DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Implementation of Communication Between AGV and Charging Station and Adjustments to the Charging Station Software

Report submitted to the Federal University of Santa Catarina

as requirement for approval of the module:

DAS 5511: Projeto de Fim de Curso

Iuri Cuneo Ferreira

Ingolstadt, April 2019

Implementation of Communication Between AGV and Charging Station and Adjustments to the Charging Station Software

Iuri Cuneo Ferreira

This monograph has been evaluated under the context of the module

DAS 5511: Projeto de Fim de Curso

and approved in its final version by the

Course of Control and Automation Engineering

Prof. Dr. Marcelo Stemmer

_____

Banca Examinadora:

M.Sc. Marius Leffler
Orientador na Empresa

Prof. Dr. Marcelo Stemmer
Orientador no Curso

Prof. Dr. Hector B. Silveira
Responsável pela disciplina

Prof. Dr. Júlio E. N. Rico
Responsável pela disciplina

Prof. Dr. Ricardo J. Rabelo
Responsável pela disciplina

Prof. Dr. Ademar G. da C. Junior, Avaliador

Ariel D. Farias, Debatedor

Kevin B. M. Martins, Debatedor

Implementação de Communicação entre AGV e Estação de Carga e Melhorias no Software de Estação de Carga

Resumo

Seguindo o objetivo de estabelecer um meio de comunicação entre um robô autônomo e sua estação de carga, as opções são estudadas e se é concluido que a melhor opção é fazer a comunicação pelos contatos de carga do robô. O software da estação de carga é alterado para aceitar a comunicação e uma placa de circuito impresso é desenvolvida para possibilitar a comunicação. A velocidade de 100 Mbit/s é alcançada em comunicação unidirecional, o que permite que o robô transmita seus logs à estação de carga dentro do tempo necessário.

Introdução

Desenvolvido na empresa arculus GmbH, na cidade Ingolstadt, estado da Baviera no sul da Alemanha, este trabalho teve como objetivo possibilitar e concretizar a comunicação entre um robô móvel automático, AGV, do inglês *Automated Guided Robot*, e sua estação de carga. O robô fica aproximadamente 1 h carregando e tem que transmitir alguns gigabytes de logs para a charging station, que transmite por cabo os arquivos a um banco de dados. Para que a comunicação seja bem sucedida, considerando uma pequena margem de segurança e com uso realizável do canal de comunicação, uma meta de 40 Mb/s é definida.

O meio de comunicação para atingir tal meta não foi estritamente definido pela empresa, mas uma solução foi sugerida com relação a usar comunicações por linha de potência, ou em inglês PLC, *Power-Line Communication*, capaz de atingir velocidades superiores a 100 Mbps com protocolos como 100BASE-T1, transmitindo dados com 100 Mbps por um único par de cabos.

A solução deve ser implementada em hardware e software e seu funcionamento deve ser, idealmente, comprovado por um protótipo. Limites são impostos pelo ambiente industrial em que os robôs devem operar, incluindo, mas não limitados a não usar ondas eletromagnéticas, a não ser que seja próximo ao espectro da luz visível em frequência.

Escolha de estratégias

Uma série de pontos precisaram ser esclarecidos antes do começo da implementação. Os mais importantes foram: cálculo de carga da bateria, detecção de fim de carga, hardware para comunicação, software para comunicação. Para o cálculo de carga, uma estimativa é feita a partir de um modelo da bateria. O fim de carga pode ser determinado pela corrente com que o robô é carregado, se ela por baixa o suficiente, a bateria está próxima de estar completamente carregada. O hardware foi definido a partir da estratégia de comunicação a ser utilizada, determinada por ser a PLC. Uma placa de circuito impresso deveria ser feita para a aplicação e testada como protótipo. Esta placa deveria permitir uma comunicação transparente entre dois processadores. A estação de carga tinha um código simples no começo do projeto. Este código deveria ser levado a um ponto de funcionamento para mais testes com o robô e, a partir do código, uma melhoria deveria ser feita, usando tanto quanto possível do código existente e necessitando de pouco tempo de implementação.

Metodologia

Inicialmente, um circuito para o hardware foi simulado utilizando componentes analógicos, com a intenção de fazer uma comunicação simples e direta com baixo delay. Este design encontrou problemas em seu desenvolvimento e, assim, um módulo para PLC, ou PLM, foi adquirido da empresa I2SE GmbH. O hardware desenvolvido conteve então o PLM, que precisa de um chip capaz de lidar com a camada física da comunicação, assim como uma parte responsável por reduzir uma tensão de alimentação de 24 V a 3.3 V para ser usado pelos chips, e a parte responsável por sobrepor o sinal gerado pelo PLM com a tensão da linha de potência, parte que deve ser capaz de ler da mesma.

O desenvolvimento em software se deu em duas etapas, a primeira trazendo o software a um estágio estável e uma segunda etapa que permite que o software se comunique com o robô. A arquitetura do software foi pensada para que a maior quantidade de código possível pudesse ser reutilizada, permitindo que o código cresça o quanto for necessário.

Solução

A solução implementada em hardware é mostrada nos anexos. A placa de circuito impresso teve três versões, sendo que a última funcionou sem problemas, se tornando o protótipo para testes.

A solução implementada em software, mesmo não tendo um design ideal, o que seria desejável, permite a comunicação com robô, modularizou as funções que já existiam e

permite que o código cresça, como será necessário uma vez que o código receberá funções adicionais realizáveis pela comunicação com a estação de carga. A estrutura do código e mais detalhes são mostrados na seção 4.3.

Resultados

Os resultados foram satisfatórios, atingindo velocidades de transmissão de quase 100 Mbps. O software não passou por um teste completo com a comunicação com o robô porque nenhum robô pode ter seu código alterado para o teste a tempo e por que o tempo não foi suficiente para que o comportamento de um robô fosse simulado. Entretanto, comandos simples como ajuste na corrente e parar carga, assim como transferência de arquivos, foram testados e funcionaram com sucesso pelo módulo de PLC desenvolvido.

Conclusão

Começando com uma pesquisa das tecnologias disponíveis para comunicação, seguindo por um teste de circuito e pesquisa sobre PLMs disponíveis no mercado, o projeto chegou ao desenvolvimento bem sucedido de um protótipo que permite a comunicação do robô com a estação de carga. Em paralelo com o desenvolvimento em hardware, esteve o desenvolvimento do software, começando com pequenos ajustes, seguido por uma grande reestruturação do código e as mudanças finais.

Com hardware e software desenvolvidos e testados na medida do possível, a maioria das metas foram atingidas, com a estação de carga funcionando com software estável e melhorado e uma tecnologia disponível para PLC com velocidades acima do requisitado. O principal ponto que faltou foi um teste geral, não feito por restrições do tempo.

Os desafios encontrados durante o desenvolvimento foram resolvidos, pelo menos parcialmente, para que permitissem o desenvolvimento do projeto. Alguns pontos ainda precisam ser repensados, como a solda do PLM, para que a tecnologia seja produzida em série.

Com o projeto bem sucedido, arculus GmbH tem a seu dispor a tecnologia de PLC e pode aplicá-la não somente ao robô e estação de carga, mas a quaisquer outras linhas de potência disponíveis em seus sistemas. Ainda assim, outras novas tecnologias estão sendo pesquisadas para as mesmas aplicações, como redes locais para comunicação usando tecnologia 5G.

Palavras-chave: Comunicação por linhas de potência, design de software, design de hardware, arquitetura de aplicações.

Abstract

Aiming to achieve a communication between robot and charging station of at least 40 Mbps, the project included making the charging station that works free of failures and developing software and hardware for it to allow the robot to send its logs to the charging station, as well as commands and data when necessary and possible. A board capable of holding power-line communication with another of itself has been developed. Thought to have one in the charging station and one in the robot, they allow for a stable communication between the main processor of the robot and the one of the charging station. The power-line is heavily filtered and this effect had to be handled by the module. The final design achieved speeds of about 100 Mbps, which would be more than enough to meet the requirements, but it is expected to achieve even more in better conditions. The time given prevented further testing and improving of the prototype, but even without them, the results were more than satisfactory. Software has been redesigned and developed to meet the requirements and increase scalability.

Keywords: Power-line communication, software design, hardware design, application architecture.

List of Figures

Figure 1 – Charging station. 24
Figure 2 – arculee docked in the charging station. 25
Figure 3 – Banana Pi model R1 used to control charging station. 27
Figure 4 – Hierarchy and components diagram from charging station. 27
Figure 5 – Initial state machine. 28
Figure 6 – Initial state machine, left side amplified. 29
Figure 7 – Initial state machine, right side amplified. 30
Figure 8 – CCCV charging curve. 33
Figure 9 – CCCV charging curve from logs, example number one. 34
Figure 10 – CCCV charging curve from logs, example number two. 35
Figure 11 – CC curve with current 36
Figure 12 – CV curve with voltage 37
Figure 13 – Voltage under CC 38
Figure 14 – Current under CV 39
Figure 15 – Forward voltage on Schottky rectifier. 41
Figure 16 – Simplified initial class diagram from charging station software 44
Figure 17 – Simplified diagram for signal coupling. 52
Figure 18 – Power and data flow diagram on a bias T-based system. 53
Figure 19 – First version of test simulation with bias T circuit. 53
Figure 20 – Second version of test simulation with bias T circuit. 54
Figure 21 – Second version of test simulation with bias T circuit, left side. 55
Figure 22 – Second version of test simulation with bias T circuit, right side. 56
Figure 23 – Successful data transfer test. 57
Figure 24 – Third version of test simulation with bias T circuit. 58
Figure 25 – Third version of test simulation with bias T circuit, left side. 58
Figure 26 – Third version of test simulation with bias T circuit, right side. 59
Figure 27 – Functional diagram for PLM applications. 60
Figure 28 – Top view of the PLC Stamp 1200 micro. 61
Figure 29 – Bottom view of the PLC Stamp 1200 micro. 61
Figure 30 – Simplified class diagram for final charging station software 64
Figure 31 – Simplified class diagram with sub-process and sub-thread data. 66
Figure 32 – Class diagram for server module. 67
Figure 33 – Class diagram for error reporter module. 68
Figure 34 – Class diagram for serial communication module. 69
Figure 35 – PLC Stamp 1200 micro Evaluation Kit. 72
Figure 36 – PLM soldering error, higher view. 74

Figure 37 – PLM soldering error, side view. 74
Figure 38 – Final design of the developed board. 77
Figure 39 – Final state machine design. 78
Figure 40 – Display of charging station in production-ready state before communication. 78
Figure 41 – Whole setup for dead-wire connection test. 80
Figure 42 – Banana Pi and evaluation kit in dead-wire connection test. 81
Figure 43 – Laptop and evaluation kit in dead-wire connection test. 82
Figure 44 – Test results for connection speed. 83
Figure 45 – Test results for connection speed using iperf. 84
Figure 46 – Test results for connection speed. 85
Figure 47 – Test results for connection speed. 86
Figure 48 – Test results for connection speed with direct connection between laptop and Banana Pi. 87
Figure 49 – Setup for test with charging, laptop communicating with Banana Pi. . . . 88
Figure 50 – Setup for test with charging, laptop communicating with Raspberry Pi. . 89
Figure 51 – Results of test with Raspberry Pi without inductors. 89
Figure 52 – Results of test with Raspberry Pi with inductors. 90
Figure 53 – Results of test with Banana Pi without inductors. 90
Figure 54 – Results of test with Banana Pi with inductors. 91
Figure 55 – Schematics of first version, RJ-45 connector. 101
Figure 56 – Schematics of first version, power supply. 102
Figure 57 – Schematics of first version, PHY chip. 103
Figure 58 – Schematics of first version, components around the PHY chip. 104
Figure 59 – Schematics of first version, coupling circuit. 105
Figure 60 – Schematics of first version, first header and bootstrap configuration. . . . 106
Figure 61 – Schematics of first version, second header, coupling transformer and zero cross detection. 107
Figure 62 – Schematics of first version, third header. 108
Figure 63 – First version of PCB, PCB design without poured copper as ground. . . . 109
Figure 64 – First version of PCB, PCB design with copper pour. 110
Figure 65 – Schematics of PLM adapter board. 111
Figure 66 – PCB design for PLM adapter borad. 112
Figure 67 – Schematics of second version, RJ-45 connector. 113
Figure 68 – Schematics of second version, power supply. 114
Figure 69 – Schematics of second version, PHY chip. 115
Figure 70 – Schematics of second version, components for PHY chip. 116
Figure 71 – Schematics of second version, coupling. 117
Figure 72 – Schematics of second version, first header and bootstrap configuration. . 118

Figure 73 – Schematics of second version, second header, coupling transformer and zero cross detection. 119
Figure 74 – Schematics of second version, third header. 120
Figure 75 – Schematics of second version, status LEDs. 121
Figure 76 – Second version of PCB, PCB design without poured copper as ground. . 122
Figure 77 – Second version of PCB, PCB design with copper pour. 123
Figure 78 – Full class diagram for first, temporary implementation of new software. . 126
Figure 79 – Class diagram for temporary software, GUI. 127
Figure 80 – Class diagram for temporary software, modules. 128
Figure 81 – Class diagram for temporary software, state machine part one. 129
Figure 82 – Class diagram for temporary software, state machine part two. 130
Figure 83 – Schematics of third version, PHY chip. 131
Figure 84 – Schematics of third version, coupling. 132
Figure 85 – Schematics of third version, adapter connectors part 1. 133
Figure 86 – Schematics of third version, adapter connectors part 2. 134
Figure 87 – Schematics of third version, power supply. 134
Figure 88 – Schematics of third version, RJ-45 header. 135
Figure 89 – Schematics of third version, PLM LEDs. 136
Figure 90 – Schematics of third version, PHY LEDs. 137
Figure 91 – Third version of PCB, PCB design without copper pour. 138
Figure 92 – Third version of PCB, PCB design with copper pour. 139

List of Tables

Table 1 – REST requests to charging station. 62

List of abbreviations and acronyms

AC	Alternating Current
AEK	Power supply in the charging stations
AGV	Automated Guided Vehicle
ARC	arculee
b	Bit
B	Byte
BMS	Battery Management System
CC	Constant Current
CCCV	Constant Current Constant Voltage
CV	Constant Voltage
CPU	Central Processing Unit
DC	Direct Current
EVK	Evaluation kit, referring to the one from I2SE for power-line communication
FSM	Finite State Machine
GIL	Global Interpreter Lock
GUI	Graphical User Interface
IC	Integrated Circuit
IP	Internet Protocol
IrDA	Infrared Data Association
JSON	JavaScript Object Notation
LED	Light-Emitting Diode
MAC	Medium Access Control
MQTT	Message Queuing Telemetry Transport
OFDM	Orthogonal Frequency-Division Multiplexing

PC	Personal Computer
PCB	Printed Circuit Board
PHY	Physical (layer)
Pi	Raspberry Pi
PLC	Power-Line Communication
PLM	Power-Line Module
REST	Representational State Transfer
RGMII	Reduced Gigabit Media-Independent Interface
Rx	Receive channel of a communication protocol, opposite to Tx
SSH	Secure Shell
SOC	State of Charge
TCP	Transmission Control Protocol
Tx	Transmit channel of a communication protocol, opposite to Rx
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol

Contents

1 INTRODUCTION 19

2 PRESENTATION OF THE PROJECT AND ITS REQUIREMENTS 23

3 ANALYSIS AND CHOICE OF STRATEGIES 31
3.1 The Problem in Details 31
3.2 Analysis of Possibilities 34
3.2.1 Charging Station 34
3.2.1.1 Controller 34
3.2.1.2 State of Charge Calculation 35
3.2.1.3 Time Left 40
3.2.1.4 Start Charge with arculee Only 40
3.2.1.5 Voltage Loss on Diode 40
3.2.1.6 End of Charge Detection 41
3.2.2 Communication 42
3.2.2.1 Hardware 42
3.2.2.2 API Between ARC and CS 43
3.2.2.3 Protocol Between CS and Log Database 43
3.2.2.4 Software 44
3.3 Chosen Approaches 45
3.3.1 Charging Station 46
3.3.1.1 Controller 46
3.3.1.2 SOC Calculation 46
3.3.1.3 Time Left 46
3.3.1.4 Start Charge with arculee Only 47
3.3.1.5 Voltage Loss on Diode 47
3.3.1.6 End of Charge Detection 47
3.3.2 Communication 48
3.3.2.1 Hardware 48
3.3.2.2 API Between ARC and CS 48
3.3.2.3 Protocol Between CS and Log Database 48
3.3.2.4 Software 49

4 METHODOLOGY 51
4.1 Hardware 51
4.1.1 Analogical Board 51

4.1.2 Power-Line Module Board 56
4.2 API Between arculee and Charging Station 59
4.3 Software 61

5 IMPLEMENTED SOLUTION 71
5.1 Hardware 71
5.2 Software 75

6 RESULTS 79
6.1 Live wire tests 81
6.1.1 Raspberry Pi without inductor 83
6.1.2 Raspberry Pi with inductor 84
6.1.3 Banana Pi without inductor 84
6.1.4 Banana Pi with inductor 85
6.1.5 Final comments 86

7 CONCLUSION 93

BIBLIOGRAPHY 97

APPENDIX 99

APPENDIX A – FIRST VERSION - SCHEMATICS AND PCB 101

APPENDIX B – PLM ADAPTER BOARD 111

APPENDIX C – SECOND VERSION - SCHEMATICS AND PCB 113

APPENDIX D – CLASS DIAGRAM FOR TEMPORARY REORGANISATION OF SOFTWARE 125

APPENDIX E – THIRD VERSION - SCHEMATICS AND PCB 131

1 Introduction

This project has been developed in the company arculus GmbH, located in the city of Ingolstadt, Bavaria, Germany. arculus GmbH offers flexible solutions on modular production for industries. The principle is based on Automated Guided Vehicles, AGVs, that are capable of carrying loaded tables around. The AGVs are controlled by a supervisory central processing unit, named FABMAN, that commands the robots around. To allow the robots to do more, they can get accessories, such as the charging stations, or special tables, called backpacks.

In one of the use cases of the system, the AGVs are responsible for carrying tables around. In one route it could, for instance, pick a table up, move it to the loading station, where the table will be loaded, then either place it somewhere, or move it to the picking station, where its contents can be picked out by an employee or by another machine.

In the beginning of the project, the charging station was in a development state. It would charge, but the code was poorly structured and full of workarounds to cover all use cases required by the charging process. The software had also never been put through an endurance test, when it would run over the course of several days. Due to that, the code should be both redesigned and thoroughly tested.

The AGV, named arculee, generates several gigabytes of logs every day. These logs are important, for they hold the data of the system, which can be necessary in the future for diverse applications. However, due to the usage limits imposed by the industrial internet available in industries and due to the industrial limits on signal frequencies, the data cannot be sent via a regular Wi-Fi connection. Hence the need for a solution that allows the arculee to save all its logs under the period of about one hour, while it charges.

The requirements given to the communication include being capable of sending several gigabytes of data in an hour and limited to physical connections or transmitting with wavelengths between infrared and ultraviolet. Plus, ideally no changes to the arculee housing and structure should be required due to the amount of rework that would be required.

Given the initial situation and the requirements, the project of this thesis can have its objective set to having a working charging station capable of communicating with the robot. For this, the steps should include:

- improvement of charging station software;
- evaluation of communication strategies;
- selection of one communication strategy;

- evaluation of products available on the chosen strategy;
- selection or development of a solution in software and hardware;
- suggestion of changes to AGV software; and
- implementation and testing of a prototype.

Under the above-mentioned topics, the specific points of work follow.

1. Finish basic functions of charging station to have it working.
   a) Get touchscreen to work properly.
   b) Calculate battery state of charge.
   c) Calculate time to full charge.
   d) Show arculee ID.
2. Improve charging station software.
   a) Design new software.
   b) Implement diode voltage lookup-table to allow full charge.
   c) Evaluate and implement strategy to detect full charge.
   d) Suggest communication strategy from charging station to database.
   e) Participate in creation of protocol for communication betweeh AGV and charging station.
3. Implement communication between charging station and arculee.
   a) Research best solution.
   b) Compare results to Power-Line Communication, PLC.
   c) Implement communication on software and hardware.
   d) Propose changes to AGV software.

As a first suggestion to the communication strategy, the local supervisor, Marius Leffler, M.Sc., pointed out the PLC option. It fills all requirements without changes to the external arculee structure. This option should be compared to other possibilities, proposed in item 3.b above.

This document has been so structured as to initially present arculus GmbH to the reader, as well as what the state of development of the charging station was in the beginning of the project. Then, the evaluation and chosen strategies for the communication are shown,

followed by the chosen strategies for implementing the changes to the software. Following that, an overall description of what has been done is proposed, along with relevant diagrams and figures, justifying how these choices fill the given requirements. Having that, a description of the final solution is given, followed by the results and a conclusion with an analysis of what has been done against the goals and future works.

2 Presentation of the Project and Its Requirements

As mentioned in chapter 1, the project was developed in the company arculus GmbH. The company is situated in the city of Ingolstadt in Germany. With around 30 employees, the company develops solutions for modular production strategies.

The core of modular production systems is to split a process into primitives [1], allowing for higher flexibility and scalability, among other advantages.

To implement such a flexible system in industries, arculus GmbH offers a system with AGVs, called arculees, which are ordered around by a central processing unit, named FABMAN. The robots can carry loads around in an area. These loads can be virtually anything, from items that can be placed on a shelf or table to other machines, the backpacks. For instance, a pick-and-place robotic arm operating on a battery could be placed on top of the arculee, so it could move around in the shop-floor as needed. The battery can be recharged when the robot recharges, or from the battery of the robot, according to time requirements.

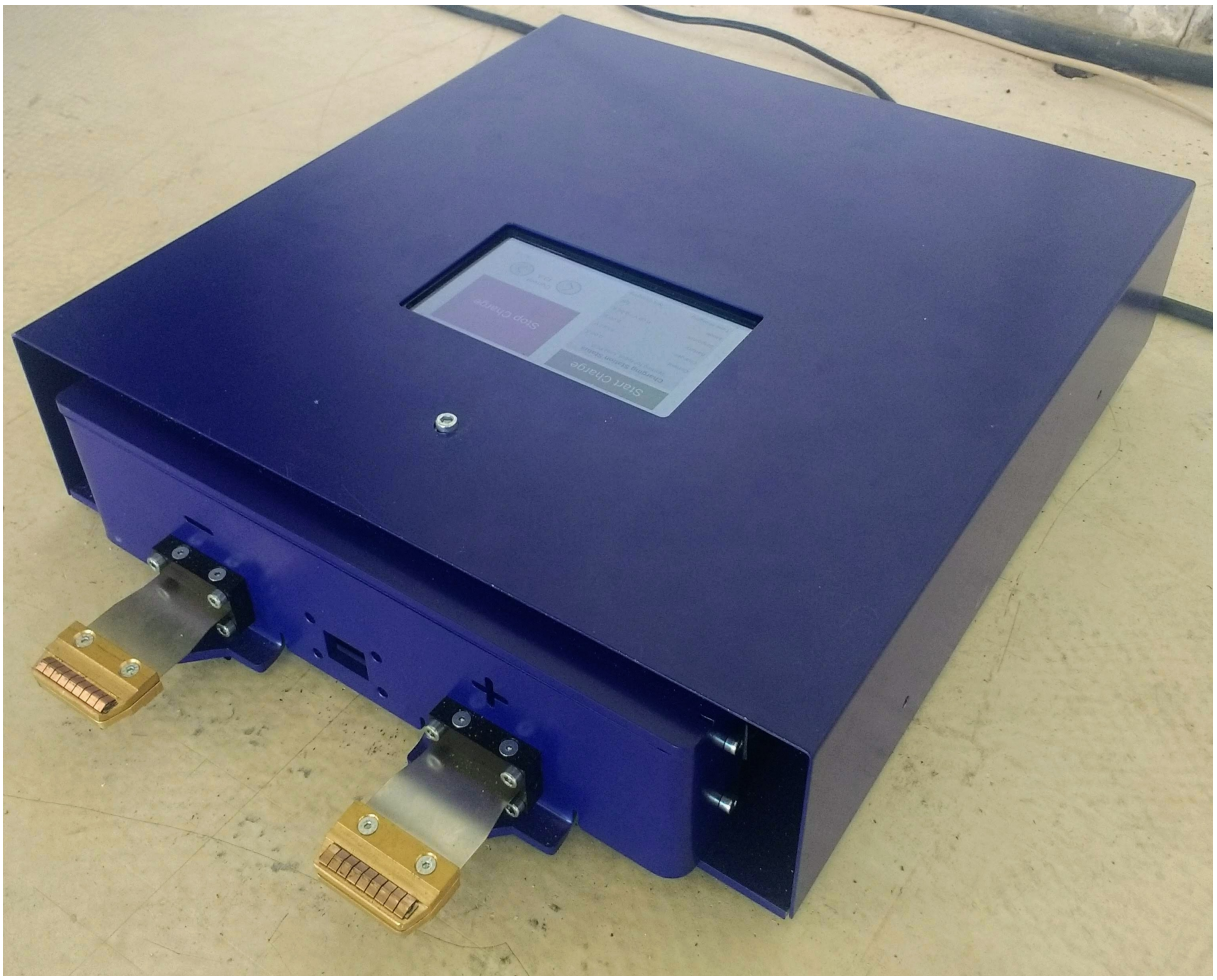
The charging station is presented in fig. 1 and the arculee charging is presented in fig. 2.

The robots have a charging station. It should be as simple as possible to operate: ideally the robot arrives, the charging station detects the presence of the arculee and starts charging. Then, when the robot leaves, the charging station should detect that the robot left, lower voltage and block current and wait for the next robot to dock.

To develop the whole system, the company has five teams, responsible for software, navigation, hardware, robotics and management. The software team is responsible for the codes from robot and charging station, as well as the supervisory processing unit and user interfaces. The navigation team works on software of the robot related to positioning, movement and safety. Hardware team works on metal and plastic structures, from their design to manufacture of prototypes. The management team deals with business and commercial contacts and partnerships, human resources and planning. And the robotics team is responsible for all electronics, the Printed Circuit Boards, PCBs, cabling and the firmwares for the microcontrollers.

The student has been hired by the company with the goal of developing a communication system between arculee and charging station that can operate on the shop-floor of industries. For that, the charging station would need to have more functions than it had in the beginning and some piece hardware would have to be added between charging station

Figure 1 – Charging station.



Source: personal archive.

and arculee, at least for the physical layer of the communication.

Even though the system works without communication, it would create possibilities of data analysis and modelling that would make up for a better, more efficient operation in the future. For instance, if the charging operation can be linked to a robot, and thus to one battery, the ageing of the battery can be estimated and, through that, it is possible to achieve a good estimation of run time. This is interesting from a process point of view as battery ageing causes a loss of over 30% of their full capacity.

Another important use case is possibility of ideally charging two different battery types. Assuming that one battery should be charged with current of 13 A and another accepts 25 A, without communication the maximum charge current is in principle limited to 13 A for both batteries, whereas with communication, the robot could tell the charging station which current to charge with according to its battery.

At last, among other use cases, a third important function brought by the communication is the certification that the correct AGV is being charged. This applies when, for

Figure 2 – arculee docked in the charging station.

Source: personal archive.

instance, the FABMAN tells the charging station that an arculee with certain ID will dock. After a short while, an arculee docks and the charging station can check its ID against the expected one and, if it is not the expected, an error can be generated to make the station free for the robot that needs to be charged.

However, the requirements for the implementation of the communication are constrained to industrial environments and to the need for high data rate, as follows.

It must not use wireless transmissions, unless the signal is in the visible light, infrared or ultraviolet ranges. This avoids any interference with other signals that may be used on the factory, also making for a more robust communication.

The robot should have as little structural changes as possible. This reduces implementation costs on existing robots and should avoid or reduce redesign on the robot and charging station housings and internal structures.

It should be able to transmit around 10 GB of data in under one hour. This makes for a baud rate of at least 22.3 Mb/s assuming an ideal communication with no header, 0 % package loss and that all bits are used for data. Assuming that the channel will not operate on 100 % of its capacity and counting with losses introduced by headers and start and stop bits, it would be reasonable to consider that a 22.3 Mb/s transmission would lead to around 6 GB/h, so a transmission of over 40 Mb/s would be best to reach the 10 GB goal in a realistic scenario.

Besides, thought should be given to the available ports of the arculee brain, its Central Processing Unit, CPU. Ideally, an Ethernet port should be used, but an USB port could be made available as well, if required. Other connectors should be avoided at first, but, if another strategy is proven to be worth it, adaptors can be used as well.

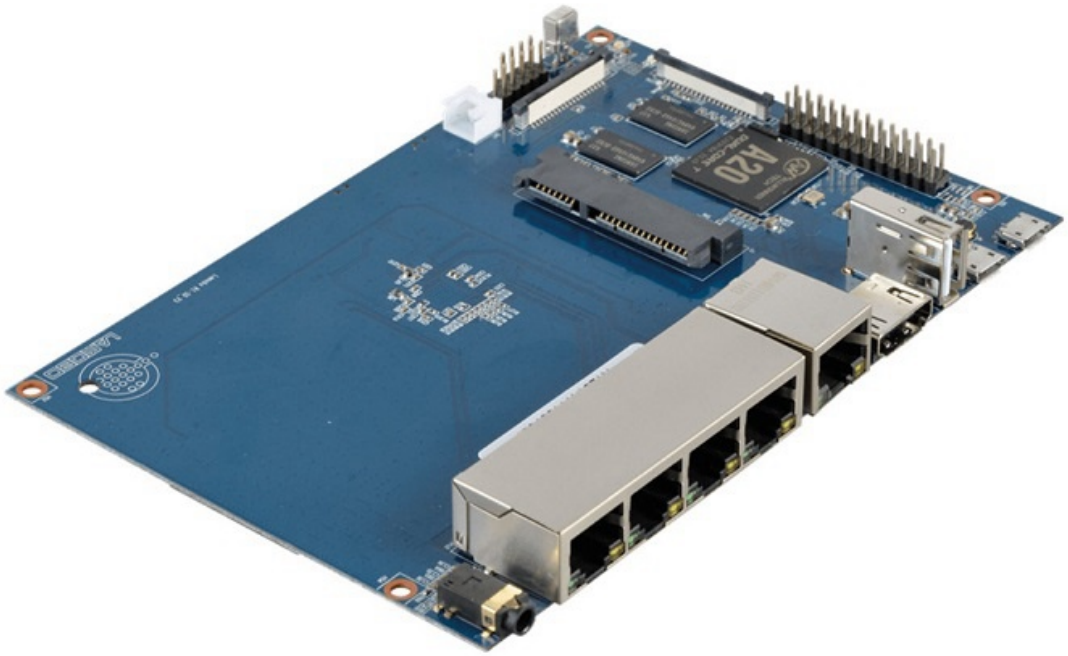
Given the requirements, an initial suggestion has been given by the local supervisor towards the use of the Ethernet standard 100BASE-T1, standardised in IEEE 802.3bw-2015. This standard refers to a physical layer of 100 Mb/s on a single twisted pair cable. The option would fulfil all the requirements, but it should still be compared to other approaches regarding product availability, costs and baud rate. Even though 100 Mb/s would be enough for how the system runs now, a faster communication would be beneficial for future applications or changes, e.g. if more logs are written. The communication would not run entirely on a twisted pair, but only within the robot, as the pair must be separated to go into the charge plugs.

The charging station has a very simple structure. It consists in three components, namely one controller, one power supply and one PCB. The PCB, named “charging PCB”, “switching PCB” or simply “PCB”, developed by arculus GmbH, is responsible for measuring voltage and opening and closing MOSFET gates that connect the AGV to the power supply. The power supply is referred to as “AEK”. The arculee might be referred to as “ARC” or “arc”, the charging station, as “CS”, and the controller used in the beginning of the project for development purposes was the Banana Pi R1, figure 3, which is similar to a Raspberry Pi; it is often referred to as “Pi” or “RasPi” and has its documentation available online [2].

The charging station must respond to the actions of the arculee and its components, the AEK and the switching PCB, must respond to the controller, the Pi. Thus, a hierarchy diagram with the components can be drawn, fig. 4.

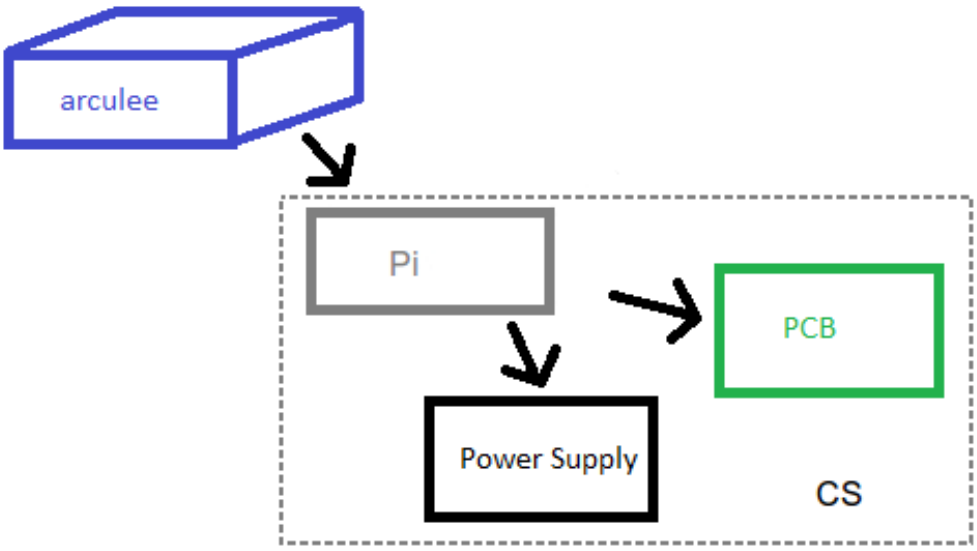
Along with the development on hardware, a rework of the code on the charging station was essential for a successful implementation of the communication. The code at first was very simple and was not programmed after any thoroughly thought project other than a simple state machine, fig. 5, 6, 7.

Figure 3 – Banana Pi model R1 used to control charging station.



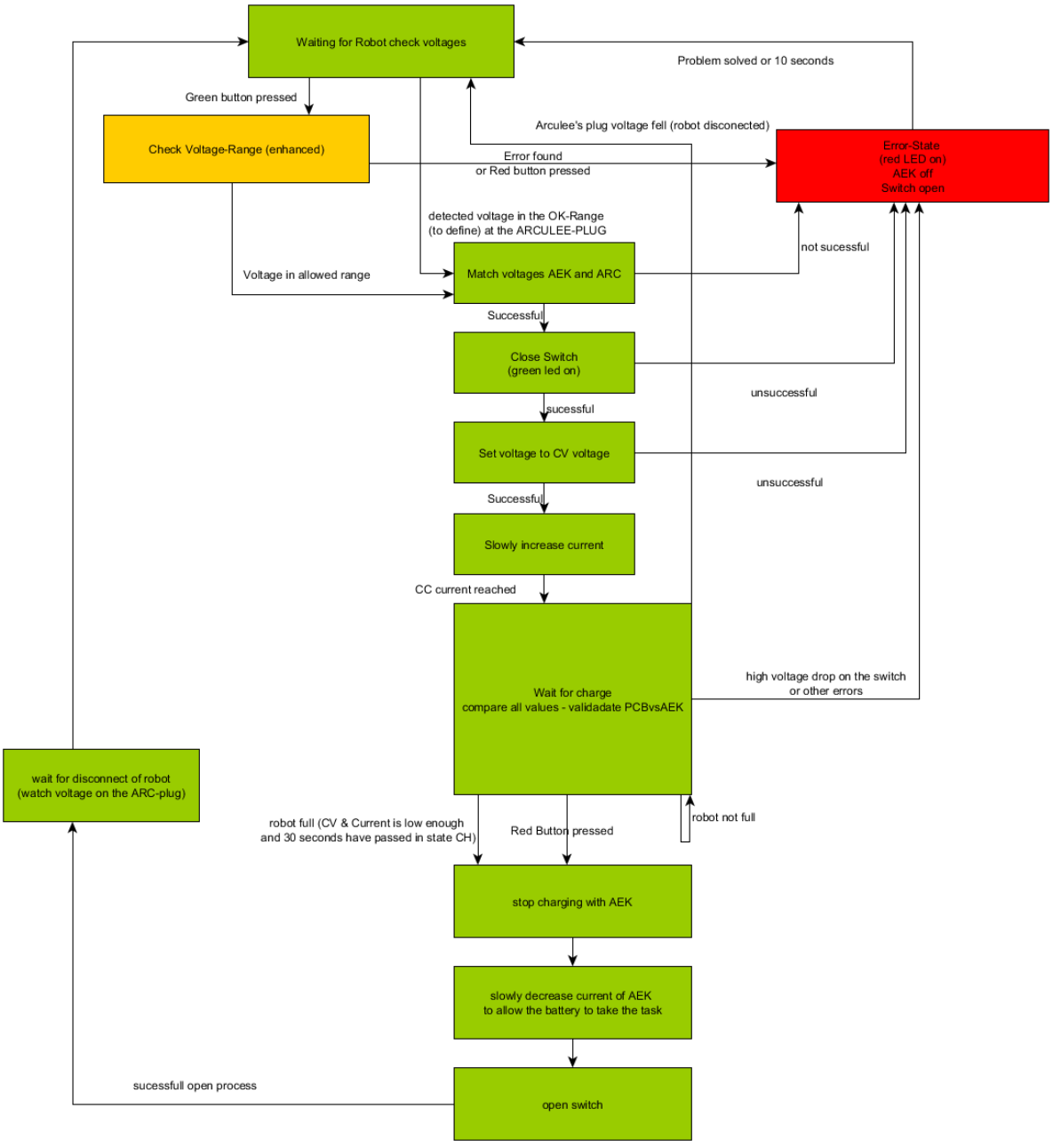
Source: Banana Pi Wiki [2].

Figure 4 – Hierarchy and components diagram from charging station.



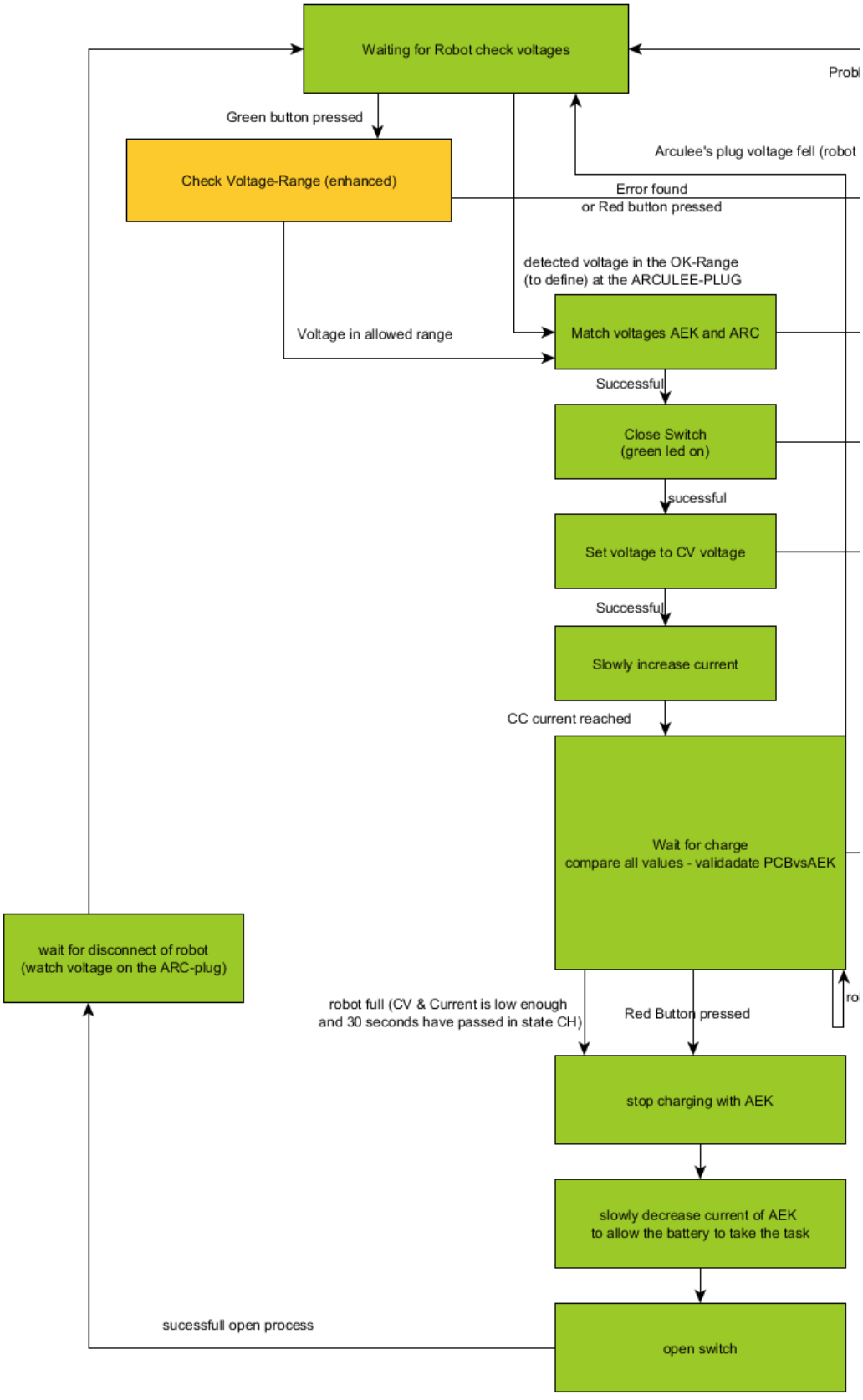
Source: personal archive.

Figure 5 – Initial state machine.



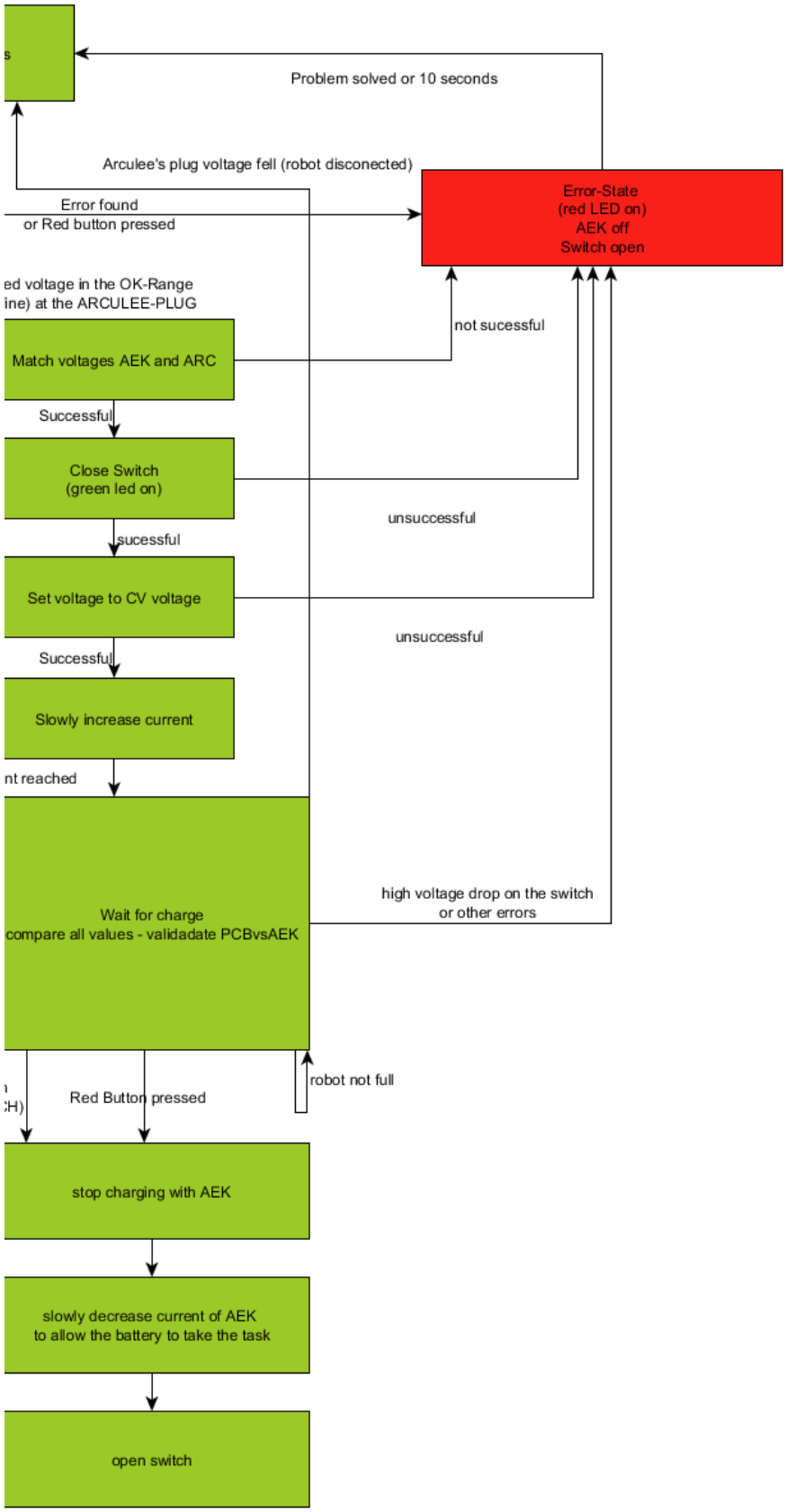
Source: personal archive.

Figure 6 – Initial state machine, left side amplified.



Source: personal archive.

Figure 7 – Initial state machine, right side amplified.



Source: personal archive.

3 Analysis and Choice of Strategies

Given the initial picture of the system in chapter 2, this chapter will present the problem in more details, then the possibilities of solutions that have been evaluated and compared and, at last, the chosen solution for each topic.

3.1 The Problem in Details

Based on the end goal of having a fully operational charging station reachable by the arculee, the first step would be to break the goal into smaller milestones, representing steps to be taken towards having the working CS.

First, the arculee and the charging station must work and, unlike the arculee, no one was working on the charging station, so the responsibility of having it working and further developing it fell into the scope of the project. With a working charging station, the communication can be developed and tested. However, the development of the communication itself is also heavily based on software, as the communication opens several possibilities of future functions on the charging station, thus the need for a well-structured software running on it.

The first task division can be made onto getting the charging station to a good state for further development and then developing software and hardware further for the communication. Then, both the communication and the charging station software can be further divided into smaller tasks that build up to a fully-operational communication between charging station and arculee.

In the early development stages of the charging station, the controller Banana Pi was responsible for controlling the charge procedure, as mentioned in chapter 2. However, it presented some issues regarding the touchscreen that was attached to it; it would show content, but not recognise touches. The display is the model RB-LCD7-2 from Joy-It, a 7 inches touchscreen display for Raspberry Pi and for some versions of Banana Pi with resolution 1024×600 pixels.

The Banana Pi R1 has been initially chosen as controller for having seven Universal Asynchronous Receiver/Transmitter, UART, ports, plus five Ethernet ports, one of which supporting Gigabit Ethernet, its transmission speed being limited by hardware at around 460 Mb/s, to be shared among all UART, USB and Ethernet ports. However, due to the display not being recognised, the Banana Pi should be replaced by another controller, as making the display work with it would take too much time and effort.

Due to that, a new controller should be selected and the touchscreen should be set up to work with it. Once that is done, some information should be printed on the screen for

the user, information which was not computed in the code yet, such as state of charge, time to full charge and a field for the arculee ID. With these tasks ready, the charging station would work and successfully charge an arculee when desired, as well as stop the charge via a button on the touchscreen.

Then, with the software running stable, the design of the next, scalable version of the software should be started. Once a satisfactory design has been achieved, it would be implemented, tested and the next changes could then be implemented on it.

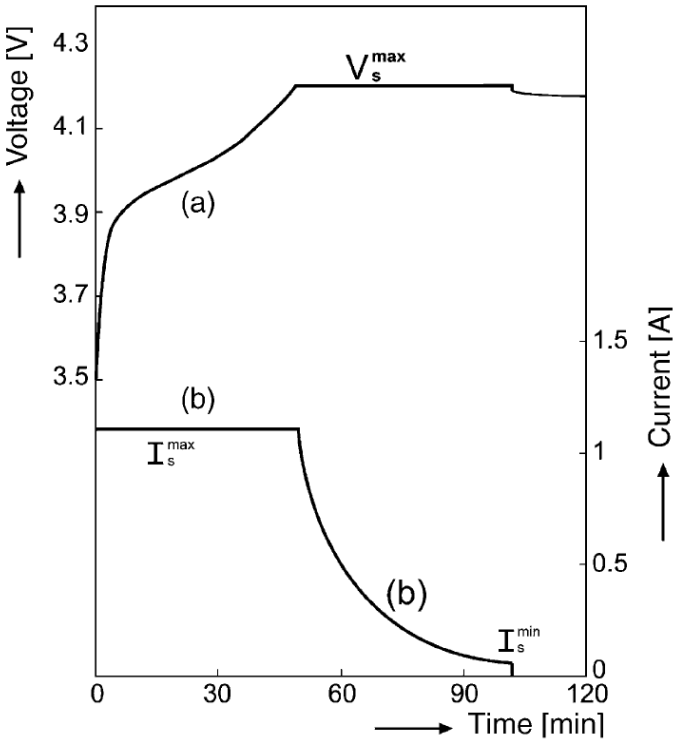
The arculee has a low-ohmic path for the current from the charge plugs to its battery. This path includes a Schottky diode, to prevent the current from potentially flowing into the charging station, or through a person who touches the contacts. This kind of diode has been chosen by the electronics specialist of arculus GmbH for low forward voltage and fast switching action. Its forward voltage depends on the current and, while not accounted for in the code of the charging station, it prevents the battery from ever fully charging. The arculee has a Lithium-ion battery and, being so, it is important that it sometimes receives a full charge to properly balance the cell voltages.

The battery has a 50.4 V charging voltage and can be charged safely with up to 13 A, safety margin included. It can be charged with higher currents, up to around 20 A, but it brings extra stress to the cells, accelerating battery ageing. Lithium-ion batteries are charged with the a charge approach known as CCCV, or Constant Current Constant Voltage [3]. For such, the charging voltage of the cells should be set by a power supply and its current should be limited to the charging current. As long as this configuration is kept, the voltage of the battery will first increase while the current stays constant and, once the charging voltage is reached, the current that the battery takes starts to fall exponentially towards zero. The charge is commonly assumed to be over when the current drops to as low as 10 % or 20 % of the charging current. If charged with lower voltage, the CV phase starts earlier than normal, causing a reduced CC phase, and, with lower charging current, the CC phase is extended, reducing the CV phase. A CCCV charging curve is shown in figure 8. Logs to a full CCCV curve are shown in figures 9 and 10. As marked by Notten in his diagram, fig. 8, curve “(a)” represents the voltage curve, plotted against the current “(b)”.

Given the behaviour of the current when charging with constant voltage and given that the forward voltage on the diode is dependent on the current, some strategy must be implemented in the code to allow for a full charge of the battery, when desired.

A full charge brings another issue, which is the Battery Management System, BMS, going into sleep mode. This happens if the BMS senses no current going in or out of the battery in over forty (40) minutes. If the BMS sleeps, it blocks any current flow to and from the battery, so if the robot is charging, that is not an issue, since it will be powered by the charging station directly, but as soon as it leaves the charging station, the robot will shut down, as the battery will not supply any power. Hence the need for a charging strategy that

Figure 8 – CCCV charging curve.



Source: Notten, P. H. L. [3].

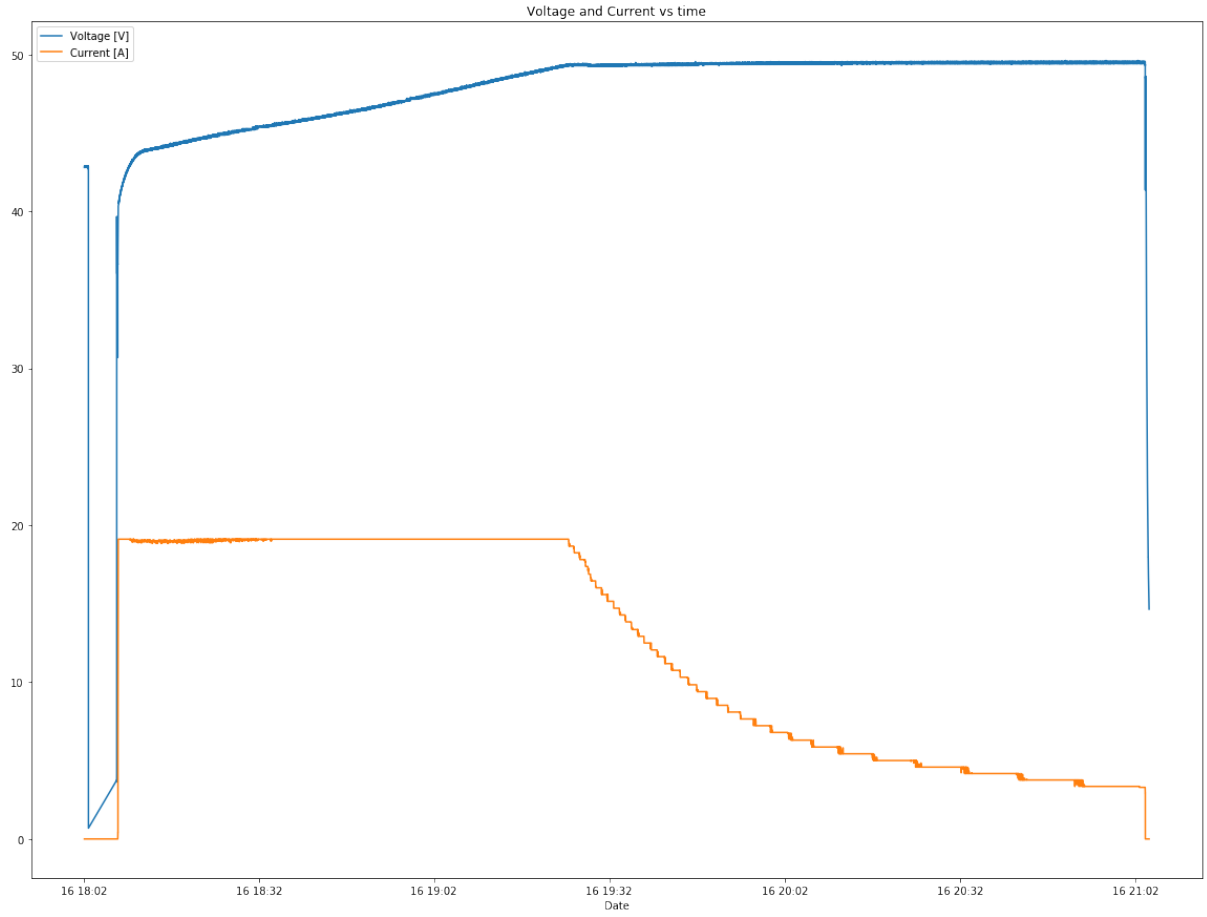
does not allow the BMS to sleep, knowing when the battery is fully charged.

Besides the hardware-related issues mentioned, the transfer of logs to the charging station also requires that the station can handle the logs and do something with them. That is, an approach must be thought out to send the data from the charging station to the central database, and an API, Application Programming Interface, must be designed to allow the robot to communicate with the charging station.

As of the communication itself, the possibilities must be studied and compared, including PLC. The solution decided to be the best should be implemented on hardware and software in a prototype, then tested. Then, the communication will bring new possibilities for charging, such as knowing robot ID and knowing exactly how much current goes to the robot and how much goes in fact to the battery. The possibilities should be mapped and changes should be proposed to the software team to be implemented on the arculee.

Ideally, after the prototype board for the communication is working without issues, it would then have its size minimised and design adapted into a final version of the board, which would be ready for production and to be mounted inside the arculees and charging stations.

Figure 9 – CCCV charging curve from logs, example number one.



Source: personal archive.

3.2 Analysis of Possibilities

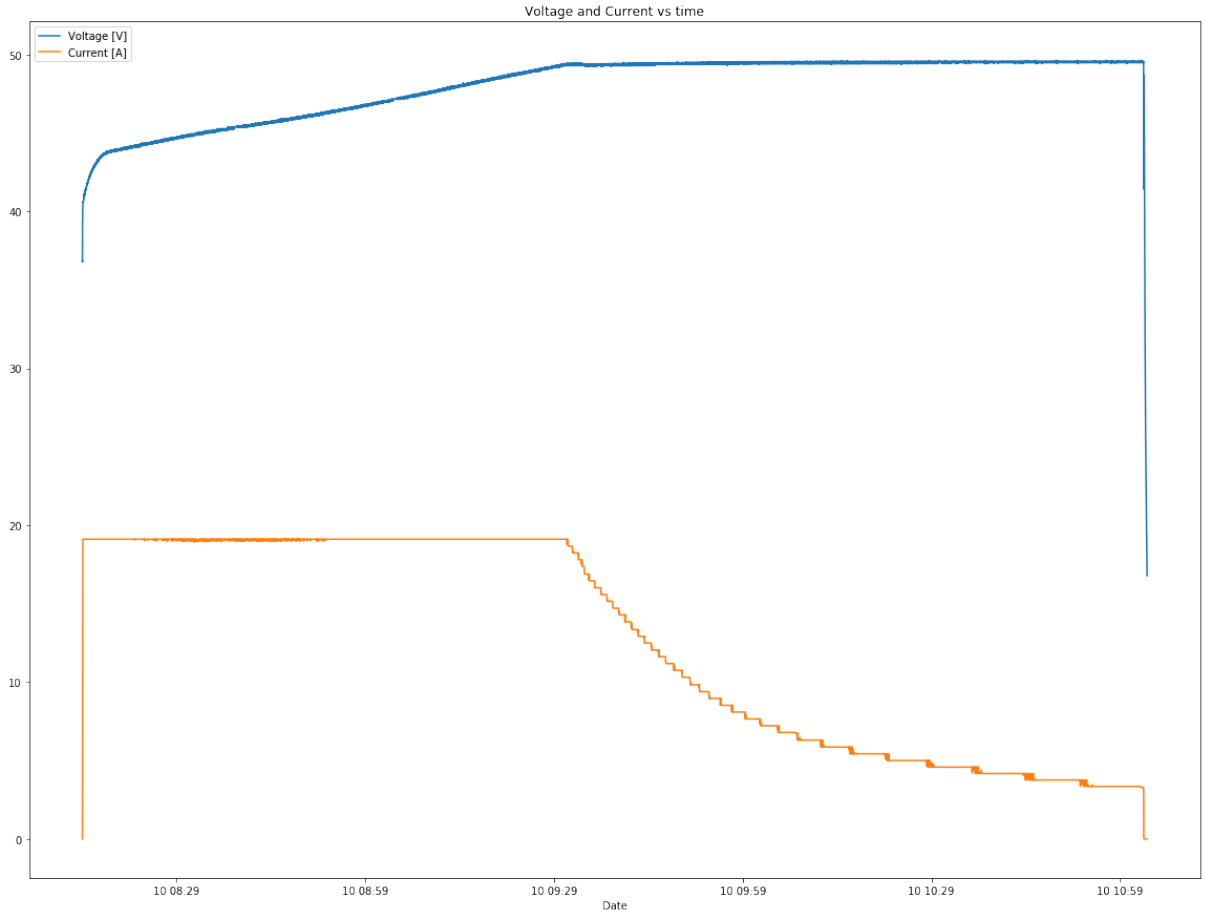
The topics will be broken into two main categories, namely communication and charging station. Communication will hold all topics directly related to the communication between charging station and arculee and Charging Station holds the remaining topics. In this section, the topics of work will have its range of solutions analysed, with its respective advantages, disadvantages, restrictions and important notes pointed out. In the timeline of this project, the charging station had its topics solved first and the works started on the communication side when the charging station was operating adequately.

3.2.1 Charging Station

3.2.1.1 Controller

The controller has not been defined by the student due to client requirements and to the options available in the company. The student had the job, however, of proposing requirements to the next controller.

Figure 10 – CCCV charging curve from logs, example number two.



Source: personal archive.

3.2.1.2 State of Charge Calculation

It was planned for the charging station to show the state of charge of the battery so the user would have an idea of the progress made in the charge process. However, computation of State of Charge, SOC, is not a trivial task. Batteries age over time, meaning they hold less charge as they are used. Ageing is non-linear and depends on charge and discharge behaviour. Assuming a battery can be charged with current of 10 A, charging with 1 A will cause it to age slower than it would if charged with 8 A. Additionally, steps in the current damage the cells, as well as charging with cold cells. These, among other points, add up and, the stronger, the more dramatic the ageing effect, i.e. a cell that is often exposed to current steps from 1 A to 10 A will age faster than a cell that receives steps from 1 A to 5 A.

The age of a battery also increases with its charge cycles. The battery used for the arculee has around 75 % of its full charge when it reaches around 500 full charge cycles, according to the datasheet of the cells. These effects are reviewed by Barré et al [4].

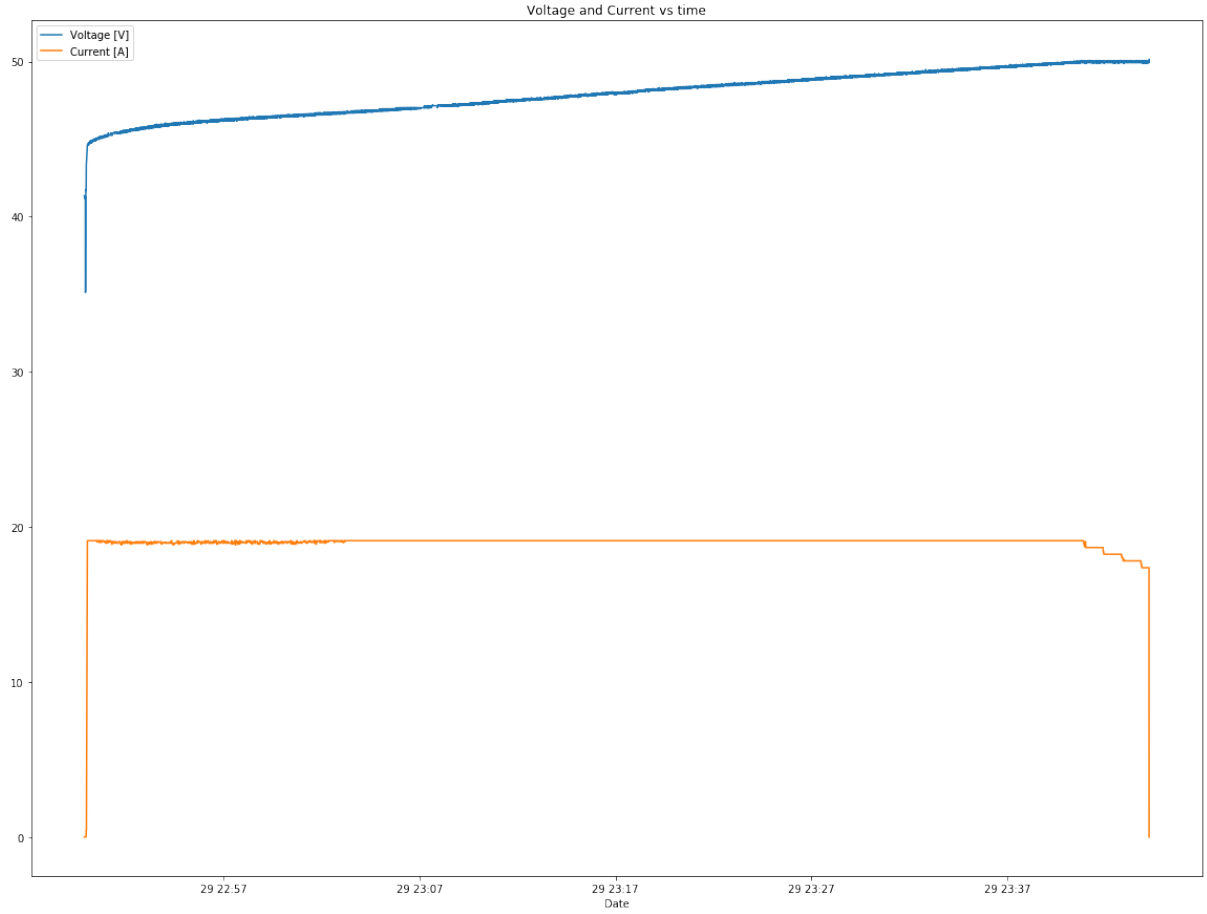
Given these behaviours, it is close to impossible for a perfect estimation of the SOC from the charging station without communication to the arculee. The charging station would

have to know the battery history or observe it through a couple charge cycles to compute the data. Such an approach is better implemented in the arculee than in the charging station, and so it has been decided in a meeting with the heads of the teams.

Knowing that the correct SOC computation will be implemented in the arculee and that its value will be exposed to the charging station via the communication, it has been decided that the station should receive a simple SOC estimation algorithm so it would show something on the screen to give the users an idea of the progress.

For the algorithm, the CCCV charge curves have been observed. A quick implementation of charging logs returned curves such as the ones in figures 11 and 12. Figures 13 and 14 show in greater detail the relevant curve in the CC and in the CV phases, respectively.

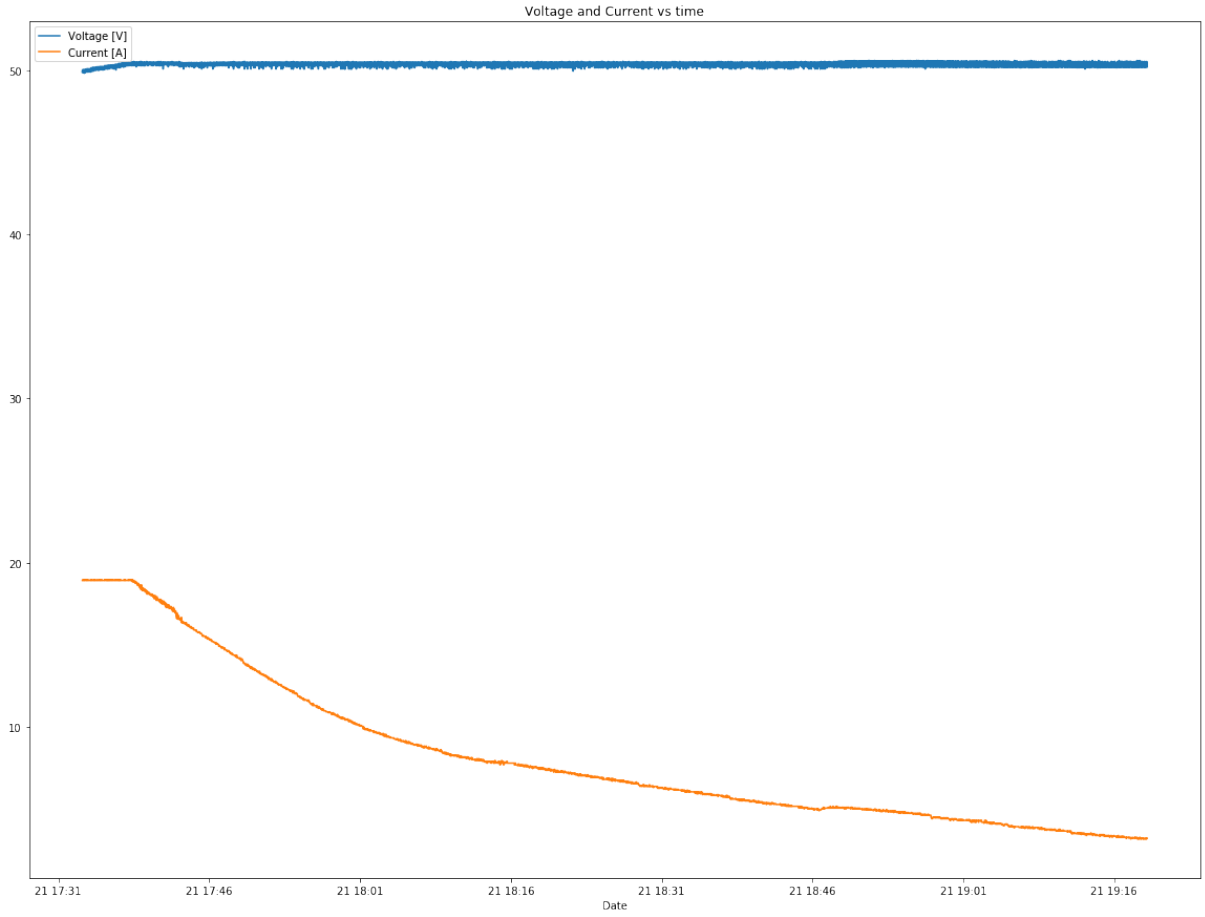
Figure 11 – CC curve with current



Source: Personal archive

From fig. 13 it can be noticed that the voltage increases linearly with time, given the exception of when the SOC is extremely low (left end of curve) or reaches CC phase (right end of the curve). This curve can be easily modelled with an equation such as eq. 3.1, with low error within the limits of the linearity. In the equation, U stands for voltage, a and b are

Figure 12 – CV curve with voltage



Source: Personal archive

parameters of the system, which need to be calculated, and t is time.

$$U(t) = a + bt \tag{3.1}$$

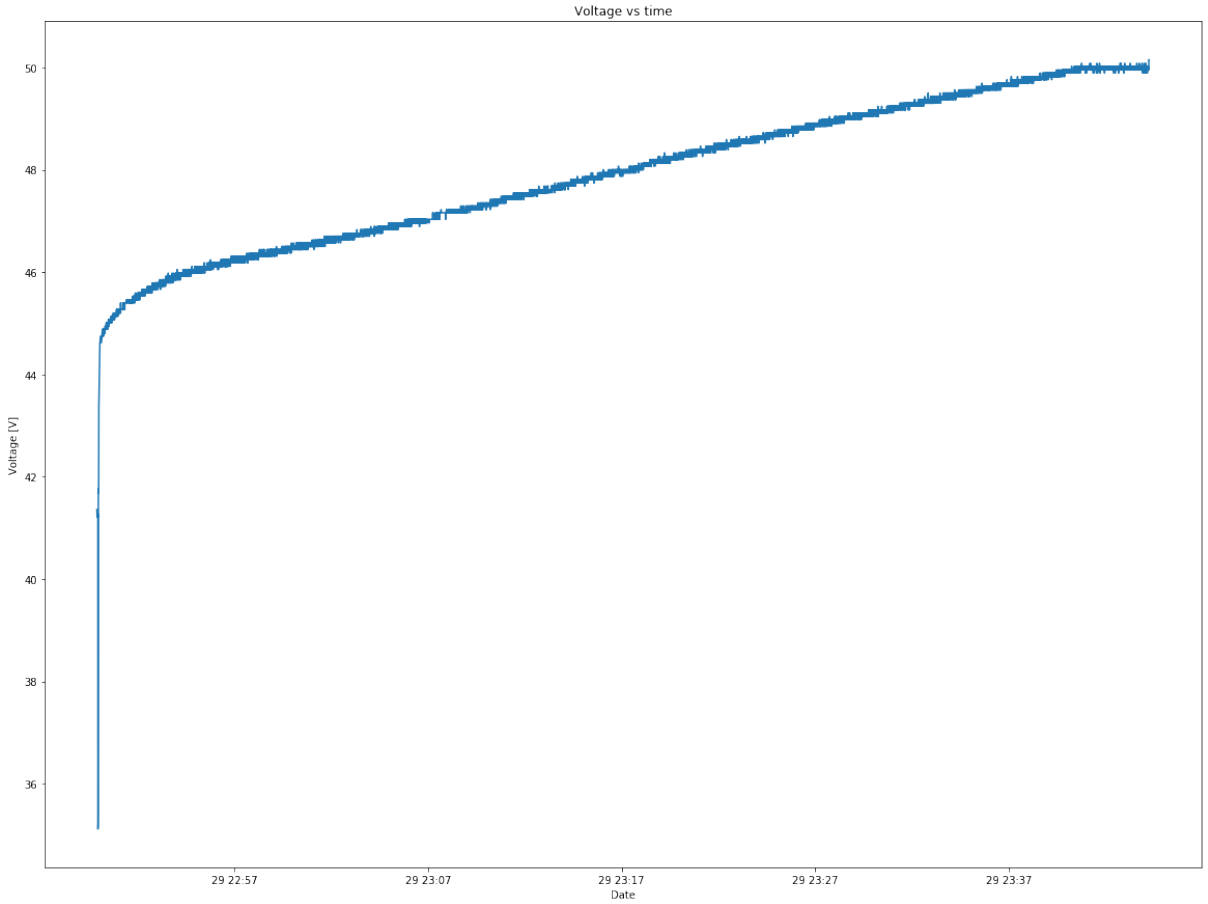
Since the current is known, the total charge C can be calculated by integrating the current I over time t, eq. 3.2.

$$C = \int_0^t I(\tau)d\tau = It \tag{3.2}$$

If the charge of the CV phase is known, C_{CV}, the SOC can be computed from eq. 3.3, where C represents the charge at a given time stamp t. The charge under CV phase, C_{CV} should also be known for SOC calculation.

$$SOC_{CC} = \frac{C}{(C_{CC} + C_{CV})}100\% \tag{3.3}$$

Figure 13 – Voltage under CC



Source: Personal archive

Since the voltage U and current I are known and the parameters a and b are previously calculated by fitting the graph to the model, the time t since the beginning of a hypothetical full charge can be calculated from eq. 3.1 and then the charge C can be calculated from the time t with eq. 3.2. So the SOC equation can be written from equations 3.1 and 3.2 as in eq. 3.4.

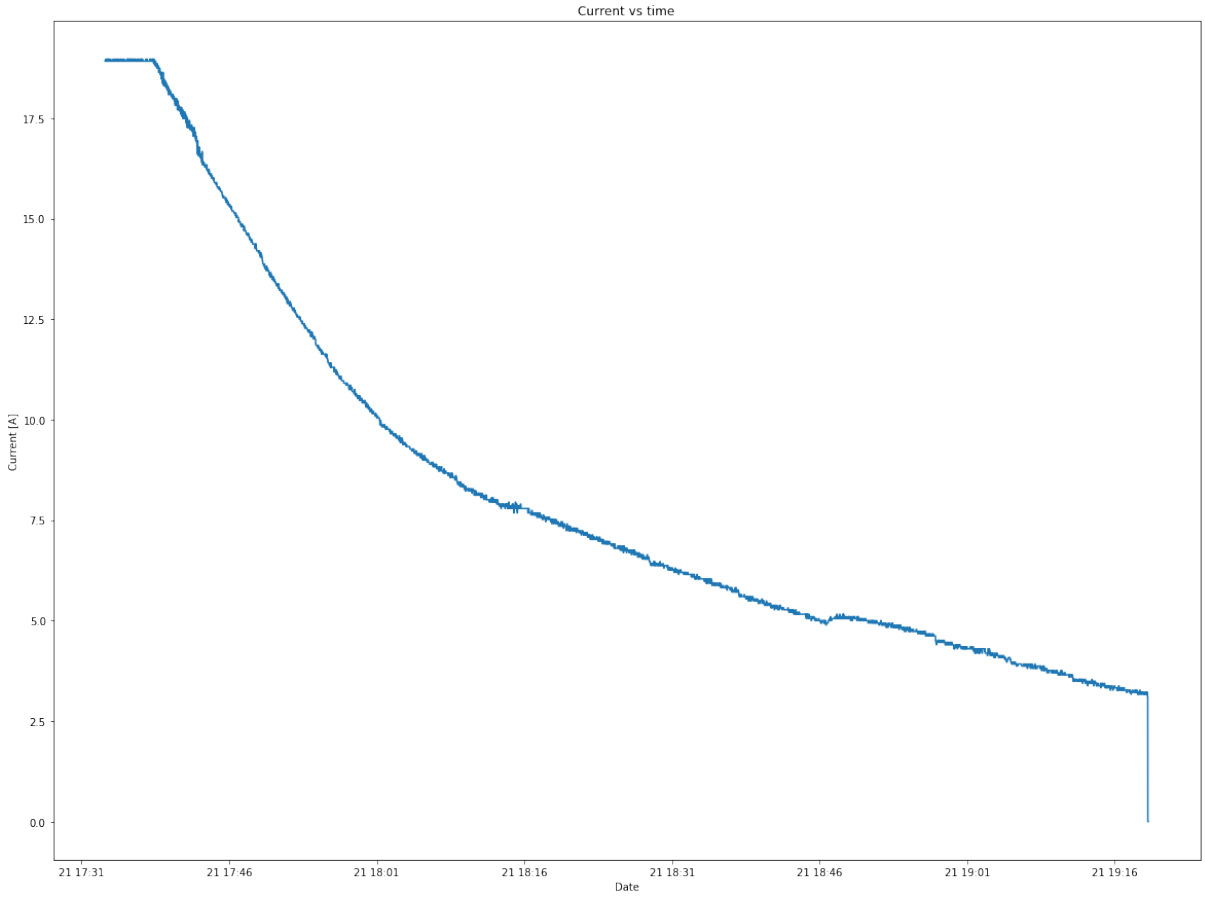
$$SOC_{CC} = \frac{I(U-a)}{b(C_{CC}+C_{CV})} \tag{3.4}$$

Now for the CV phase, fig. 14 shows that the current I falls exponentially over time t. This curve can be easily modelled with an equation such as $I = k_1 + k_2 e^{-k_3 t}$, or, for more flexibility and lower error, eq. 3.5 applies, where k_n represents coefficients to be calculated by fitting log data to the model.

$$I = k_1 + k_2 e^{-k_3 t} + k_4 e^{-k_5 t} \tag{3.5}$$

Similarly to the CC phase, the charge under CV can be calculated from the integral

Figure 14 – Current under CV



Source: Personal archive

of the current curve, which is given by eq. 3.6.

$$C_{CV} = \int_0^t I(\tau)d\tau = k_1 t - \frac{k_2}{k_3}e^{-k_3 t} - \frac{k_4}{k_5}e^{-k_5 t} + \frac{k_2}{k_3} + \frac{k_4}{k_5} \tag{3.6}$$

And with the charge at a given time t, the SOC can be calculated from eq. 3.7.

$$SOC_{CV} = \frac{C_{CC}}{C_{CC} + C_{CV}}\left(k_1 t - \frac{k_2}{k_3}e^{-k_3 t} - \frac{k_4}{k_5}e^{-k_5 t} + \frac{k_2}{k_3} + \frac{k_4}{k_5}\right) \tag{3.7}$$

However, the inverse of equation 3.5 is required to find the exact time t since start of the CV phase. This could be calculated if the charging always started in CC phase, but that is not always the case, so a look-up table is calculated to get an estimate of the time under CV charge t from the current. The precision of the current is very limited, it increases and decreases in steps of about 0.065 A, which brings errors to the estimation. These errors are specially meaningful when nearing the end of charge, where the exponential curve gets flatter regarding the current axis. The error is increased due to the current that feeds the robot, current which does not go to charge the battery. This current has been observed to

revolve around 1.2 A, with maximum around 2 A and minimum around 0.7 A. Given these values, the current that is used for the calculations must have the arculee current subtracted from it.

Two more strategies were proposed to reduce the error, first being the implementation of a Kalman Filter. It would help filtering out the measurement steps, but would not bring much regarding the arculee current. The second strategy proposed calculates the SOC from the voltage. From the CCCV charge model and the open circuit voltage curve of the cells, which is known from the datasheet, a voltage difference curve can be calculated with respect to time and, based on an initial SOC estimation, the actual, correct SOC can be calculated within some iterations by measuring the voltage, adding a voltage difference from the curve and matching it against the datasheet curve. With some signal processing based on expert knowledge of the system, as well as on results from tests, the error caused by the robot current should be reduced by a considerable margin and the current precision would not be an issue.

3.2.1.3 Time Left

Time left refers to the time left from the current point in time until the estimated end of charge. For such, it has to be calculated from some sort of progression measurement, which, for the charging station, is the SOC. Fortunately for the calculations, the computation of the SOC itself is dependent on charge time and a thorough model of the charge process is required. This means that the time left must be calculated from the battery model, along with the SOC. Given that, the strategy chosen for the time left will mirror the strategy chosen for SOC estimation.

3.2.1.4 Start Charge with arculee Only

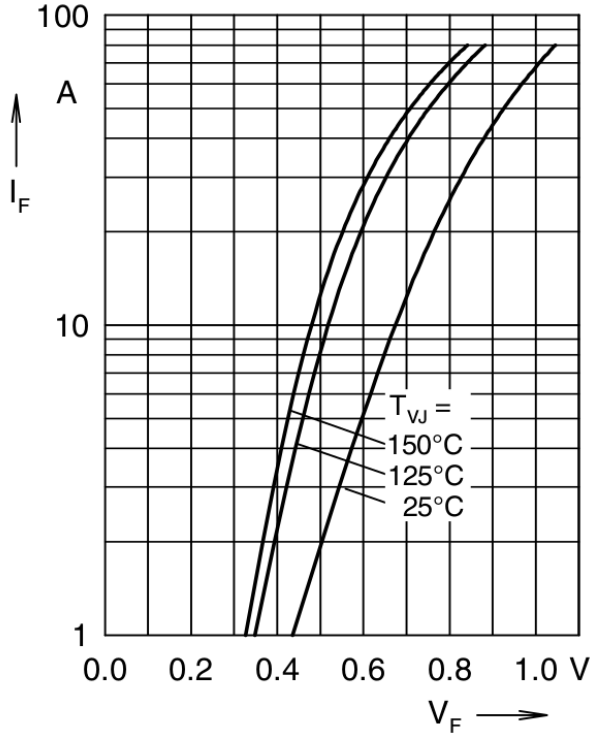
In order to start the charge only in case the arculee is connected, the charging station must look for characteristics of the arculee in the environment before charging. Options include looking for its shape and colour, battery voltage and metal casing. Since hardware changes to the robot and to the charging station must be avoided, a camera or sensor for shape and colour recognition are not available option, neither is a sensor for the metal casing. This leaves one option left, which is checking the battery voltage before charging.

3.2.1.5 Voltage Loss on Diode

There is a safety diode placed before the charging contacts on the robot, which prevents the battery from supplying energy through the charging contacts, for safety reasons. This diode, however, includes a voltage drop on the power supplied by the charging station before it reaches the battery. To be able to fully charge the battery, this voltage must be accounted for on the charging station logic.

The datasheet of the diode presents the diagram shown on fig. 15 to the user.

Figure 15 – Forward voltage on Schottky rectifier.



Source: IXYS, datasheet for model DSS 2x41-01A

The temperature on the diode can reach over 25°C, so the curve for 125°C should be read. The easiest and fastest option to implement it is via a look-up table, where the forward voltage can be obtained from the current. The other option being to make a fit of an equation to the curve. In both cases, a safety margin should be given for the arculee current, measurement error and small deviations from translating the diagram into the table.

3.2.1.6 End of Charge Detection

The strategies can be broken into two groups, one for full charge and one for stopping before the battery is full. Both are important and would be eventually implemented, as the battery must sometimes be fully charged.

The strategies to stop the charge below 100% SOC are basically dependent on the current and on how long the robot should in fact be charging. The stopping current influences the SOC that the robot will have when it leaves the charging station. The time not charging can be adapted. On the one hand, charging too few causes the robot to run out of battery fast. On the other hand, the closer to 100% charge, the higher the risk that the BMS will shut down.

For an idea of dependency of the time of charge on the stop-charge current, the following estimation has been carried out. As a note, when these estimations were made,

the logs went down to 3.15 A only. Every value after 200 min or 3.15 A is an estimation extrapolated from the CCCV model extracted from the data.

A full charge with 19 A takes around 200 min, from 20 % robot SOC down to 3.15 A, around 96 % SOC, a initially chosen stop-charge value. Then, a reduction from 3.15 A to 2.2 A adds 40 min to the total charge time, making it 240 min, or 4 hours. Further reducing it to 1.5 A would add around 190 min to the charge process, requiring 430 min or 7 h 10 min for the whole charge cycle.

The other group of strategies refer to charging the arculee fully, without allowing the BMS to sleep. One of the possibilities is to detect when the current that goes to the battery stopped decreasing and, then, stop charging. This could, in principle, be implemented by observing the standard deviation of the current and if it drops below a threshold, the current could be assumed to be steady. The issue with this approach is that the current is never really steady due to the variation of the current that the robot needs, which oscillates with an amplitude of approximately 0.6 A around an average. This is aggravated by the end of the exponential curve, which may take over half an hour to decrease one 0.06 A step of the current.

Another possibility would be to make short stops in the charge when the current is considered dangerously low for the BMS, for instance, after charging 30 min under 2 A, a short pause is made, before the charging station restarts charging. This approach is independent from the current, but requires a slightly more sophisticated logic to be implemented. The problem with this approach is that, at some point, the time not charging should consume more charge than it would be charged over 30 min, so a true 100 % charge would be impossible, although, theoretically, the current that flows to the battery would never be zero due to its exponential behaviour.

3.2.2 Communication

3.2.2.1 Hardware

For the communication hardware, the first possibility considered was infrared light. Due to its hardware restriction, the advantages would have to be considerable over the others for it to be the chosen strategy. However, high speed transceiver modules are rare and expensive. Most of the modules are limited to 4 Mb/s with the protocol Infrared Data Association, IrDA. Another option would be ultraviolet, but the technology is mostly in development phase still, with a few uses for military [5]. With these options out, the transmission is restricted to physical a medium.

Given that the communication is restricted to a physical medium and, therefore, should be transmitted over two cables, the protocol 100Base-T1 suggested comes into question. After some research, the technology HomePlug AV2 [6] has been found and, with

it, data rates of up to 1 Gb/s could theoretically be reached. No commercially available 48 V transceiver has been found for the application that would work with the required current. Some Integrated Circuits, ICs, were found that can build a link for the communication, but with no coupling interface nor physical layer for the normal RJ-45 connector.

Besides that technology, the other main option would be USB or serial bus. To send such signals in two cables, one option would be to send the voltage over the charge contacts. This would work for USB 2.0 as it is half-duplex and, with transmission rate of 480 Mb/s, it would be fast enough. The problem again comes with the products available, as there are not many producers of USB over PLC and the products available still require a coupling circuit.

Other protocols are not as fast or would require hardware changes, so the possibilities were limited to these due to the satisfying possibilities available. Still, the student proposed one solution with analogical processing of the USB signal with coupling to the power-line. The removal of the digital processor would reduce costs and delays. The requirements were high and the development would not be simple, but, if successful, the simplicity of the solution would be beneficial for future development.

3.2.2.2 API Between ARC and CS

The API between the arculee and charging station was set to use JavaScript Object Notation, JSON, by the software team. The messages required are to be decided by the student depending on the needs of the charging station. The possibilities regarding the content are bound to what the robot can measure, that is all of its relevant internal values, such as currents, voltages, ID, histories.

3.2.2.3 Protocol Between CS and Log Database

A protocol should be chosen for the charging station to deliver the logs to the log database.

In a development scenario, where the arculee is running on a regular Wi-Fi connection, the logs can be sent over Wi-Fi. In this state, log saving is realised on the robot side with the command *rsync*, which transfers files from one Linux machine to another.

On the shop-floor, the strategy could be applied to the charging stations, which could be connected to a database or server via an Ethernet cable. Another protocol could be implemented, such as RS-485, which is, in principle, serial for multiple peers. Ethernet has the advantage of being flexible and not requiring changes to the database, with high data rates and most of the other points that RS-485 offers, with higher flexibility, e.g. it allows the creation of sub-networks. RS-485 is more robust than Ethernet for its lower baud rate and it has a deterministic behaviour.

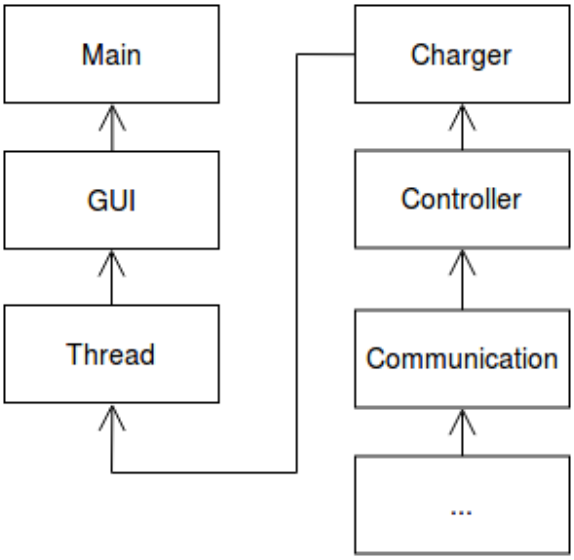
3.2.2.4 Software

The charging station software was initially running on Python with an user interface running on the framework Kivy. It was running on two threads, one for the Graphical User Interface, GUI, and one for the state machine.

Due to the amount of software changes required for the communication and due to the initial, disorganised state of the software, a full redesign was proposed. The software is mostly based on two parts, one state machine, that exchanges data with the serial bus and with the GUI, and the communication-related modules, for the serial busses.

What made the software so disorganised was mostly the lack of modularisation, where, for instance, the whole state machine was implemented in one long method. This was likely implemented as it was in order to avoid dealing with data exchange between the states and the communication modules, as the states were inside a single main controller class. A simple diagram showing class hierarchy within the program is shown in figure 16. Main started the program GUI, which started the charger thread, with the Finite State Machine, FSM, which will be initialised and updated by the Charger module. The code for the FSM lies in the Controller module, which would then also hold the communication-related objects, for serial and Ethernet.

Figure 16 – Simplified initial class diagram from charging station software



Source: personal archive.

The problem in the design is that the Controller becomes a so-called “God Object”, mostly considered a bad practice in programming [7]. Even though this is not unanimously seen as a poor coding practice [8], the lack of modularisation causes the need for several little changes in the code every time it needs to be modified, which leads to several errors and issues. Given that, the next design would have to hold a state machine that would easily

take new states, while allowing for easy modifications to the existing states. It would also have to be better modularised for improved readability, version control and to make it easier to find and fix errors. The new design should also consider the whole data flow, making data from serial bus easily available to the display.

The code was somewhat based on a Model-View-Controller architectural pattern [9], as it would implement the Observer pattern [10], but the function called to update the data on the model, which would be the FSM, was the function to update the state, which would run on new data. Thus, no actual observer existed, as the model and the controller were on the same module, Controller, which would update the view, GUI.

One option for the new design would be to keep the same base structure, with a controller, a FSM, the communication modules and the same base design pattern, only more organised. This would allow the reuse of most of the code, reducing the need for writing and testing. Another option would be to fully redesign the code and reuse only the pieces of code that apply. However, a design that goes too far from what was implemented would not add much to the project, but cost plenty of time. The end result could be more comprehensible, but at the cost of time for planning, programming, testing and fixing the code.

From the behavioural design patterns, the Mediator pattern [11] is worth mentioning for its similarity to the Observer pattern, as it would facilitate the implementation, while organising the code and braking the God Class into a few classes, e.g. one Mediator between state machine and communication modules, and one supervisor for the FSM, plus the states and some side modules such as a SOC estimator.

Once implemented and tested, the design could be further improved in a couple aspects. For instance, the communication could be run in parallel to the charger thread, reducing waiting time in each iteration. Furthermore, an Ethernet communication class, thought to hold the communication to the robot, would have to be adapted to the chosen communication strategy and API. The charging station should also communicate internal errors and failures to the system supervisor, FABMAN, so no arculee gets sent to a defective charging station, this is dependent on the technology chosen by the software team. At last, Kivy is not the most robust GUI framework available for Python, it is mostly used for smartphone touch applications. Other user interface frameworks should be considered, such as PySide and PyQt, PySide has a licence that allows for commercial use, while PyQt does not. Coding for both is very similar and both are well maintained.

3.3 Chosen Approaches

With the problem presented in section 3.1 and the possibilities of solutions explored in section 3.2, in this section the chosen solutions will be presented for each individual topic along with the reason why they were chosen.

3.3.1 Charging Station

This subsection will present the choices made for the charging station.

3.3.1.1 Controller

One controller should be chosen to handle charging while using the display. After the software team and clients discussed the requirements on the controller, no agreement has been found regarding one that would be allowed to access the industrial network on the site. Thus, given the display support, mentioned in section 3.1, and given the availability in the company, the Raspberry Pi 3B has been chosen to temporarily substitute the Banana Pi for development. The display works on it and the same code could be kept.

3.3.1.2 SOC Calculation

Initially decided to be left with considerable uncertainties due to time constraints, the SOC calculation was based on the first, basic model for a while, whose equations were presented. It has been decided that the most accurate SOC calculation should be implemented on the arculee, which would pass it on to the charging station when charging. So, it is pointless to spend more time implementing a sophisticated solution for the SOC estimation on the charging station. When more data had been gathered, the algorithm that calculated the model was improved to generate a better model from the new data, reducing the error with little time spent.

3.3.1.3 Time Left

Given the choice for the simple approach for the SOC calculation, the time left can be easily calculated by knowing duration of the CC and the CV phases, which can be extracted from log data. The time left would come from the eq. 3.8, where t_{CC} and t_{CV} represent time for a full CC charge phase and time for a full CV charge phase respectively, and t represents the time that it would take to charge up from 0 % to the current SOC, which comes from the inverse of equations 3.1 and 3.5.

$$T_{left}(t) = \begin{cases} t_{CV} + t_{CC} - t, & \text{if charging CC} \\ t_{CV} - t, & \text{otherwise} \end{cases} \tag{3.8}$$

The error of time left estimation is similar to the error of the SOC estimation due do its dependence on the model used for the SOC, which will vary according to several factors such as battery ageing. The better the model, the better the estimation.

Finally, due to its connection to the SOC, this would eventually be calculated on the robot in future implementations after the most accurate SOC estimation algorithm.

3.3.1.4 Start Charge with arculee Only

Being the only option, the function must be implemented based on the voltage measured from the battery before charging. When the robot connects, a minimal threshold must be set, that would already differentiate a robot battery from other objects.

Due to the circuit formed by the charging station and the robot electronics, the voltage measured is dependent on the voltage of the power supply. So, when the robot first connects to the charging station, the voltage read is around 3.5 V, instead of over 40 V, which would be expected from the battery in an open circuit. Then, an initial matching voltage step is done to read the actual voltage of the battery, in which an iterative approach would be implemented, where the voltage will be set to the voltage read until the difference between them is little enough. Once this is achieved, the voltage is checked against an expected voltage level and, if the voltage is not within the preset thresholds, the charge start up is halted.

The threshold voltages are set to charge voltage and to the minimum voltage observed, with a tolerance margin.

When testing for implementation of this function, connecting another battery, e.g. a 9 V battery, would cause the voltage to be always read at 9 V, independent from the voltage at the power supply.

3.3.1.5 Voltage Loss on Diode

Since both solutions available would bring to the same result, a look-up table has easiest implementation as it would also be required in order to obtain a model. Plus, the model may have fitting errors in relation to the curve, while a well-built table requires no fitting. The table should be implemented following the voltages for the 125°C curve and the voltage must only be added while charging. The implementation must undergo an observed test as an over-voltage would cause the BMS to isolate the battery from it, turning the robot off when it leaves the charging station.

3.3.1.6 End of Charge Detection

For the end of charge below 100 % SOC, the options are limited to the current when the charging stops, as this is the only parameter changing in the charge process. Initially a charge down to 3.15 A has been set as enough charge for the robot, as further charges do not increase the SOC as much and would take too long.

Regarding the full charge approaches, given the fluctuation of arculee current, the short stops are seen as ideal for avoiding risks of having a robot shut down while charging. Even though the charge would not go to a true 100 %, if the stops are short enough, the BMS is capable of handling the balancing itself. In this case, the most robust approach must be chosen as the risk of losing a robot must not be taken.

3.3.2 Communication

This subsection will present the choices made for the communication.

3.3.2.1 Hardware

Initially, independent of the protocol, an analogical transmission of the signal would be tested in simulations. Should the results be unsatisfactory, the approach should be changed to building a coupling circuit that handles ideally the HomePlug AV2 standard or, if not possible, the standard 100Base-T1. An Ethernet port is available on each robot and on each charging station and could be used for the purpose of this project. Furthermore, the German company I2SE GmbH offers HomePlug AV2-compliant PLC modules and presents a coupling circuit with evaluation kits. The evaluation kits would work as a base for the design and it would be possible to make a proof of concept with the boards. There are a couple more options available that do the same, but this was chosen for being made in Germany and for the data available from the modules in the datasheets.

3.3.2.2 API Between ARC and CS

With Ethernet connection between arculee and charging station, the software team proposed that each charging station should host a local server. It would receive REST, Representational State Transfer, commands from the robot and handle them. Given that, the charging station profits from knowing the robot ID, battery current, battery temperature, SOC, time left and charging voltage and current. More information regarding the battery can be passed as well, if required for fast charging different battery models.

As defined by the software team, the REST API should be based on JSON.

The charging station should also be able to charge without communication, in case there is some problem with the module. This brings two different types of current data to be handled by the charging station, one current only available with communication, which is the current that flows to the battery only, and one current available from the AEK, which is the one that is shared by the battery and the arculee electronics.

3.3.2.3 Protocol Between CS and Log Database

The path will be used primarily to send logs, considered low-priority data, the Ethernet approach is implemented on TCP, which has a checksum and packet repetition in case of error, and a deterministic behaviour does not bring much to the system in general, as there should not be plenty of charging stations per site and the data has no latency requirements. From these points and given that Ethernet is has higher transmission rate than UART, the protocol has been chosen.

That being, the same approach should be implemented on the charging station, using *rsync* to transmit the logs. This choice can be reconsidered in the future.

3.3.2.4 Software

Regarding the redesign, the chosen pattern was the Mediator pattern, which has an advantage in this case for its similarity to the code that was already implemented. The code was working, so any change should be as small as possible and should take as few time as possible. Still, the changes should allow for scalability and bring modularisation. The observer pattern would require more time for redesigning the data flow inside the code and the required updates, so it should be avoided. Other patterns require plenty of code changes and, therefore, plenty of time. For the state machine, the State pattern [12] should be implemented.

The server, required from section 3.3.2.2, should be programmed using Swagger, a tool for generating servers based on REST API and joined to the code, for it is the tool used by the company. Any time-consuming communication should be run in parallel to save time and be able to iterate faster in case of errors or unexpected situations, increasing safety by reducing risk of electric shocks. The Ethernet communication class can be updated to handle the interface between server and the rest of the program. A module responsible for handling and reporting errors must be included and made available to third party software like the FABMAN. At last, Kivy has been decided to be kept until further notice. It should be changed to PySide in the future for its robustness and license, but the change has lower priority than this project.

In meeting with the software team, it has been decided as well that not everything should be reported as error, so alerts should be sent to indicate state changes such as software being closed, start-up done, charging started, charging done and the transition between CC and CV charging phases. A heartbeat should be regularly sent as an alert with time left to charge. Missed heartbeats mean that the charging station stopped operating. The alerts should be sent using MQTT, Message Queuing Telemetry Transport, a lightweight messaging protocol developed for internet of things, based on a broker approach with channels that peers can subscribe to. Errors, on the other hand, are set to be sent as REST requests with JSON content to the supervisory system.

4 Methodology

After the smaller topics being thoroughly covered in chapter 3, this chapter should focus on the main topics for the project, namely the hardware for the PLC, the software redesign and the API between arculee and charging station.

4.1 Hardware

Chosen the power-line communication, initially a couple weeks were given for the student to develop a simulation and evaluation of a circuit design without digital processors. The future path of the hardware project would then be chosen according to the success of the simulation.

4.1.1 Analogical Board

For the analogical board, some designs and implementations of PLC were studied. From them, it could be noted that some points deserved attention, namely filtering, coupling strategy, amplification and signal modulation.

The idea of the implementation was to be as simple as to send a differential signal over a power and a ground, so no specific modulation would be implemented on hardware. Regarding filtering, low-pass circuits should be avoided and, should that not be realistic, the cut-off frequency should be over ten (10) times the baud rate, planned at around 100 MHz, ideally over 200 MHz for some margin to ensure a good-quality signal.

Both Ethernet and USB implement normal pulses to send the signals, so a square wave. Knowing that and following the 200 MHz frequency goal, all components must be able to switch fast enough so that the signal reaches a steady state before the middle of each pulse. This increases the probability that the microcontroller will sample the signal when it is already steady. Delay, on the other hand, is not a problem, as it will simply shift the signal in time, but keeps its shape. With this frequency requirement, any component that may alter the signal, such as operational amplifiers for signal amplification, would need to reach a steady state in under half the period of the signal. Given the frequency requirement, the components would, ideally, need to reach steady state in under 2.5 ns, eq. 4.1. Another important implication brought by this requirement is a high slew-rate, meaning that the components would be required to change their output voltage in over 5 V in a time as short

as 2.5 ns.

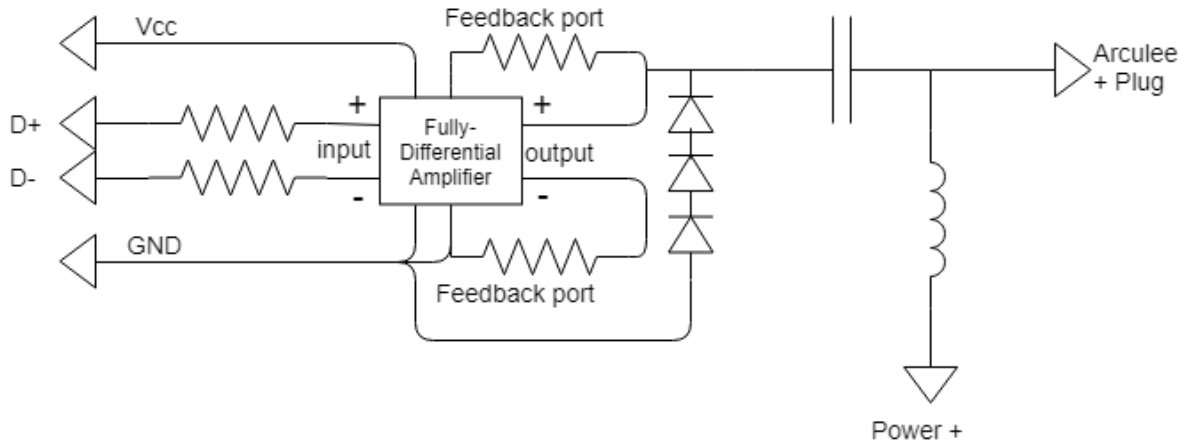
$$T = \frac{1}{f_{sw}} = \frac{1}{0.2 \times 10^9} = 5 \times 10^{-9} = 5 \text{ ns} \rightarrow \frac{T}{2} = 2.5 \text{ ns} \tag{4.1}$$

Regarding coupling, initially an inductive coupling [13] has been designed to work according to the frequency requirements, but the signal failed to be coupled onto the DC power-line. Then, a capacitive coupling [13] has been designed, again watching for its frequency requirements, but the signal again would not be coupled on to the DC power-line. Then, a bias T circuit [14] has been implemented to couple the signals, with satisfactory results.

Some of the circuits tested to reach the final design include the proposed by the scholars in [15], [16], [17], [18] and [19], but the best result still came from the bias T solution.

After testing the coupling strategies, the goal was set to generate one single-ended signal from the differential signal in the port. This signal would be coupled to the positive DC power-line and decoupled on the other end. The coupling would be done via a bias T circuit. A simplified diagram representing the idea is shown in fig. 17.

Figure 17 – Simplified diagram for signal coupling.

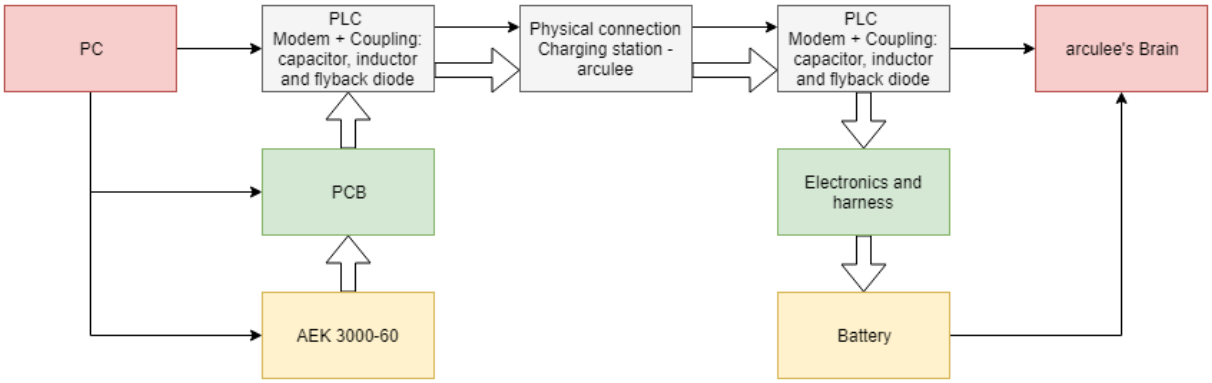


Source: personal archive.

After this design, the whole picture would look as shown in fig. 18, where the thick arrows represent high-current power-line and the thin arrows represent communication signals.

Knowing what would be done, the actual simulation could be started. The first test, fig. 19, simulated on the software PSIM, shows a simplified idea of the system. On the left side, the charging station circuit can be seen. On the top left lies the transmitting end, bottom left, the receiving end. In the middle, the power-lines are represented, with the power supply on the charging station side and the battery on the right side, close to the circuit that would be

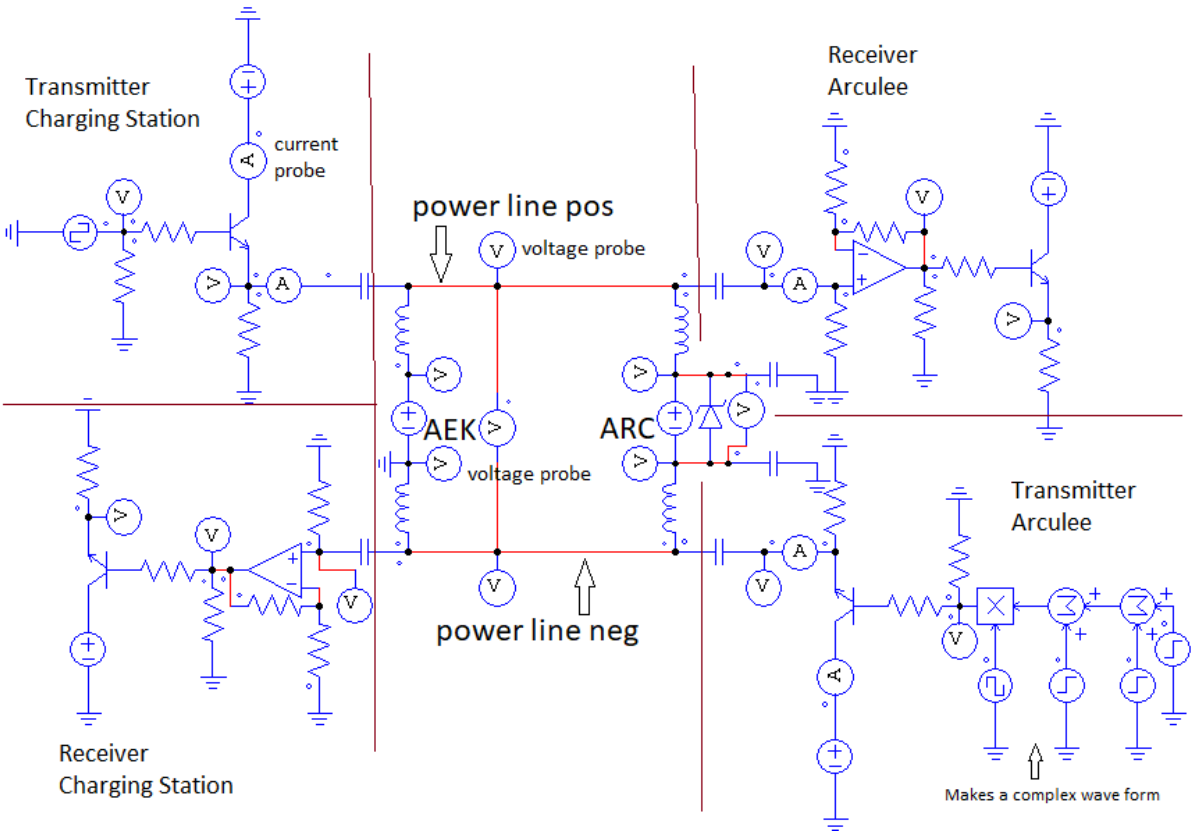
Figure 18 – Power and data flow diagram on a bias T-based system.



Source: personal archive.

placed on the arculee. On the top right, the receiving end for the arculee and on the bottom right, the transmitting end, with some signal mathematics to generate the transmission of a more complex signal than a simple square wave.

Figure 19 – First version of test simulation with bias T circuit.



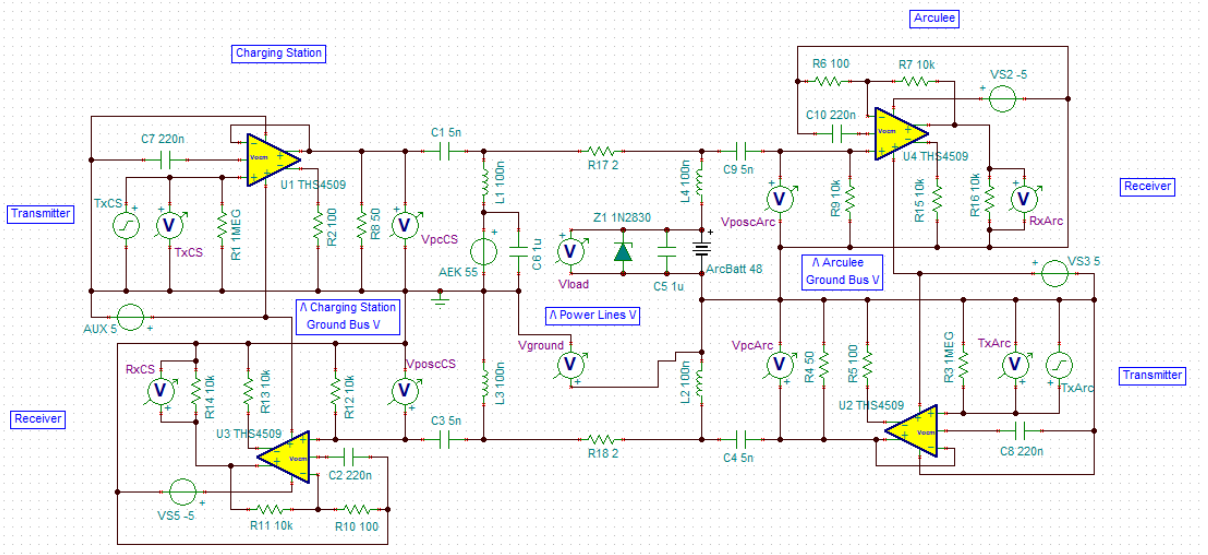
Source: personal archive.

On the transmitting end, first the signal is amplified by a MOSFET, so the power required for the signal transmission does not come from the processor. The signal is then coupled by the bias T circuit to the power line and reaches the receiving end. There, a

bias T makes the decoupling, sends the signal to a high-speed operational amplifier for amplification and this signal has then its voltage level adjusted by a MOSFET switch to be read. This design has shown promising results on PSIM, but the software was not simulating real components nor the real system, it worked simply as an initial proof of concept.

The following simulations were developed in the TINA-TI SPICE software, an electronics simulator developed by Texas Instruments that has the components from the brand. Since high-speed operational amplifiers were required, Texas Instruments offers good options for the chips and, therefore, their software was used. The second design was then proposed, fig. 20. Similar to the other simulation, the signal coming from the processor, marked with "Tx" on the transmitting end, would be adjusted by a buffer operational amplifier. This would then be coupled onto the power line, now with a resistance simulating the real system, and then decoupled and amplified on the receiving end. Other changes added to this simulation are the output capacitor on the power supply, C6, and the arculee having a different ground than the charging station, as the robot is not grounded. The figure is shown in greater detail on figures 21 and 22.

Figure 20 – Second version of test simulation with bias T circuit.

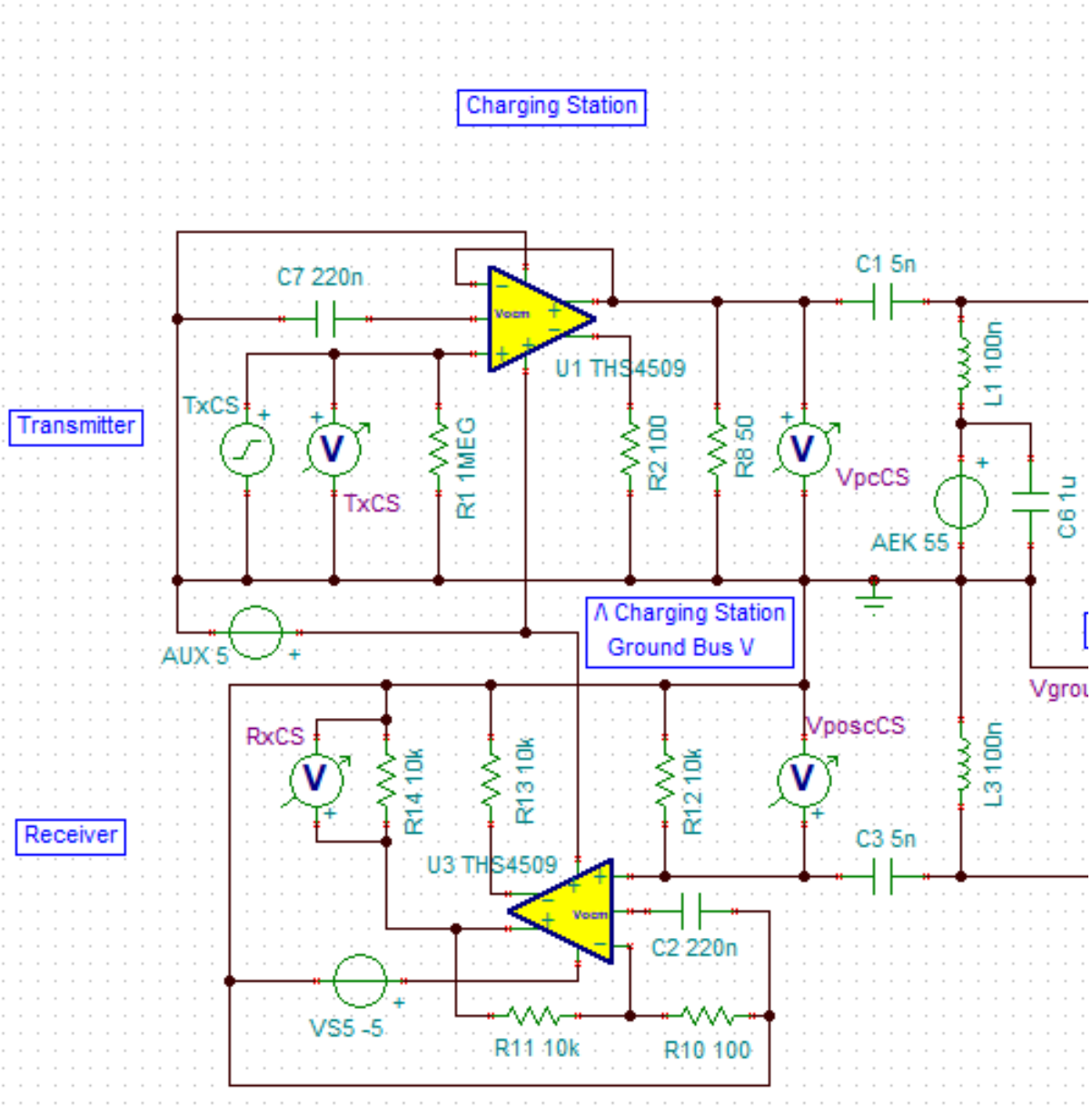


Source: personal archive.

This circuit presented good results with a synchronised input of Tx on both ends. If the inputs were not synchronised, the output on the receiving end would not reflect what had been transmitted. This was mainly caused by the ground difference, as the removal of the power-line resistance would remove the effect. A successful transmission is shown in figure 23.

However, several implementations of the literature use inductive coupling and it was suggested by the electronics expert, plus most commercial applications send both signals on the same bus, leaving the ground at 0 V, which could reduce the interference on an

Figure 21 – Second version of test simulation with bias T circuit, left side.

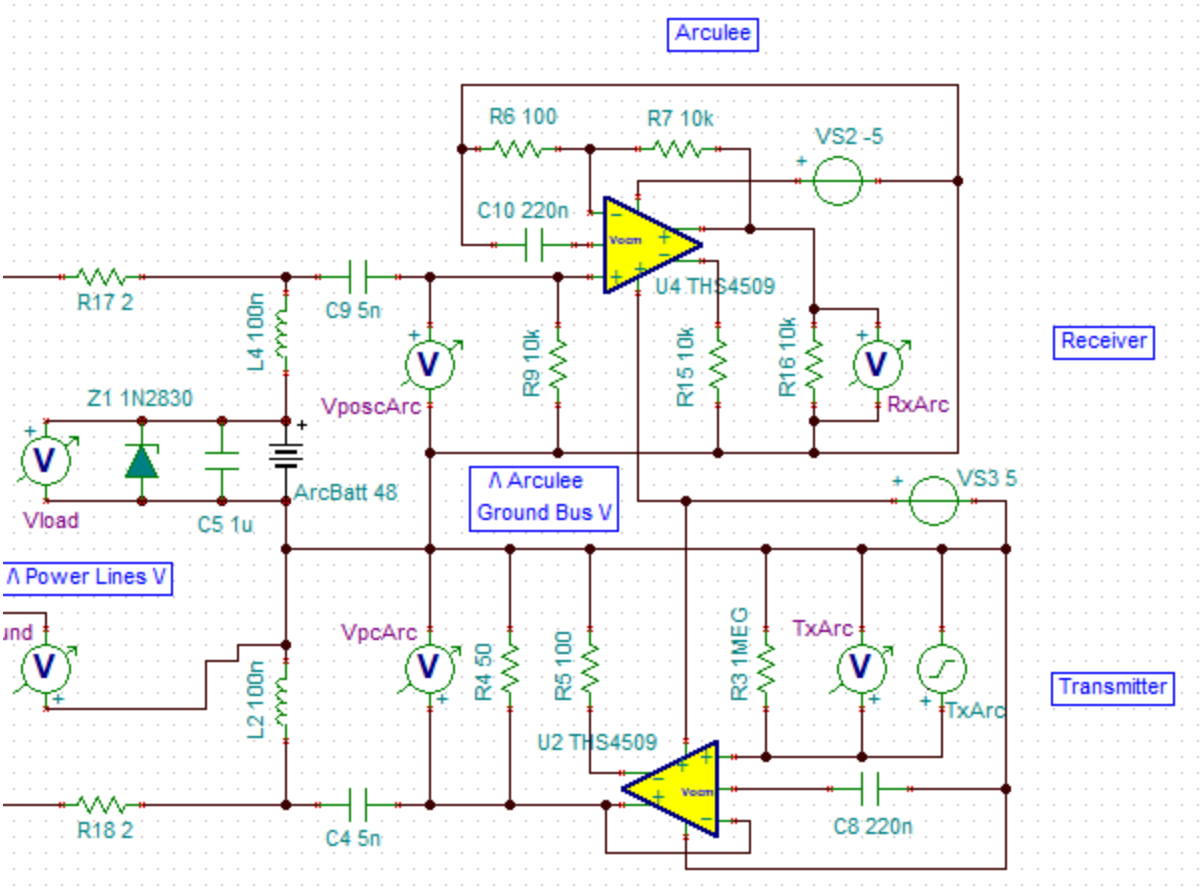


Source: personal archive.

asynchronous transmission. The design is based on sending the signal and subtracting the sent signal from the read one. The chosen operational amplifier has a 4.5 ns propagation delay and its response is not the fastest, making it considerably rounded at 100 MHz. Texas Instrument has one model with a 700 ps delay and a much faster response that could be used in place, but this model was enough for a second proof of concept. The circuit is presented on fig. 24 and shown with greater detail on figures 25 and 26. The total price for these components would be under 20 €, half the price for the PLC module offered by I2SE GmbH alone.

To make it work as it should, a half-duplex protocol would be needed, as sending

Figure 22 – Second version of test simulation with bias T circuit, right side.



Source: personal archive.

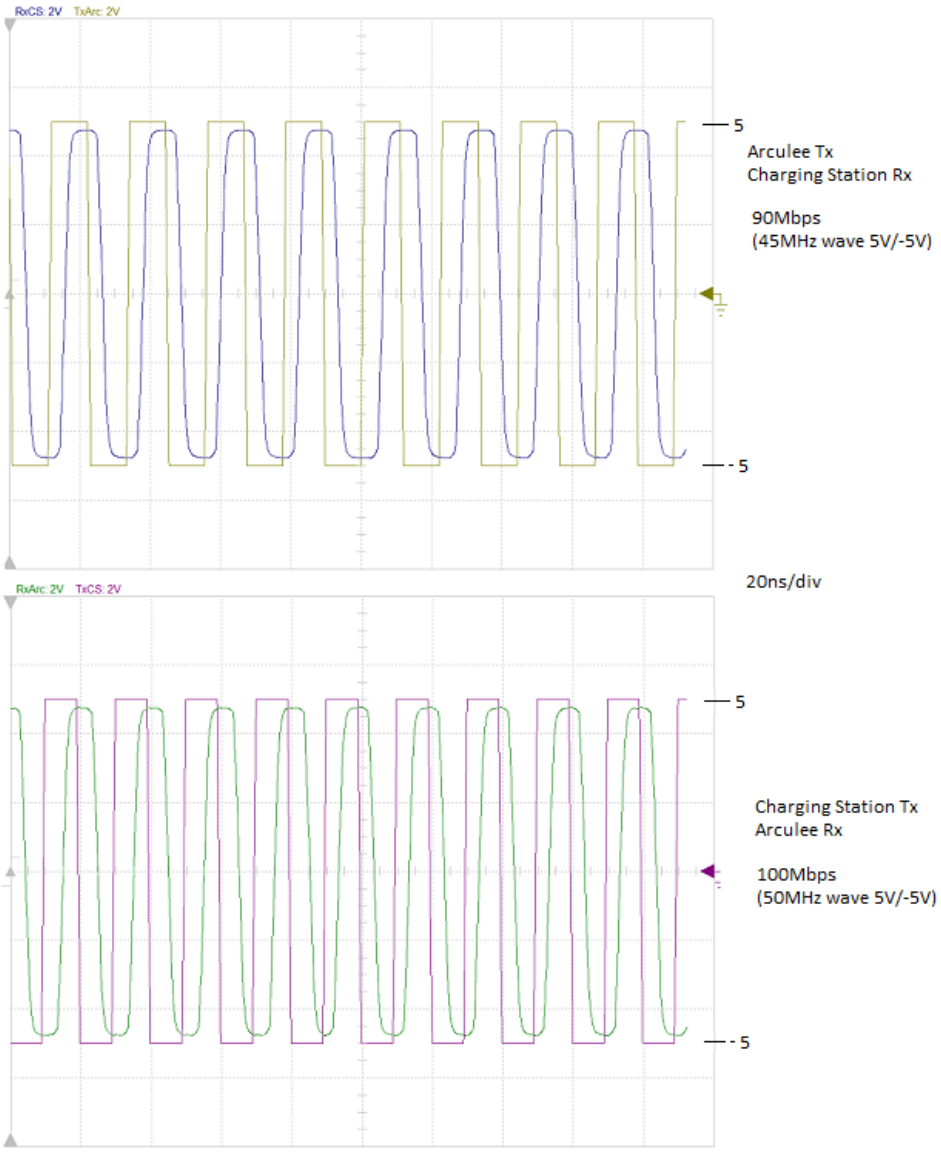
and receiving asynchronously still brought some interference, however developing such a system was discouraged by the electronics expert as the time and effort would not be worth spending when a modem can be bought for a reasonable price, missing only the coupling. On the plus still, developing these circuits gave the student more knowledge and experience on coupling strategies, which would help on the development of the coupling for the system.

4.1.2 Power-Line Module Board

It has been then opted to go for the Power-Line Module, PLM, offered by the company I2SE GmbH. Two evaluation kits have been bought. The evaluation kits are composed of the PLM plus the coupling circuit, plus a chip responsible for the Ethernet physical layer. This chip will receive data from a PC and send it to the PLM, which operates as MAC, Medium Access Control. The power supply must also be handled for the application.

For the coupling circuit, I2SE offers its own transformer and suggests one coupling design, giving values for capacitors and resistances. They also recommend a zero cross detection circuit. Then, the physical layer, PHY, is missing, along with the power supply and connectors.

Figure 23 – Successful data transfer test.

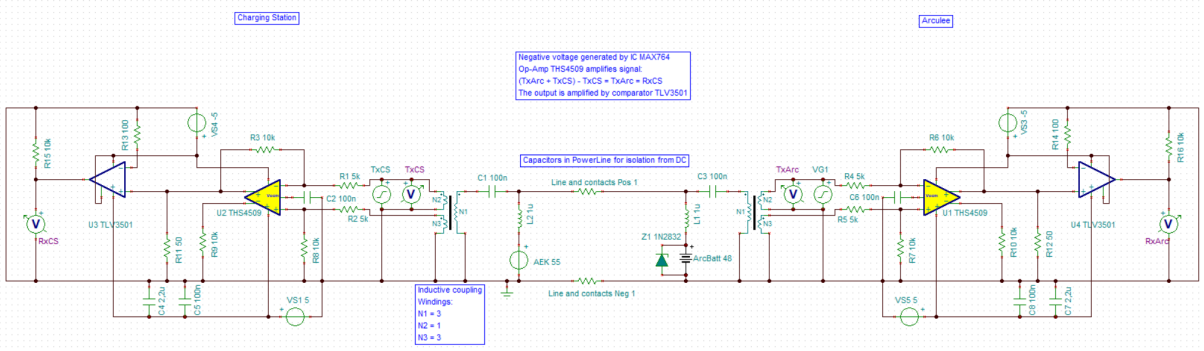


Source: personal archive.

On the evaluation board, the chip Atheros AR8035 is responsible for the PHY layer, so the chip should be used on the designed board. The power supply should transform the available 24 V into 3.3 V for the modules and the connectors need to be one RJ-45 for the Ethernet port and one high-current, four-poles connector for the signal. The high-current factor is important, because the board should hold an inductance to decouple the high frequency part from the low-pass filters on the power supply and battery, as in fig. 20.

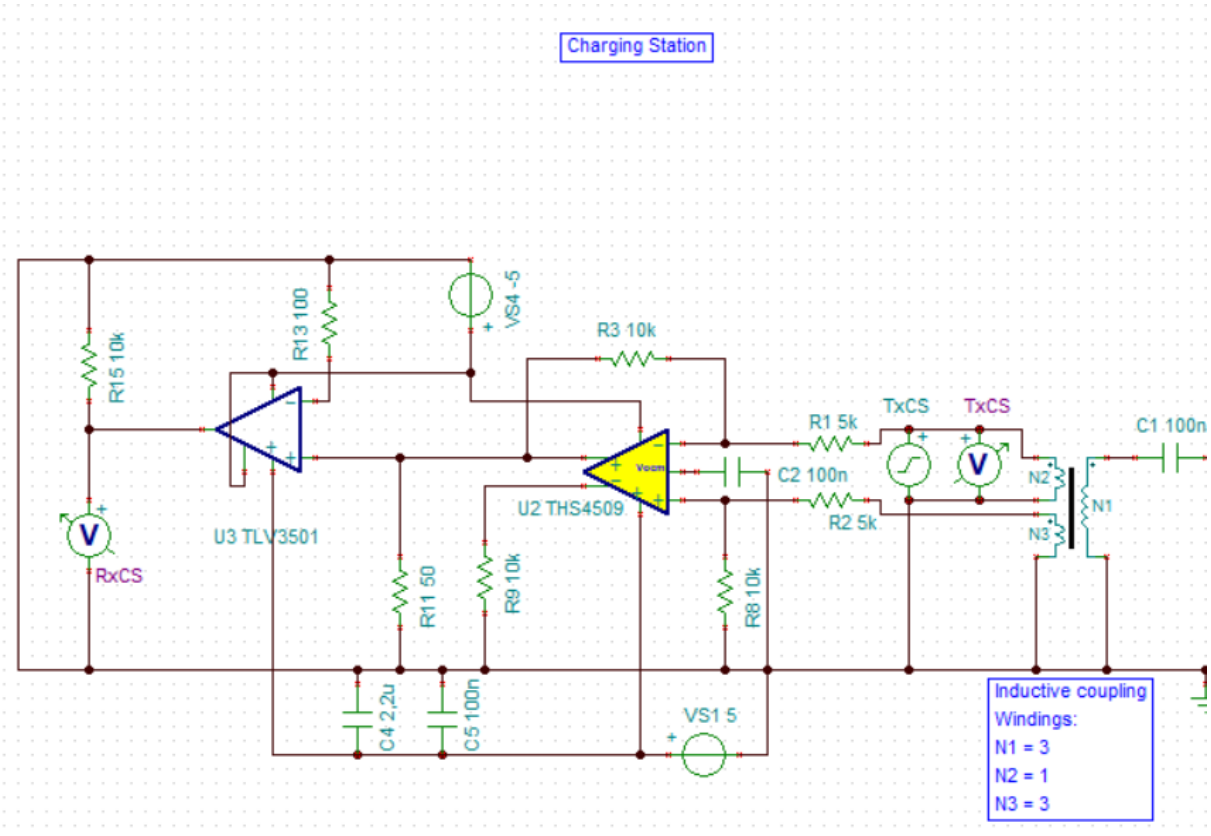
A block diagram, as well as a diagram with the components is provided by I2SE in the datasheet of the evaluation kit. The PLM offered is named PLC Stamp 1200 micro, and the functional diagram for its usual application is available on the block diagram of the datasheet, presented in fig. 27.

Figure 24 – Third version of test simulation with bias T circuit.



Source: personal archive.

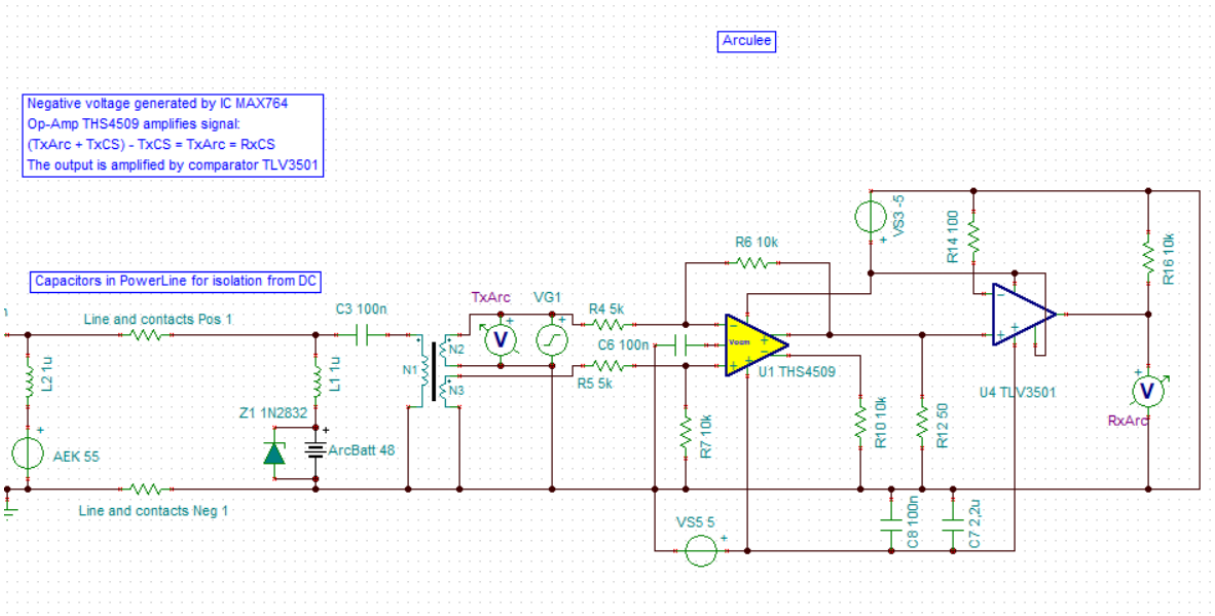
Figure 25 – Third version of test simulation with bias T circuit, left side.



Source: personal archive.

The tracks are expected to hold frequencies of up to 68 MHz and should have little phase shifting due to the use of OFDM, Orthogonal Frequency-Division Multiplexing, [20]. Additionally, the student, even though supervised by an expert, has little experience on design of PCBs. Given these factors, is it expected that the board will present design flaws. To handle that, the PLM should be kept in a separate board, reducing costs in case more than one version of the main board is needed. The PLM, fig. 28 and fig. 29, has big pads

Figure 26 – Third version of test simulation with bias T circuit, right side.



Source: personal archive.

on the bottom, as seen on fig. 29. Due to those, moving the component should be avoided, as soldering it onto a PCB must be done by a third party company. Thus the advantage of keeping it on a separate board with connectors to the main board.

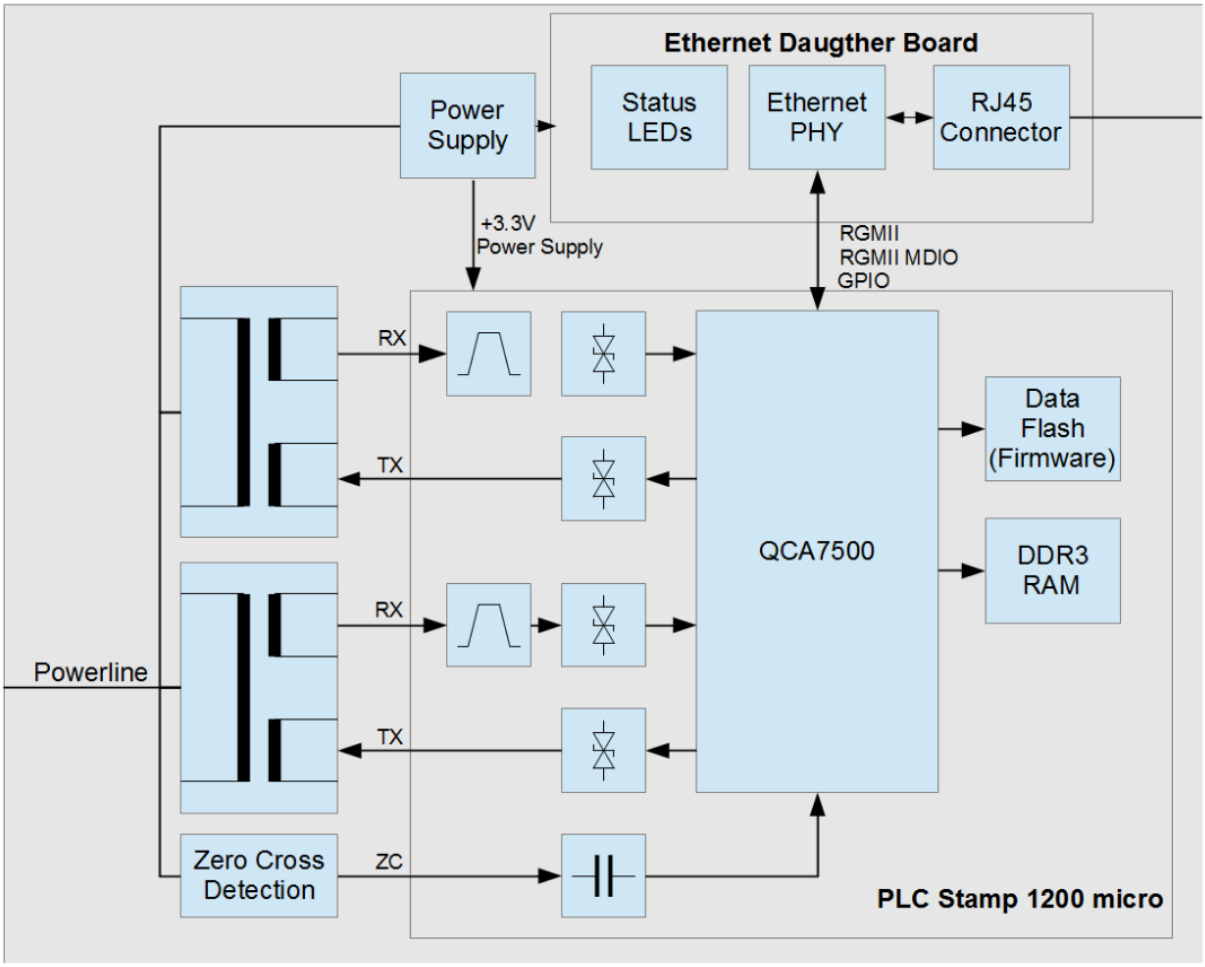
Before the beginning of the design, one proof of concept should be held with the robot and the charging station while charging, to ensure that the modules can communicate. This should be done with the evaluation kits. This is supposed to test the modules for their operation on different grounds, with the low-pass filtering and with the regular noise from the charge process.

4.2 API Between arculee and Charging Station

As mentioned in section 3.3.2.2, the most important data that the robot should send to the charging station are: its ID, the current that flows into the battery, the battery temperature, SOC, time left in charging station, charging voltage and current and data from the battery model. Besides that, other useful information could include the goal SOC and data on whether the robot wants to leave or not.

Given the little, yet existing inductive behaviour of the batteries [21], it is not recommended to cut the current abruptly. This, however, cannot be avoided without communication between arculee and charging station, as the latter cannot know when the former wants to leave. On the other hand, if the communication exists, the arculee may request the charging station to stop charging whenever necessary and, once done, the robot may leave, reducing ageing effects on its battery.

Figure 27 – Functional diagram for PLM applications.

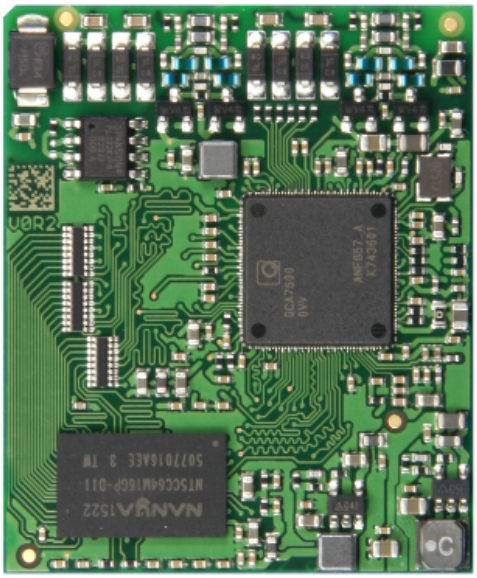


Source: PLC Stamp 1200 micro Evaluation Kit Datasheet, I2SE GmbH, August 31st 2017.

Another function enabled by the communication is the possibility of adapting the charging current based on what actually gets to the battery. Assuming that the robot uses between 0.9 A and 1.5 A, charging current could be increased to somewhere close to 0.9 A, allowing the robot to charge faster. Besides that, the battery temperature can be controlled so that, if too high, the current can be reduced, preventing the BMS from shutting down. SOC and time left information allow the charging station to show correct data on the display, ID allows for tracking the robot charge history in the charging logs and charging data allow for different types of battery to be handled by the charging station.

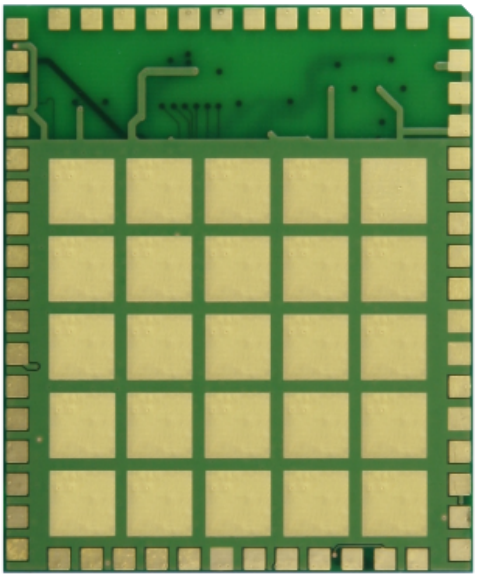
For that, the arculee should send a handshake to the charging station. This handshake would hold charge data, ID and other static information. Then, the arculee would periodically send updates with dynamic data, such as current, SOC, time left and temperature. As long as the communication is held, the charging station can run the functions mentioned above. Strong points of this application include the periodic updates working as a heartbeat from arculee to charging station and that it sends static data only once, because

Figure 28 – Top view of the PLC Stamp 1200 micro.



Source: PLC Stamp 1200 micro Datasheet, I2SE GmbH, January 15th 2018.

Figure 29 – Bottom view of the PLC Stamp 1200 micro.



Source: PLC Stamp 1200 micro Datasheet, I2SE GmbH, January 15th 2018.

there is no need to send static data more often.

For that, three (3) REST requests should be expected by the charging station server, shown on table 1.

4.3 Software

The requirements for the software are given below.

1. Host a server that receives commands.
2. Communicate via serial.
3. Control PCB.
4. Control power supply.
5. Display charging status and data on the screen.
6. Receive commands from screen to stop or restart charge.
7. Host a state machine.
8. Write logs.
9. Report errors.
10. Send alerts.
11. Upload logs to server.
12. Differentiate charge with communication to robot from without it.
13. Use as much of the existing code as possible.
14. Estimate SOC.
15. Estimate time left.
16. Avoid charge other objects other than arculee.
17. Increase voltage while charging to cover diode voltage.
18. Stop charge without causing BMS to shut down.

Table 1 – REST requests to charging station.

Request name	Content
Handshake	arculee ID Charging current Charging voltage
Update	Current Temperature SOC Time left
Stop	None

Source: personal archive.

To help separate the requirements into modules for the software, the requirements can be separated according to their nature, keeping related ones together. With this approach, one possibility of grouping is the following, where the numbers represent the requirements as enumerated above.

- Communication - 1, 2, 3, 4, 9, 10, 11
  - Server - 1
  - Serial - 2, 3, 4
  - REST requests - 9
  - MQTT messages - 10
  - Ethernet/*rsync* - 11
- Screen - 5, 6
- Charge - 7, 16, 17, 18
- Others - 8, 12, 13, 14, 15

The communication modules were separated according to the way they affect the code and the situations in which they are required. Even though it separates the communication modules throughout the code, they can be individual modules working on their own, so keeping them separated would not pose a heavy negative effect on code readability, but rather improve it for avoiding forcing data to flow through several modules in the code.

Following this strategy, it is important to know how each communication module interacts with the rest of the code, so they can be best placed. Starting with the server, it will change the behaviour of the code. The state machine will decide when to read and write to the serial. Alerts will only be written, and always by the state machine, and errors will always be written by several modules throughout the code. At last, logs need to be constantly watched so they get uploaded when enough has been written, and this function can be detached from other modules for it is independent from the charging process.

Now, all the charge-related modules somehow revolve around the state machine, which is responsible for the charging process. Being so, all “Charge” items could fit inside states of the FSM.

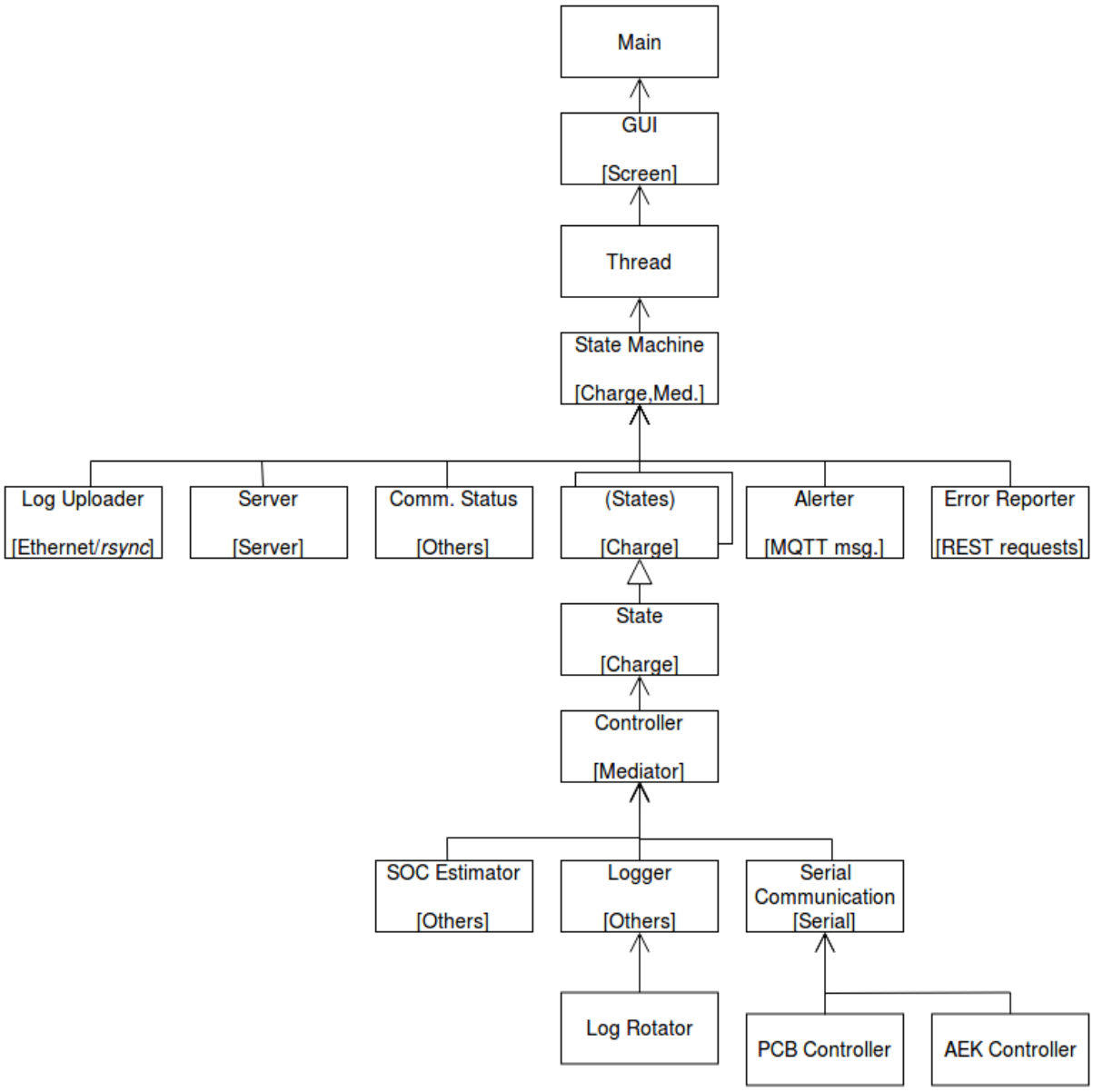
The program could then start from the screen as it did in the initial code. The screen could read data by reaching through its attributes to update the display, as in the initial code, fig. 16.

Regarding other requirements, SOC and time left can be calculated together, and so they should be. The number 13 does not refer to a specific function of the program, so no

module is needed, and writing logs and handling the status of the robot communication can be done by two separate modules.

Based on this division, on the initial structure and on the Mediator and State patterns, an initial class diagram could be drawn, fig. 30.

Figure 30 – Simplified class diagram for final charging station software



Source: personal archive.

The controller module works as an actual mediator, reading data from the serial and from other modules and making it available for the rest of the code. The state machine and the individual states will then request its data and methods, which are bound to serial communication, SOC estimation and logging. The state machine will not hold any data, but it is responsible for receiving requests and distributing them through the system as well as sending alerts when needed.

Besides, in the state diagram, the modules tied to the requirements show their grouping in the bottom of their box, and, aside from the classes mentioned above, the class named as “Log Rotator” has been added. This class is responsible for log rotation, function that does not allow the logs to accumulate indefinitely in case it is not possible to have them be uploaded.

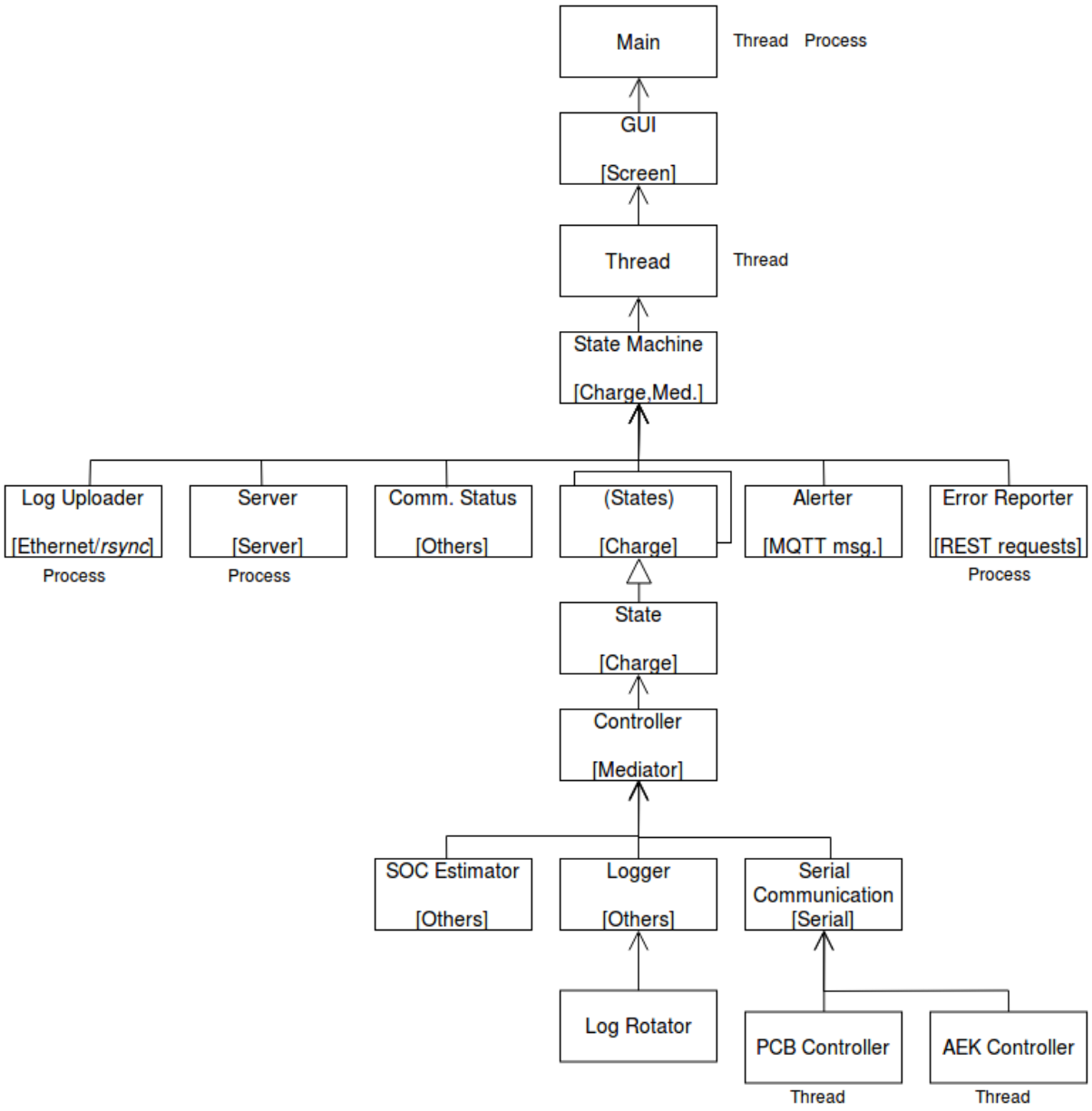
The State pattern for Python is commonly implemented programming each individual state as one class and implementing their *__init__* and *__del__* methods, called automatically when an object is created and deleted, respectively. Each state should also receive one update function, to be called by the state machine to run a normal iteration on the state. Since the state classes share data, such as the same controller object and attributes, each state inherits the general State class.

It is still important to define which modules will run on separate threads and processes, as well as whether they will run on threads or on processes. Processes on Python have the advantage that they can be terminated with one command, but run on separate Global Interpreter Locks, GILs. The GIL controls memory access within a python process, e.g. it allows different, concurrent threads to access the same variables without issues. However, two different processes on Python do not share a GIL, so they cannot share variables with each other. Threads, on the other hand, have their variables available to be read and written by other threads on the same process. They cannot be terminated as a process, but there are workarounds on how to build a kill-able thread generating manually an exception inside of it. Also, some Python modules may not run on separate threads, they must run on separate processes, such as when using TCP sockets.

For purposes of better modularisation, it has been decided that one module should run on a separate process whenever it does not have to read or write data to the main process. Given this decision, figure 30 can be redrawn to add process and thread information, obtaining fig. 31. The point of creation of new threads and processes have the respective tag by their side, so, for instance, when “Main” is started, it runs on the main thread and main process, which are there started. The GUI will then host the second thread, with the state machine and all its dependencies, one of which being the Server, which runs on a separate process, just as the “Log Uploader” and “Error Reporter” modules. At last, “Serial Communication” will host two threads, for the PCB and the AEK. Every class or module associated to the tagged class runs on its thread or process. That meaning that, for instance, the error reporter process is started from the secondary thread.

This design allows for the screen to read data from the state machine and from the controller easily via reaching through the attributes, so the GUI does not need to be modified much. It allows for the state machine to easily handle and forward requests from the server, and for the states to read all data that they need to know to operate, which come from the serial communication. The voltages from arculee and from power supply are read

Figure 31 – Simplified class diagram with sub-process and sub-thread data.



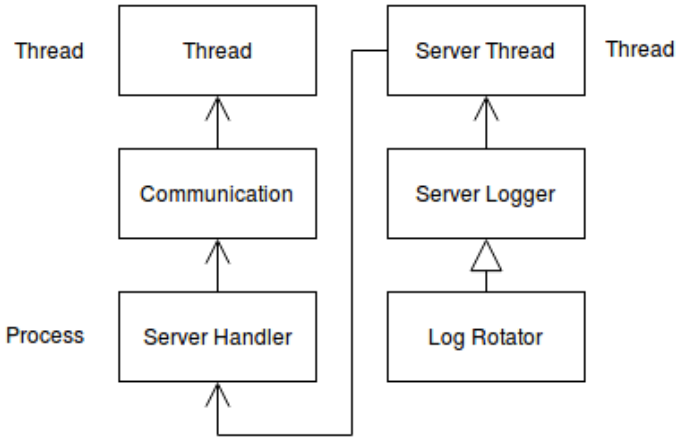
Source: personal archive.

from the PCB and current is read from the power supply. Besides that, this design allows for scalability with respect to adding modules, by expanding the reach of the state machine or of the controller, and with respect to number of states by adding states to the state machine. This is achieved while keeping most of the structure of the initial code, which can be seen by the central structure of fig. 31 in comparison to fig. 16.

It is important to mention, however, that the individual modules do require a bigger inner structure in comparison to what has been presented so far and that is the reason why the class diagrams presented so far were called “simplified”. For instance, to create an abstraction interface between the state machine and the server, several classes inside

the module are required. The goal for the server is to have a polling function, from which the state machine can read and handle one request at a time. For that, the class diagram presented on figure 32 has been proposed.

Figure 32 – Class diagram for server module.



Source: personal archive.

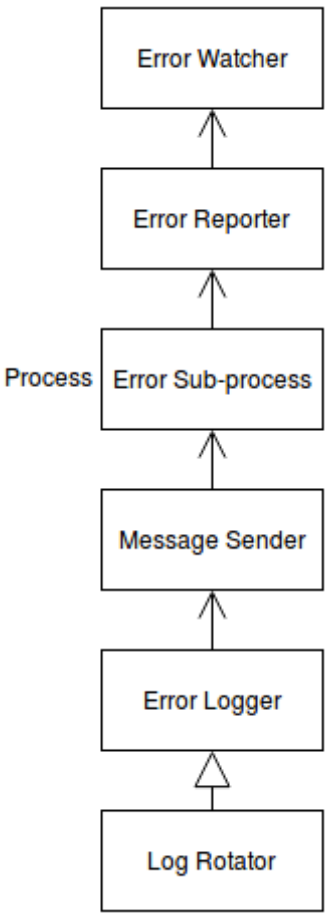
The initial thread starts up the module, calling the communication class, that will start the process and wait for commands from it. Whenever a command arrives, it is made available and a flag is set on the communication class. Once the state machine wants to check for requests, it can call a method in the thread class that will handle the poll request by calling methods from the communication class and processing the returned values. Further down, the server handler, run as a sub-process, will start the thread for the server itself. It will also forward the requests received from the server to the communication class whenever a request is available. A communication pipe is required for inter-process data exchange. The server will be started with the thread and will host its own logger, where general information from the process will be logged. The logger implements the rotating log handler, “Log Rotator” from the main process.

The “Log Uploader” module can be implemented in a single class plus one interfacing class to handle start-up and shut-down of the sub-process and the API. The communication status module can be one single class. The alerter module could be split in two classes, one for the MQTT protocol and one for handling the messages. The SOC estimator can be a single class, as well as the log rotation handler. For the error reporter and serial communication, their structure is shown in figures 33 and 34.

The error watcher is responsible for receiving error requests and formatting them to forward to the error reporter, which will then forward them to the sub-process that will send the message and write logs.

The serial communication, on the other hand, has one interfacing class, which starts

Figure 33 – Class diagram for error reporter module.

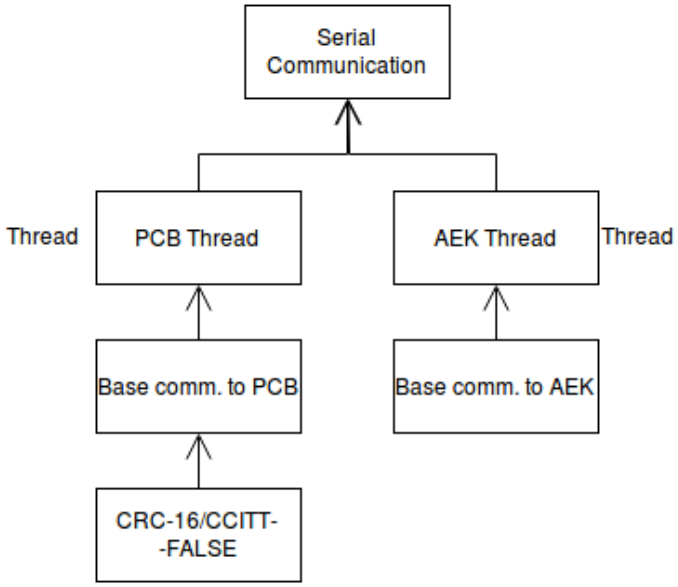


Source: personal archive.

two threads for the actual communication. Each thread will hold a sequence of reads and writes to be repeated in a loop. An initial error handling is performed in the threads. Then, the actual read and write-related methods are implemented on the base communication classes, which hold the serial objects. The communication to the PCB implements a checksum, namely the CRC-16/CCITT-FALSE cyclic check.

With the software then designed and running, a setup file should also be programmed to take one Raspberry Pi from a fresh firmware installation to having the right settings and code to run the software. To achieve such a goal, the script must install all dependencies and update all configuration files, plus enable functions such as Secure Shell, SSH, and bash aliases that are used on the Raspberry. This means that all configuration files should be stored with the code, which will be hosted on the distributed version-control system that is already in use at arculus GmbH, a GitLab hosted locally.

Figure 34 – Class diagram for serial communication module.



Source: personal archive.

5 Implemented Solution

After designing and projecting the solutions, they could then be implemented. The timeline of the implementation started with the software due to the need for a functional charging station. The software was initially brought to a stable and working stage, then, the hardware was developed. For its development, iterations were made where the student would send the design to the electronics expert of the company for corrections and tips, then rework the design and send again. There was time to further work on the software while the electronics expert was reviewing the work and while the components and boards were ordered but had not yet arrived.

It is important to note, however, that the software was working initially. Still, it was unstable due to the lack of thorough, long-time testing and the new use cases being found required modifications that were too time-consuming. A first reorganisation was performed to get it to run stable, as well as to facilitate other small changes to come.

During the development of the software, new use cases and requirements were found and corrected or added to the software. For instance, during the winter, the temperature falls to around -20 °C and charging a battery with such cold cells damages its internal structure. Therefore, a warm-up state has been proposed by the battery expert, during which the battery would have charge current limited to 5 A for 5 min, before increasing to full charge current.

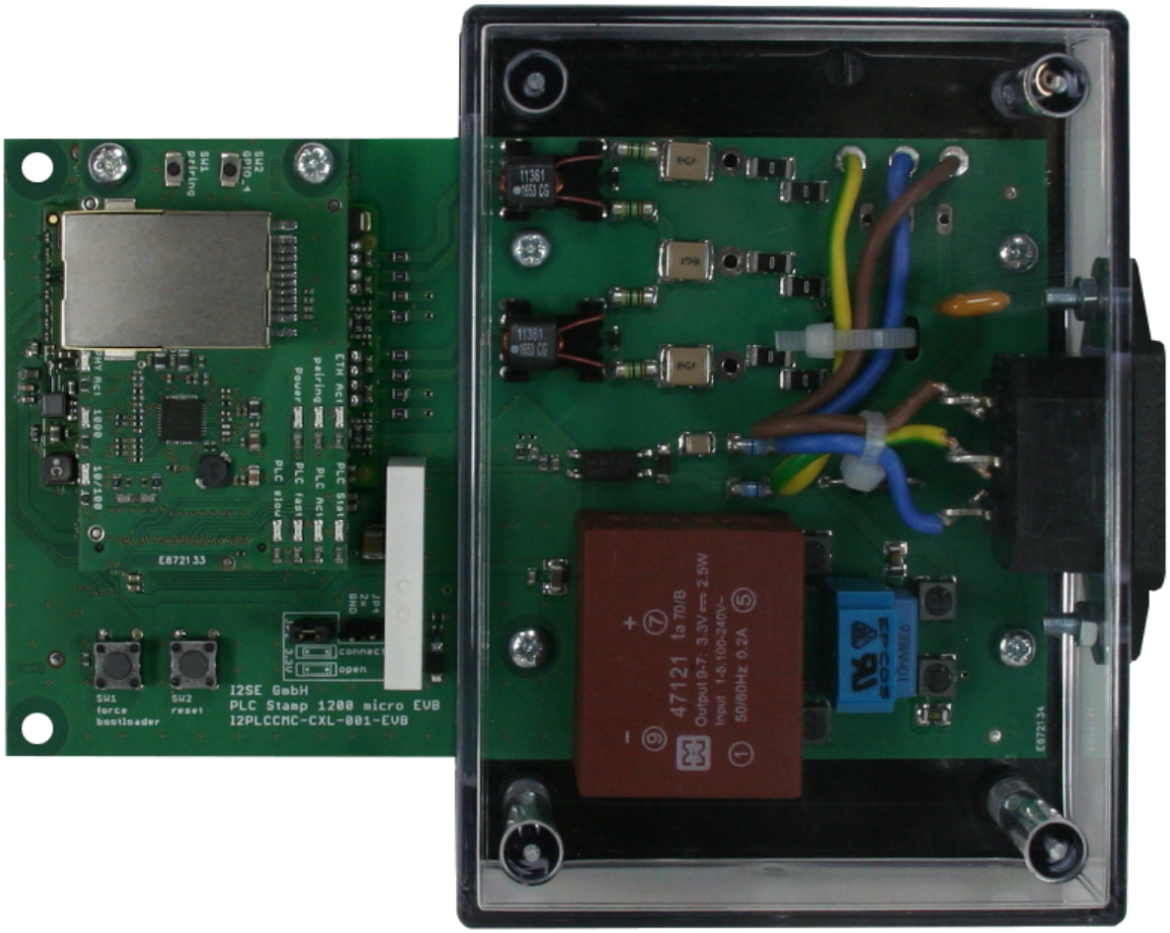
In this chapter, additional use cases and requirements, as well as the remaining topics of the theory regarding the solution will be presented, with details to the implementation.

5.1 Hardware

As mentioned in section 4.1.2, a proof of concept should be made to test the functionality of the system. The proof of concept should use the evaluation kits, fig. 35, one connected to the charge plugs in one arculee and the other, to plugs in a charging station. The arculee should charge and the communication should hold. The proof of concept was performed with the Banana Pi, for its Gigabit Ethernet port and one laptop connected to the evaluation kits. The results were satisfactory, even though it achieved a baud rate of just about 35 Mb/s. The speed in fact oscillated between 25 Mb/s and just over 200 Mb/s, staying mostly between 30 Mb/s and 40 Mb/s.

Such a baud rate is enough for the current system, however not ideal for the future, as stated in chapter 2. These results are believed to be caused by the strong filtering in the output of the power supply. Hence the importance of adding an inductance to the board,

Figure 35 – PLC Stamp 1200 micro Evaluation Kit.



Source: PLC Stamp 1200 micro Evaluation Kit Datasheet, I2SE GmbH, August 31st 2017.

which will isolate the high frequency signal from the low-pass filter while allowing the DC current to flow to the battery, after the design of a bias T circuit.

The proof of concept showed also that the charging station will not start charging with both modules connected and operating, so the modules should have a reset pin available.

After the success of the concept, two prototype versions were developed for the hardware. The first has its data shown in appendix A. It holds the PHY chip, along with power supply, coupling and RJ-45 connector. The PLM is soldered to a second board, as mentioned in chapter 4.1.2. This adapter board is presented in appendix B, it connects to the main board via the headers with IDs U6, U7 and U8. This version did not work, as the PLM module was found to be defective, causing a short-circuit in its power input, and, once the chip had been replaced, the PHY chip was found to be damaged, as the expected voltages in its pins did not match the actual voltages. The PHY chip was most likely damaged while testing the power supply components of the board. This part was found to contain a design error in the voltage regulator, which was designed as adjustable voltage and had in fact fixed output. In the PCB design, the PHY chip is the IC, Integrated Circuit, that sits in the middle

of the three board connectors, with many tracks connecting to it.

The PHY chip had some issues to be ordered, needing to be imported from the United States. Only two were ordered, and one had been damaged. Then, to avoid risking a damage to the one chip left, it had been decided to design a second version of the board. The design and schematics of the board are presented in appendix C. The board was designed with corrected tracks, measurement points and status LEDs, Light-Emitting Diodes. The measurement points for voltage were found to be essential, as the PHY IC lays under the adapter board while in operation, making it close to impossible to get safe measurements for certain pins. The measurement points are regular, empty pads connected to specific, known voltages, such as the busses with 1.1 V, 2.5 V and 3.3 V created by the PHY. These pads lay outside of the area covered by the adapter board, allowing for an easy measurement of the voltages with a voltmeter. Status LEDs show the PLC status, e.g. if it is paired, connected or active.

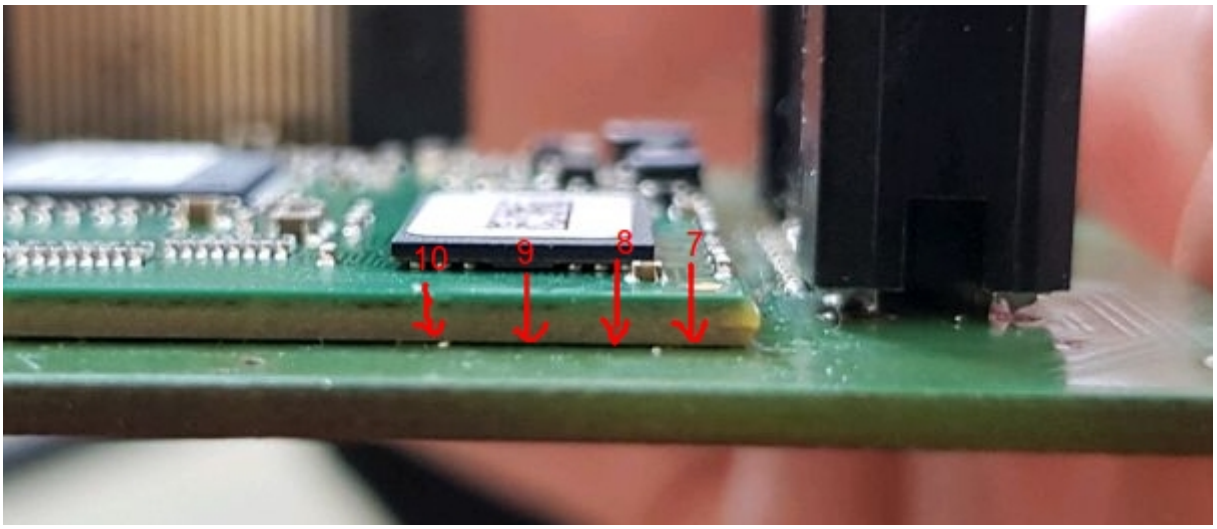
After testing, problems were seen with the second version as well. The communication protocol between the PHY chip and the PLM is called RGMII, or Reduced Gigabit Media-Independent Interface. This protocol is based on 6 channels for the master, which is the MAC, and 6 channels for each slave, the PHY. The channels are one control channel, one clock and 4 data channels. However, the PHY chip is a transceiver and such chips usually have the incoming messages in the bus named transmit (Tx) and the outgoing messages in the bus marked as receive (Rx), with relation to the PHY. The board, however, just as the first design, had the incoming messages in the receive bus and the outgoing in the transmit bus. After an attempt of fixing the issue without the need for a new version, the board still did not work due to several issues, including wrong configuration bootstraps.

The changes noted for the third design included: heavier capacitors in a few pins, lighter in others, more capacitors in general, more beads in the power tracks, even more probing pads, status LEDs for the PHY, resistors in the Rx lines of the RGMII protocol for slight signal filtering, a capacitor for high-pass filtering in clock input of PHY coming from MAC and updated bootstrap configurations. The last design is presented in appendix E.

With this board soldered, it initially did not work. Analysing where signals were missing, a problem in the soldering of the adapter board has been found, as the PLM did not have all of its pads soldered onto the board. The issue is shown in figures 36 and 37. These are GPIO pads and, if some pads were poorly soldered, it raises a warning that more pads could have the same issue. The GPIOs where the problem was seen are 8 and 9, their position is marked in the related figures. Even though it is hard to see, in the positions of the pads for GPIOs 8 and 9, no solder can be seen, while it is visible in pads 7 and 10.

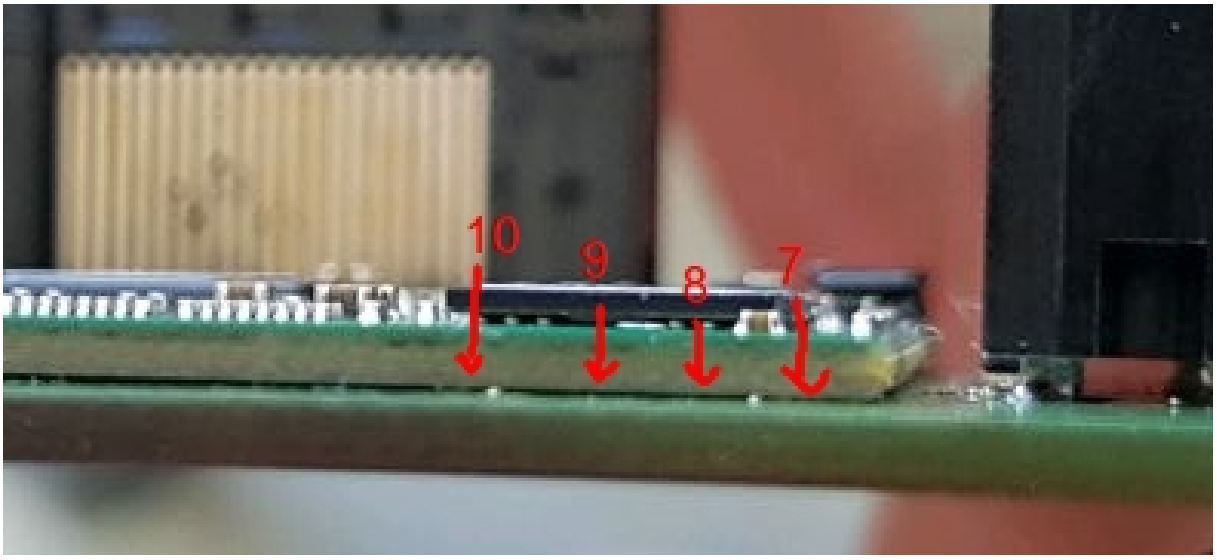
The board has then been re-soldered and everything seemed to be working as it should, according to the sequence after which the LEDs lit up, as it matched the one from the EVK without fail.

Figure 36 – PLM soldering error, higher view.



Source: Personal archive.

Figure 37 – PLM soldering error, side view.



Source: Personal archive.

A few tests were performed to find the issue, including checking if the power-line communication was in fact connected, as shown in the LEDs. For this, a PLC testing software provided by Qualcomm has been used, as the company is the manufacturer of the chip used in the PLM. The output showed that the connection was held, but it would only work when requested from the side of the evaluation kit, not when requested from the developed board. This information, plus some signal measurement and Ethernet package tracking and matching informed that the problem lied either in the communication between PHY and PLM, or in the configuration of either chips. From that, the problem has been found to be in one of the data tracks of the RGMII Tx, as a pad had come off the board, interrupting the signal.

With that fixed, the module worked. The third design was then the prototype to be used for the tests. At last, the board was also found to not have the casing of the RJ-45 header grounded, fixed with a simple cable soldered to it and to a ground pin.

The final design of the board is shown in figure 38.

5.2 Software

As mentioned in section 5, initially the software was brought to a working, stable state. This state was a simple reorganisation of the old software, taking a step towards the whole redesign, but without spending much time on it. The structure of this intermediary state is shown in appendix D. Some modules were kept or implemented for testing purposes, such as the standard deviation and the logging thread, but they were removed afterwards for the new design. This software has a class named “Data Set”, it is equivalent to the class “Logger” from fig. 30. It also had a reading voltage state separated from the matching voltage state, as the voltage was, initially, first read and then matched, meaning that the voltage on the power supply would be raised to charging voltage, then the battery voltage would be read, and then power supply voltage would be decreased to match the battery voltage. This was later modified to the iterative voltage matching approach, which was implemented in the reading voltage state, removing the need for a matching voltage state. This was initially mentioned in section 3.3.1.4, explained in greater detail below.

The voltage read depends on the voltage set to the power supply. This happens due to the circuit formed when the arculee docks in the charging station, as it contains a voltage divider where the voltage is measured. Meaning that, in order to read the correct voltage of the arculee, the voltage on the charging station must be either above or close to it and the voltage read increases with the voltage on the power supply. Being so, it is possible to iteratively set the voltage of the power supply to the voltage read from the battery until a point where the battery voltage will not increase anymore, when the power supply and the battery will in fact have their voltages matched.

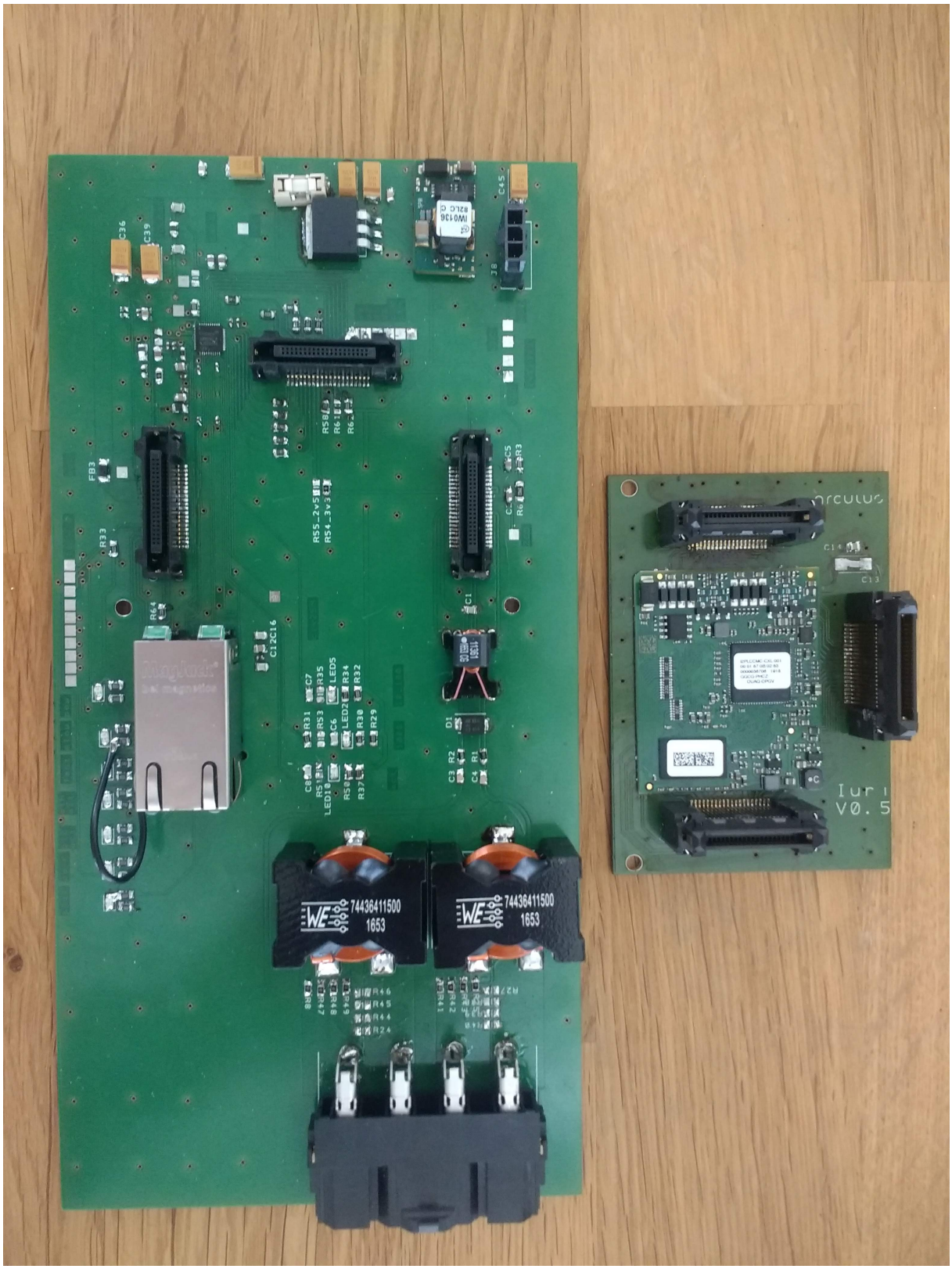
This initial version of the software was modified over the course of the project to, in the end, have the structure presented on figure 30. The final state machine is shown in fig. 39, it is started at state “Waiting for Robot”.

The state machine has a warm-up state, as mentioned in the introduction of the present chapter and has no matching voltage state as previously for its iterative voltage reading strategy.

Figure 40 shows the display of the charging station in a production-ready state before the communication has been implemented. This previous state was required to be implemented in the client location before the communication was ready to be implemented.

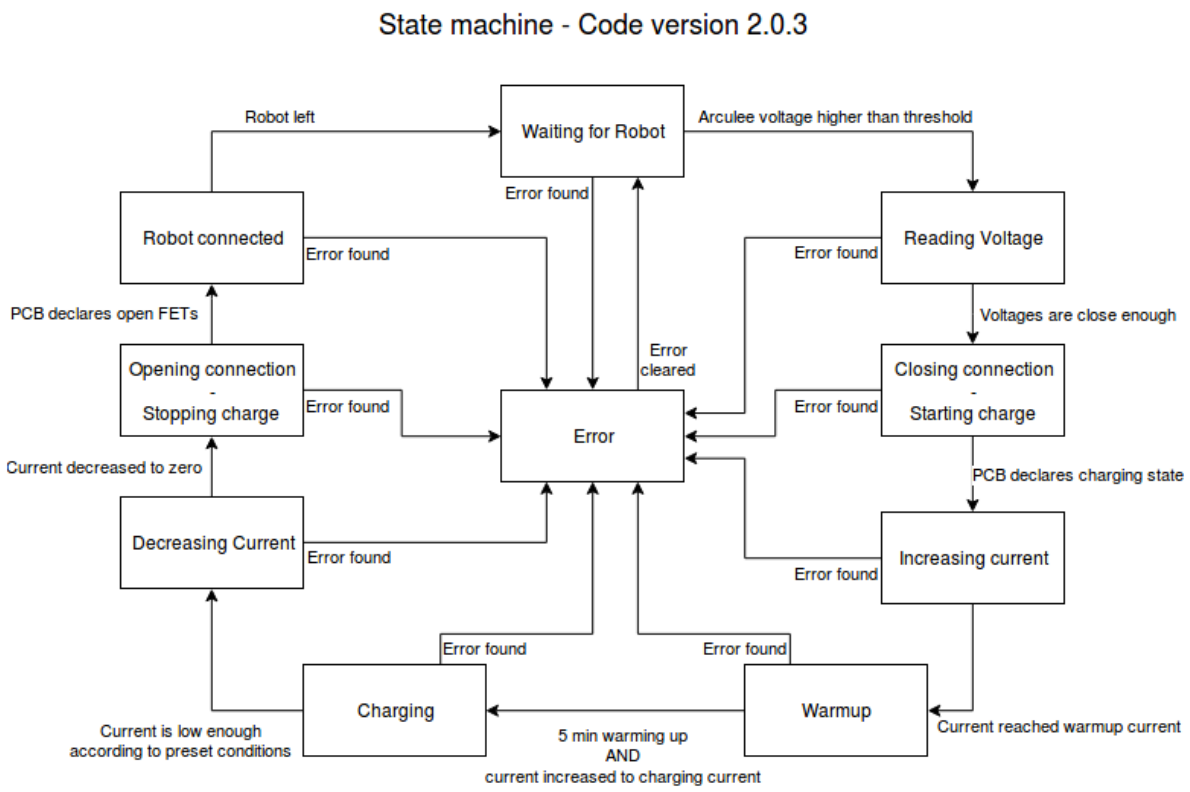
Due to the error and uncertainties, the time left and state of charge were left out, and the arculee ID is not shown due to lack of communication.

Figure 38 – Final design of the developed board.



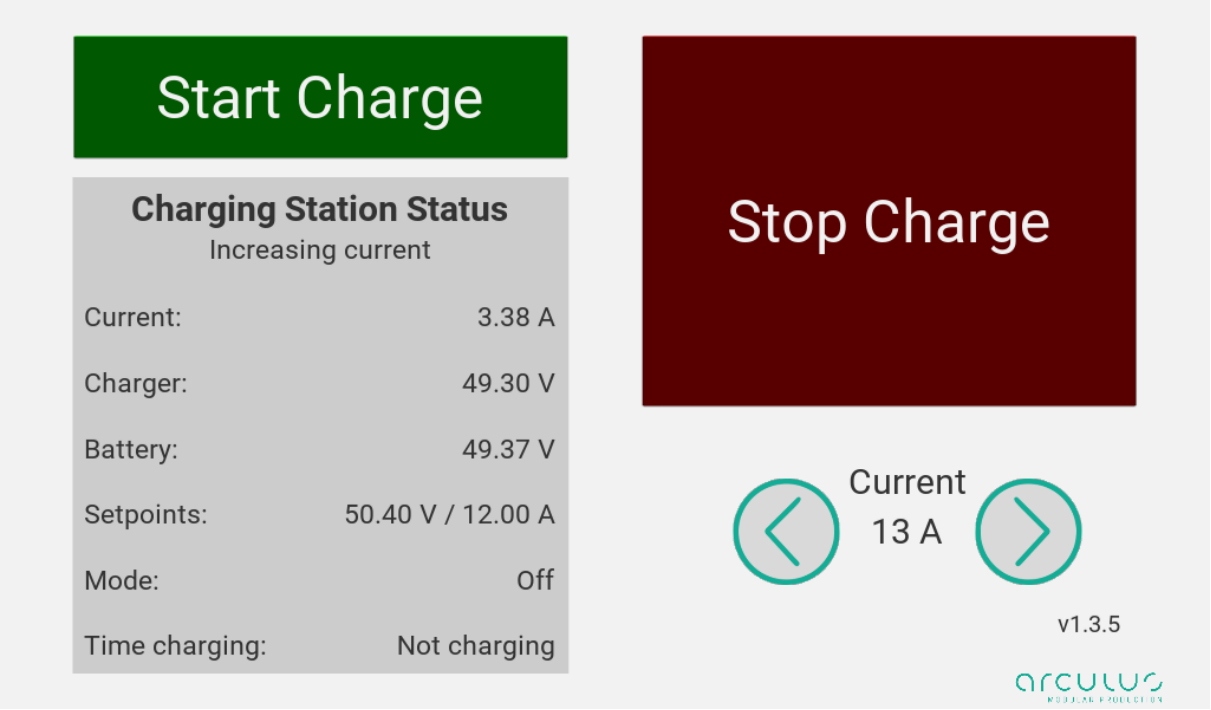
Source: Personal archive.

Figure 39 – Final state machine design.



Source: personal archive.

Figure 40 – Display of charging station in production-ready state before communication.



Source: personal archive.

6 Results

Even though there was no available time to test the whole system with the robot, both the software and the hardware had their individual tests. The test of the software was straightforward, based on running the software in a charging station and sending requests to the server endpoints as a robot would and observing that the system entered the expected modes without stopping charge. The code has a fallback mechanism that, in case the communication fails for any reason, it can still charge further without the communication. As no full charge test has been run with the communication, some functions were tested separately, such as current setpoint update and full charge requests, both successfully generating a change in the charge current and a full charge request in the software.

Regarding the PCB, however, the test setup was more complex. Initially, the module has been tested on cables with no voltage nor current, called dead cables. The developed hardware was connected by the cables simulating the power-line to one evaluation kit. A laptop connected to the developed board had its Ethernet settings adapted to have a fixed IP, 10.0.0.2, with a netmask of 24. Connected to the evaluation kit was the Banana Pi by its Gigabit-capable Ethernet port. The Banana Pi was set to have the fixed IP of 10.0.0.1 with same netmask. Both the evaluation kit and the developed board were powered by a laboratory power supply. The cable from the laptop was category 5e and the cable from the Banana Pi was category 6, both categories are adequate for Gigabit Ethernet. Both the laptop and the Banana Pi run Linux-based systems, the laptop being Ubuntu and the Banana Pi, Armbian.

The Banana Pi, being powered by the USB 3.0 port of the laptop, was turned on and a direct ping has been tested by running the command "ping 10.0.0.1" in the laptop. This returned, in average, a delay of 0.45 ms for each ping, without lost packages, making sure that the Banana Pi and the laptop have the correct settings. Then, the direct connection would be separated, the laptop being then connected to the board and the Banana Pi, to the evaluation kit.

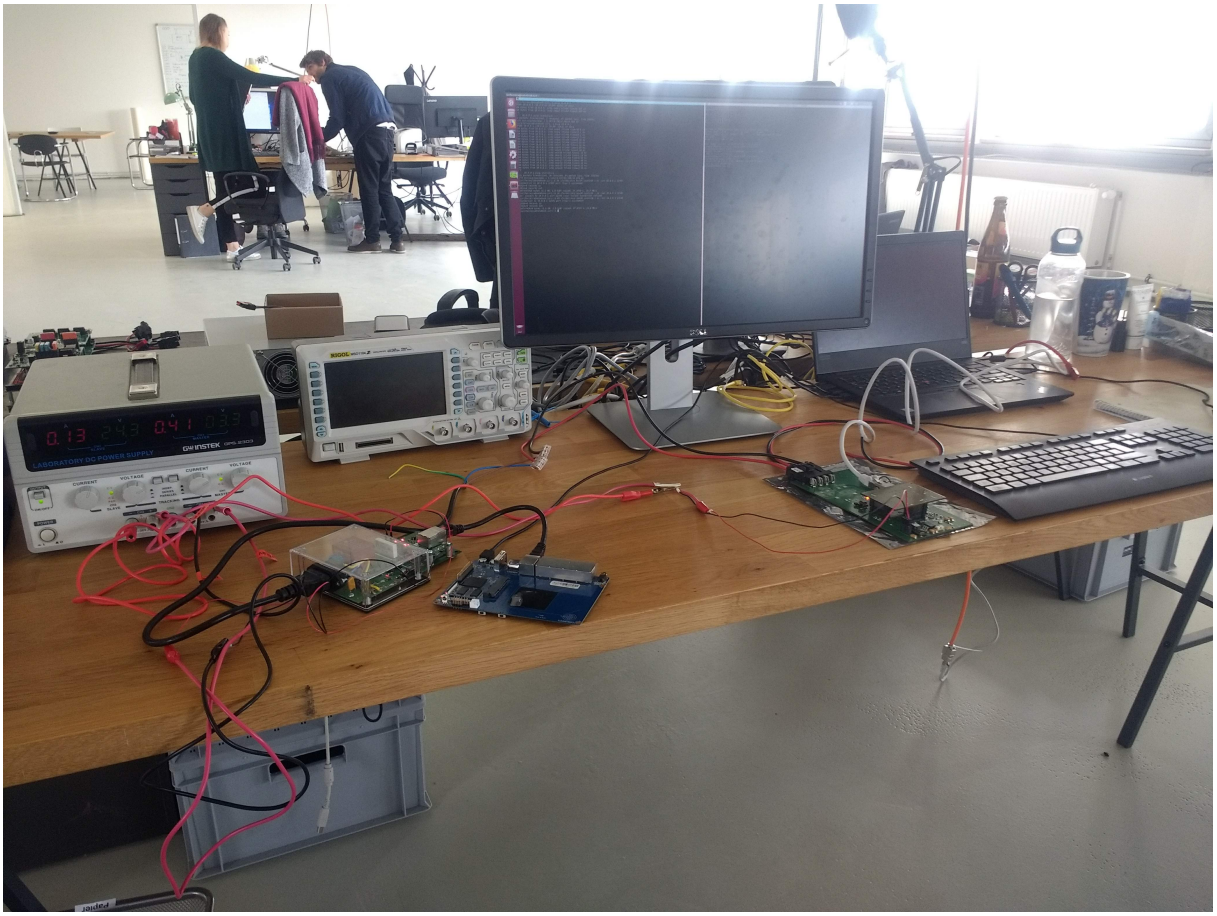
Turning on the power supply, both boards turned on and communicated. LEDs lit up as they should in both boards synchronised, until settling down. Once settled down, by running the same ping command, the PLC connection LED "PLC Slow" lit up and, the ping command returned a delay of about 3 ms.

With the connection built and tested, the connection speed would be tested next. For such, a TCP/IP server was held in the laptop. The Banana Pi was accessed via SSH and would send packets full of the number "0" to the laptop to test the bit rate of the connection.

The test setup is shown in figures 41, 42 and 43. The result of the connection speed

test is presented in figure 44.

Figure 41 – Whole setup for dead-wire connection test.



Source: personal archive.

After that, the ping command was run, shown in the lower window of fig. 44. For the two requests sent, the delays were 3.21 ms and 3.22 ms, as shown in the end of the lines.

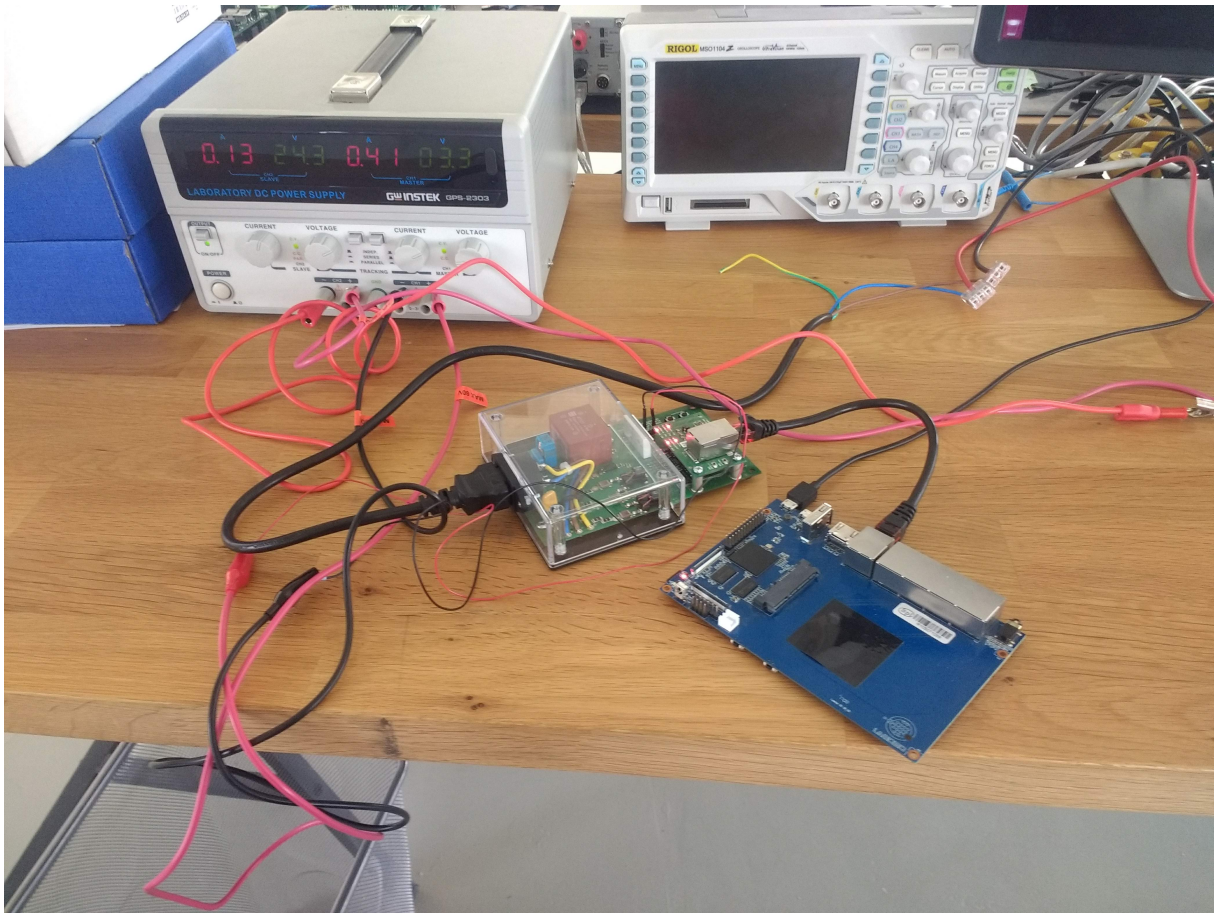
Then, the command "nc -vvlnp 12345 > /dev/null" will start a server on port 12345 and wait for a connection. Received data is discarded into the path */dev/null*. With the server running, the Banana Pi has been accessed with SSH and a command has been sent to test the connection speed, in this case, sending 1.1 GB of zeros to the laptop. The speed achieved was about 29 MB/s, or 230 Mbits/s.

With the speed of 230 Mbps, the goal of 40 Mbps has been achieved. Still, the conditions were close to perfect, as there was nothing else using the power-line and the module has potential for reaching up to 600 Mbps.

With the same setup, a second test has been performed using the software *iperf*, which tests Ethernet connections using protocols TCP or UDP. 293 Mbps and 294 Mbps were achieved using UDP with maximum speed, fig. 45.

The reason why such high speeds were not reached may lie on the fact that the PLM

Figure 42 – Banana Pi and evaluation kit in dead-wire connection test.



Source: personal archive.

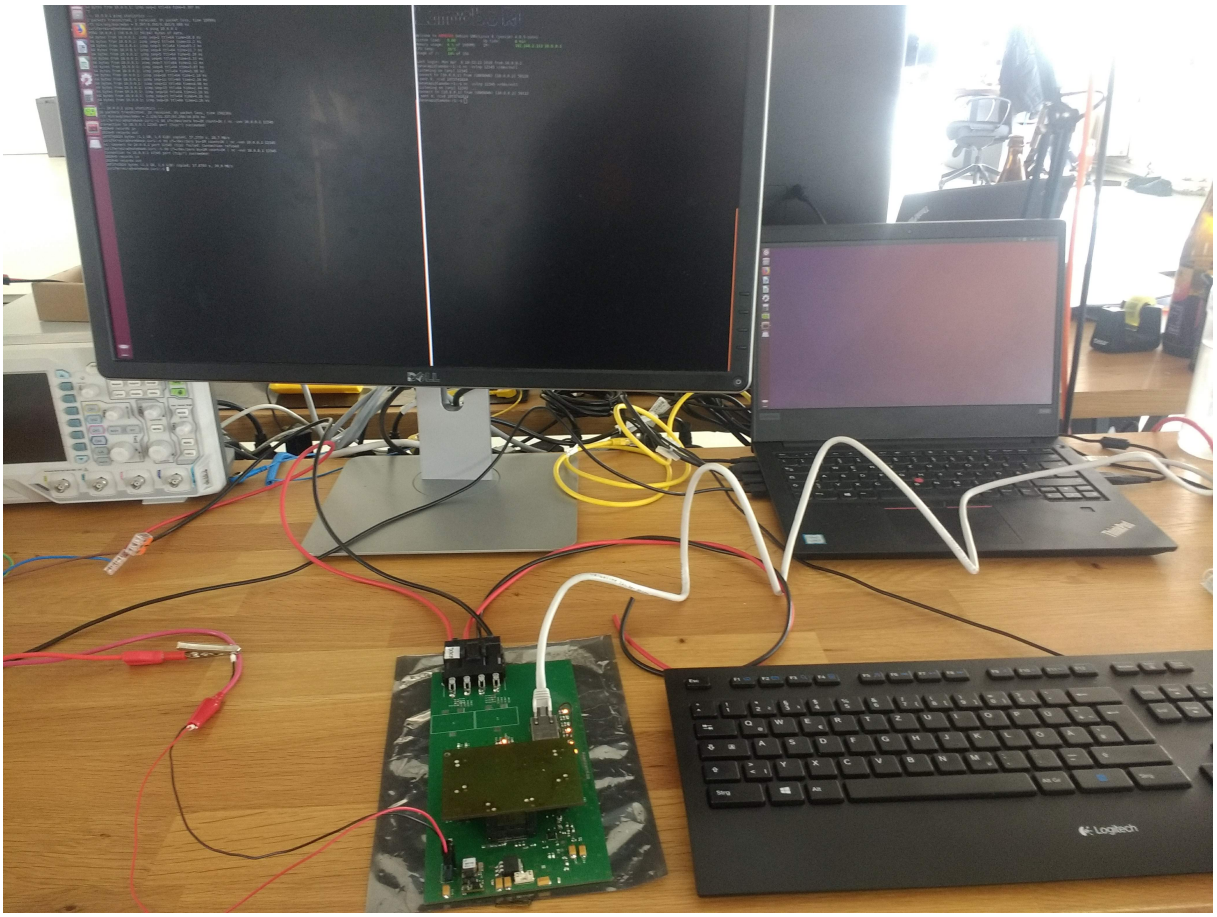
stays in a separate board. The connectors produce some interference, as shown in figures 46 and 47, measured with an antenna 2 cm away from the connectors, with a gain of about -28 dB, or 0.04. Plus, the longer tracks required by the connectors make the signals more susceptible to interference and damping.

The other possibility would be a limitation of hardware in the Banana Pi used. Using a single cable to connect the laptop to the Banana Pi resulted in 357 Mbps, fig. 48. The Ethernet bus in the Banana Pi is shared with the serial ports and other internal busses and the whole bus usage is limited to just about 460 Mbps.

6.1 Live wire tests

Testing on live wires, the results were severely reduced, as expected. Two tests setups were used, shown in figs. 49 and 50, with the Banana Pi and Raspberry Pi, respectively. The company politely asked to avoid showing the inside of the charging station and of the arculee, so they have a blue square drawn over.

Figure 43 – Laptop and evaluation kit in dead-wire connection test.



Source: personal archive.

The tests were again performed using the developed board in its third version, which communicated with an evaluation kit. The laptop would communicate to one of the processors, either the Raspberry Pi of the charging station with its Fast Ethernet port, i.e. with a theoretically maximum speed of 100 Mbps, or with the Banana Pi, with maximum tested speed or 357 Mbps.

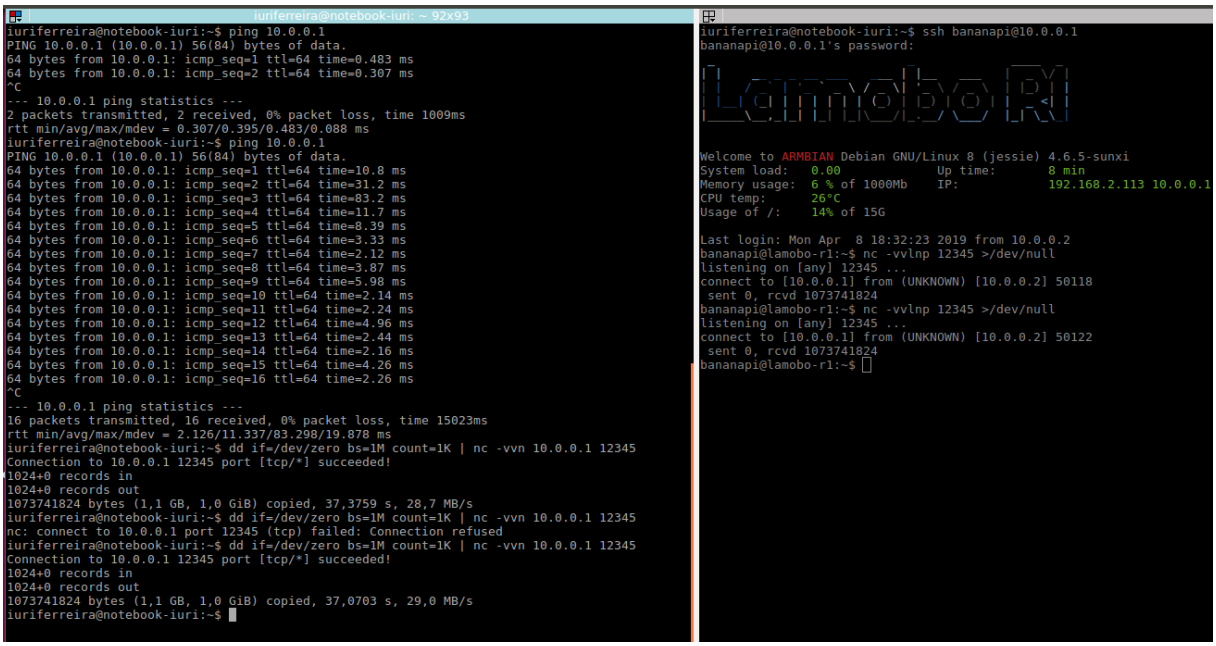
Four tests were performed in total, two with the Raspberry Pi and two with the Banana Pi. The two tests are one without inductors soldered to the board and other with them. First the two tests without inductors were performed. Then, the board was disconnected from the cables, had the two inductors soldered onto it and the 0-ohm resistors were switched to make the current to flow through the coil. The board was reconnected and the tests with inductors were performed. Charging current was limited to 5 A.

The steps for preparing the test were as follows:

1. Turn on arculee.

2. Turn on charging station.

Figure 44 – Test results for connection speed.



Source: personal archive.

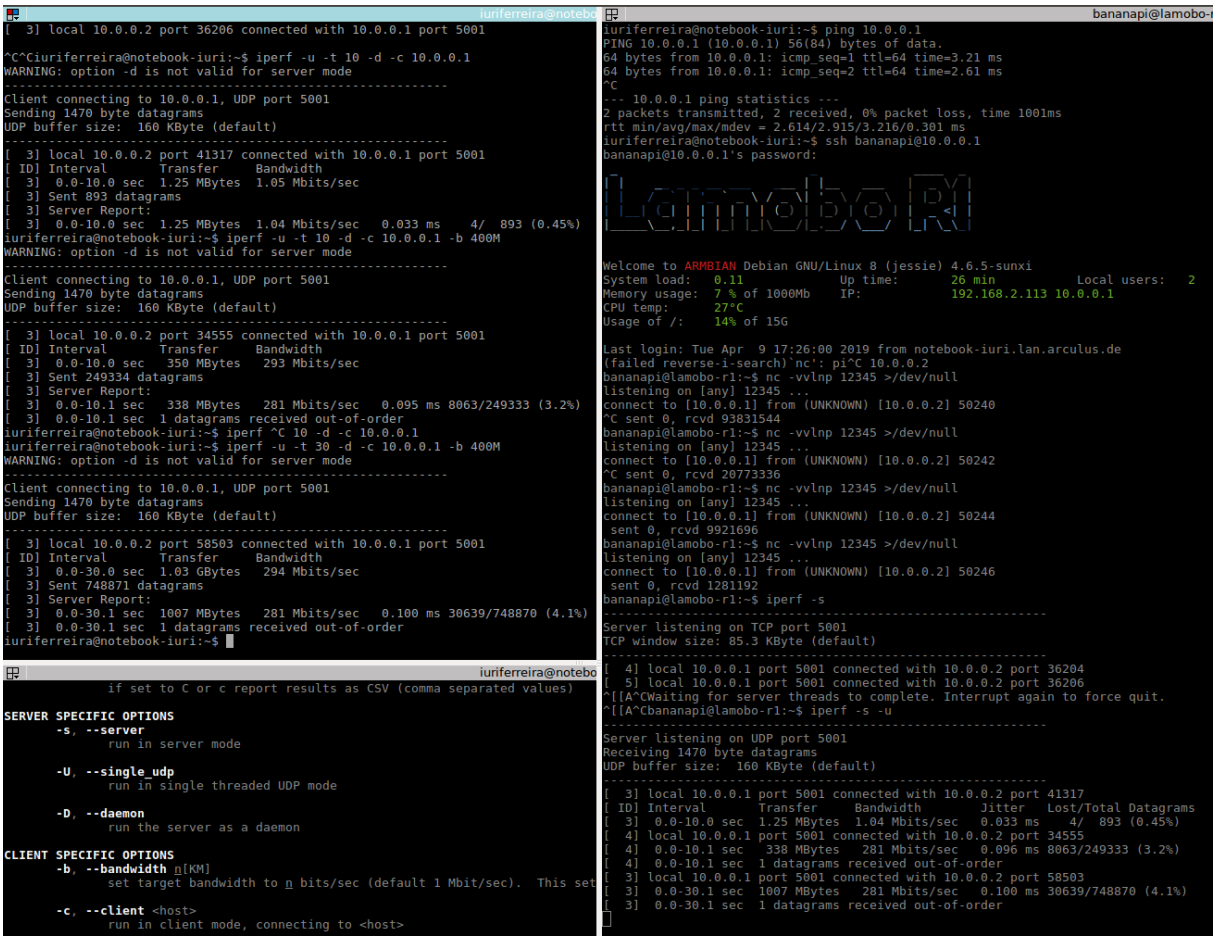
3. Drive arculee into charging position.
4. Wait until charging station starts charging.
5. Connect EVK to robot charge plugs.
6. Turn on power supply to power both power-line modules.
7. Send ping command from laptop to prove communication.
8. SSH into Pi.
9. Run speed tests.
10. Reduce current until 0 A.
11. Disconnect everything.

6.1.1 Raspberry Pi without inductor

Results of the test shown in fig. 51. They show a maximum connection speed of 10.2 MBps, or 81.6 Mbps.

The laptop, connected to the evaluation kit, would send 100 packages of 1 MB each from the robot side, through the charging contacts, to the charging station side, passing by the developed board before going to the charging PCB and AEK.

Figure 45 – Test results for connection speed using iperf.



Source: personal archive.

6.1.2 Raspberry Pi with inductor

Results of the test shown in fig. 52. They show a maximum connection speed of 11.9 MBps, or 95.2 Mbps.

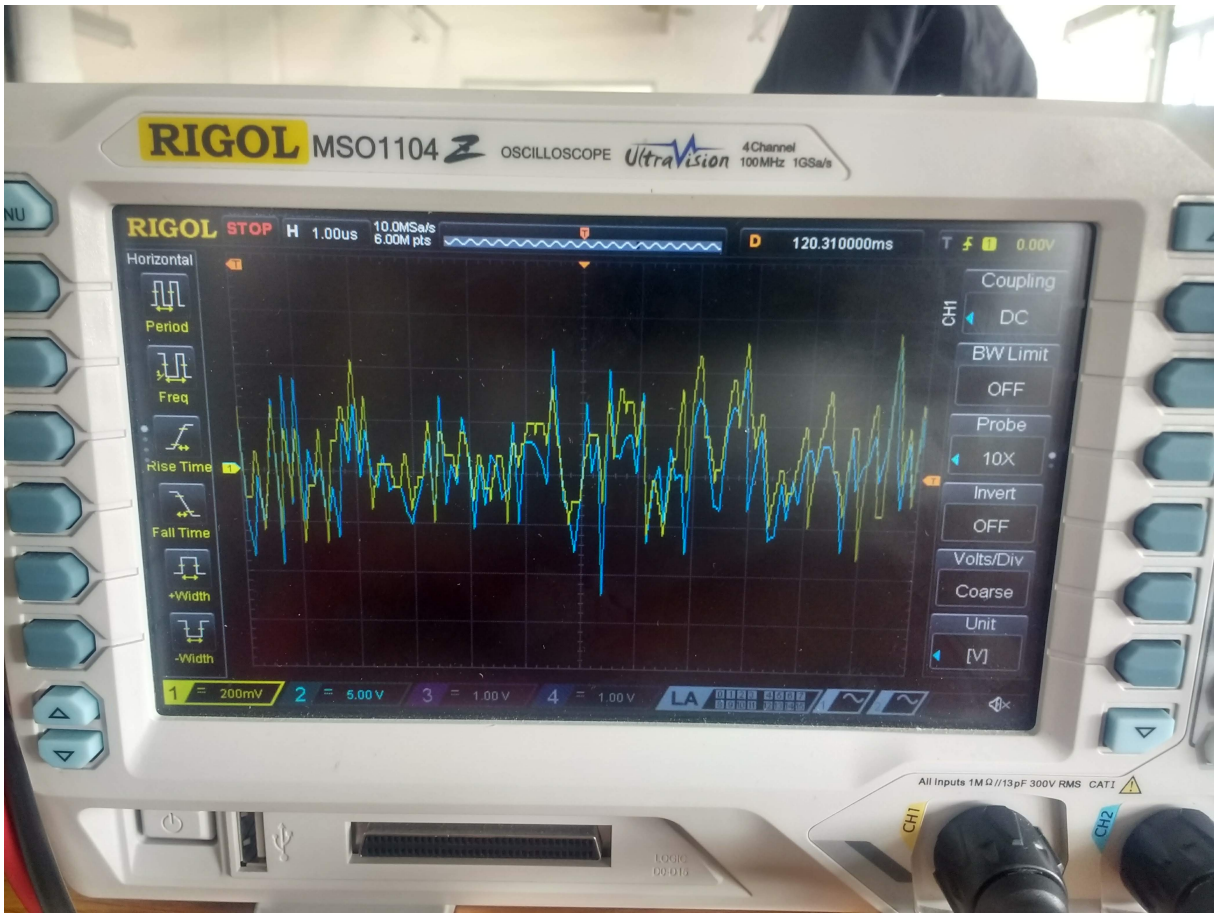
Test method was equal to test without inductor and Raspberry Pi, only with the inductors soldered and 0-ohm resistors changed on the board.

6.1.3 Banana Pi without inductor

Results of the test shown in fig. 53. They changed a lot. In a first sub-test, 76 Mbps were reached in one direction, while 9.3 Mbps were reached in the other. In a following second sub-test, 47.1 Mbps in the one direction and 32.8 Mbps in the other. The first sub-test sent data through 30s in each direction, while the second sub-test sent data for 10s in each direction, both using UDP. The higher speeds were in direction CS to arculee, the signal being sent by the PCB and read by the EVK.

The laptop, connected to the board on the charging station side, would send data for

Figure 46 – Test results for connection speed.



Source: personal archive.

the some time to the Banana Pi, connected to the EVK on the arculee side. At the same time, the Banana Pi would send data for the same duration to the laptop.

The Raspberry Pis were not analysed with the bidirectional method because they do not have the software *iperf*.

6.1.4 Banana Pi with inductor

Results of the test shown in fig. 54. In a first sub-test using UDP, the measured transmission speeds were 71.1 Mbps in one direction with 44.8 Mbps in the other, transmitting for 10s in each. A sub-test using TCP transmitting for 20 seconds resulted in 78.1 Mbps with just under 1 Mbps in the other direction. The higher speeds were in direction CS to arculee, the signal being sent by the PCB and read by the EVK.

The method was equal to test without inductor and Banana Pi, only with the inductors soldered and 0-ohm resistors changed on the board.

Figure 47 – Test results for connection speed.

Source: personal archive.

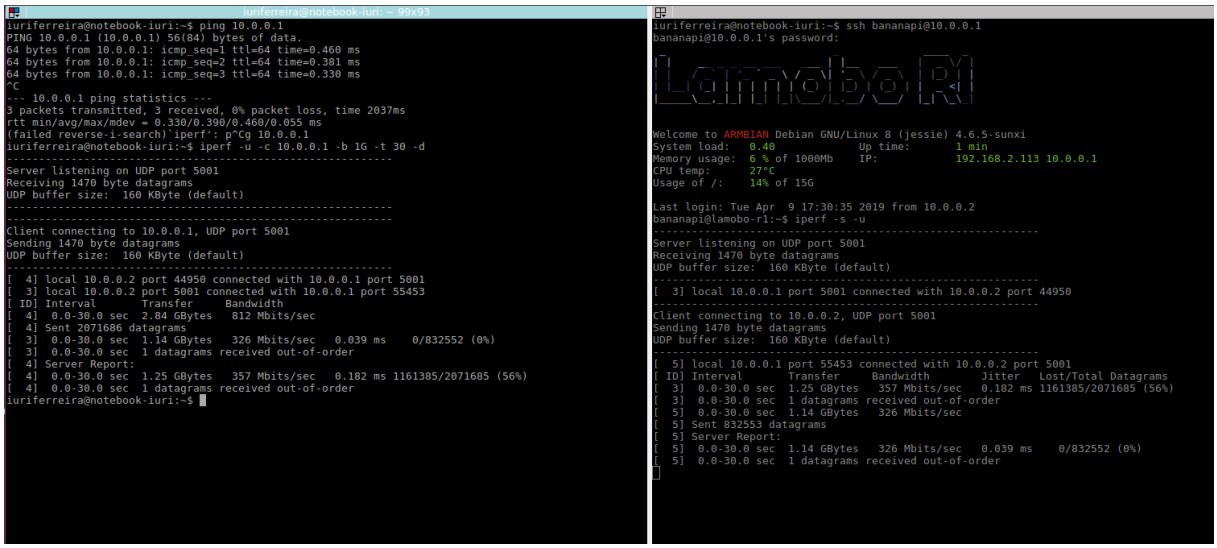
6.1.5 Final comments

Most importantly, the general objective of 40 Mbps has been achieved on most of the tests. Still, more tests should have been done with two of the developed modules communicating with coils on both ends, but the project ended up being put on hold by the company, as it will be discussed in the next chapter. The tests results could have been measured by a tool with a internationally recognised certification to increase their value, but, due to the change on the status of the project, there was no more budget or time for it.

Besides that, four main remarks can be made at this point to the project, first is the lack of a freewheeling diode in the coils. If the robots simply leave the charging station, there can be over 10 A flowing through the coil before it being cut to an open circuit. A freewheeling diode would help prevent any damages to equipment.

Besides the freewheeling diode, as it was already known from the proof of concept in section 5.1, the charging station will not start charging with the modules communicating. What was noticed, however, is that the module on the robot side cannot be connected to the contacts, while the one in the charging station can be connected and operating, without

Figure 48 – Test results for connection speed with direct connection between laptop and Banana Pi.



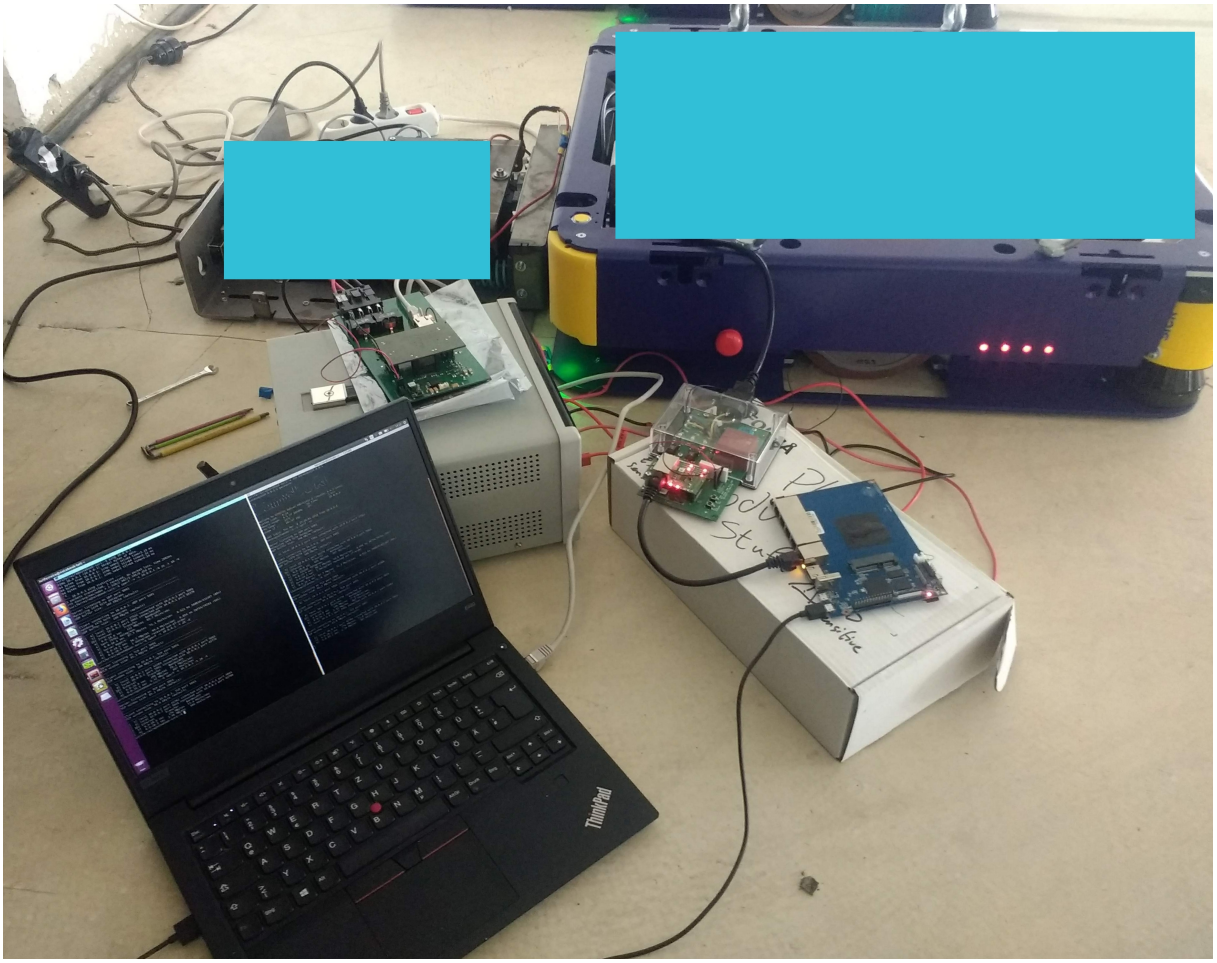
Source: personal archive.

disturbing the charge process.

Then, the tests showed poor communication with bidirectional transmission. This can be caused by the interference in the connector. Finding the reason for this belongs to future works to be done following the project.

At last, the addition of an inductance increased the connection speed by 17 %, maybe limited by having reached the maximum of the protocol available. Being Fast Ethernet instead of Gigabit Ethernet in the Raspberry Pi, the maximum baud rate is 100 Mbps, or 12 MBps; 11.9 MBps were achieved.

Figure 49 – Setup for test with charging, laptop communicating with Banana Pi.

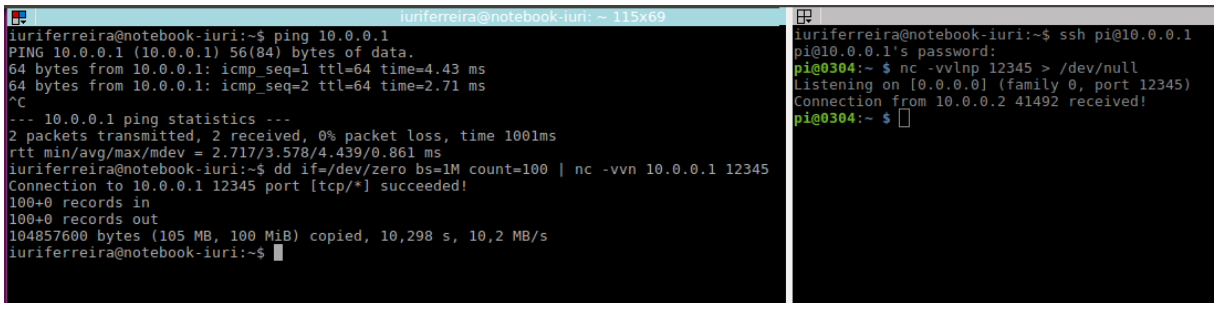


Source: personal archive.

Figure 50 – Setup for test with charging, laptop communicating with Raspberry Pi.

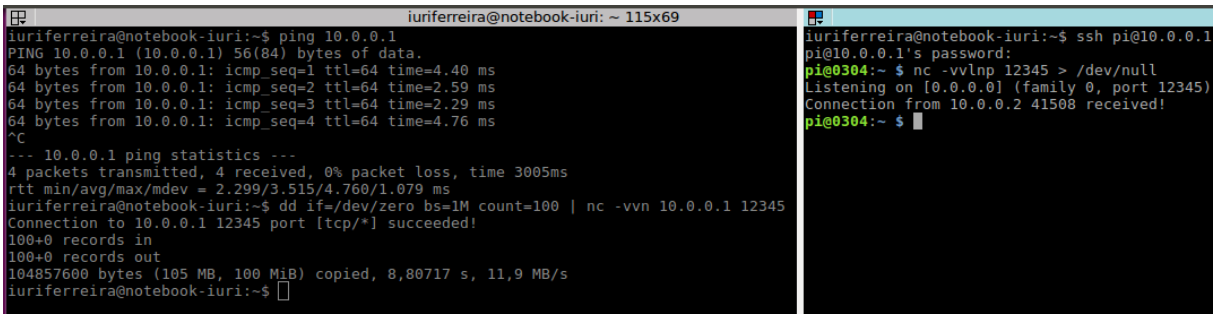
Source: personal archive.

Figure 51 – Results of test with Raspberry Pi without inductors.



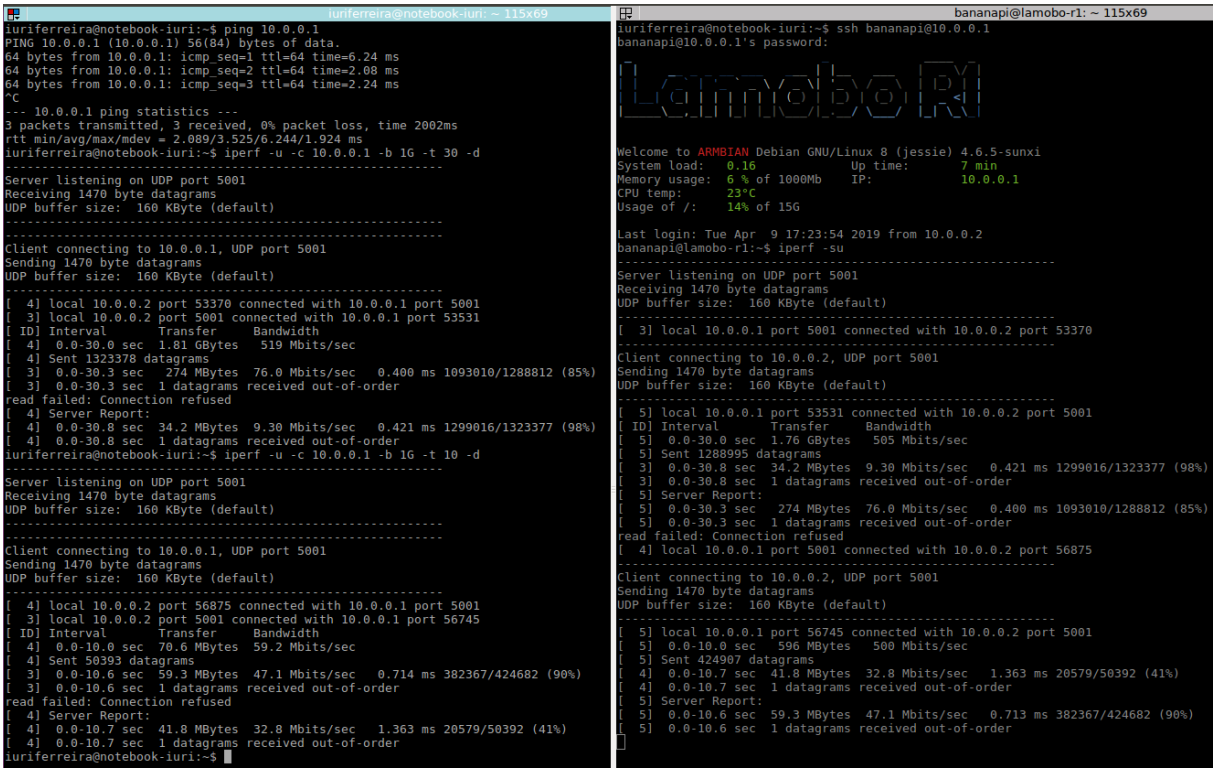
Source: personal archive.

Figure 52 – Results of test with Raspberry Pi with inductors.



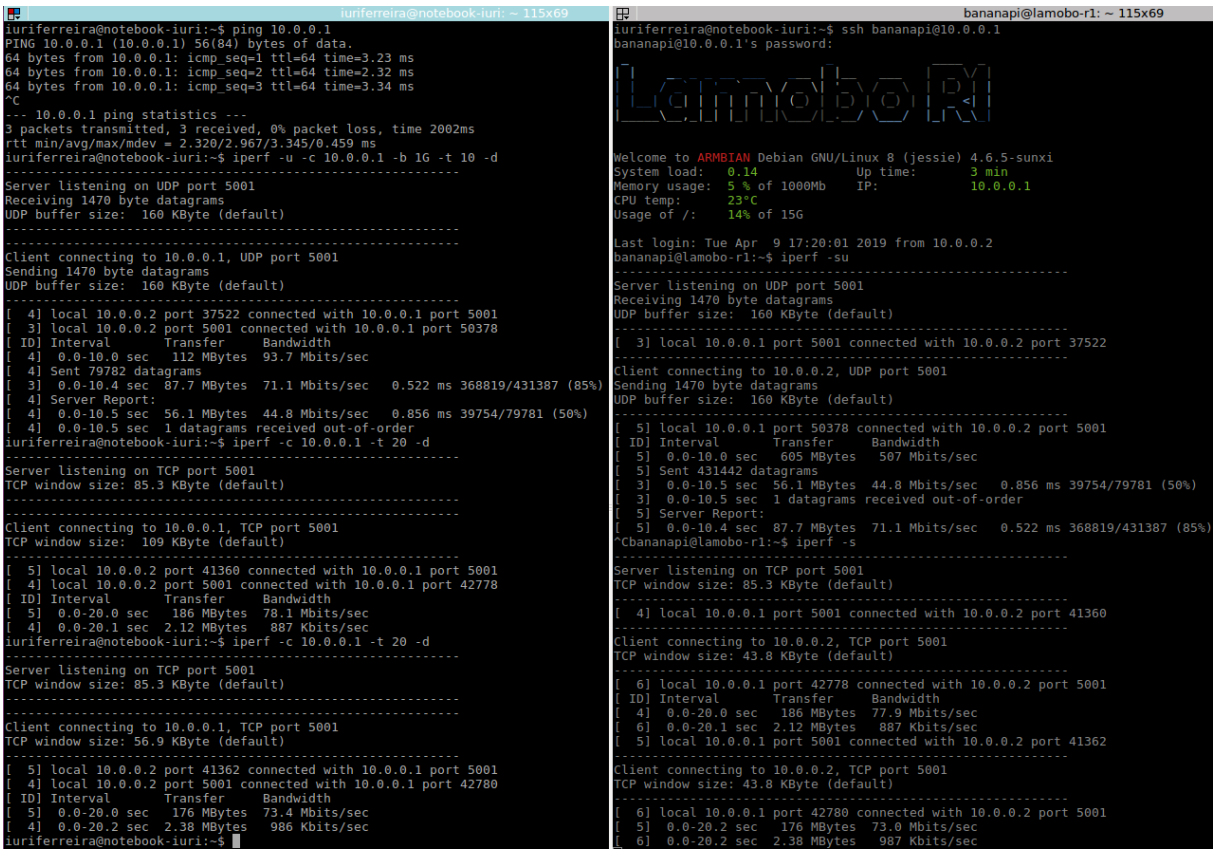
Source: personal archive.

Figure 53 – Results of test with Banana Pi without inductors.



Source: personal archive.

Figure 54 – Results of test with Banana Pi with inductors.



Source: personal archive.

7 Conclusion

Starting the project with a research of the available technologies in the literature for PLC gave the student a basis knowledge on the topic and on what to look for. The following phase of trying to design a circuit capable of holding a communication over power-line further expanded the knowledge of the student, even though it did not really bring any concrete results to the project. Then, the research of the available technologies in the market is what really set the base for the PLC and thus for the project. Following the market research came the proof of concept with the evaluation kits and the actual design of the circuit boards until a working prototype.

Flowing almost in parallel to this were the advancements on the software side. Initially with a simple improvement of what had already being implemented, followed by a bigger restructuring of the whole code, which then allowed for it to be expanded to the new, basic use cases brought by the communication with the robot, and allowing for a smoother addition of the more advanced use cases in the future.

With regard to the goals of the project, most of them were achieved, with the charging station having a working, improved software and a working power-line-capable communication bus with speeds well above the requirements. The goals which were not achieved were proposing changes to the AGV software based on the new use cases and a final test of the whole system as a final proof of its working. The priority of the proposed changes has been reduced, as the whole communication protocol is expected to undergo alterations, besides the fact that this task would now be executed by the team as a whole when the need for extra functions come, instead of the student alone. The final, full test is most likely to be performed in the following weeks, probably still by the student, but not under the time or contract for the thesis and thus has been left out and should be seen as a future work.

The challenges faced during the development of the project started with the proof of concept, when the suspicion was confirmed that the power supply, the AEK, has a low-pass filter that severely reduces the communication baud rate. To handle this, the principle of the bias T circuit [14] was added with the inductances. The capacitor for the high-pass filters are already part of the inductive coupling of the PLC circuitry. This, as seen in the section 6, has been well overcome with the addition of the inductors and could be further improved by having two more inductors on the other side, which would isolate the effects of the battery and of the Schottky diode from the communication line.

Besides that, the next greater challenge faced was soldering the PLM onto the board. The board of the PLM, fig. 28 and 29, has a little curvature to it due to its imperfections. This curvature causes some pads to be left unsoldered to the board underneath the module,

issue discussed in chapter 5. This is hard to spot and one possibility of solving this issue could be soldering little metal cubes to each pad and then soldering these cubes onto the board. This makes it easier to find issues as it allows for the solderer to see if any pad is floating or if there are any short-circuits between the pads. These can be caused by applying too much solder paste on the pads or by pressing the PLM against the board while the paste is still fluid. The problem was handled in the project by soldering it once, unsoldering the PLM, adding more solder paste to the pads without solder and re-soldering the PLM onto the board, then passing it through a reflow oven several times with flux. This process is extremely costly and therefore cannot be used on production scale.

As a last challenge was the time taken for each iteration. Alone the manufacturing of the each version of the board took over one week, plus the time to discuss the design with the supervisors and the time to find errors, it all summed up to just over one month for each version of the board. On the beginning of the project, this time allowed the student to work on the software in parallel to the hardware, however, already early in the development the software was ready and the project went through numerous waiting phases. The effect of this in the final project was mostly the lack of time for a thorough test.

Regarding the challenges found, it can be said that they did not affect the project itself as much as they will affect the future implementations of it in productive scale. Thus leading into the future works section, which include the previously-mentioned full test, a remodelling the prototype board to have the PLM soldered onto the same board as the PHY, as well as to reduce its size into a board that fits inside of the robot. Plus the implementation of the additional functions and changes to the protocol as needed in the charging station, and implementing all functions and the protocol needed in the AGV software. Other items to be added to the board are a ground connection to the casing of the RJ-45 connector and a freewheeling diode to each coil.

A final note about the effect of the project in the company and in the process should be given. To this project, it must be said that the company is still pondering the use of this technology for the charging station, as there are other technologies being researched, such as installing a local 5G network. Unlike 4G technology for telecommunications, the 5G technology allows the creation of closed local networks with all the advantages of 5G technology, such as low latency and high bandwidth [22]. Still, with many robots charging and all using the same network to transmit gigabytes of data, it may cause some issues and slower connections and for such cases the power-line option is very interesting. Besides that, the technology can be applied to virtually any other power-lines or two cables in use or not, allowing for a high-speed, 2-cable connection based on TCP and UDP-based protocols. In summary, even though it is still unsure whether the technology will be in fact applied to the robot or not, arculus GmbH has now the technology and can apply it if the need for it comes. Other applications could also allow for 3-cable connections, which allow for doubled

baud rate with the addition of a second power-line channel, available in the PLM.

Bibliography

1 ROGERS, G. G.; BOTTACI, L. Modular production systems: a new manufacturing paradigm. *Journal of Intelligent Manufacturing*, Springer, v. 8, p. 147–156, 1997. Cited on page 23.

2 BANANA PI WIKI. *Banana Pi BPI-R1*. http://wiki.banana-pi.org/Banana_Pi_BPI-R1#Resources/, 2018. Access on 08/02/2019. Cited 2 times on pages 26 and 27.

3 NOTTEN, P. H. L.; VELD, J. H. G. O. het; BEEK, J. R. G. van. Boostcharging li-ion batteries: A challenging new charging concept. *Journal of Power Sources*, Elsevier, v. 145, p. 89–94, 2005. Cited 2 times on pages 32 and 33.

4 BARRé, A. et al. A review on lithium-ion battery ageing mechanisms and estimations for automotive applications. *Journal of Power Sources*, Elsevier, v. 241, p. 680–689, 2013. Cited on page 35.

5 CARRANO, J. C.; MALTENFORT, A. J. Semiconductor ultraviolet optical sources for biological agent detection. *Unattended Ground Sensor Technologies and Applications IV*, SPIE, v. 4743, 2002. Cited on page 42.

6 YONGE, L. et al. An overview of the homeplug av2 technololgy. *Journal of Electrical and Computer Engineering*, Hindawi, v. 2013, 2013. Cited on page 42.

7 BOIS, B. D. et al. Does god class decomposition affect comprehensibility. Universiteit Antwerpen, 2006. Cited on page 44.

8 OLBRICH, S. M.; CRUZES, D. S.; SJøBERG, D. I. K. Are all code smells harmful? a study of god classes and brain classes in the evolution of three open source systems. In: *2010 IEEE International Conference on Software Maintenance*. [S.l.: s.n.], 2010. p. 1–10. ISSN 1063-6773. Cited on page 44.

9 MODEL-VIEW-CONTROLLER. *Enterprise Solution Patterns Using Microsoft .NET*, Microsoft, 2003. Cited on page 45.

10 OBSERVER. *Enterprise Solution Patterns Using Microsoft .NET*, Microsoft, 2003. Cited on page 45.

11 HANNEMANN, J.; KICZALES, G. Design pattern implementation in java and aspectj. In: *OOPSLA '02 Proceedings of the 17th ACM SIGPLAN conference on OOP, sys., lang., and app.* [S.l.]: SIGPLAN, 2002. v. 37, n. 11, p. 161–173. Cited on page 45.

12 JAEGER, T. *The State Design Pattern vs. State Machine*. https://www.codeproject.com/Articles/509234/The-State-Design-Pattern-vs-State-Machine, 2013. Access on 13/02/2019. Cited on page 49.

13 COSTA, L. G. d. et al. Coupling for power line communications: A survey. *Journal of Communication and Information Systems*, v. 32, n. 1, Mar. 2017. Cited on page 52.

14 BAUMGARTNER, A. R.; KORNBICHLER, J. W. W. High-power high-bandwidth linear driving circuit for vlc applications. *IEEE 802.15.7 D2*, IEEE, Mar. 2010. Cited 2 times on pages 52 and 93.

15 Grassi, F.; Pignari, S. A. Coupling/decoupling circuits for powerline communications in differential dc power buses. In: *2012 IEEE International Symposium on Power Line Communications and Its Applications*. [S.l.: s.n.], 2012. p. 392–397. Cited on page 52.

16 COSTA, L. et al. Coupling for power line communication: A survey. *Journal of Communication and Information Systems*, v. 32, 01 2017. Cited on page 52.

17 BILAL, O. et al. Design of broadband coupling circuits for power line communication. 01 2004. Cited on page 52.

18 Pinomaa, A. et al. Homeplug green phy for the lvdc plc concept: Applicability study. In: *2015 IEEE International Symposium on Power Line Communications and Its Applications (ISPLC)*. [S.l.: s.n.], 2015. p. 205–210. Cited on page 52.

19 Lienard, M. et al. Modeling and analysis of in-vehicle power line communication channels. *IEEE Transactions on Vehicular Technology*, v. 57, n. 2, p. 670–679, March 2008. ISSN 0018-9545. Cited on page 52.

20 Tomba, L. On the effect of wiener phase noise in ofdm systems. *IEEE Transactions on Communications*, v. 46, n. 5, p. 580–583, May 1998. ISSN 0090-6778. Cited on page 58.

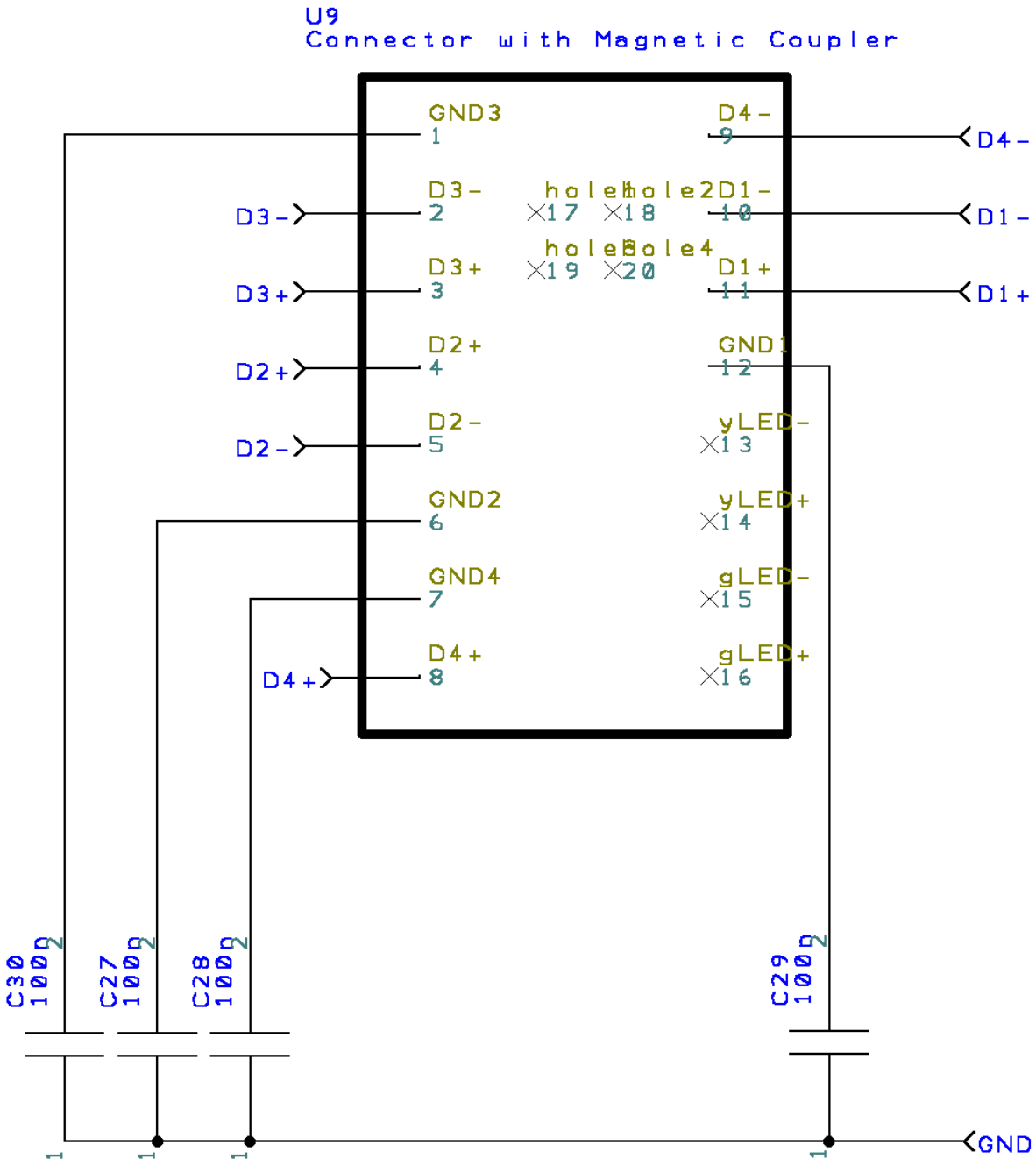
21 SAIDANI, F. et al. Lithium-ion battery models: a comparative study and a model-based powerline communication. In: *Advances in Radio Sciences Open Access Proceedings*. [S.l.: s.n.], 2017. p. 83–91. Cited on page 59.

22 HOSSAIN, S. 5gwireless communication systems. *AJER*, AJER, v. 02, n. 10, p. 344–353, 2013. ISSN 2320-0847. Cited on page 94.

Appendix

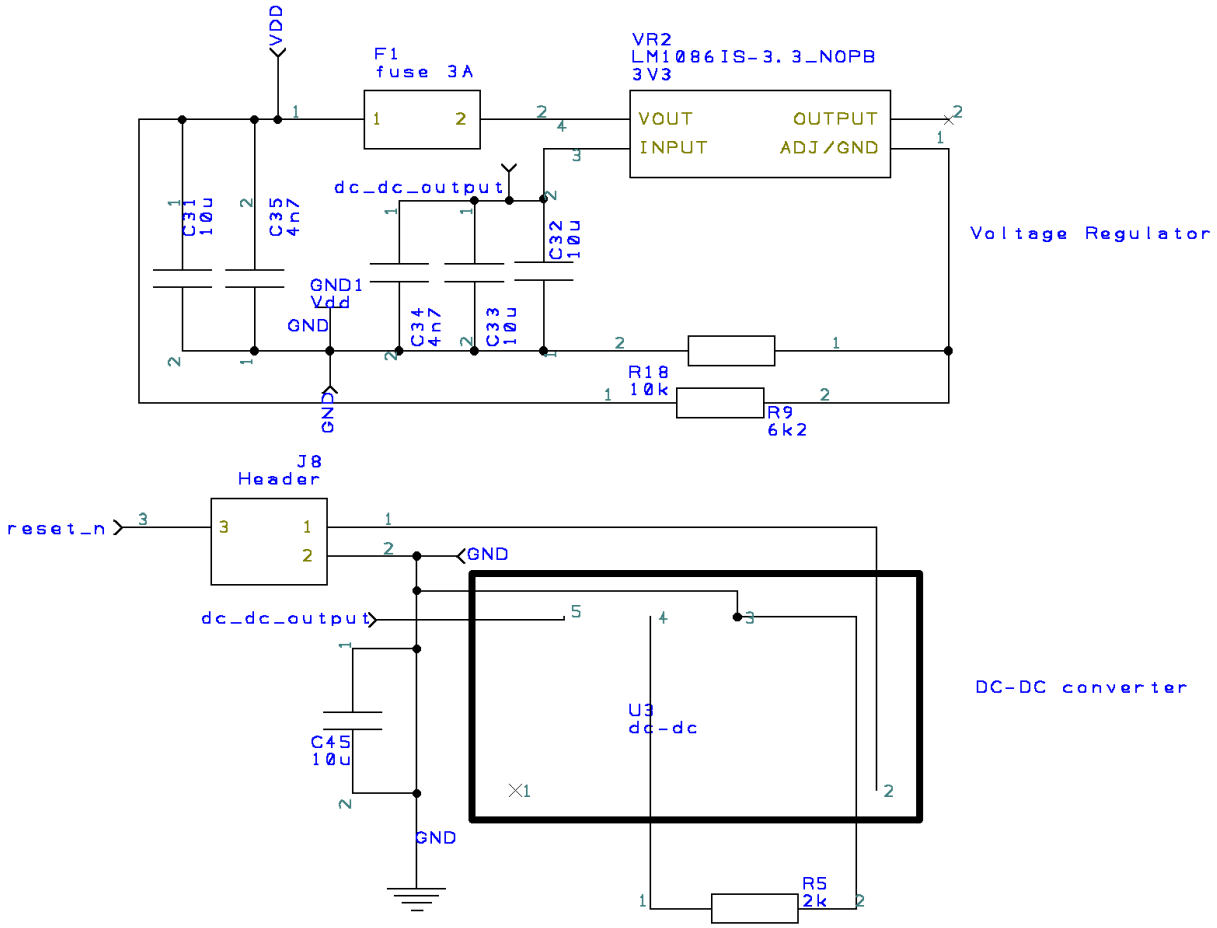
APPENDIX A – First version - Schematics and PCB

Figure 55 – Schematics of first version, RJ-45 connector.



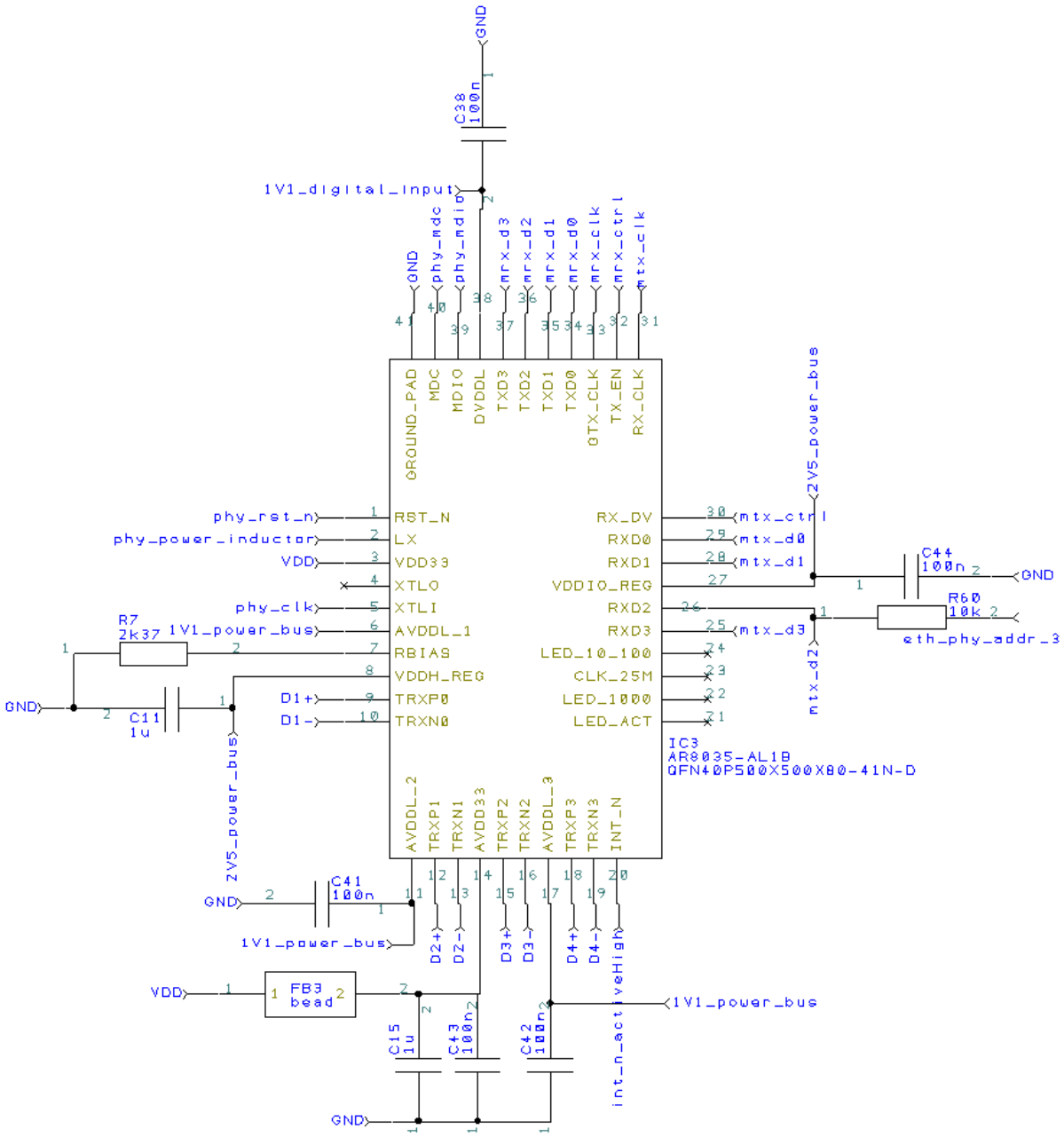
Source: personal archive.

Figure 56 – Schematics of first version, power supply.



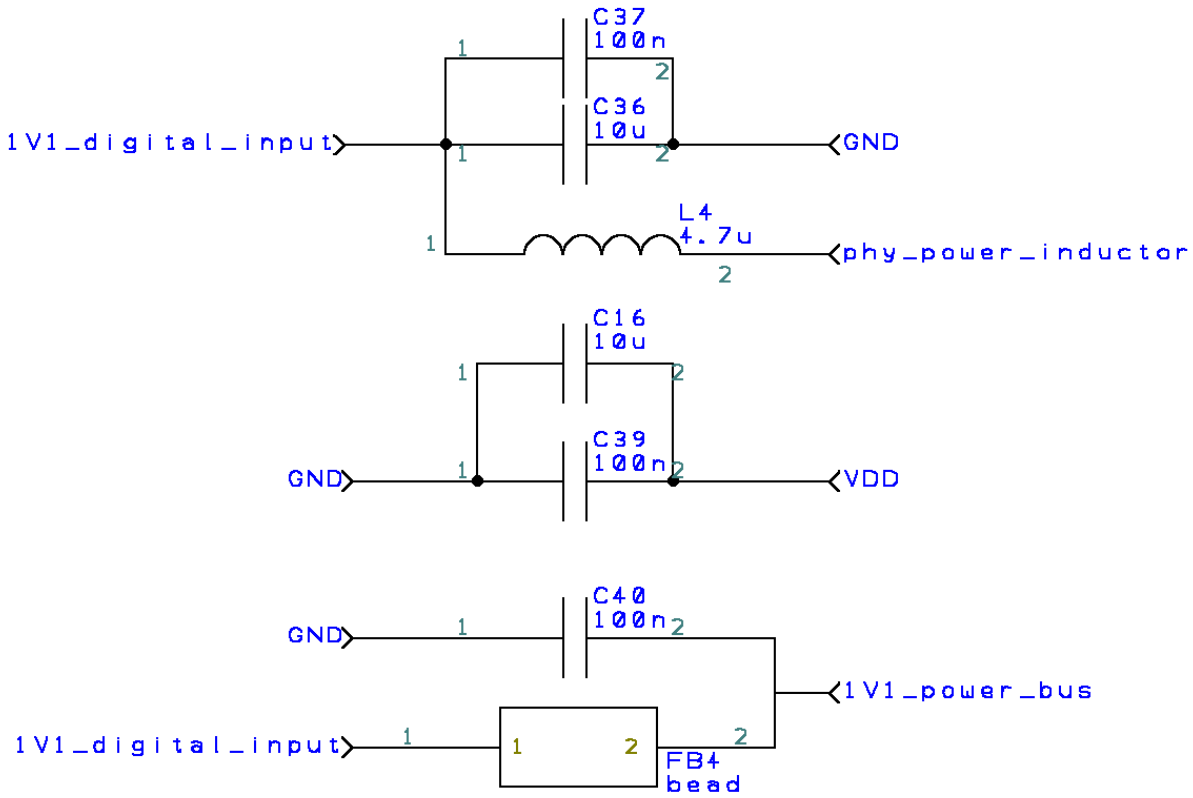
Source: personal archive.

Figure 57 – Schematics of first version, PHY chip.



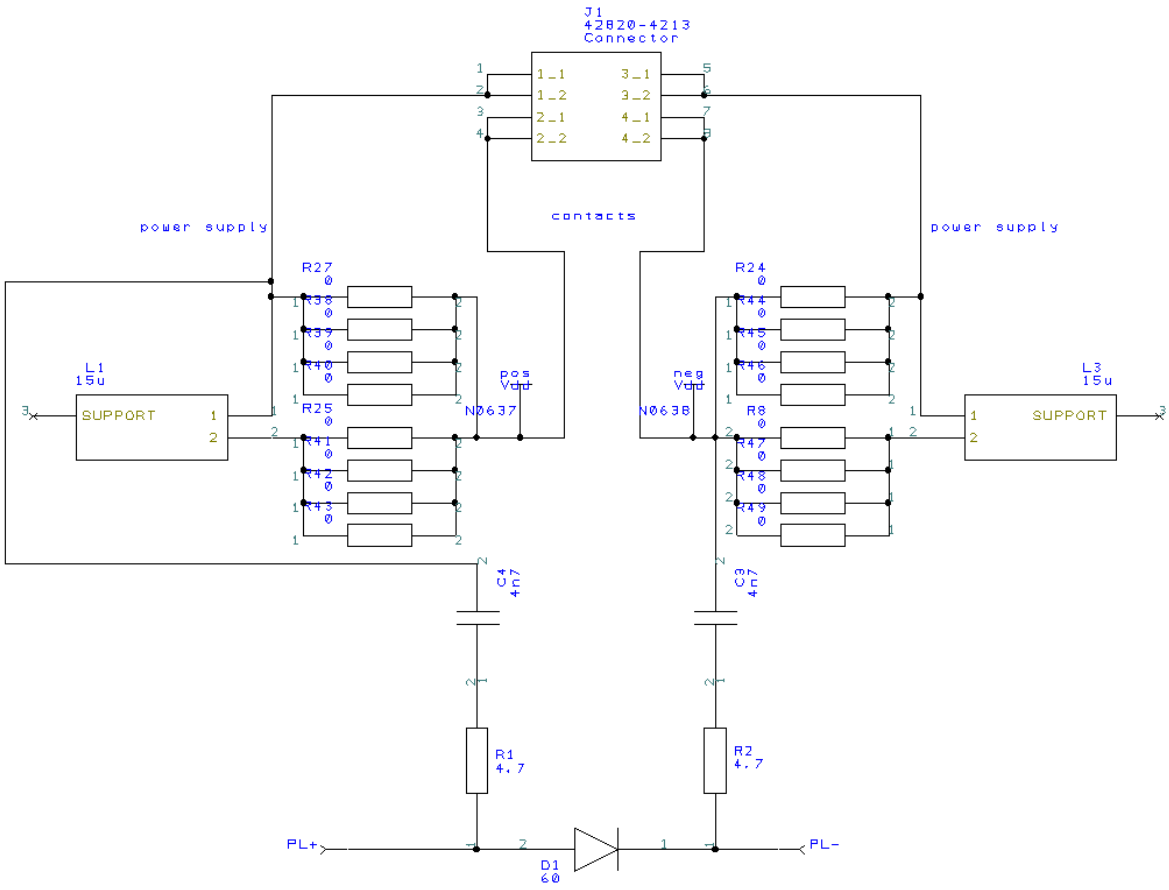
Source: personal archive.

Figure 58 – Schematics of first version, components around the PHY chip.



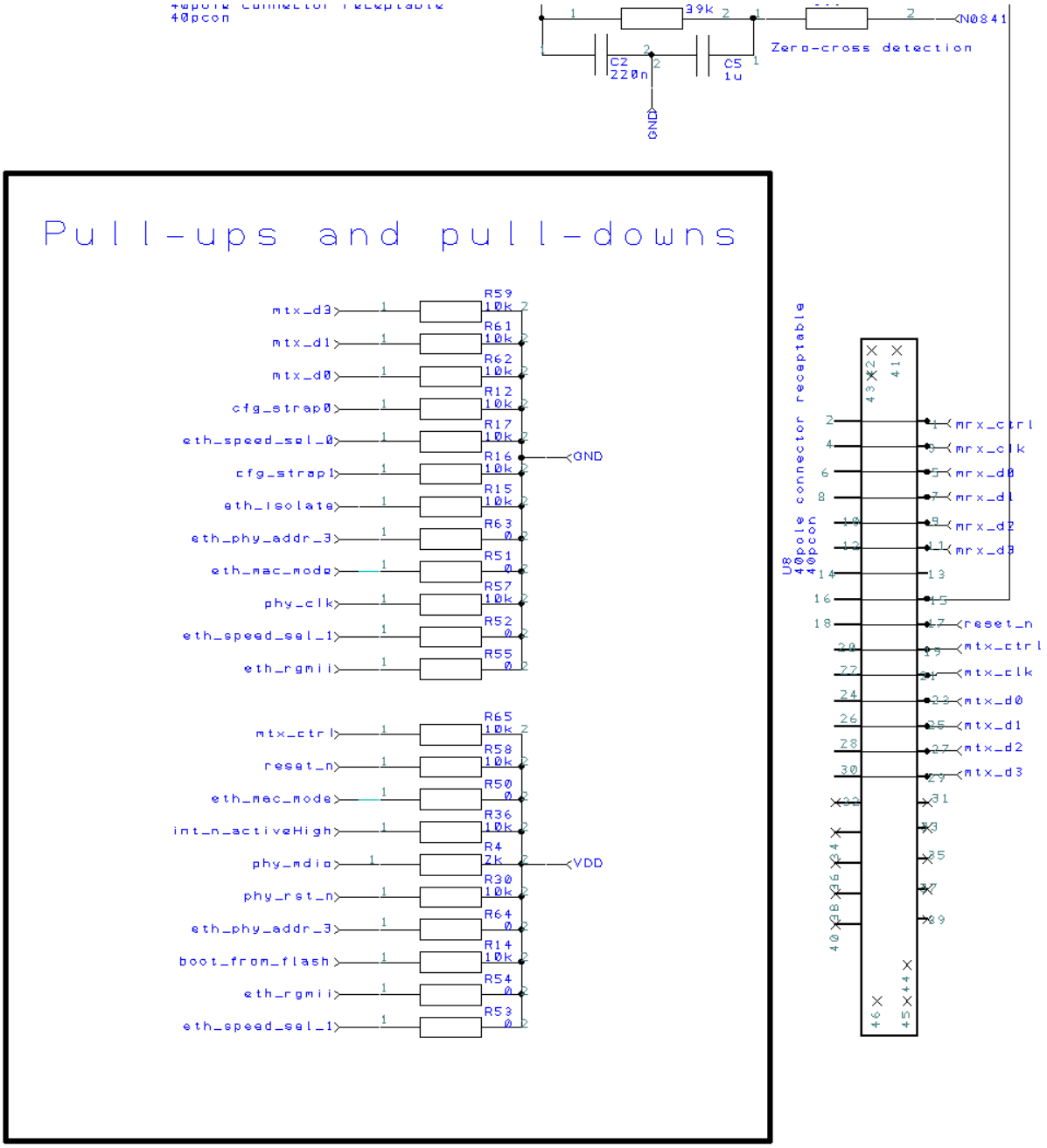
Source: personal archive.

Figure 59 – Schematics of first version, coupling circuit.



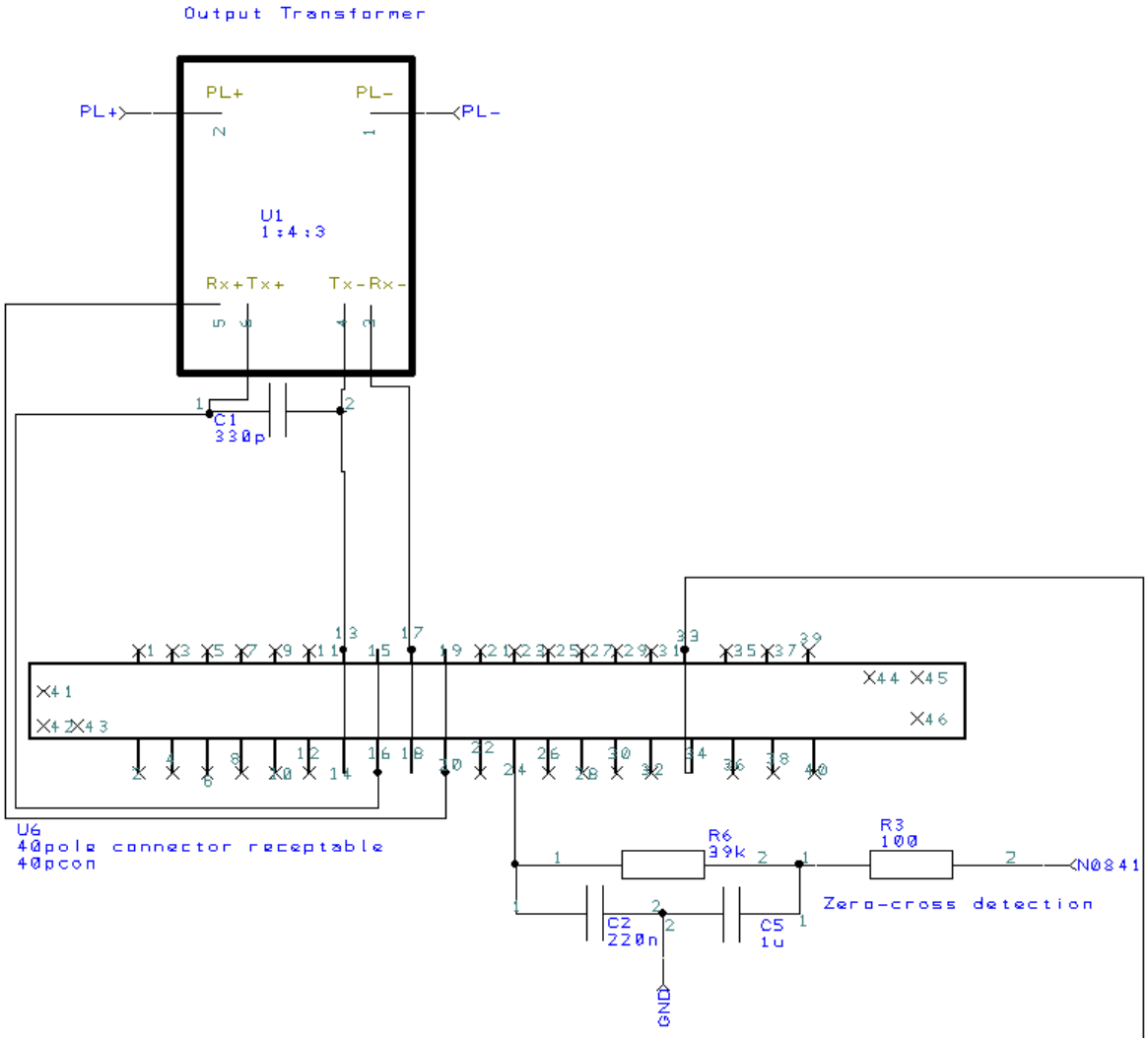
Source: personal archive.

Figure 60 – Schematics of first version, first header and bootstrap configuration.



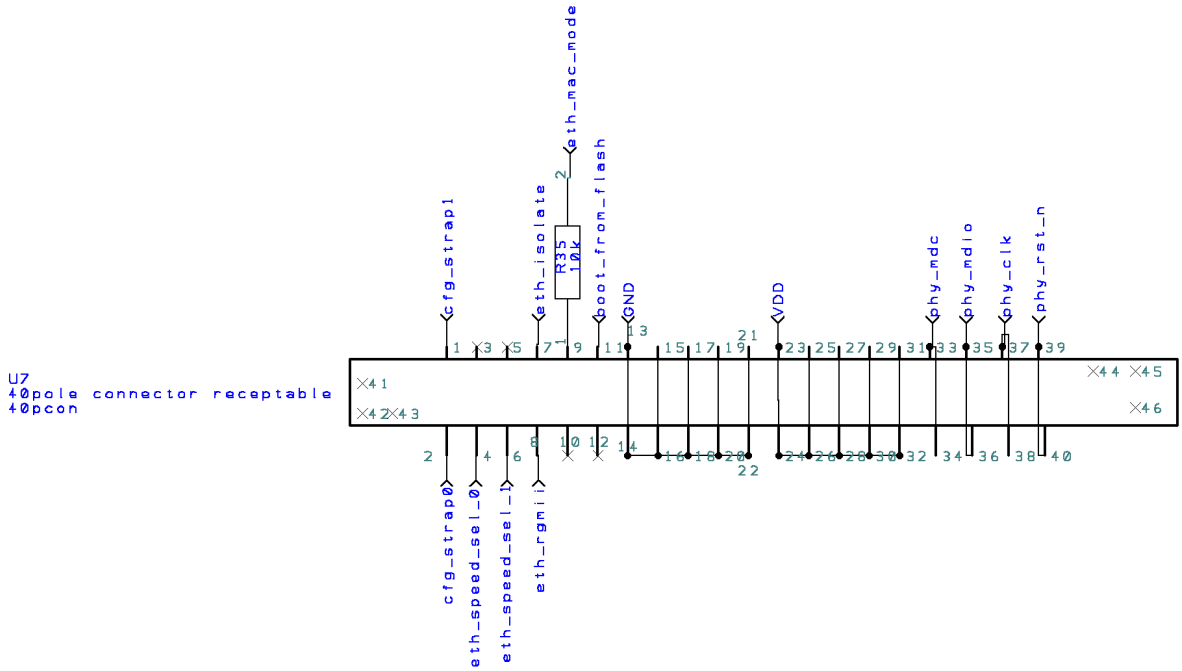
Source: personal archive.

Figure 61 – Schematics of first version, second header, coupling transformer and zero cross detection.



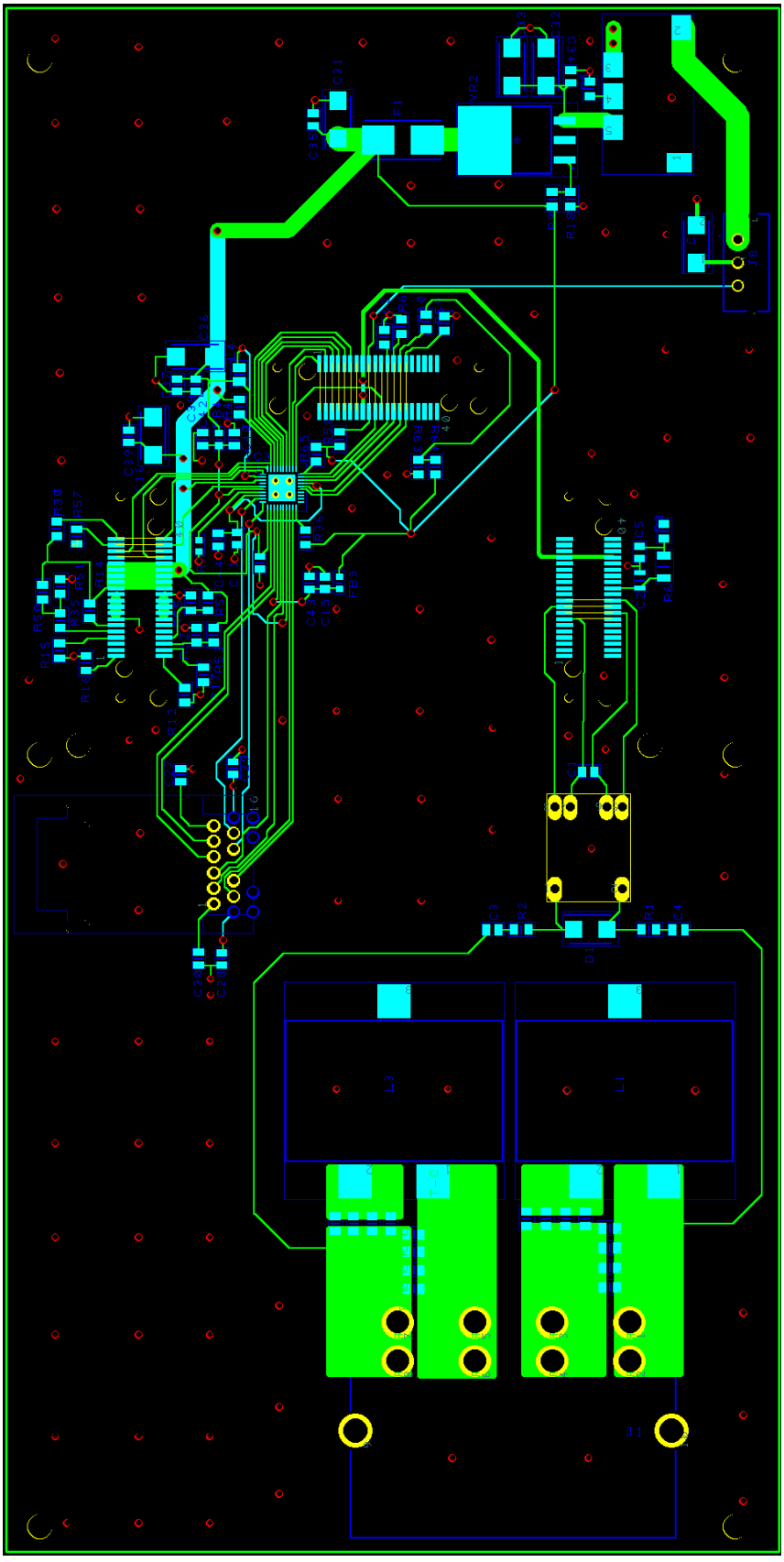
Source: personal archive.

Figure 62 – Schematics of first version, third header.



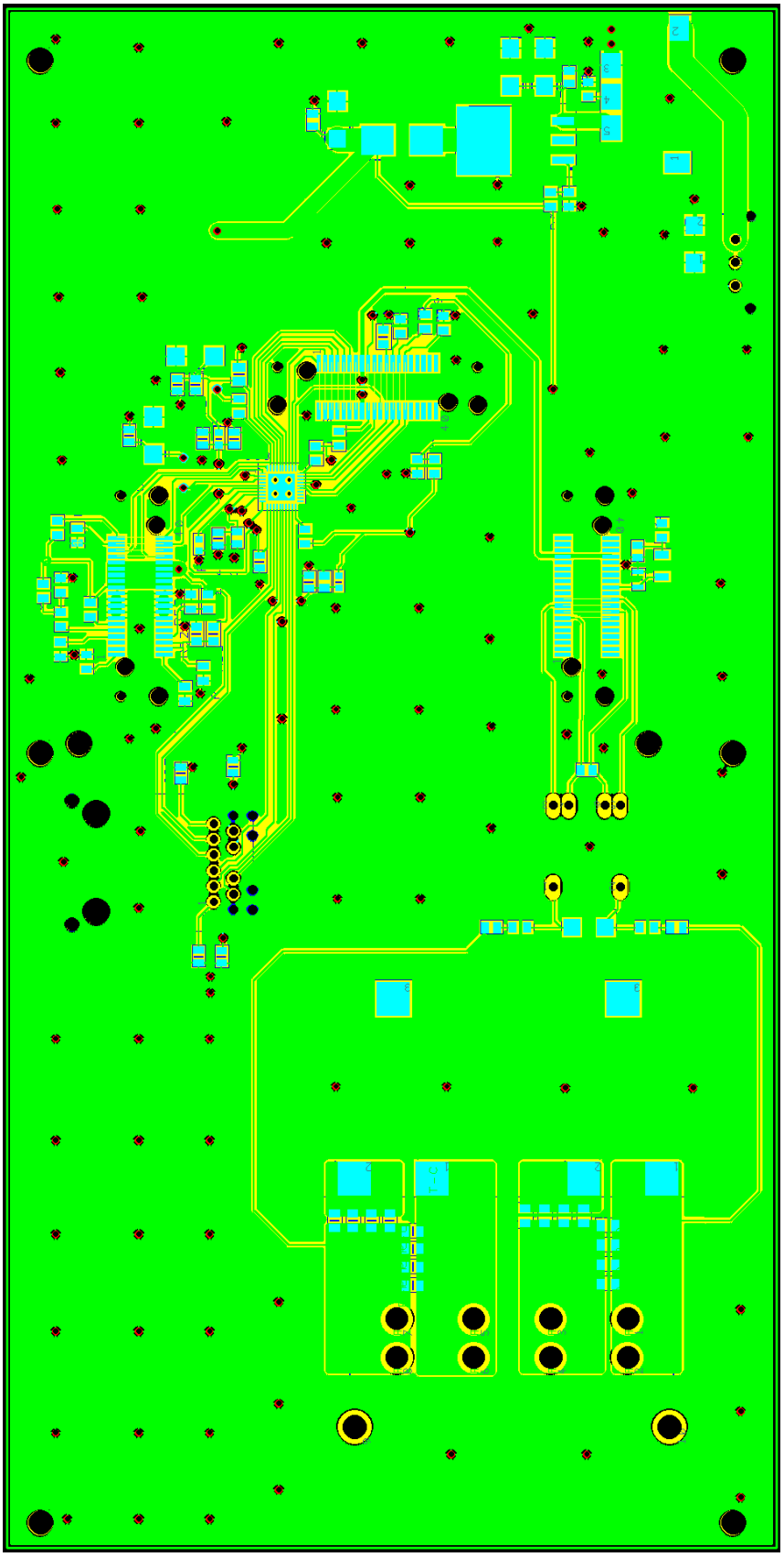
Source: personal archive.

Figure 63 – First version of PCB, PCB design without poured copper as ground.



Source: personal archive.

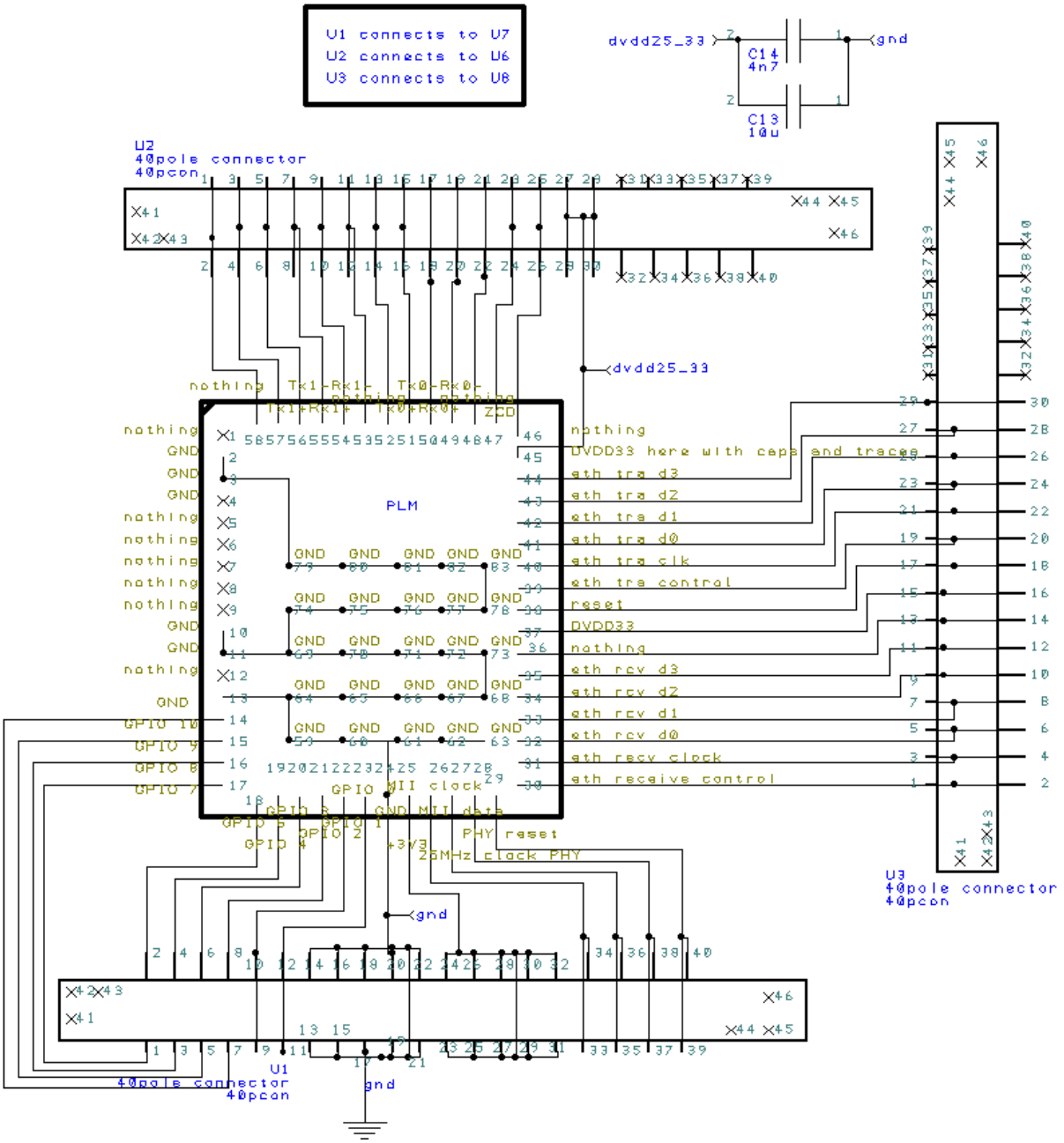
Figure 64 – First version of PCB, PCB design with copper pour.



Source: personal archive.

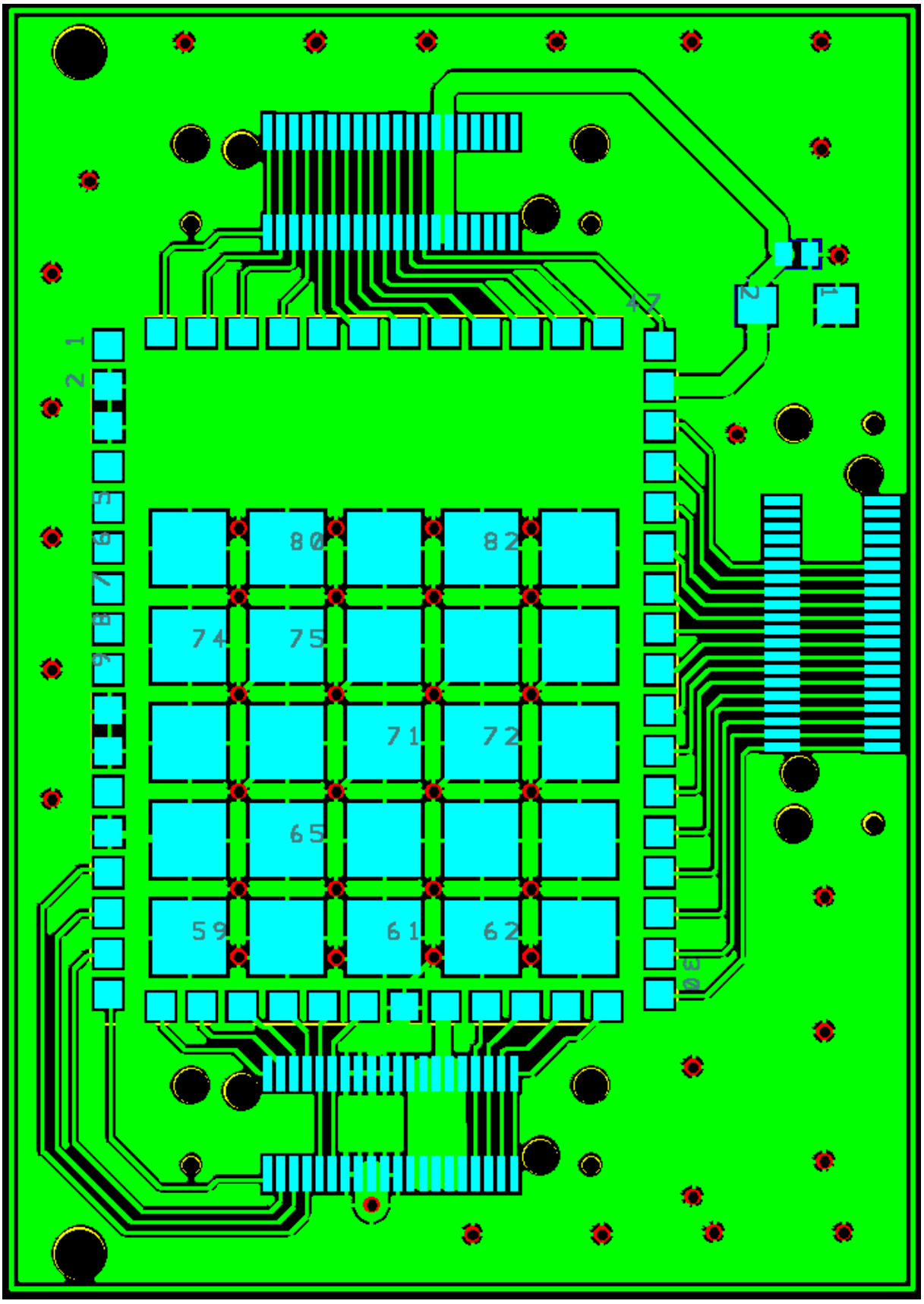
APPENDIX B – PLM Adapter Board

Figure 65 – Schematics of PLM adapter board.



Source: personal archive.

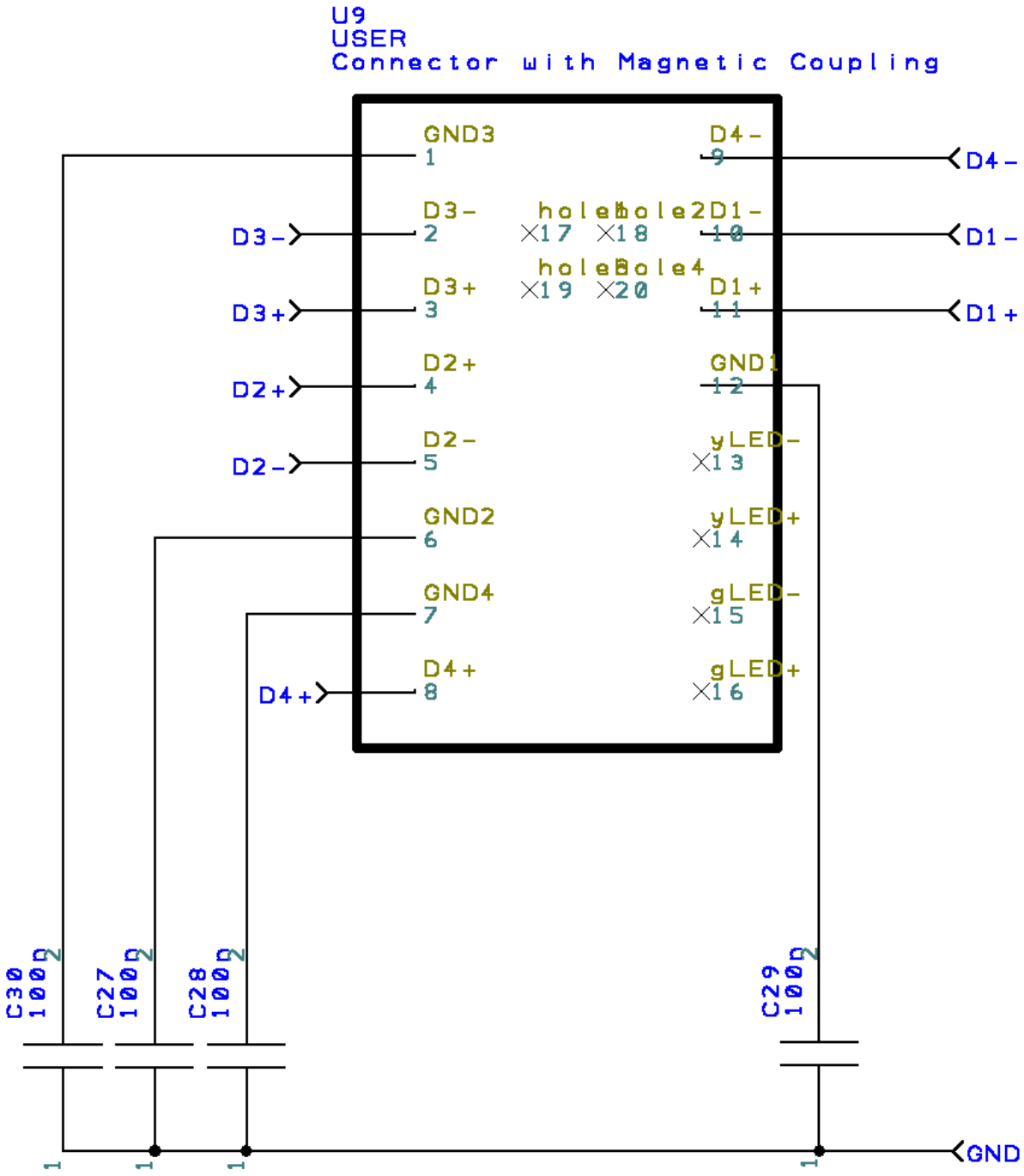
Figure 66 – PCB design for PLM adapter borad.



Source: personal archive.

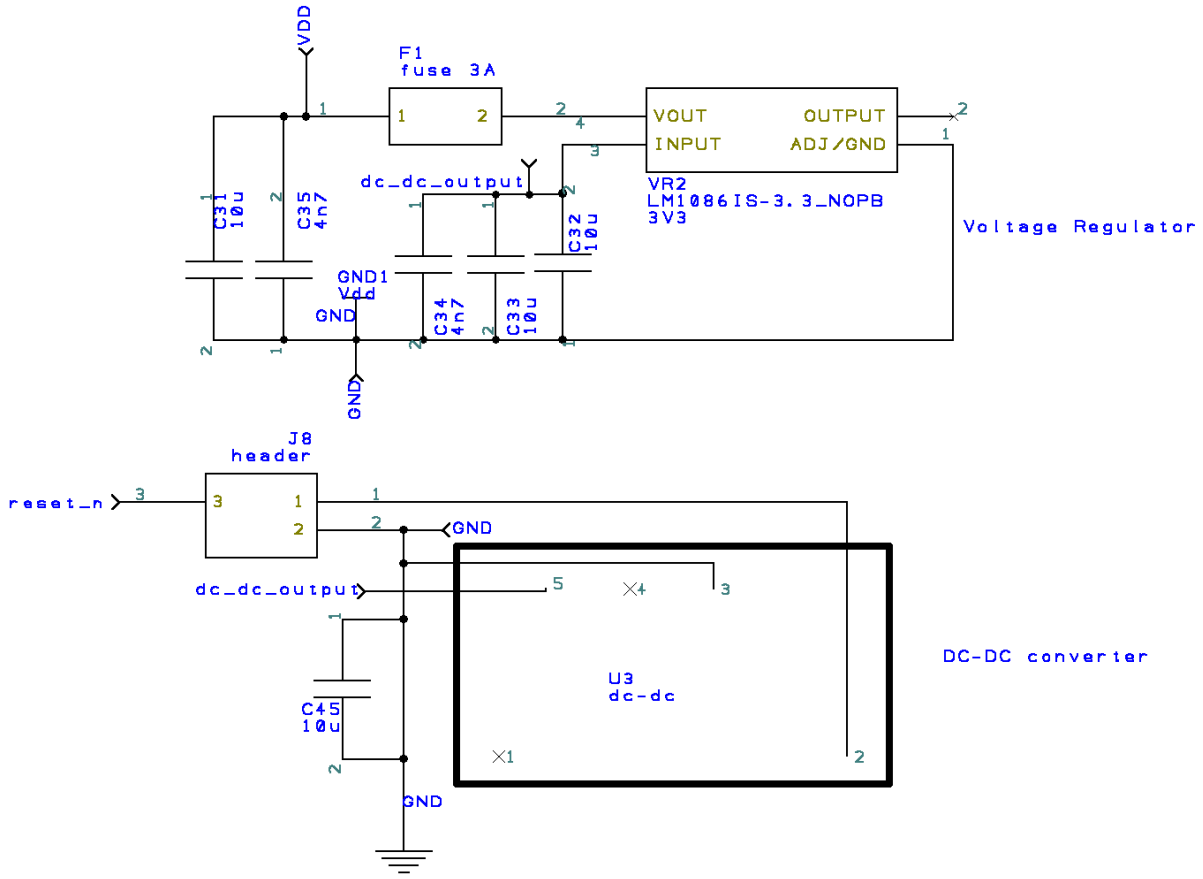
APPENDIX C – Second version - Schematics and PCB

Figure 67 – Schematics of second version, RJ-45 connector.



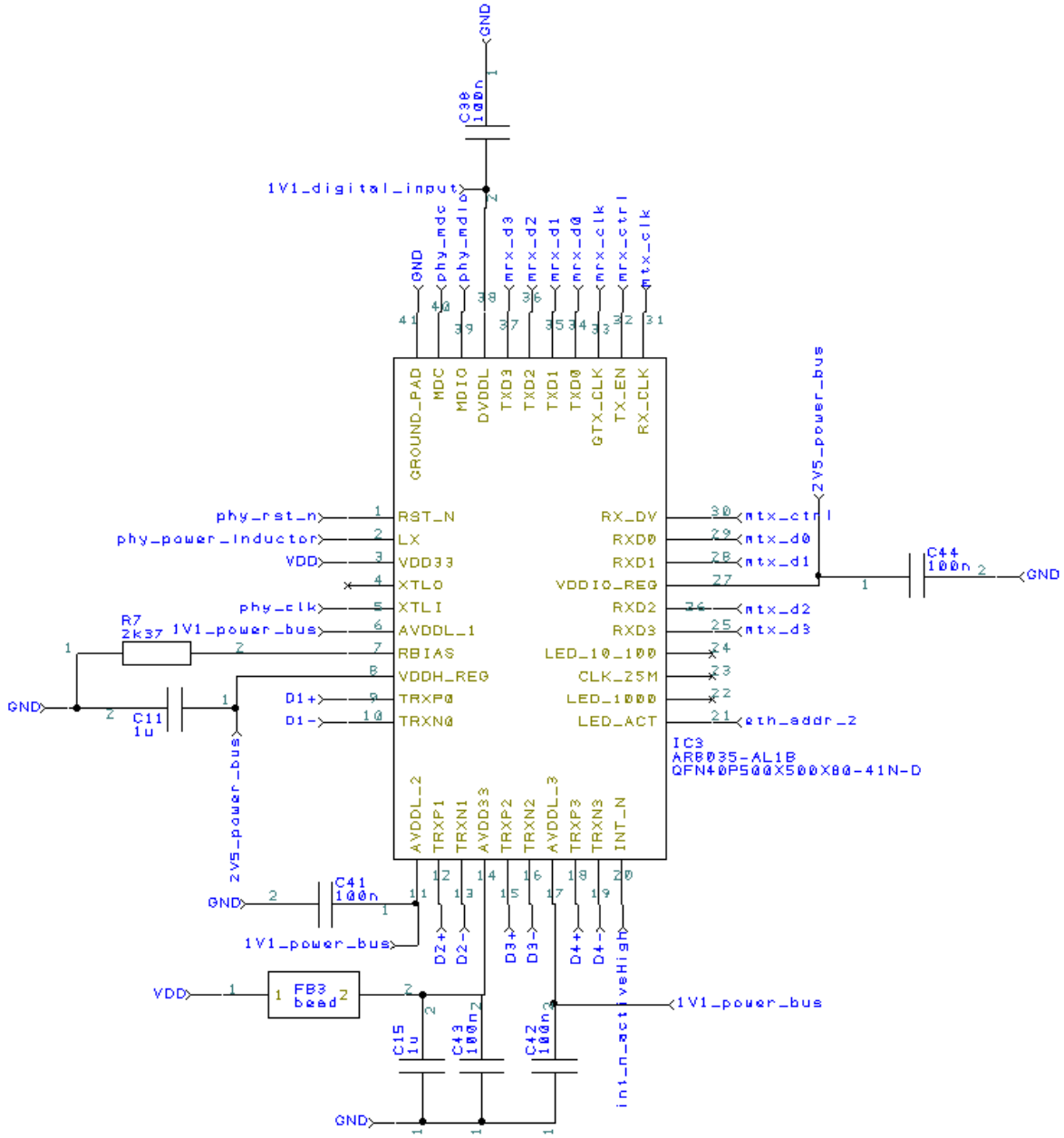
Source: personal archive.

Figure 68 – Schematics of second version, power supply.



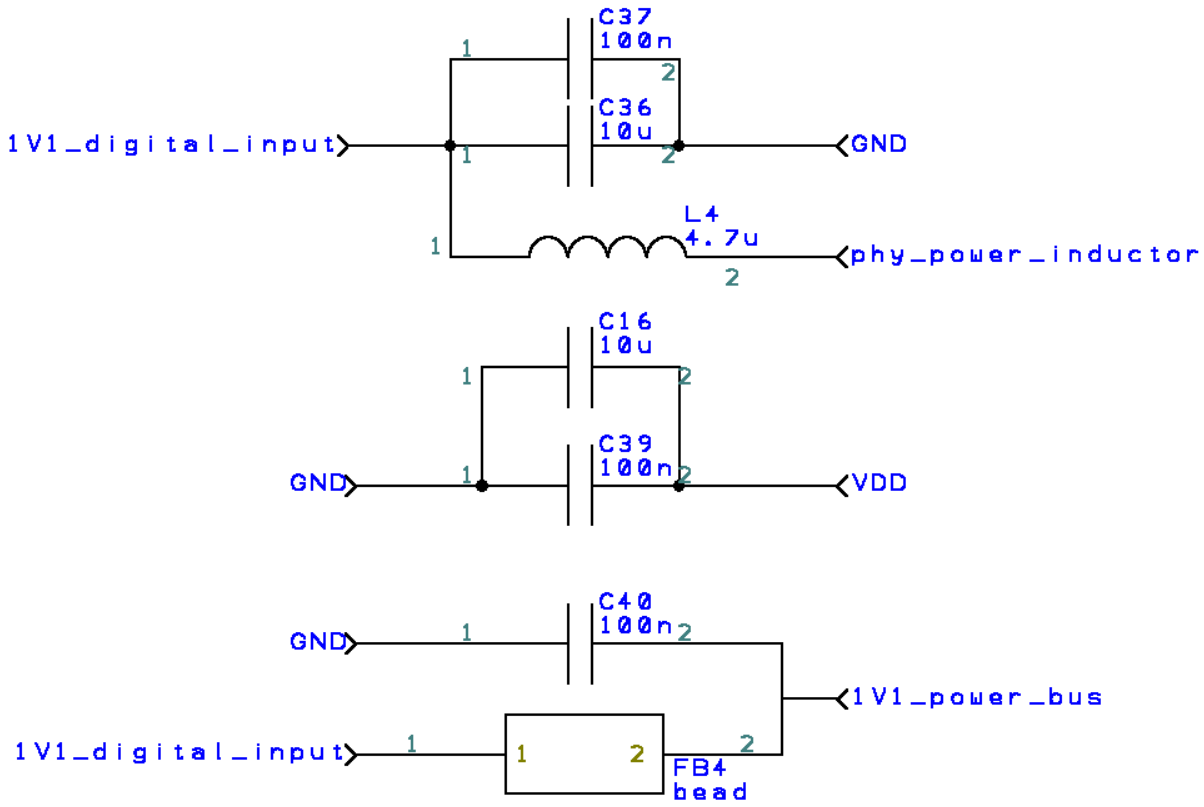
Source: personal archive.

Figure 69 – Schematics of second version, PHY chip.



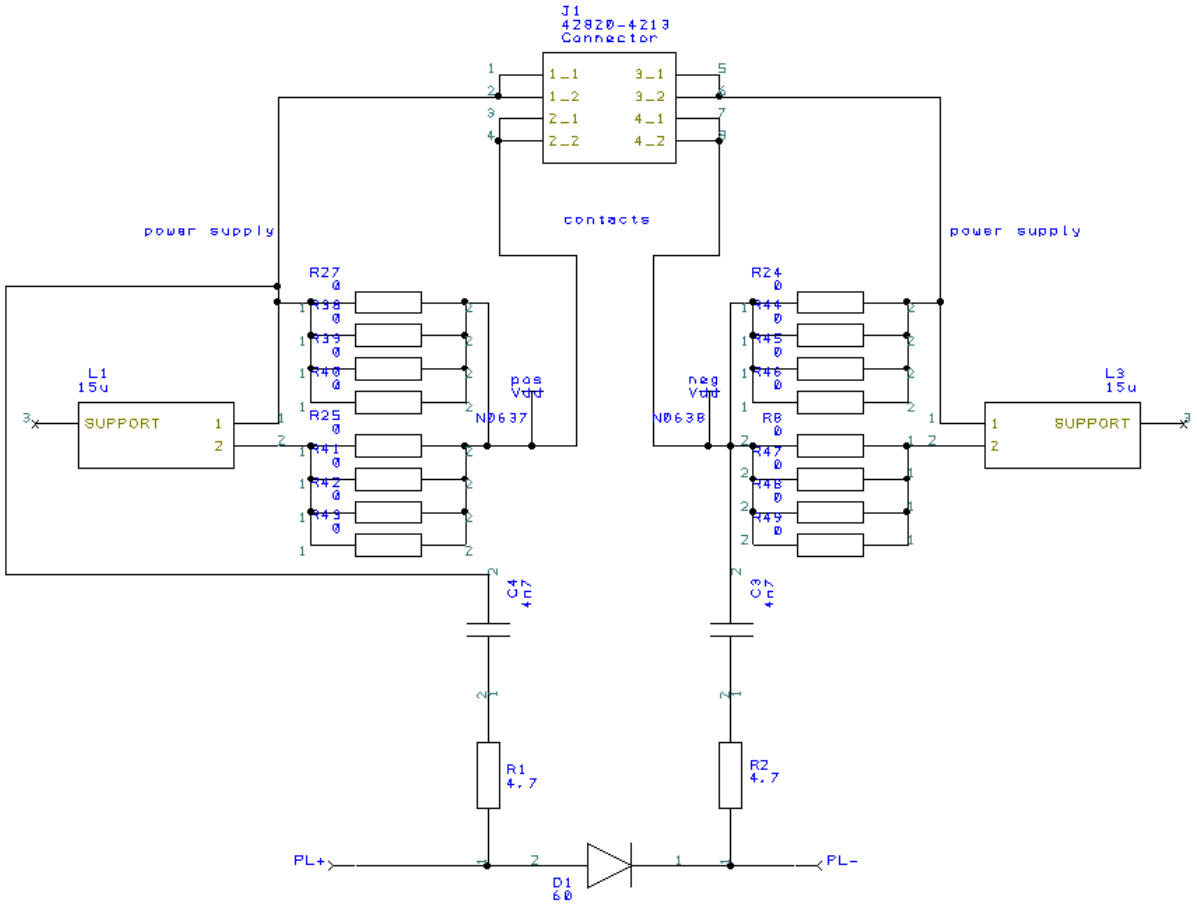
Source: personal archive.

Figure 70 – Schematics of second version, components for PHY chip.



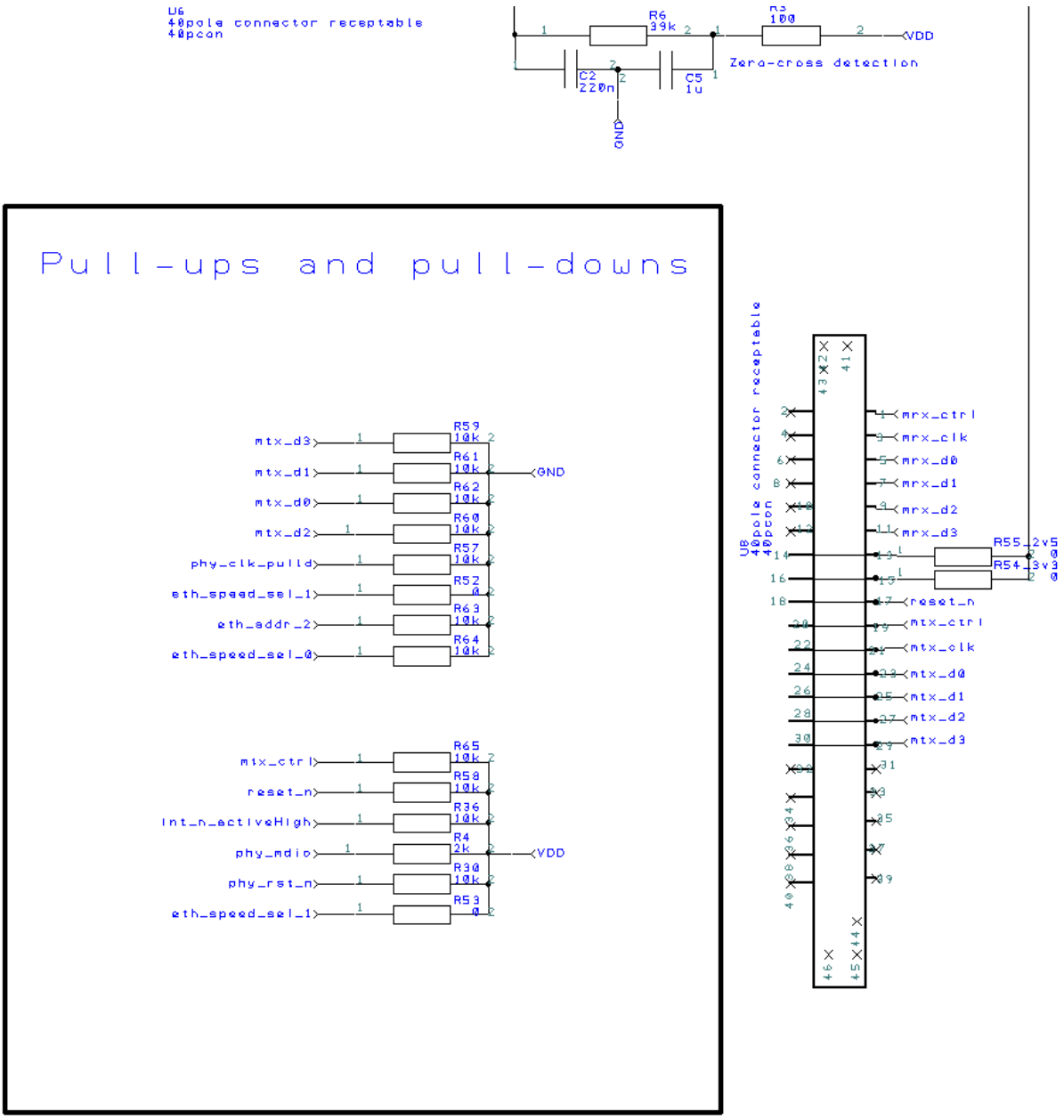
Source: personal archive.

Figure 71 – Schematics of second version, coupling.



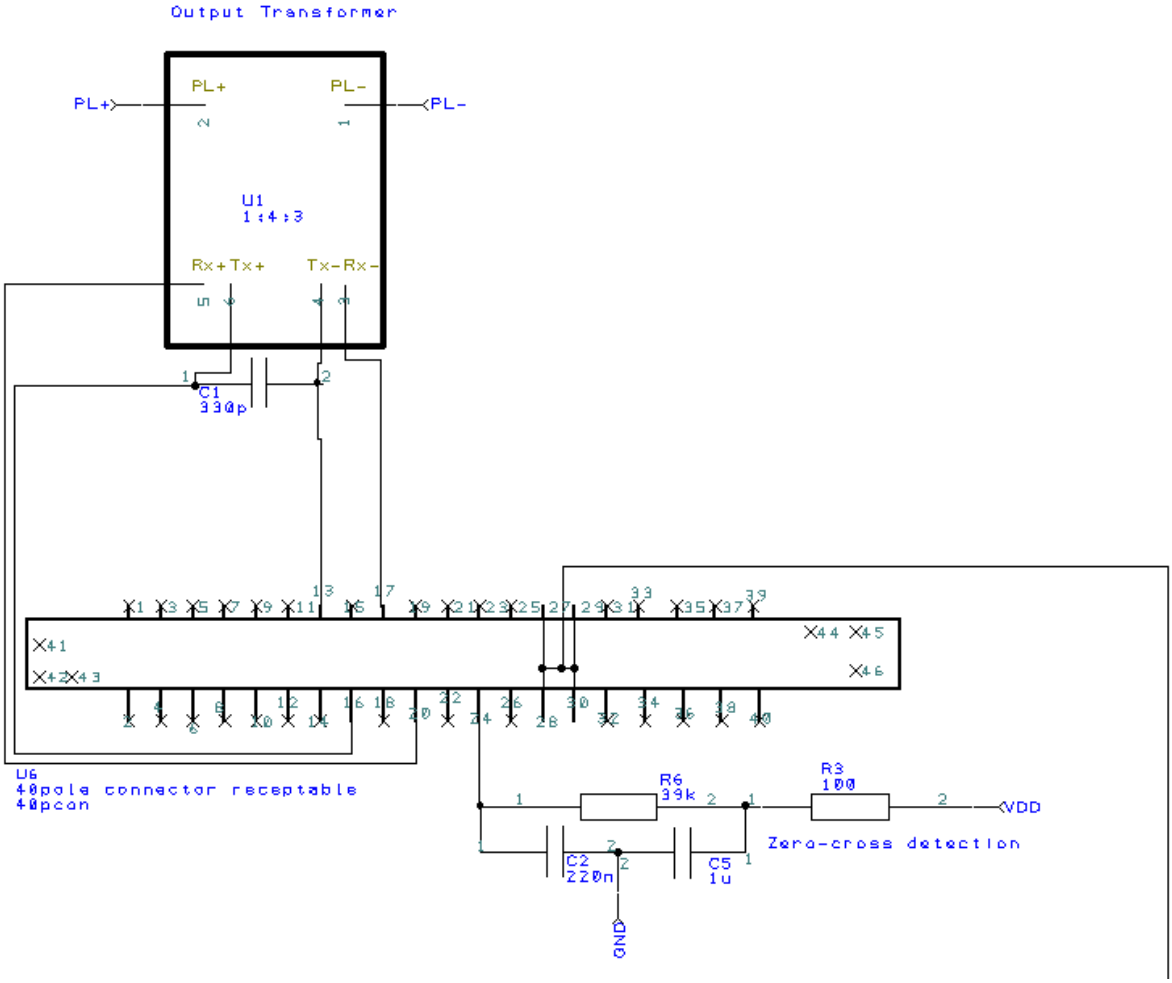
Source: personal archive.

Figure 72 – Schematics of second version, first header and bootstrap configuration.



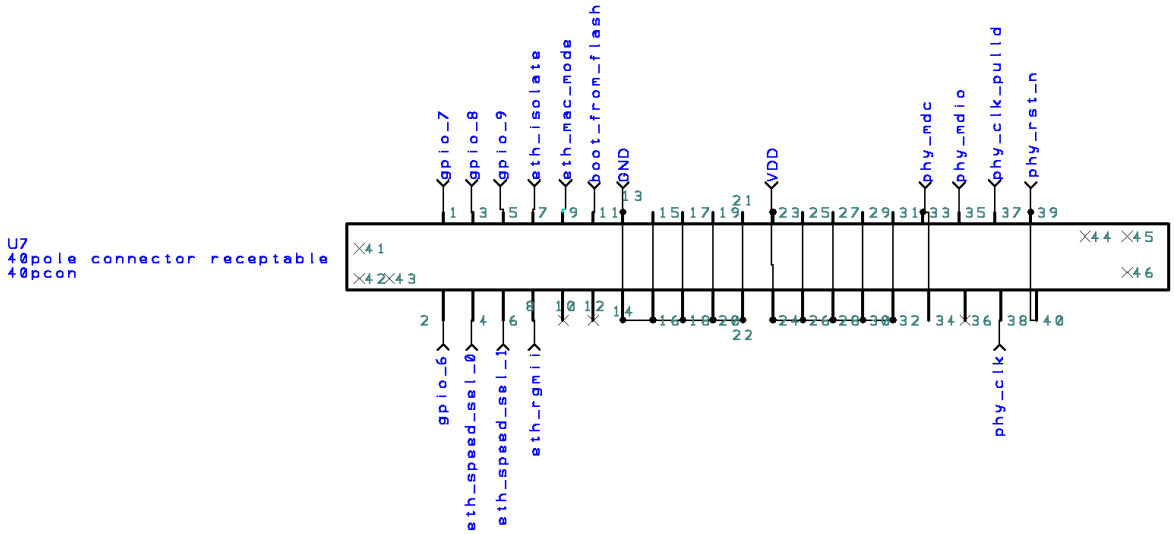
Source: personal archive.

Figure 73 – Schematics of second version, second header, coupling transformer and zero cross detection.



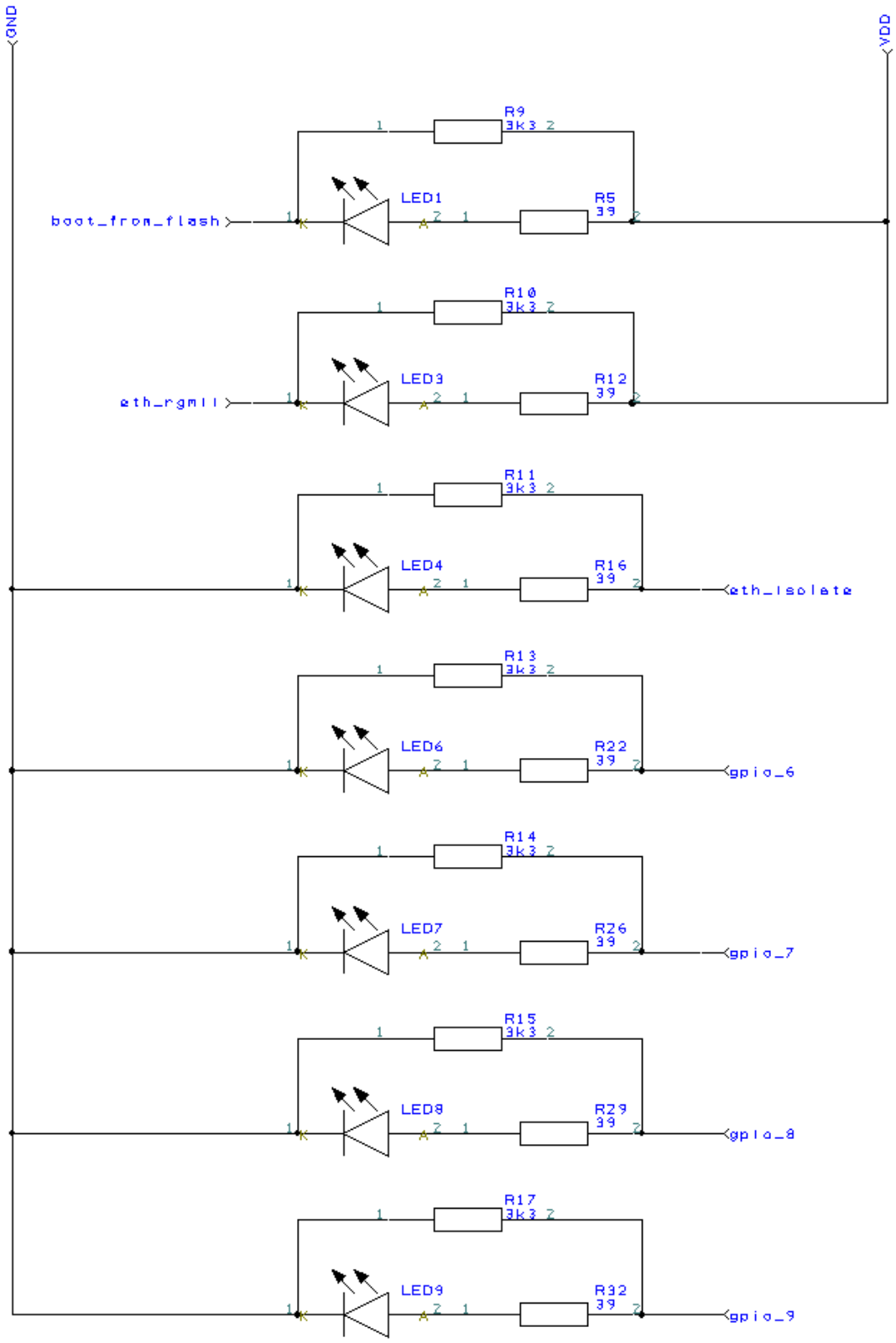
Source: personal archive.

Figure 74 – Schematics of second version, third header.



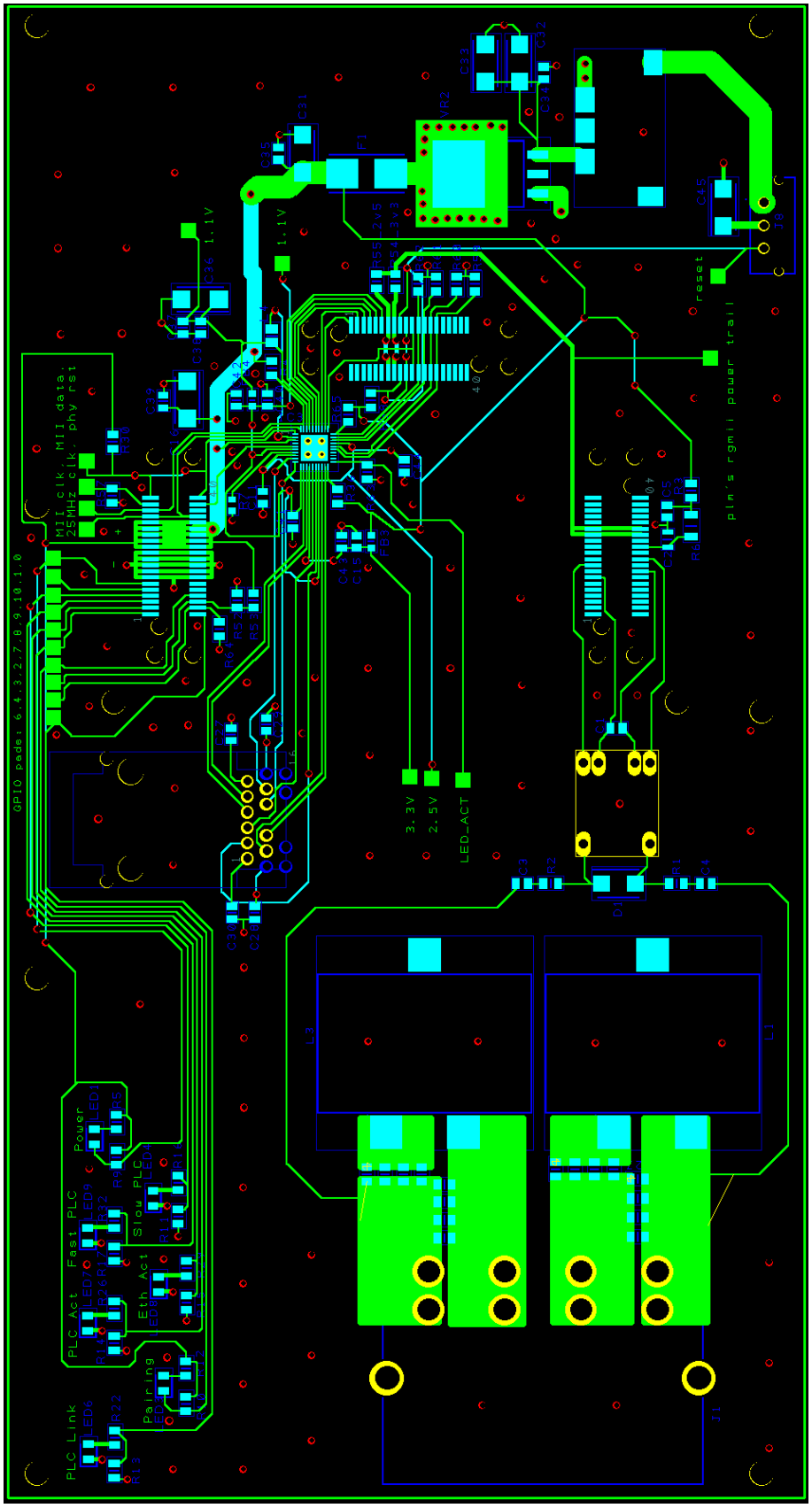
Source: personal archive.

Figure 75 – Schematics of second version, status LEDs.



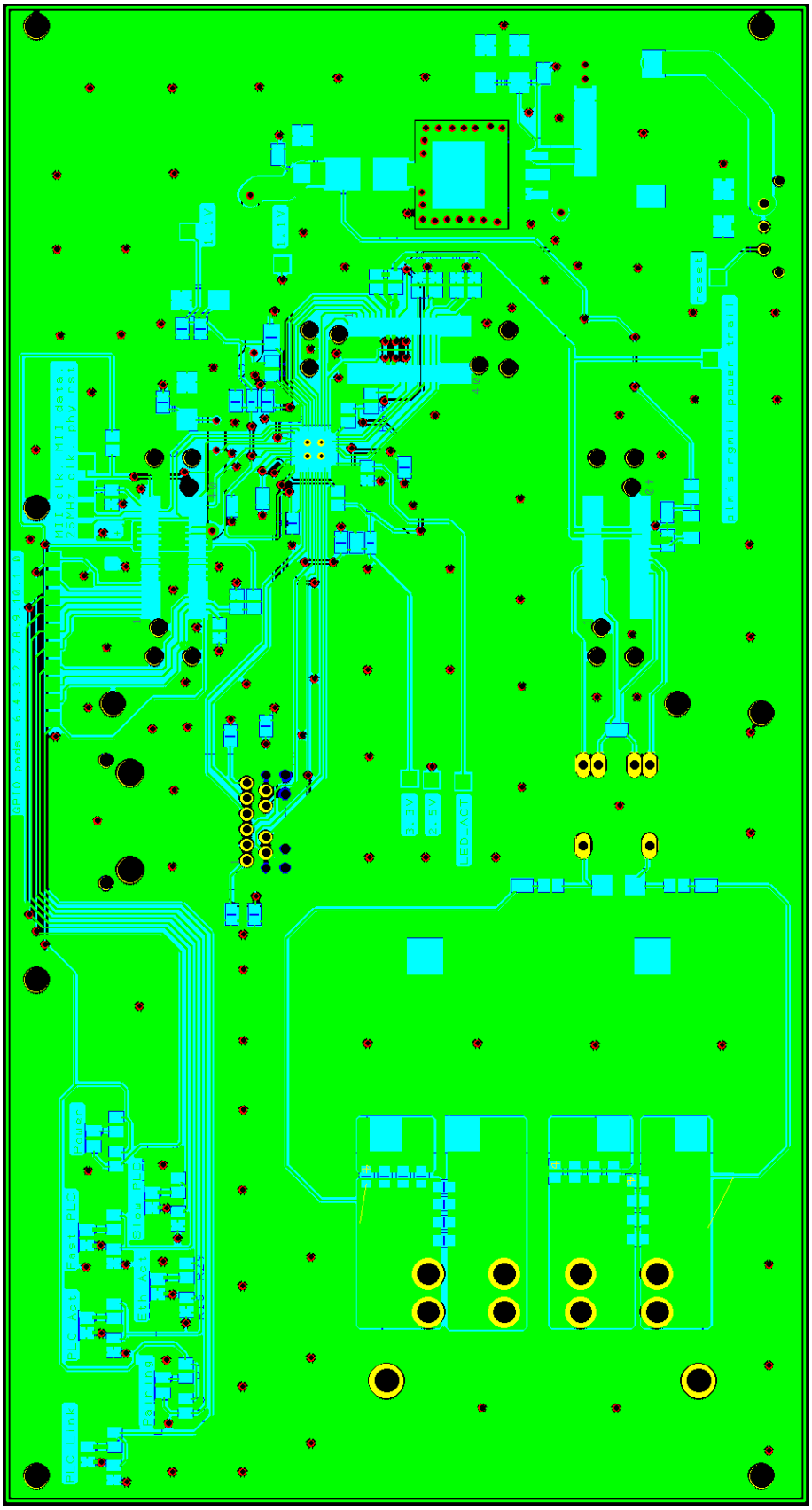
Source: personal archive.

Figure 76 – Second version of PCB, PCB design without poured copper as ground.



Source: personal archive.

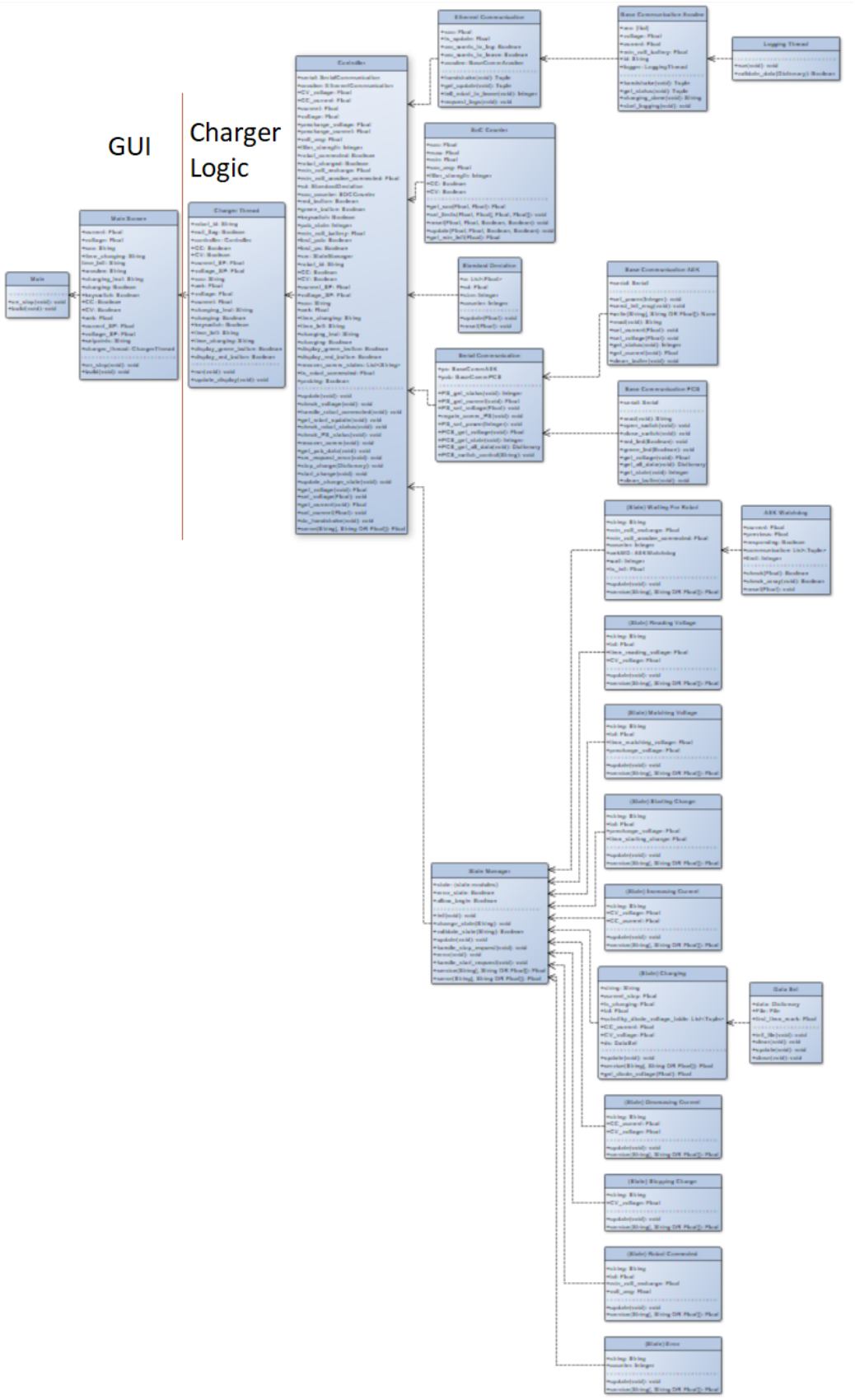
Figure 77 – Second version of PCB, PCB design with copper pour.



Source: personal archive.

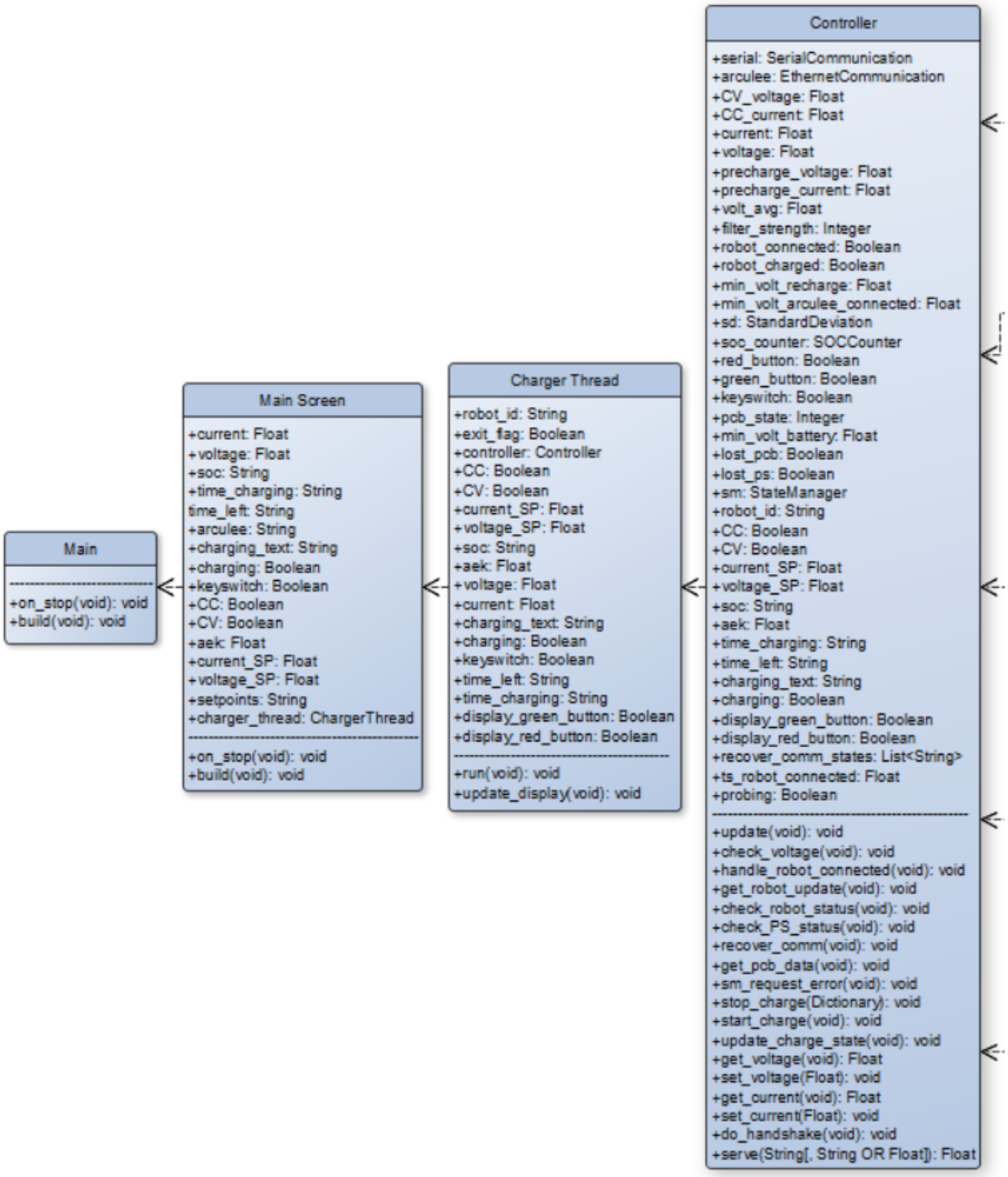
APPENDIX D – Class Diagram for Temporary Reorganisation of Software

Figure 78 – Full class diagram for first, temporary implementation of new software.



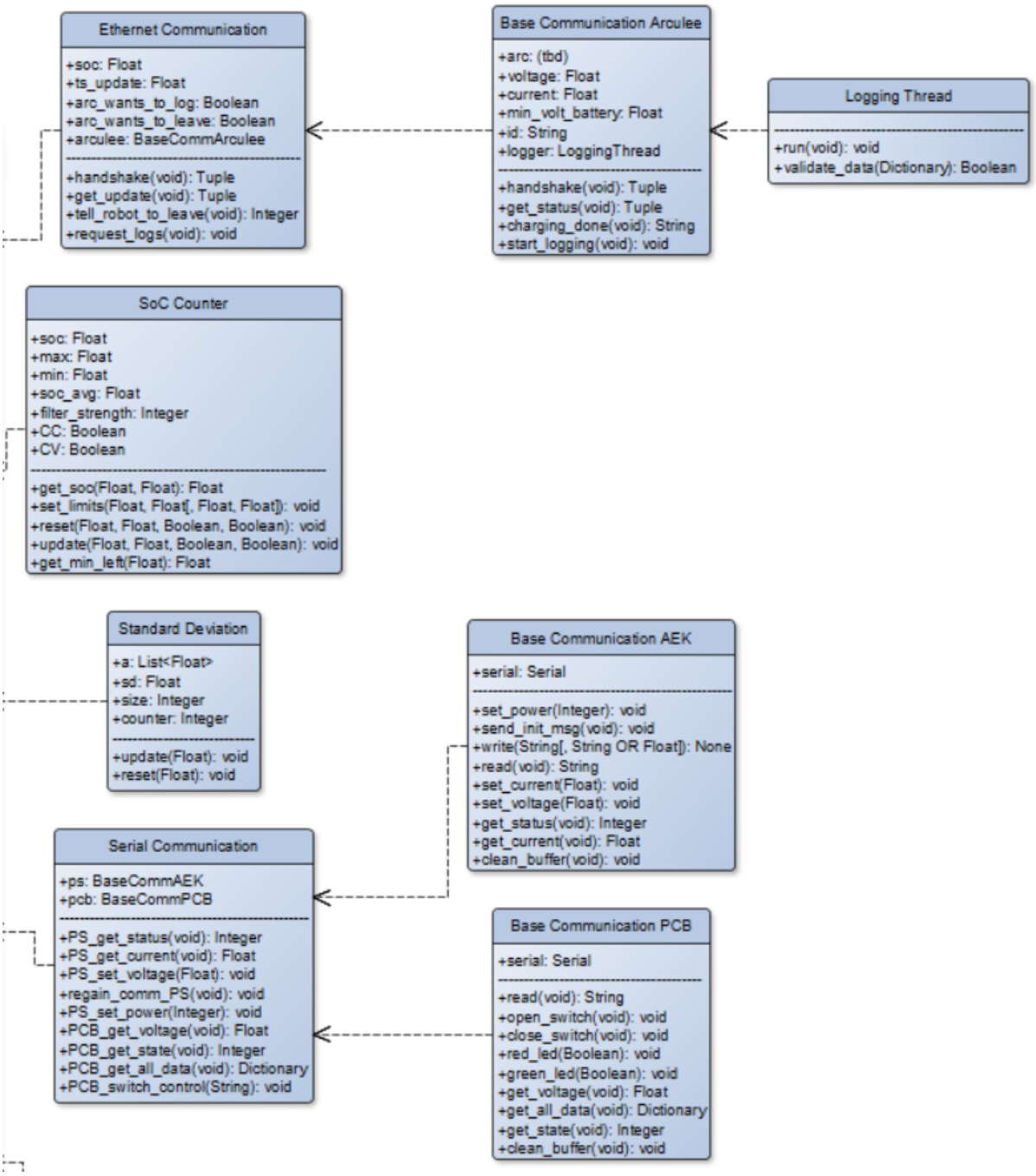
Source: personal archive.

Figure 79 – Class diagram for temporary software, GUI.



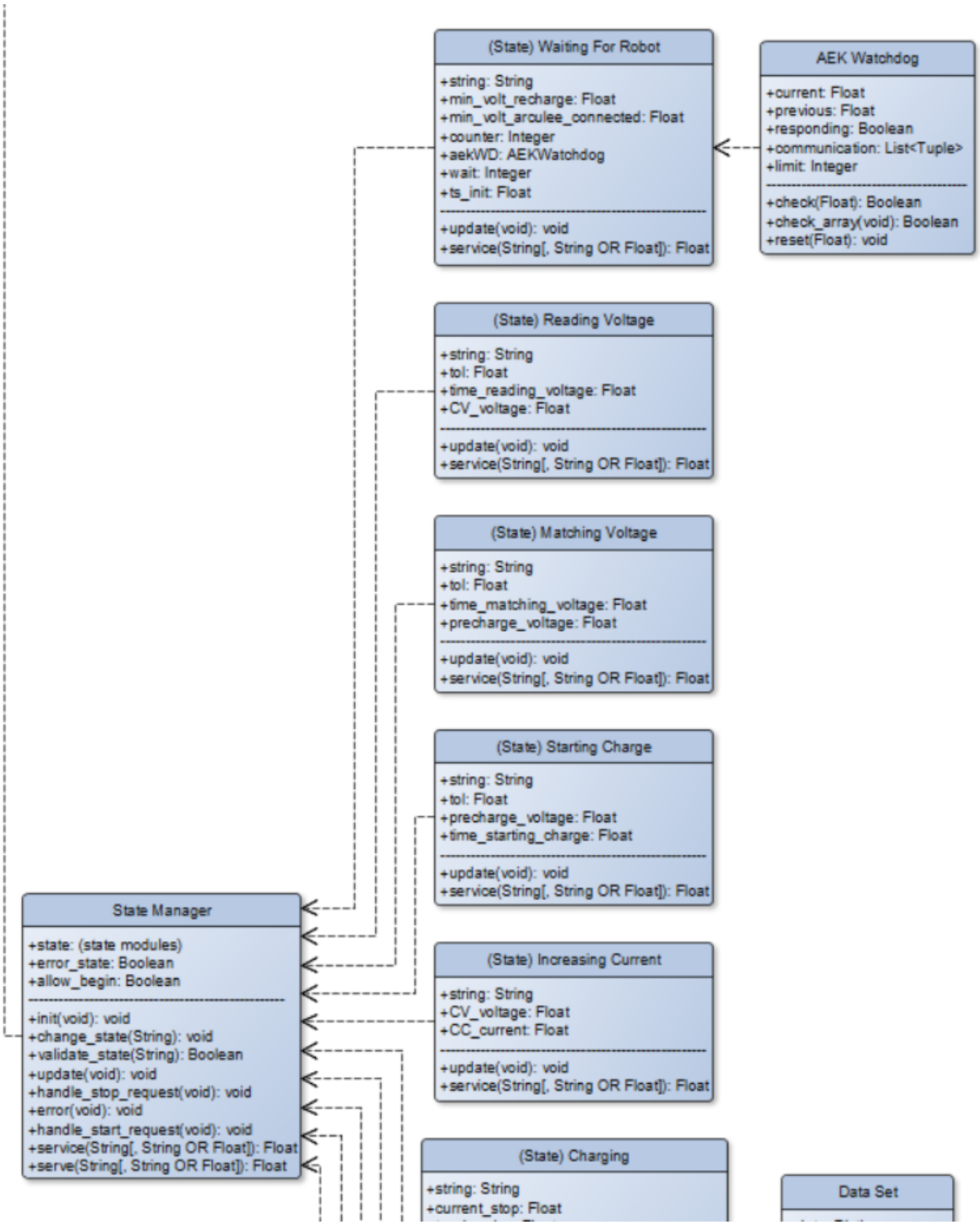
Source: personal archive.

Figure 80 – Class diagram for temporary software, modules.



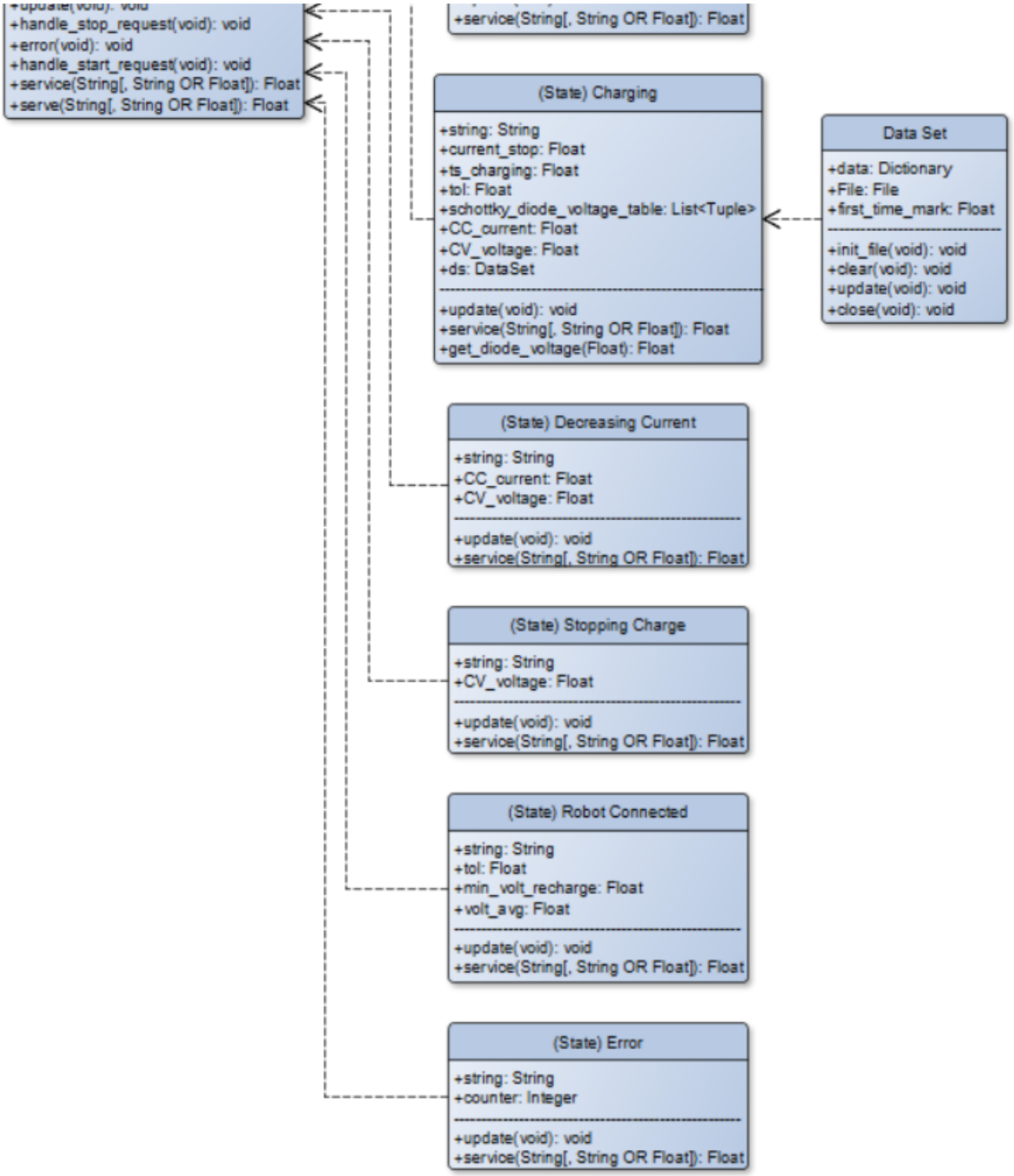
Source: personal archive.

Figure 81 – Class diagram for temporary software, state machine part one.



Source: personal archive.

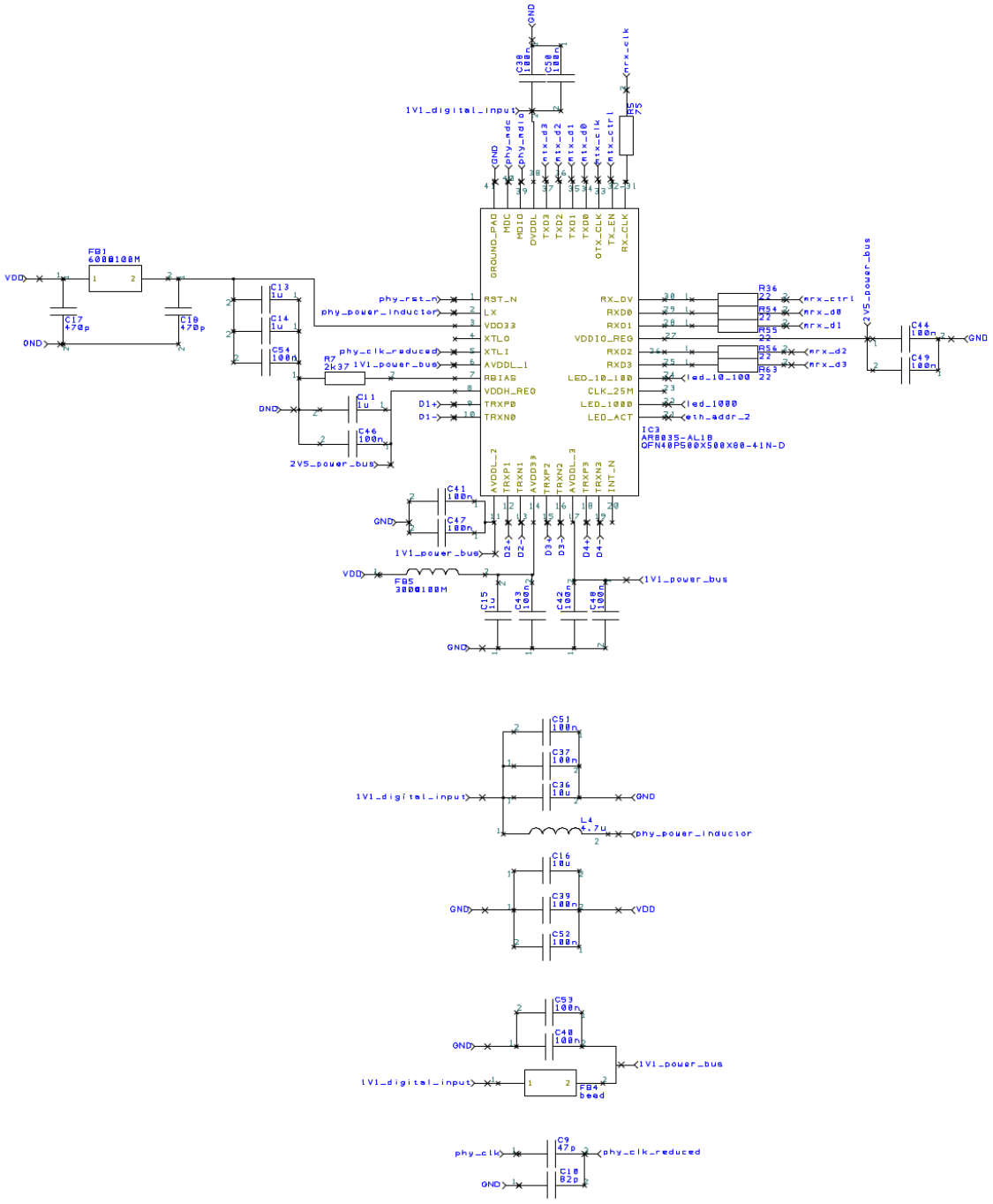
Figure 82 – Class diagram for temporary software, state machine part two.



Source: personal archive.

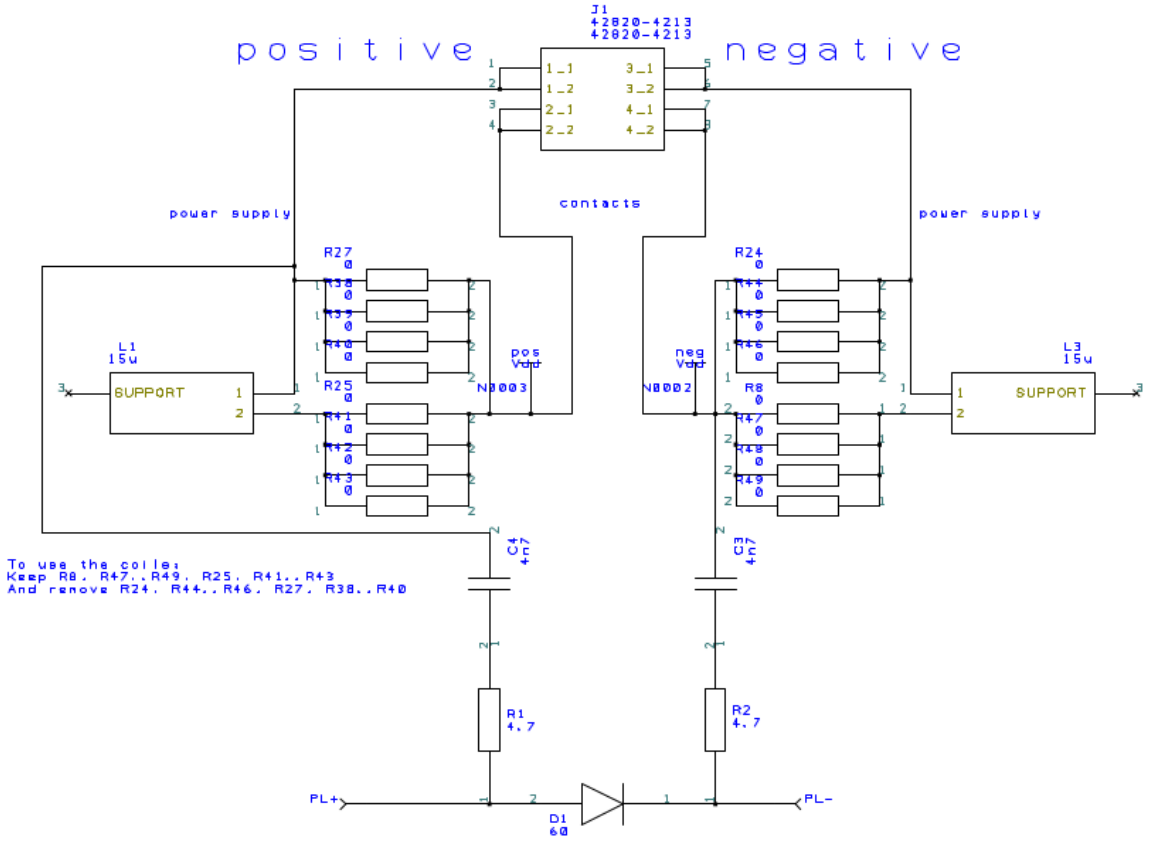
APPENDIX E – Third version - Schematics and PCB

Figure 83 – Schematics of third version, PHY chip.



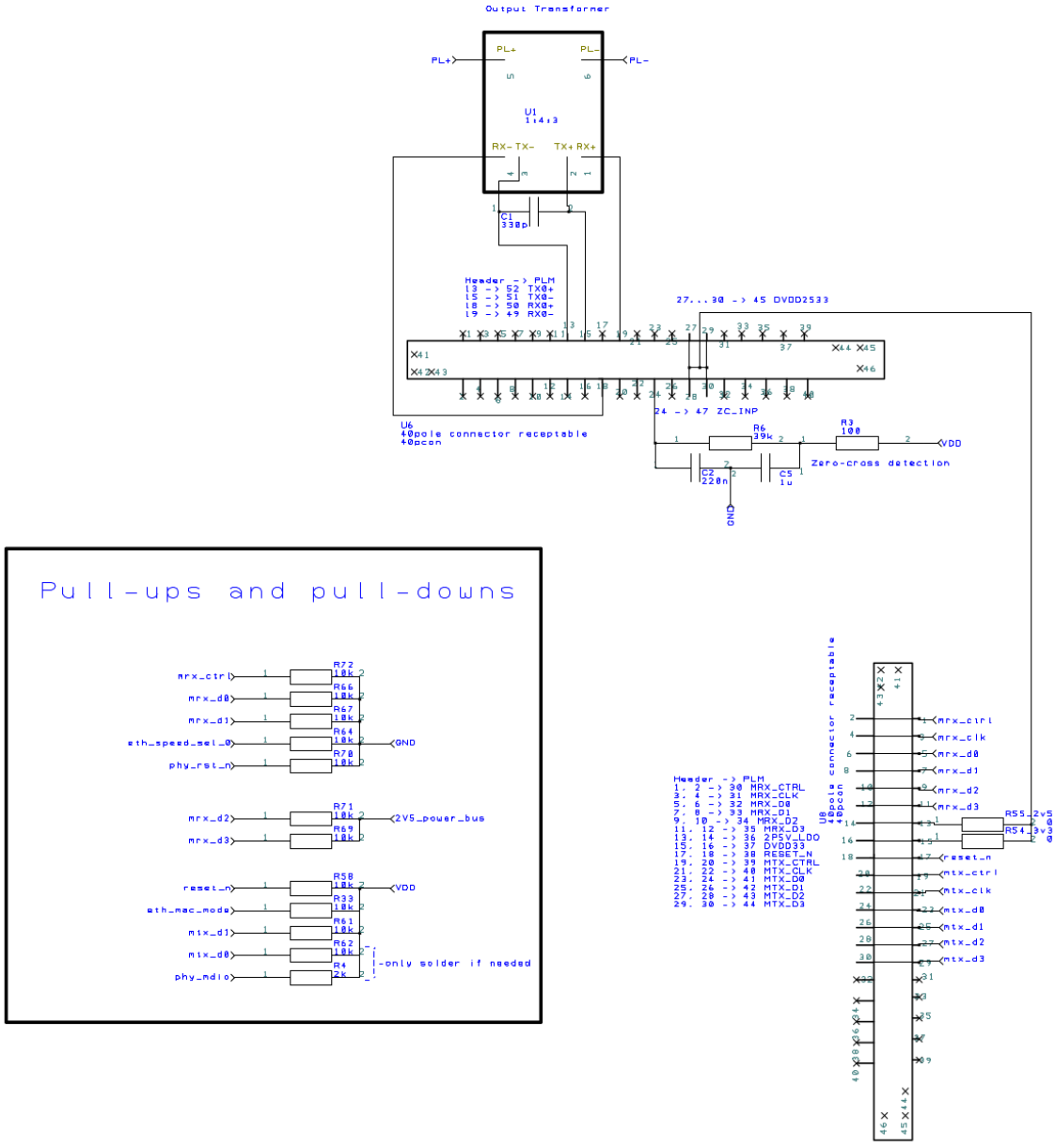
Source: personal archive.

Figure 84 – Schematics of third version, coupling.



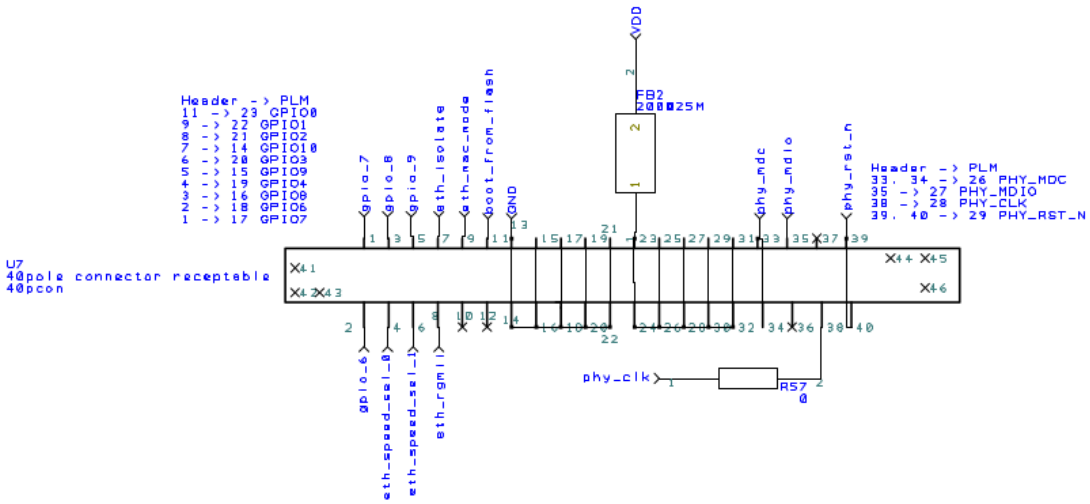
Source: personal archive.

Figure 85 – Schematics of third version, adapter connectors part 1.



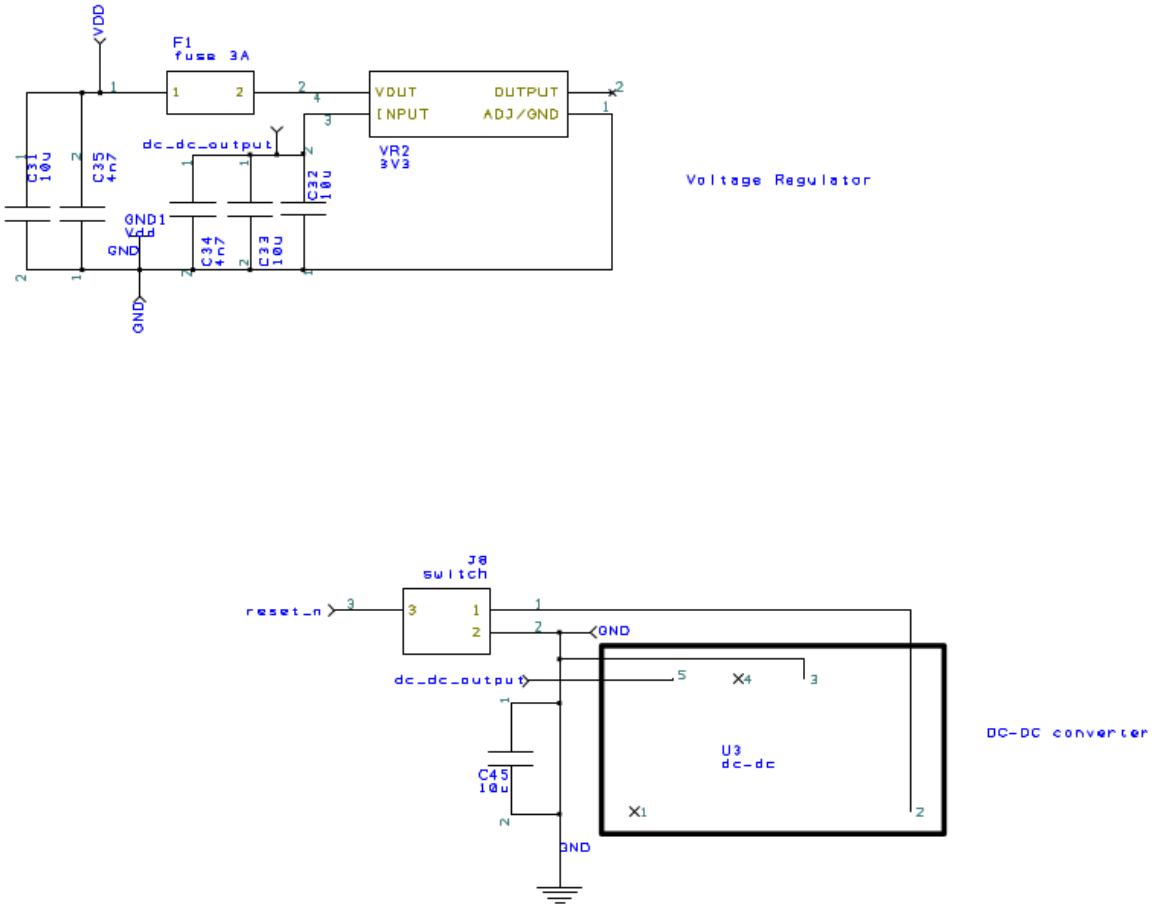
Source: personal archive.

Figure 86 – Schematics of third version, adapter connectors part 2.



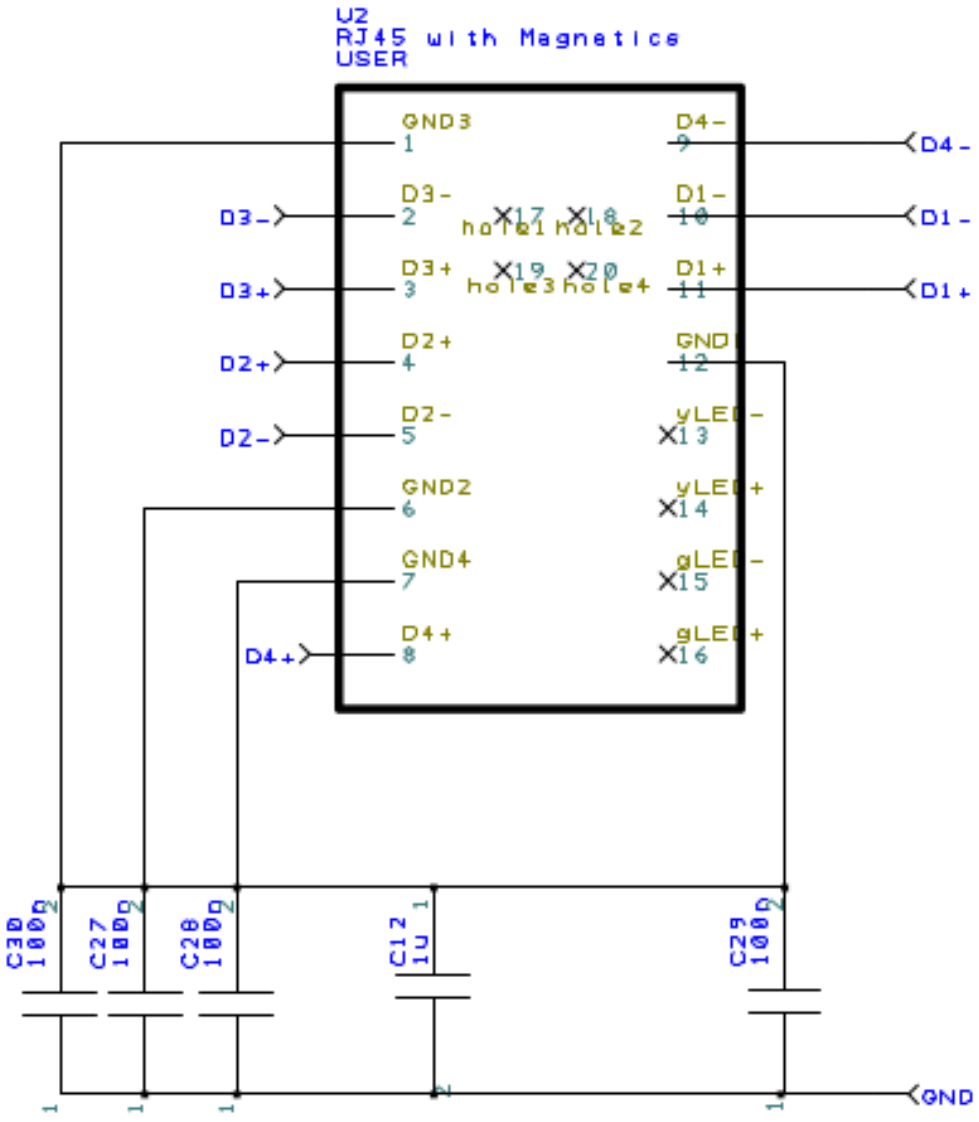
Source: personal archive.

Figure 87 – Schematics of third version, power supply.



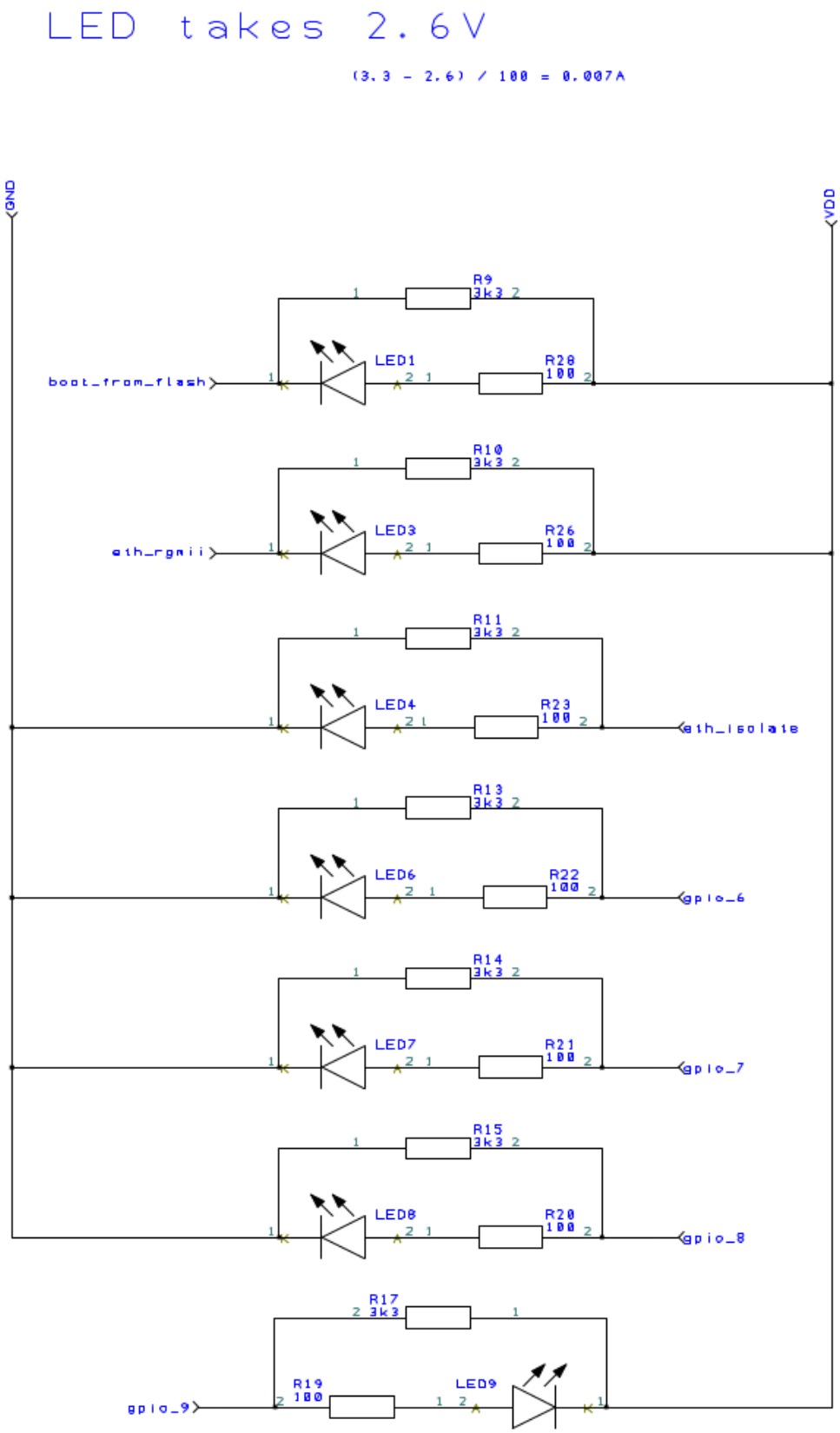
Source: personal archive.

Figure 88 – Schematics of third version, RJ-45 header.



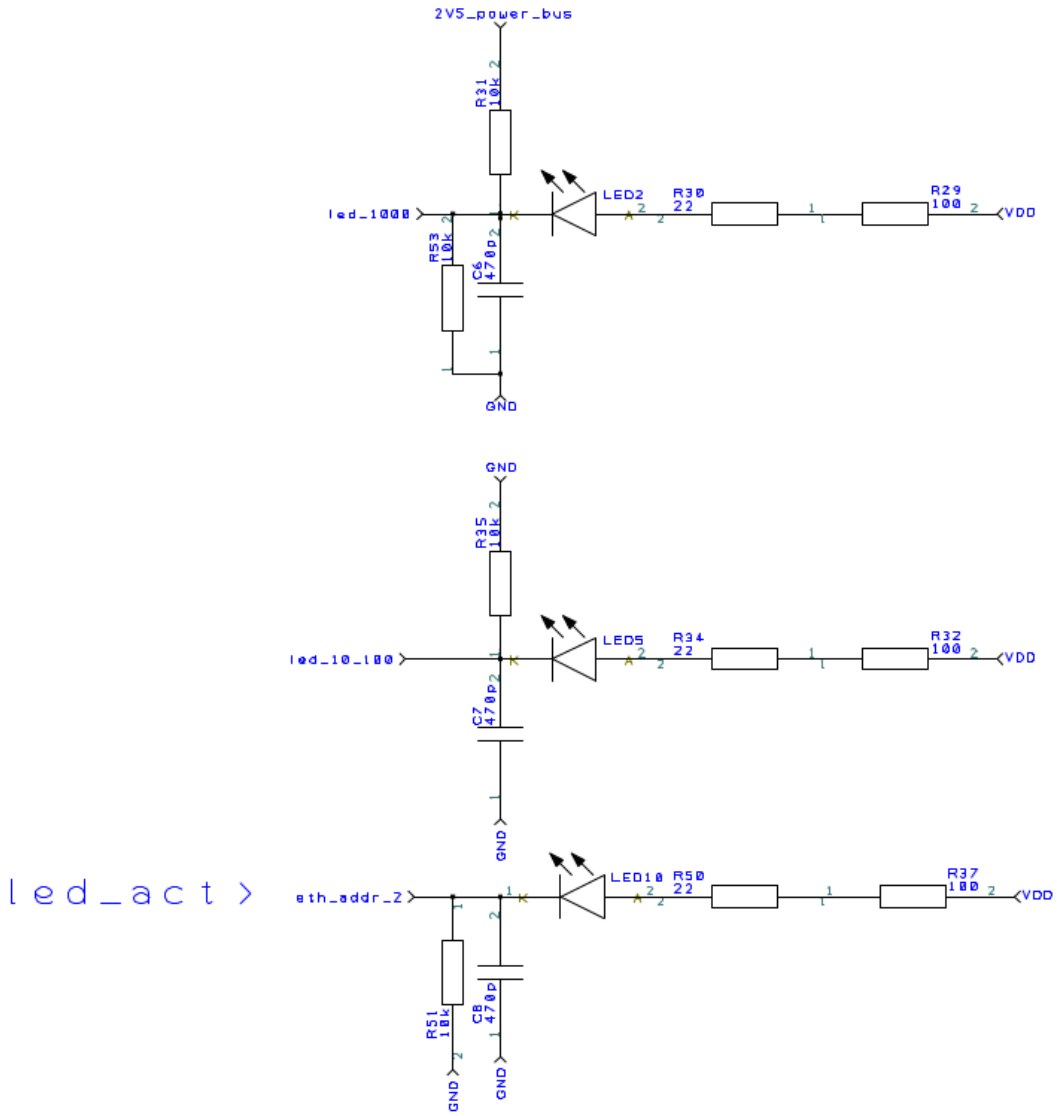
Source: personal archive.

Figure 89 – Schematics of third version, PLM LEDs.



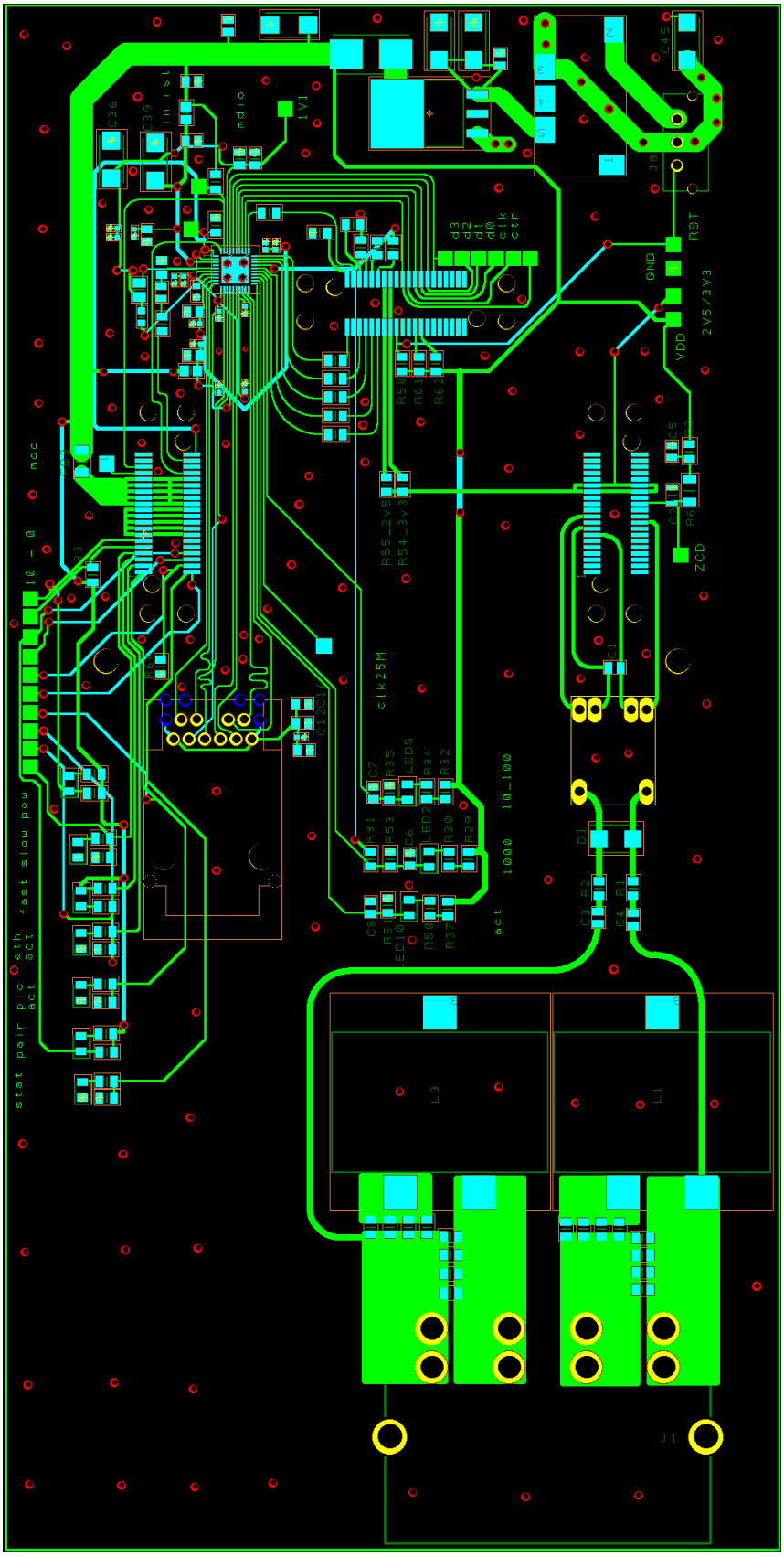
Source: personal archive.

Figure 90 – Schematics of third version, PHY LEDs.



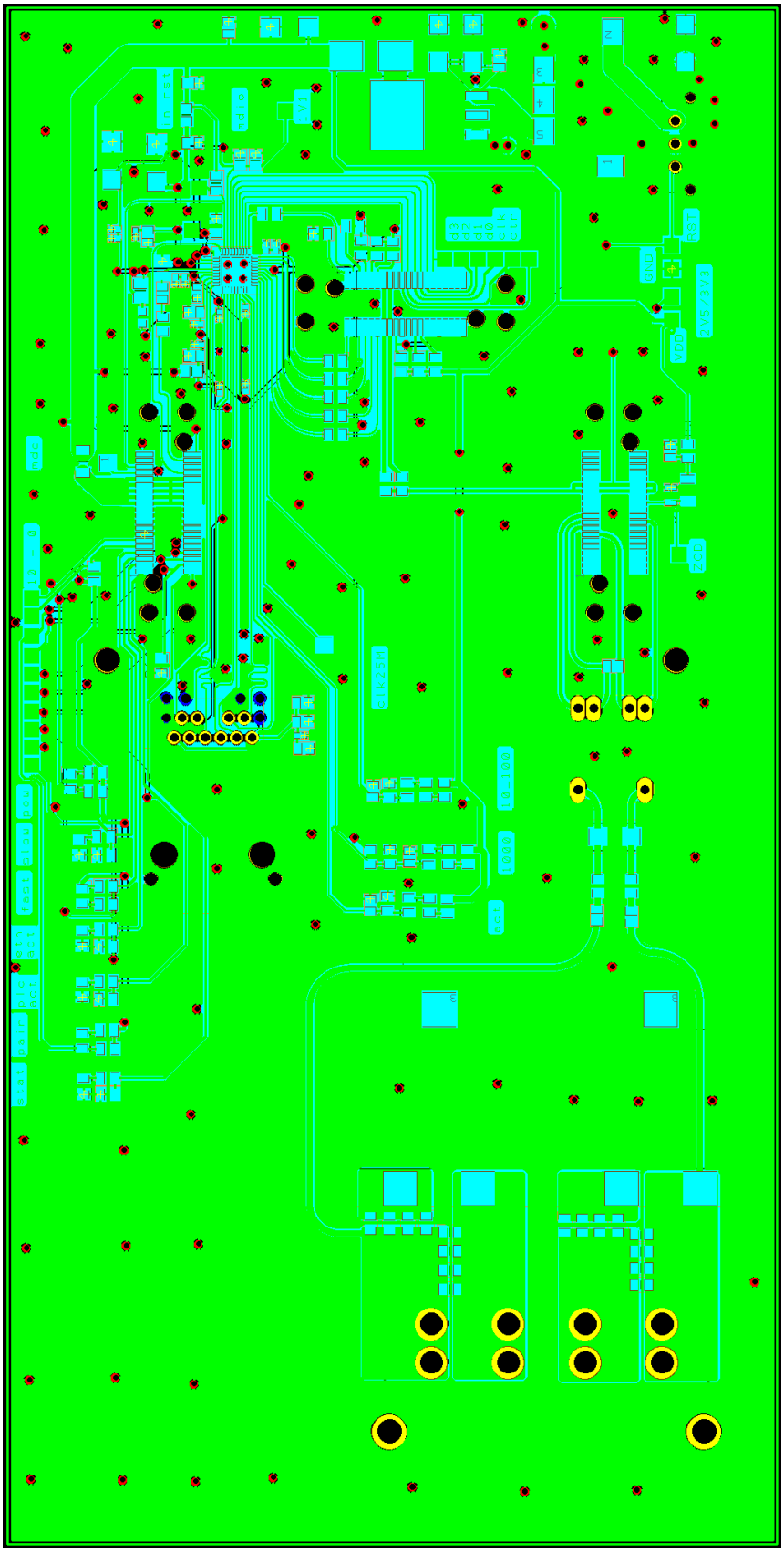
Source: personal archive.

Figure 91 – Third version of PCB, PCB design without copper pour.



Source: personal archive.

Figure 92 – Third version of PCB, PCB design with copper pour.



Source: personal archive.