04.2020	7	Documentation: MDB/ICP proper explanation with own generated graphics
18.04.2020	1	Documentation: MDB/ICP proper explanation with own generated graphics
19.04.2020	1	Meeting 5 Preparations
22.04.2020	5	Documentation: Progress with MDB/ICP explanation. **Meeting Protocol 5**
23.04.2020	6	Documentation: Generate cashless device FSM graph. Reading trough and correct.
24.04.2020	0.5	**Meeting Protocol 5**. Update working journal
30.04.2020	10	Documentation: Rewriting Introduction Chapter and Abstract
01.05.2020	12	Documentation: Finish Introduction Chapter
02.05.2020	12	Documentation: Rework Conceptioning Chapter
03.05.2020	14	Documentation: Rework Prototyping and Conclusion chapter. Finish preliminary study
04.05.2020		**Finish preliminary study**
	185	Total working hours

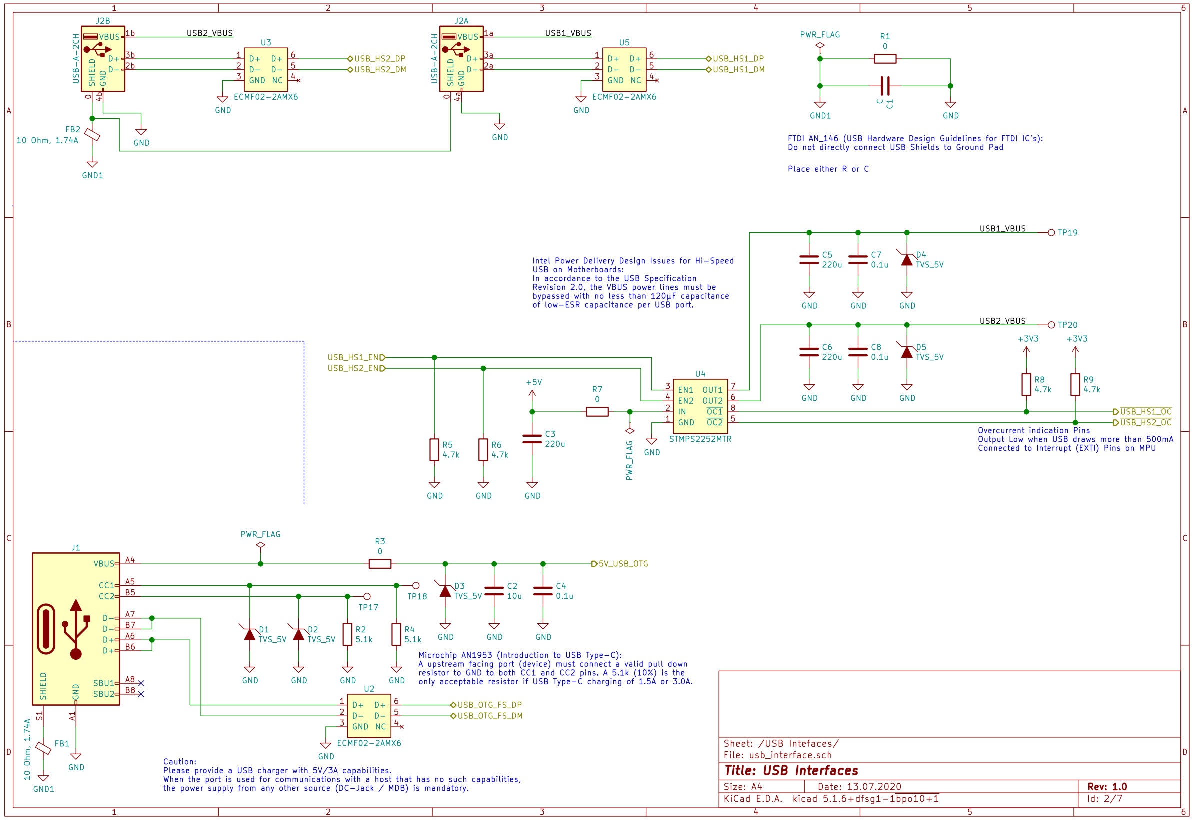
Page 1

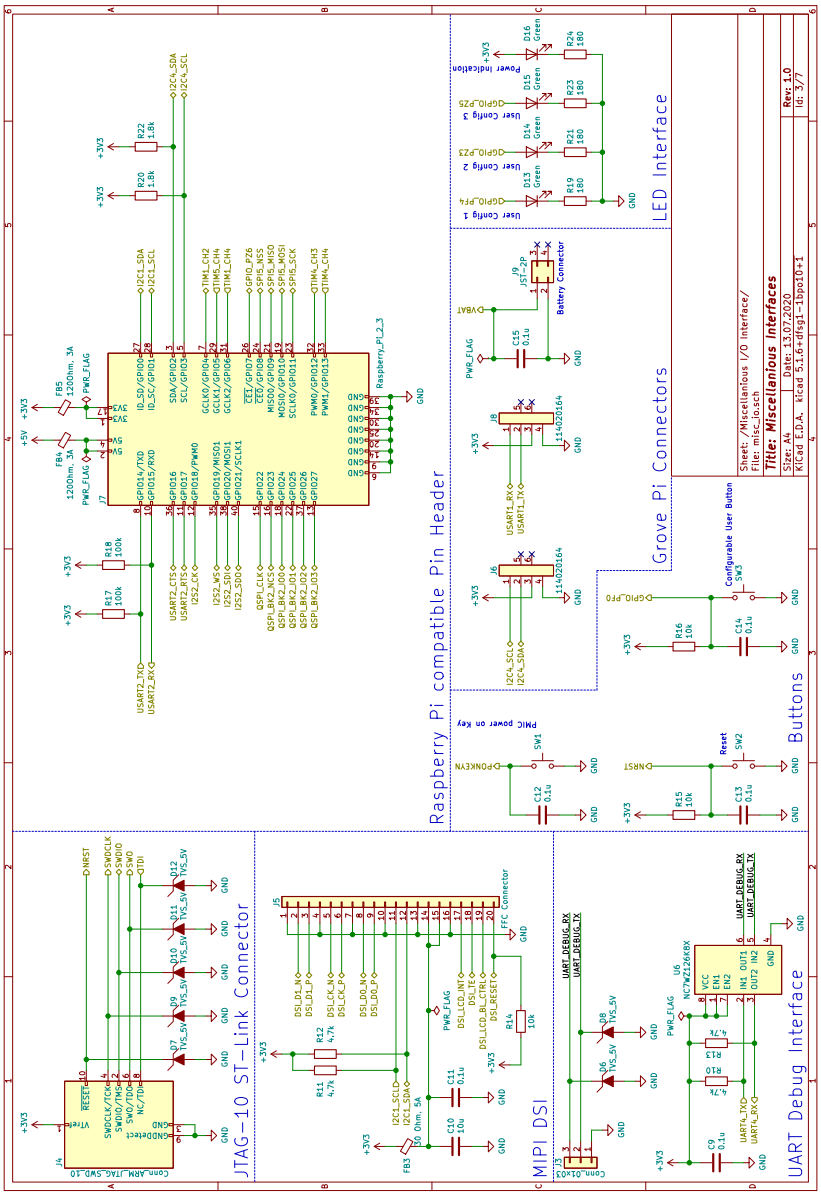
C. MDB/ICP Gateway Schematic

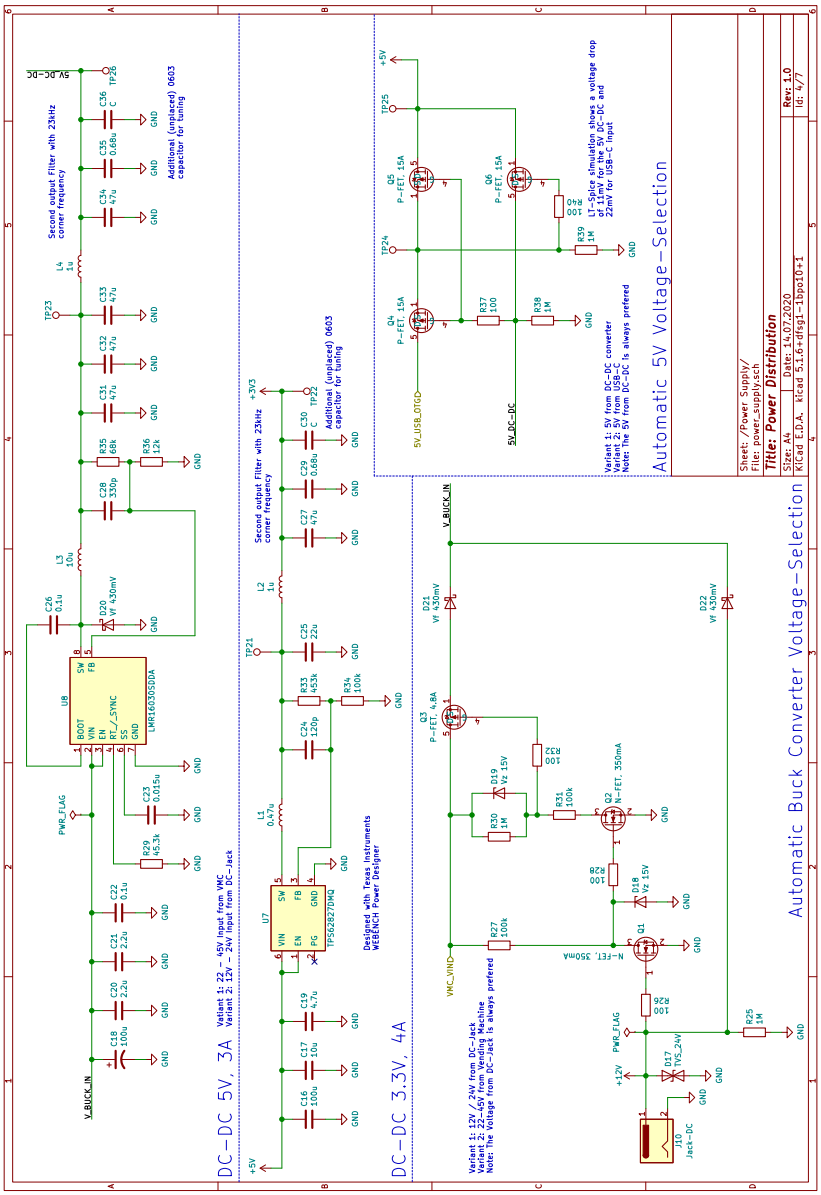

Title: MDB/ICP Raspberry Pi extension board
Size: A4
Date: 12.05.2020
Rev: A
Id: 1/1
KiCad E.D.A. eeschema 5.1.6+dfsg1-1bpo10+1
Sheet: /
File: mdb-icp_converter_module.sch
BFH
Raspberry_Pin_Header
ST-Link_Connector
UART to USB Type A conversion
Microcontroller
Micro_SD_card connector
MDB/ICP connector to VMC
MDB/ICP data decoupling
Buck Converter 34V to 5V
STM32G070KBTx
FT234XD
LMR14030SDDAR
HCPL-0600-000E
MDB_Connector
Micro_SD_Card
Schottky_0.42V
LED_AMBER
LED_RED
LED_GREEN
Zener_39V
Ferrite_Bead
USB_A
SW_Push
TestPoint
SWDCLK_BOOT0
UART_PI_Rx
UART_PI_Tx
UART_USB_Tx
UART_USB_Rx
UART_USB_RTS
UART_USB_CTS
UART_MDB_Rx
UART_MDB_Tx
SPI_SD_NSS
SPI_SD_SCK
SPI_SD_MOSI
SPI_SD_MISO
SPI_PI_MOSI
SPI_PI_MISO
SPI_PI_SCK
I2C_PI_SDA
I2C_PI_SCL
VMC_Rx
VMC_Tx
VMC_VCC_34V
PWR_FLAG
RESET
SWDIO
LED_1
LED_2

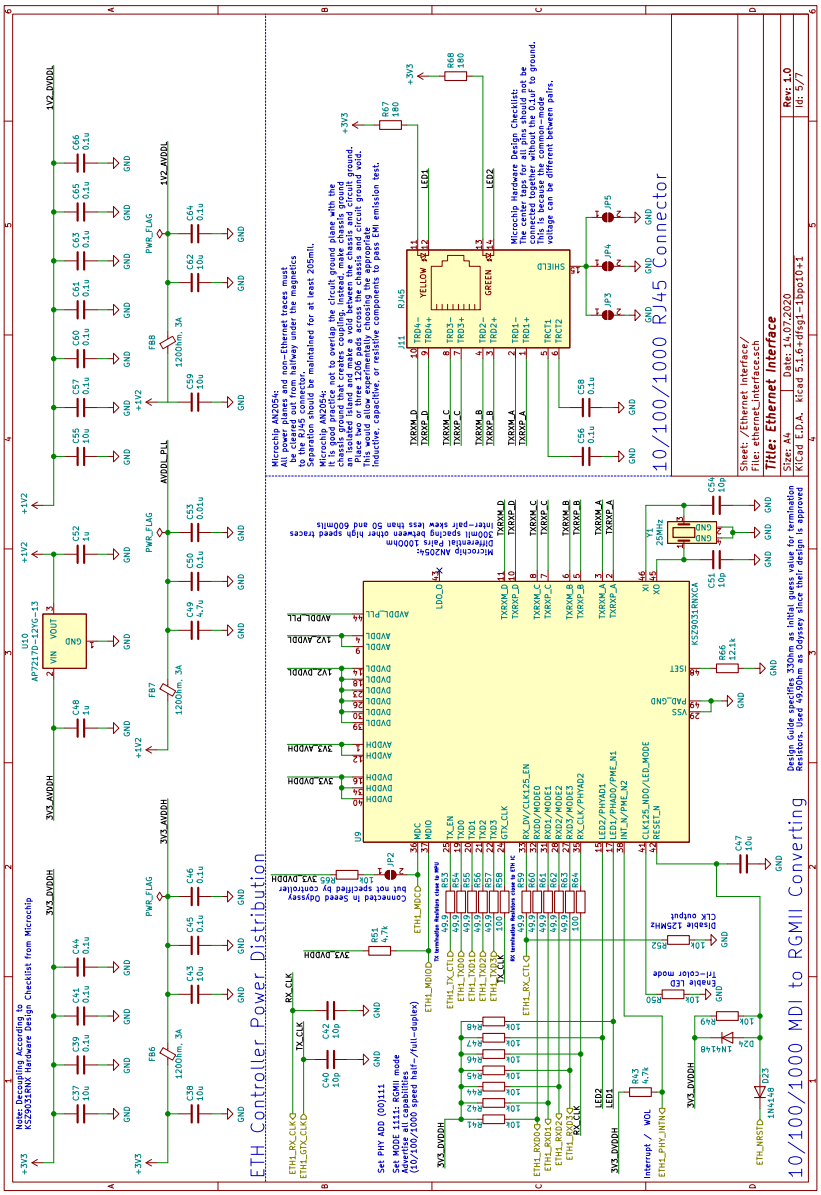
D. Embedded Taler MDB/ICP Platform Schematic

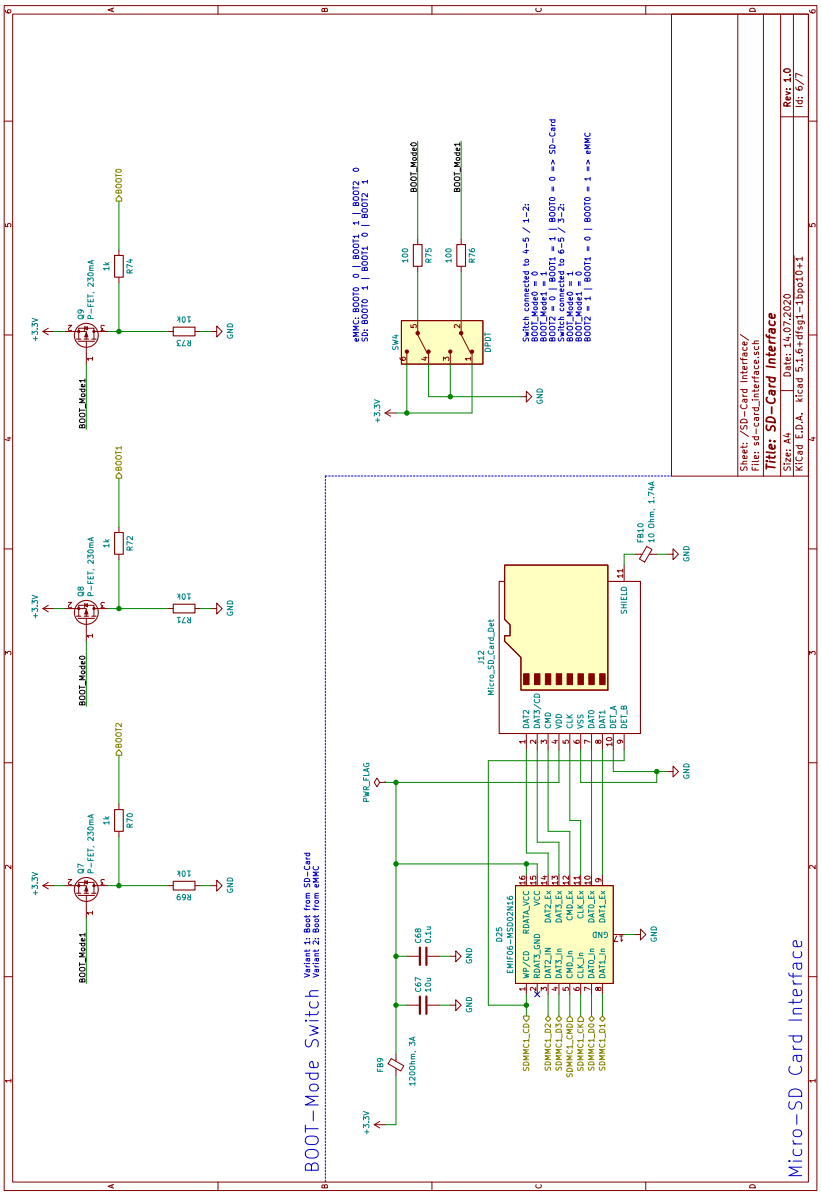

Sheet: Power Supply
File: power_supply.sch
Sheet: MDB/ICP Interface
File: mdb-icp_interface.sch
Sheet: SD-Card Interface
File: sd-card_interface.sch
Sheet: USB Intefaces
File: usb_interface.sch
Sheet: Miscellanious I/O Interface
File: misc_io.sch
Sheet: Ethernet Interface
File: ethernet_interface.sch
U1A
U1B
U1C
Seeed_STM32MP157C_SoM
NOT CONFIGURED IN CubeMX!
Sheet: /
File: embedded_taler_mdb-icp_platform.sch
Title: Embedded Taler MDB/ICP Platform
Size: A3
Date: 14.07.2020
Rev: 1.0
KiCad E.D.A. kicad 5.1.6+dfsg1-1bpo10+1
Id: 1/7

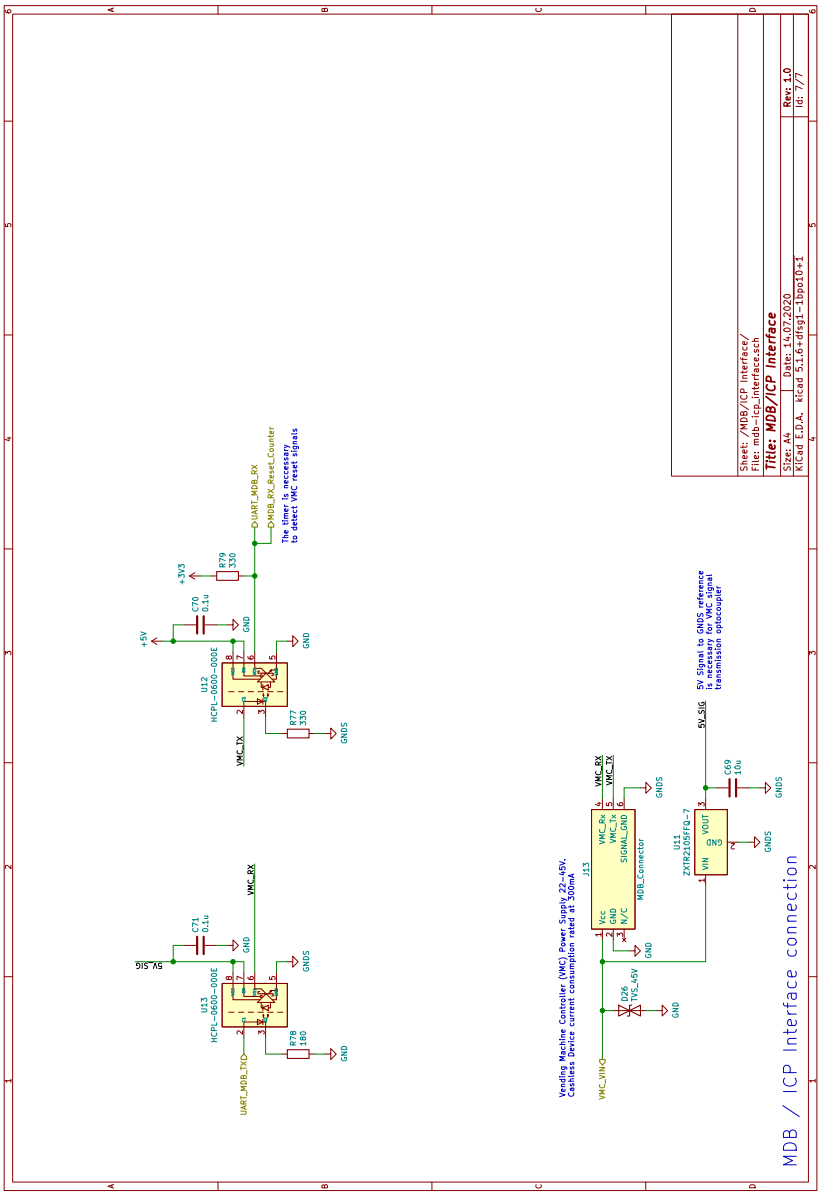

J2B
USB-A-2CH
VBUS
SHIELD
GND
USB2_VBUS
U3
ECMF02-2AMX6
USB_HS2_DP
USB_HS2_DM
FB2
10 Ohm, 1.74A
GND1
GND
J2A
USB1_VBUS
U5
USB_HS1_DP
USB_HS1_DM
PWR_FLAG
R1
0
C1
FTDI AN_146 (USB Hardware Design Guidelines for FTDI IC's):
Do not directly connect USB Shields to Ground Pad
Place either R or C
Intel Power Delivery Design Issues for Hi-Speed USB on Motherboards:
In accordance to the USB Specification Revision 2.0, the VBUS power lines must be bypassed with no less than 120µF capacitance of low-ESR capacitance per USB port.
TP19
C5 220u
C7 0.1u
D4 TVS_5V
TP20
C6 220u
C8 0.1u
D5 TVS_5V
+3V3
R8 4.7k
R9 4.7k
USB_HS1_EN
USB_HS2_EN
+5V
R7 0
C3 220u
R5 4.7k
R6 4.7k
U4
STMPS2252MTR
EN1 OUT1
EN2 OUT2
IN OC1
GND OC2
USB_HS1_OC
USB_HS2_OC
Overcurrent indication Pins
Output Low when USB draws more than 500mA
Connected to Interrupt (EXTI) Pins on MPU
J1
VBUS
CC1
CC2
SBU1
SBU2
R3 0
5V_USB_OTG
TP17
TP18
D1 TVS_5V
D2 TVS_5V
D3 TVS_5V
R2 5.1k
R4 5.1k
C2 10u
C4 0.1u
Microchip AN1953 (Introduction to USB Type-C):
A upstream facing port (device) must connect a valid pull down resistor to GND to both CC1 and CC2 pins. A 5.1k (10%) is the only acceptable resistor if USB Type-C charging of 1.5A or 3.0A.
U2
USB_OTG_FS_DP
USB_OTG_FS_DM
FB1
10 Ohm, 1.74A
Caution:
Please provide a USB charger with 5V/3A capabilities.
When the port is used for communications with a host that has no such capabilities, the power supply from any other source (DC-Jack / MDB) is mandatory.
Sheet: /USB Intefaces/
File: usb_interface.sch
Title: USB Interfaces
Size: A4 Date: 13.07.2020 Rev: 1.0
KiCad E.D.A. kicad 5.1.6+dfsg1-1bpo10+1 Id: 2/7


Raspberry Pi compatible Pin Header
LED Interface
Grove Pi Connectors
Buttons
JTAG-10 ST-Link Connector
MIPI DSI
UART Debug Interface
Power Indication
User Config 1
User Config 2
User Config 3
Battery Connector
Configurable User Button
PMIC power on Key
Reset
Sheet: /Miscellanious I/O Interface/
File: misc_io.sch
Title: Miscellanious Interfaces
Size: A4
Date: 13.07.2020
KiCad E.D.A. kicad 5.1.6+dfsg1-1bpo10+1
Rev: 1.0
Id: 3/7


Automatic Buck Converter Voltage-Selection
Automatic 5V Voltage-Selection
DC-DC 5V, 3A
DC-DC 3.3V, 4A
Title: Power Distribution
Sheet: /Power Supply/
File: power_supply.sch
Size: A4
Date: 14.07.2020
Rev: 1.0
KiCad E.D.A. kicad 5.1.6+dfsg1-1bpo10+1
Id: 4/7
Designed with Texas Instruments WEBENCH Power Designer
LT-Spice simulation shows a voltage drop of 11mV for the 5V DC-DC and 22mV for USB-C input
Variant 1: 5V from DC-DC converter
Variant 2: 5V from USB-C
Note: The 5V from DC-DC is always prefered
Variant 1: 12V / 24V from DC-Jack
Variant 2: 5V from USB-C Backline
Note: The Voltage from DC-Jack is always prefered
Variant 1: 12V - 45V input from VMC
Variant 2: 12V - 24V input from DC-Jack
Second output filter with 23kHz corner frequency
Additional (unplaced) 0603 capacitor for tuning
5V_DC-DC
5V_USB_OTG
V_BUCK_IN
VMC_VIN
PWR_FLAG
LMR16030SDDA
TPS62827DNQ
N-FET, 350mA
P-FET, 4.8A
P-FET 15A


10/100/1000 MDI to RGMII Converting
ETH Controller Power Distribution
10/100/1000 RJ45 Connector
Title: Ethernet Interface
Sheet: /Ethernet Interface/
File: ethernet_interface.sch
Size: A4
Date: 14.07.2020
Rev: 1.0
KiCad E.D.A. kicad 5.1.6+dfsg1-1bpo10+1
Id: 5/7
Note: Decoupling According to KSZ9031RNX Hardware Design Checklist from Microchip
Microchip AN2054:
All ground planes and non-Ethernet traces must be cleared out from halfway under the magnetics to the RJ45 connector.
Separation should be maintained for at least 20mils.
Microchip AN2054:
It is good practice not to overlap the circuit ground plane with the chassis ground plane. The chassis ground plane should be surrounded by an isolated island and make a void between the chassis and circuit ground. Place the chassis mounting holes in the isolated chassis ground void. This would allow experimentally choosing the appropriate inductive, capacitive, or resistive components to pass EMI emission test.
Microchip Hardware Design Checklist:
The center tap of the magnetics must not be connected together without the 0.1uF to ground. This is to provide isolation between the two voltage can be different between pairs.
Microchip AN2054:
Differential Pairs 100Ohm
Inter-pair skew less than 50 and 600mils
30mil spacing between other high speed traces
Connected in Speed Odyssey but not specified by controler
Set PHY ADD (00)111
Set MODE 1111: RGMII mode
Advertise all capabilities
(10/100/1000 speed half-/full-duplex)
Disable 125MHz CLK output
Enable LED tri-color mode
Interrupt / WOL
Design Guide specifies 330hm as initial guess value for termination Resistors: Used 49.90hm as Odyssey since their design is approved


BOOT-Mode Switch
Micro-SD Card Interface
Variant 1: Boot from SD-Card
Variant 2: Boot from eMMC
Title: SD-Card Interface
Sheet: /SD-Card Interface/
File: sd-card_interface.sch
Size: A4
Date: 14.07.2020
Rev: 1.0
KiCad E.D.A. kicad 5.1.6+dfsg1-1bpo10+1
Id: 6/7
eMMC: BOOT0 0 | BOOT1 1 | BOOT2 0
SD: BOOT0 1 | BOOT1 0 | BOOT2 1
Switch connected to 4-5 / 1-2:
BOOT_Mode0 = 0
BOOT_Mode1 = 0
BOOT2 = 1 | BOOT1 = 1 | BOOT0 = 0 => SD-Card
Switch connected to 6-5 / 3-2:
BOOT_Mode0 = 1
BOOT_Mode1 = 1
BOOT2 = 1 | BOOT1 = 0 | BOOT0 = 1 => eMMC
SW4
DPDT
R75
R76
R74
R73
R72
R71
R70
R69
Q9
Q8
Q7
P-FET, 230mA
J12
Micro_SD_Card_Det
SHIELD
FB10
10 Ohm, 1.7A
FB9
120Ohm, 3A
C68
0.1u
C67
10u
D25
EMIF06-MSD02N16
PWR_FLAG
GND
+3.3V
BOOT0
BOOT1
BOOT2
BOOT_Mode0
BOOT_Mode1


MDB / ICP Interface connection
Sheet: /MDB/ICP interface/
File: mdb-icp_interface.sch
Title: MDB/ICP Interface
Size: A4
Date: 14.07.2020
Rev: 1.0
KiCad E.D.A. kicad 5.1.6+dfsg1-1bpo10+1
Id: 7/7
U12
HCPL-0600-000E
U13
HCPL-0600-000E
R79
330
R77
330
R78
180
C70
0.1u
C71
0.1u
+3V3
+5V
5V_SIG
GND
GNDS
UART_MDB_RX
UART_MDB_TXD
MDB_RX_ResetCounter
VMC_TX
VMC_RX
The timer is neccessary
to detect VMC reset signals
J13
MDB_Connector
Vcc
GND
N/C
SIGNAL_GND
VMC_Rx
VMC_Tx
Vending Machine Controller (VMC) Power Supply 22-45V.
Cashless Device current consumption rated at 300mA
VMC_VIN
D26
TVS_45V
U11
ZXTR2105FFQ-7
VIN
VOUT
5V_SIG
C69
10u
5V Signal to GNDS reference
is necessary for the signal
transmission optocoupler