FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

A Teaching Platform for Embedded Systems Engineering

Bachelor's Thesis

to achieve the academic degree

Bachelor of Science

in

Computer Engineering

by

Arvid Staub
Registration Number 0726421

at the
Department of Computer Engineering of the Vienna University of Technology

Supervisor: Univ.Ass. Dipl.-Ing. Armin Wasicek

Vienna, 6th July 2011

(Signature of Author)

(Signature of Supervisor)

Erklärung zur Verfassung der Arbeit

Arvid Staub
Aichholzgasse 18/7, 1120 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasser)

Abstract

This work presents a teaching and learning hardware platform for hands-on training in the *Embedded Systems Engineering* lab course. A Real-Time Network of four micro-controller nodes is provided along with a rich set of peripherals: A thermal control path, a wireless sensor network bridge and various digital and analog user interfaces. The board includes a flexible micro-controller programmer and a debugger with an integrated 32-channel logic analyzer. A host PC can access this functionality using a single USB interface.

Acknowledgements

I want to thank all people who contributed to this work, knowingly and unknowingly. In no particular order, these are my supervisor Armin Wasicek of the Department of Computer Engineering of the Vienna University of Technology, who guided this work into the best possible path while still allowing creative exploration and taking risks in favor of re-using old-fashioned, but proven designs. Raphael Charwot has been of great help in critical discussions about the proposed system structure. Finally Victoria Klang, for giving me unfailing support every day. Without her, I wouldn't have had the opportunity to put the effort into this work it deserved.

My gratitude also extends to the numerous folks who contribute to the development of the Free and Open Source Software used to produce this thesis.

Table of Contents

List of Figures

Table of Acronyms

ARM	Advanced RISC Machine
BCD	Binary Coded Digit
BOM	Bill of Materials
CAD	Computer-Aided Design
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
DBI	Display Bus Interface
DMA	Direct Memory Access
EDA	Electronic Design Automation
ESE	Embedded Systems Engineering
FPGA	Field-Programmable Gate Array
HCI	Human-Computer Interaction
I^{2}C	Inter-Integrated Circuit
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
ISP	In-System Programming
JTAG	Joint Test Action Group
KiB	Kilobyte = 1024 Byte
LCD	Liquid Crystal Display

LED	Light Emitting Diode
LSB	Least Significant Bit
MIPI	Mobile Industry Processor Interface
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
PC	Personal Computer
PCB	Printed Circuit Board
PLL	Phase Locked Loop
PWM	Pulse Width Modulation
RISC	Reduced Instruction Set Computer
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TAP	Test Access Port
TFT	Thin Film Transistor
TQFP	Thin Profile Plastic Quad Flat Package
TTP/A	Time Triggered Protocol A
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
XML	eXtensible Markup Language

CHAPTER 1

Introduction

Embedded Systems have become our ubiquitous companions over the past years. Nearly all electronic devices we use on a daily basis contain digital processors, which implies that customer experience is becoming more and more dominated by software design decisions. Success or failure in the market is determined by well-crafted embedded software. At the same time, the functional demands to this software rise. Devices are expected to be "smart" and unintrusive, which requires them to communicate with each other to exchange information about their environment.

The course *Embedded Systems Engineering* held by the Department of Computer Engineering of the Vienna University of Technology teaches methods for design and implementation of real-time distributed embedded computing applications. It offers students an opportunity for hands-on training with a real-world networked embedded system.

1.1 Problem Statement

This bachelor thesis' objective was to design and prototype a new hardware platform that is rugged enough to endure everyday use in the lab by computer engineering students. The key requirements are:

- Mechanically and electrically rugged construction
- Realistic Control Path with support for Sensor Fusion
- Multiple Options for HCI[1]
- Wireless Sensor Network Integration
- Support for multiple micro-controller architectures
- Extensive Debugging Facilities for Students

[1]Human-Computer Interaction

1.2 Structure of this Thesis

The following chapter gives a brief introduction to the context of this thesis. It presents a predecessor implementation and other works that explain underlying concepts or provide supplemental information. A bachelor's thesis that complements the proposed hardware platform with the necessary system software implementation is also introduced.

A feasible system design is established in chapter 3. The modular composition of the system is detailed along with various logical concepts to fulfill all requirements from the problem statement.

In chapter 4, an implementation for each module is set forth. This chapter covers the electronic components employed and the circuits that put them to use. It also shows selected pre-layout circuit simulations and explains the schematic entry and layouting process.

The result of this work is discussed in chapter 5, which reveals the assembled prototype and outlines measurement results of key electrical characteristics verified against the results obtained by simulation from the previous chapter.

The last chapter concludes this thesis with a critical reflection about the practical and theoretical work as well as its outcome. It summarizes the challenges faced during the development of the presented hardware platform.

Supplementary documentation in the form of schematics and source code for test programs can be found in Appendix A.

CHAPTER 2

Related Work

2.1 A Platform for Teaching and Researching Distributed Real-Time Systems

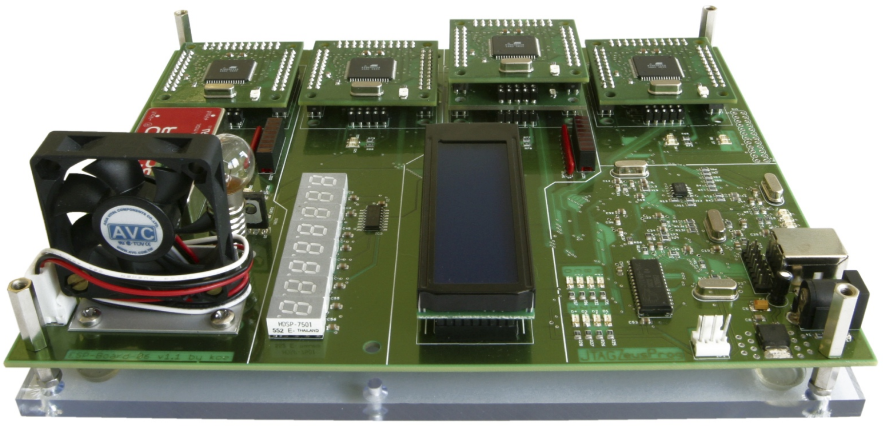


Figure 2.1: Alexander Kößler's ESE board (source: [28])

This works builds upon the concepts presented in Alexander Kößler's master thesis. The basic modular design philosophy is inherited from his work. Some concepts have been dropped, like the explicit support for a remote workplace setup or the CPLD[1]-based multiplexed programming chain. Others have been modified to make them more flexible and user-friendly. The exam evaluation concept has evolved into a logic analyzer that reduces manual tasks in teaching and automated grading. The electrical module interface was redesigned from scratch to adapt to the updated requirements and new peripheral devices.

[1]Complex Programmable Logic Device

2.2 A Modular, XML-based JTAG Programmer for Embedded Devices

One objective of this work was to integrate an existing micro-controller programmer implementation. Martin Schmölzer's programming software provides a flexible framework for programming different kinds of micro-controllers through the well-defined JTAG[2] interface. The key feature is an advanced communication protocol that moves the complexity to support multiple micro-controller families into the PC[3]-based host application. A legacy system approach using a serial communication channel provided by an USB[4]-to-serial bridge is proposed. [28]

This work will foster the porting of Martin Schmölzer's source code to a modern ARM[5]-based 32-bit micro-controller with an integrated USB device interface. While reducing component count and hence manufacturing costs, this measure will also streamline the system design and allow several add-on features to be implemented without requiring additional hardware. Removing the USB-to-serial bridge also removes the need for a proprietary device driver on the host PC, therefore allowing a pure open-source implementation that works on virtually all operating systems without any kernel drivers.

2.3 A modular software package for the Embedded Systems Engineering Board

Jürgen Galler's bachelor's thesis presents the necessary software to operate the hardware outlined in this work. It includes a ported version of Martin Schmölzer's JTAG device software that is binary compatible with the original host PC application. It additionally features are a command-line driven logic analyzer utility that can acquire digital waveforms from the hardware and format them for viewing in the popular GTKWave application for Linux. Last but not least, two virtual serial ports are provided by the device software that allow direct access to the two on-board communication buses.

The presented USB device implementation relies on kernel drivers to provide virtual serial ports that connect the programmer and the buses. This limits its compatibility to the Linux operating system, since it employs "unusual" device descriptors that are supported badly on Microsoft Windows and not at all on Apple Macintosh. A future enhancement could provide an updated version of the device software that does not depend on kernel drivers. The resulting system would show improved performance and stability at the cost of requiring an adaption of the host PC applications. An example of this user-space-only approach is given by the logic analyzer utility which is based on the open-source `libusb` library.

[2]Joint Test Action Group
[3]Personal Computer
[4]Universal Serial Bus
[5]Advanced RISC Machine

CHAPTER 3

System Design

The presented hardware platform hosts four independent micro-controller nodes connected to a Real-Time network. Different peripheral devices for each node provide an extensive learning opportunity for students. A shared communication bus is included for Real-Time data transfer between the four nodes. This concept was introduced by Alexander Kößler's ESE[1] board. [17]

This system extends the basic concept by providing a connection to a wireless sensor network via an included Zigbee interface.

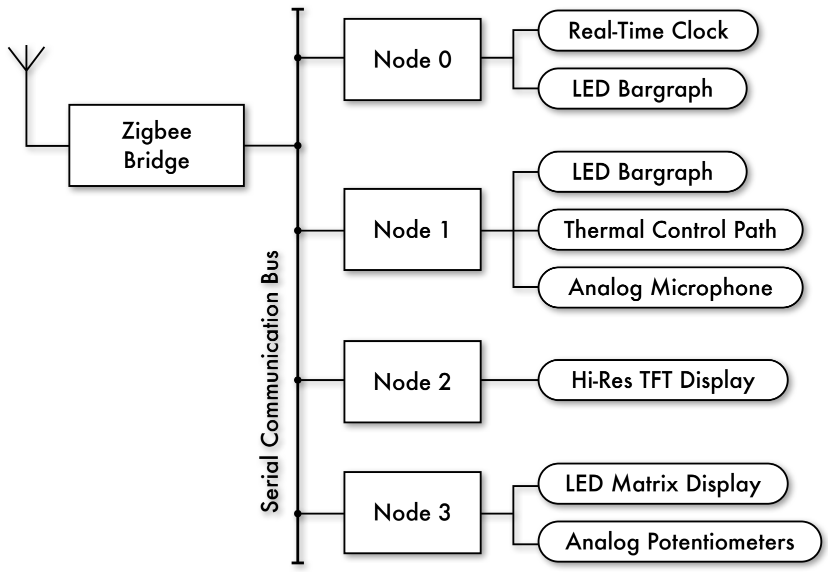


Figure 3.1: Network and Peripheral Devices

[1]Embedded Systems Engineering

Figure 3.1 presents the logical structure of the hardware platform. A large PCB[2] shall host all peripheral devices and provide sockets for pluggable Micro-controller Nodes. The various peripheral devices available are summarized in Table 3.1.

Node 0	2 Push Buttons with LEDs[3] 8-LED bargraph Real-Time Clock with battery-backed oscillator and static memory
Node 1	2 Push Buttons with LEDs 8-LED bargraph Resistor Heating (see section 3.1) Fan 3 Temperature Sensors Analog Microphone
Node 2	2 Push Buttons with LEDs Modular LCD[4] or TFT[5] display ESE analog temperature and luminosity measurement module
Node 3	2 Push Buttons with LEDs 2 analog thumbwheel potentiometers 6 digit LEDs matrix

Table 3.1: Node Peripheral Devices

The modular structure makes the system flexible to accommodate different micro-controller architectures by designing pluggable CPU[6] modules according to the specification in section 4.1 of this thesis.

3.1 Thermal Control Path

The system's primary focus is on the thermal control path which provides a learning platform for control theory in combination with Real-Time networking. This control path is formed by a board-mounted heat sink. Its temperature can be influenced by two devices:

1. Resistors for heating
2. A fan for cooling

Three digital temperature sensors provide redundant temperature feedback. All relevant control signals and sensors are connected to NODE 1. The digital temperature sensors output a signal to indicate an over-temperature condition if the measured temperature crosses a user-defined threshold. This signal is used to shut down the heating circuit as a safety measure against excessive heating caused by erroneous node software.

[2]Printed Circuit Board
[6]Central Processing Unit

Since this threshold is configurable via the node interface, an extra safeguard against malicious student programming has been introduced in the form of a fourth temperature sensor which is only accessible by the system's supervisory controller (see section 3.5). This fourth sensor can set a safety limit as well as report the measured temperature to a PC-based monitoring software.

An analog microphone is mounted close to the fan. It is connected to NODE 1 to allow loudness and frequency domain analysis of the fan noise. This provides an independent channel for measuring the rotational speed and the radiated noise of the fan.

This setup provides a realistic control path that can be used for a variety of control theory exercises, for example

- Fan control – Keeping the heat sink at a predefined temperature with random heat ingress
- Temperature profile tracking
- Creating a steady airflow with a predefined temperature
- "Silent" fan control with a fan noise limit
- ...

3.2 Real-Time Communication

The system includes a Real-Time communication bus exclusive to the four Micro-controller Nodes. It employs a wired-AND communication scheme with one *recessive* and one *dominant* state. A logic high level is defined as the recessive state. This state is maintained by termination pull-up resistors. Any node can drive the bus into the dominant state by pulling it to ground potential. This can be achieved with open-drain output drivers. Each node is connected to the bus through a bus coupling unit that translates the transmit line to open-drain and decouples the receive line.

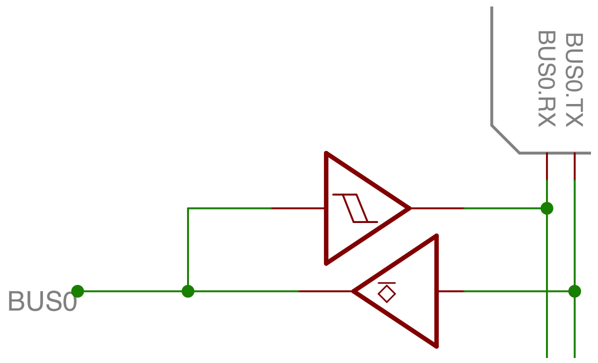


Figure 3.2: Bus Coupling Unit

The open drain driver (◇) makes sure that no high level can be driven onto the bus. The Schmitt Trigger (⎍) isolates the *receive* path and improves the noise immunity of the serial receiver. An example of bus communication is presented in Figure 3.3. It shows all relevant states that can arise during node communication.

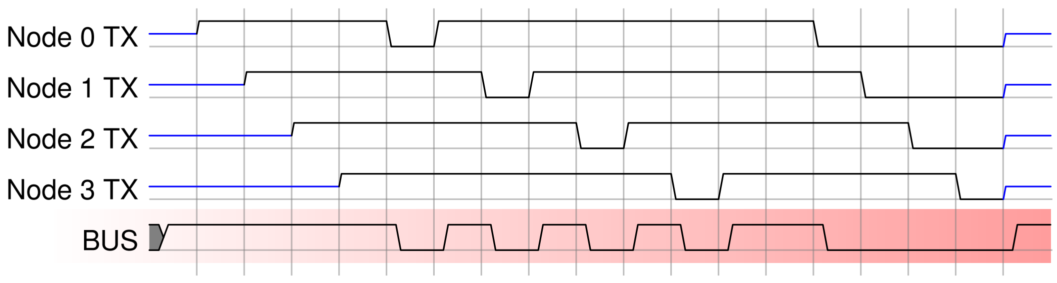


Figure 3.3: Bus communication example

`BUS1`

`BUS1` is an exact copy of `BUS0`, except it also connects the wireless sensor network bridge. It can be used for a number of purposes, including:

- Backup channel for Real-Time communication
- Communication with a wireless network
- Runtime debugging

3.3 Wireless Bridge

The system includes a wireless sensor network node that acts as a bridge into a wireless IEEE[7] 802.15.4 network. This node is a participant of `BUS1` to allow all Micro-controller Nodes to communicate with other boards or wireless sensor nodes via a dedicated relay application running on the wireless bridge.

3.4 User Interfaces

Each Micro-controller Node is equipped with a set of simple digital user I/Os[8]. Two push buttons with one LED each can be used for basic user interaction. Using the visual feedback from the LED, more elaborate functions like toggle or multi-state buttons can be implemented in software. Continuos set-points are entered via two potentiometers associated with NODE 3.

Bargraphs

NODE 0 and NODE 1 have additional arrays of 8 LEDs arranged as bar graphs. These can be used as a part of the user interface or as visual state indicators for debugging purposes.

[7]Institute of Electrical and Electronics Engineers
[8]Input/Outputs

LCD/TFT display

NODE 2 has an interface that is capable of driving an industry standard LCD panel. The display panel itself is attached as a modular PCB, which allows different panels to be used - see section 4.2.

LED Matrix

NODE 3 is equipped with a 6-digit LED matrix display. Each character is composed of 5 x 7 pixels, giving an overall number of 210 LEDs, each of which is independently addressable. A shift register holds the pixel data for a full row while a 3-bit row-select signal is used to specify which row is addressed. A multiplexing sequence is required to illuminate all rows continuously. This sequence must be generated by NODE 3's software.

3.5 Programming and Debugging Interface

An on-board companion controller provides a management interface to a host PC which offers programming and debugging services via an integrated USB device interface. The design is tailored towards easy integration of an existing modular software implementation. This software contains an implementation of the low-level programming tasks to be run on the companion controller as well as a PC-based downloading application which is highly configurable and extensible. [28]

Figure 3.4 shows the hard-wired connections of the companion controller. Designated connections for a micro-controller programming interface, a thermal monitor, a logic analyzer and two serial bus monitors are provided.

Programming Interface

The companion controller offers a program download feature that is used to update the internal flash memory of Micro-controller Nodes and the included Zigbee bridge. A programmer providing the vendor-independent IEEE 1149.1-2001 (JTAG) interface has been implemented. The "JTAG" standard is designed to support multiple devices sharing a single programming interface. Each device to be programmed exposes a TAP[9], which is composed of these signals:

TCK	Test Clock
TMS	Test Mode Select
TDI	Test Data In
TDO	Test Data Out

The standard supports a very flexible wiring scheme. The most commonly used implementation is a daisy-chained setup that allows the number of connected device to be discovered at run-time by software. [1, 28]

[9]Test Access Port

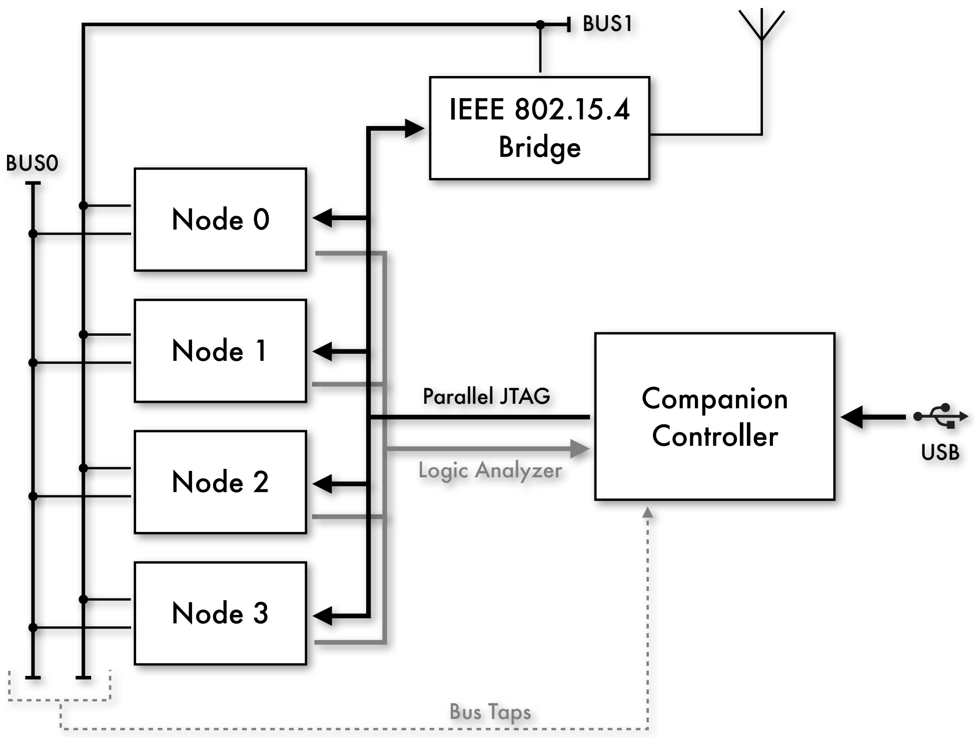


Figure 3.4: System Programmer Overview

Although most practical and flexible, this approach is affected by an erratum in the targeted ATMega128 family of micro-controllers which makes it impossible to use page oriented data transfer commands. Inability to use these commands results in severely degraded programming performance. [3]
In an effort to avoid potential problems and performance penalties associated with software based work-arounds, a parallel wiring scheme has been devised.

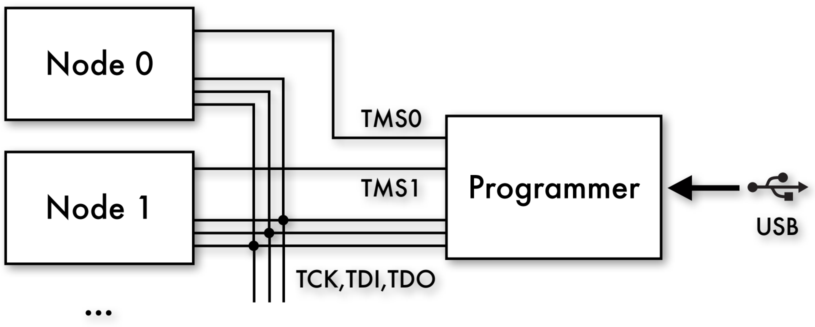


Figure 3.5: Parallel JTAG scheme

The CLOCK, DATA IN and DATA OUT lines are connected to all devices in parallel while a dedicated TEST MODE SELECT line to each programmable node is used to select the appropriate device for programming. All inactive devices that share the same clock and data lines must be kept in the TEST-LOGIC-RESET state. This state causes standard compliant devices to disable their output drivers on DATA OUT to avoid interfering with the active device. [1]

As it can be seen in Figure 3.6, applying a logic high (`1`) to the TEST MODE SELECT input leads the TAP state machine into the TEST-LOGIC-RESET state within at most 5 TCK cycles. The state machine keeps idling in this state as long as the input is held at logic high. The programmer can therefore selectively access each device's TAP without using any external

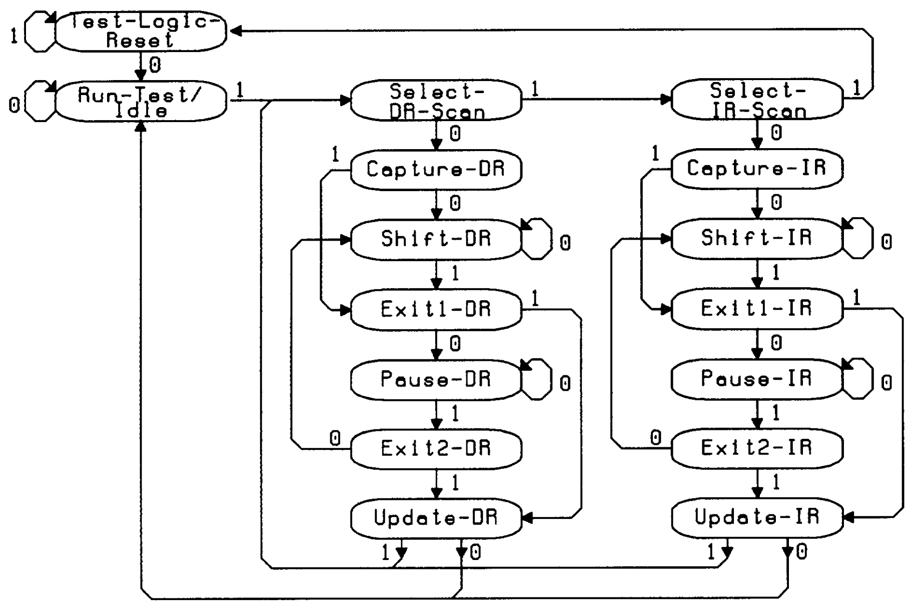


Figure 3.6: IEEE 1149.1-2001 TAP State Machine controlled by TMS (source: [1])

multiplexing hardware by using the associated TMS line and leaving all others at their inactive (high) level.

TMS0	Node 0 Microcontroller
TMS1	Node 1 Microcontroller
TMS2	Node 2 Microcontroller
TMS3	Node 3 Microcontroller
TMS4	Zigbee Bridge
TMS5	External JTAG Chain

Table 3.2: TMS Association to Programmable Devices

Bus Taps

The companion controller also serves as a serial bus bridge to aid with debugging serial data protocols. `BUS0` and `BUS1` are connected to hardware UART[10] cores inside the companion controller. Data on `BUS0` and `BUS1` can be read and written, which allows for bus snooping (debugging) as well as fault injection and protocol robustness analysis.

Logic Analyzer

The companion controller reserves 32 dedicated data lines for capturing data from the four Micro-controller Nodes. Each node has a directly connected 8-bit bus wired to the companion controller. For added flexibility, bus switches have been added between each 8-bit bus and selected dedicated function signals like PWM[11] or the communication bus I/O signals of each node. These are controlled by a shared active-low LASEL signal. While LASEL is asserted, the primary logic analyzer bus on all nodes should be placed in a *High Impedance* state to avoid data corruption. Unused lines (see Table 3.3) may be driven regardless of the state of LASEL.

Bit	*Node 0*	*Node 1*	*Node 2*	*Node 3*
7	Bus 0 TX			
6	Bus 0 RX			
5	Bus 1 TX			
4	Bus 1 RX			
3	–	–	–	–
2	–	HEATREQ	LCD_DDIR	LEDMATRIX_SCL
1	–	FANPWM	LCD_TEAR	LEDMATRIX_SD
0	Clock IRQ	FANSENSE	LCD_BACKLIGHT	LEDMATRIX_LATCH

Table 3.3: Mapping of Special Node Functions to Logic Analyzer Bits

[10] Universal Asynchronous Receiver Transmitter
[11] Pulse Width Modulation

CHAPTER 4

Implementation

This chapter describes the three modules the system is composed of. The *mainboard* hosts the system power supply, Programming and Debugging Interface, Thermal Control Path, Wireless Bridge, two Bus Coupling Units per node and all other node peripherals. It offers sockets for four *CPU module*s and one LCD/TFT display.

The *CPU module* is a small pluggable PCB that hosts a micro-controller along with power supplies and clock generation. It plugs into the *mainboard* and is held in place by the connectors. One module is required for each Real-Time network node. The *LCD module* is a separate pluggable PCB that mounts and interfaces the LCD panel.
A complete system requires one *mainboard*, four *CPU module*s and one *LCD module*.

4.1 CPU Module

This module provides an abstraction layer between the *mainboard* and the employed micro-controller architecture. It contains a micro-controller along with its power supply and a crystal oscillator for clock generation.

Electrical Connection

The *CPU module* is connected by two keyed[1] Hirose DF9-31 connectors. Table 4.1 and Figure 4.1 descibe all I/O signals as seen from the CPU module.

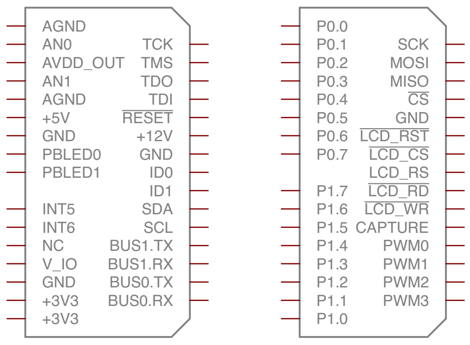
[1] keying ensures that a connector will not be inserted the wrong way

AGND	Analog Ground
AN$[0,1]$	Analog Inputs
AVDD_OUT	Analog supply voltage. Supplied by CPU module.
PBLED_$[0,1]$	Push Button LED drive outputs
INT$[5,6]$	micro-controller Interrupt inputs
RESET	low-active micro-controller reset input. Should be driven open-drain.
NC	no connection
V_IO	I/O supply voltage. Supplied by CPU module.
GND	Ground (0V)
+3V3	CPU module supply (+3.3V)
+5V	CPU module supply (+5V)
+12V	CPU module supply (+12V)
TCK	JTAG clock input
TMS	JTAG test mode select input
TDO	JTAG test data output
TDI	JTAG test data input
ID$[0,1]$	socket identification bits
SDA,SCL	I^{2}C interface
BUS$[0,1]$.TX	Bus Transmit outputs
BUS$[0,1]$.RX	Bus Receive inputs
P0.[0..7]	Port 0 (8 bit)
P1.[0..7]	Port 1 (8 bit)
SCK	SPI^2 clock output
MOSI	SPI master out slave in
MISO	SPI master in slave out
CS	SPI active low chip select output
LCD_RST	LCD interface: controller reset output
LCD_CS	LCD interface: chip select output
LCD_RS	LCD interface: register select output
LCD_RD	LCD interface: read strobe output
LCD_WR	LCD interface: write strobe output
CAPTURE	signal edge capture input
PWM[0..3]	PWM outputs 0..3

Table 4.1: CPU module interface

Power Considerations

The CPU module can draw power from any of the supplied $+12V$, $+5V$ or $+3.3V$ lines. The combined power consumption must not exceed $1W$.



(a) left connector (b) right connector

Figure 4.1: CPU Module Pin Locations

Implementation Details

The existing TTP/A[3] implementation is designed to run on Atmel's ATMega micro-controllers. To avoid porting the existing source code, a micro-controller of this family must be used. Additionally, a second communication interface (`BUS1`) must also be provided by the *CPU module*. This requires a second UART, which is only included in the larger ATMega models.

The *CPU module* is populated with Atmel's ATMega128A or ATMega1281 controller in the 64-lead TQFP[4] package. These micro-controllers offer all necessary peripheral cores (SPI, two-wire serial, multiple PWM units, two UARTs) and a large internal flash memory of 128KiB[5]. [8]
The main clock is generated by an external 14.7456MHz crystal oscillator. This clock frequency is required to ensure compatibility with the existing TTP/A implementation. A supply voltage of at least 4.5V is required to run at this clock speed. [2]
The micro-controller draws power from the supplied $+5V$ rail and derives an analog power supply from the $+12V$ rail using a ferrite bead for decoupling and a $5V$ low-noise linear voltage regulator. The module further contains one red and one green LED for immediate status indication.

4.2 LCD Module

This module is a small PCB that hosts the LCD panel assembly 32NHF0H by Seiko Instruments Inc, which contains a high resolution TFT panel and an integrated driver IC[6]. [29]

[3]Time Triggered Protocol A
[4]Thin Profile Plastic Quad Flat Package
[5]Kilobyte = 1024 Byte
[6]Integrated Circuit

Electrical Connection

The module is connected to the *mainboard* via a single Hirose DF9-31 header. This header exposes an MIPI[7]-DBI[8] Type B compliant interface.

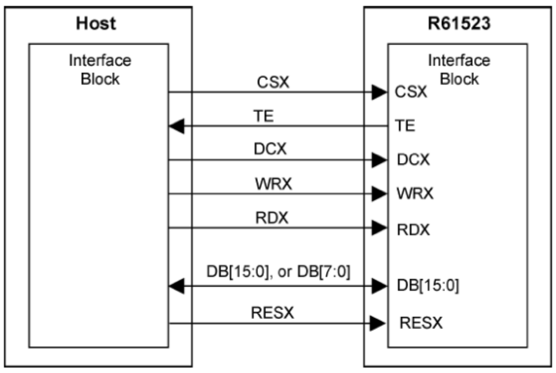


Figure 4.2: LCD Controller Interface [27]

In this design, the 8-bit variant of the MIPI-DBI Type B Bus is used. The following signals are routed through to the *mainboard* header.

CSX	Chip Select
TE	*Tear Effect out (optional)*
WRX	Write Strobe
RDX	Read Strobe
DB[0..7]	Bi-directional Data bus
RESX	Controller Reset

The recommended I/O voltage ranges from $1.65V$ to $3.6V$, while a voltage of $1.8V$ is stated as typical. [27, 29]
Since the micro-controller I/O voltage is flexible and the display controller inputs are not over voltage tolerant, this interface cannot be driven directly by the node I/O ports. A voltage translator has been introduced to avoid destroying the display controller when using incompatible voltage levels. While this component does require an additional control signal to specify the data flow direction on the (translated) data bus, the gained flexibility outweighs the additional design complexity. An optional circuit has been included that derives this control signal from the read strobe input. In cases where this is undesired, the data direction signal can be driven via the node interface – the automatic generation circuit can be disabled by removing a solder jumper.

The system-side interface contains the following signals (as seen from the LCD module):

[7]Mobile Industry Processor Interface
[8]Display Bus Interface

VIO	I/O voltage input
GND	Ground ($0V$)
+3V3	Power Supply $+3.3V$
+5V	Power Supply $+5V$
+12V	Power Supply $+12V$
BL_EN	Backlight Control
DDIR	Data Direction input
TEAR	Tearing effect output
NC	no connection
DATA[0..7]	Data Bus (bi-directional)
WR	Write Strobe input
RD	Read Strobe input
DSEL	Data/Command Select input
CS	Chip Select input
RST	Reset input

Table 4.2: LCD module interface

Backlight

The display panel is illuminated by a string of 6 LEDs connected in series. This provides the best brightness uniformity, but also requires a high drive voltage of more than 20 Volts. Since the highest available voltage is 12 Volts, this drive voltage must be generated by a step-up regulator.

Power Considerations

The *LCD module* can draw power from any of the supplied $+12V$, $+5V$ or $+3.3V$ lines. The combined power consumption does not exceed $3W$.

Implementation Details

The *LCD module* contains an I/O voltage level conversion circuit that translates between the Micro-controller Node and the display controller. The conversion is done by an NXP Semiconductors 74ALVC164245, a 16 bit dual-supply translating transceiver. [20]

One of the two available 8-bit blocks is used in a fixed-direction configuration to translate the display control signals. The second 8-bit block is used for the display data bus. Its direction can either be configured by a dedicated signal from the Micro-controller Node or derived from the read strobe signal.

The backlight drive voltage is generated by Fairchild Semiconductor's FAN5343, a monolithic constant current boost converter with a single-wire digital control interface. LED string current (and therefore brightness) can be adjusted in 32 steps by applying digital pulses to the module's BL_EN input. [12]

4.3 Mainboard

The *mainboard* hosts the system power supply, programming and debugging facilities, and all node peripheral devices. It provides a number of external interfaces:

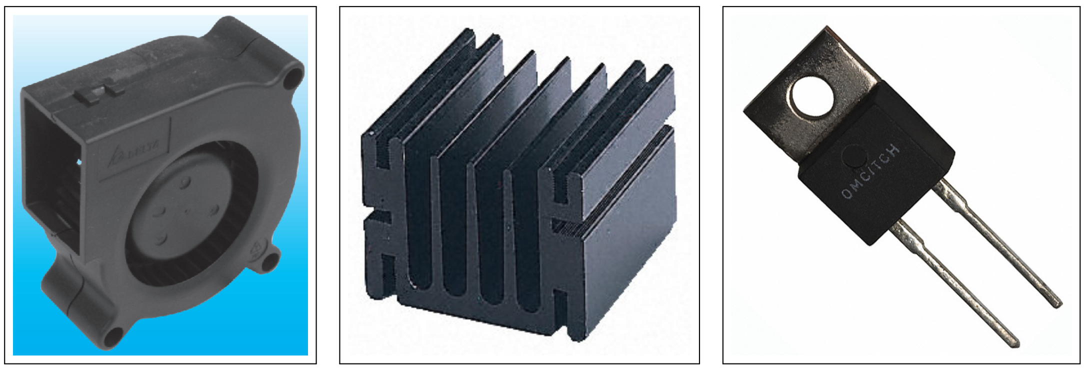
- Power supply inlet. Requires a stabilized +12V (nominal) source with at least 40W.
- USB Type B connector for device programming
- A JTAG interface for programming the companion controller [10]
- An additional ISP[9] header for the companion controller (+3.3V serial port)
- An extension header with bus access and programming support (see schematics in Appendix A)
- A header exposing `BUS0` and `BUS1` for debugging

The *mainboard* provides two regulated supply voltages to its components and submodules: +3.3V and +5V. Each voltage is generated by a non-isolated Point-of-Load synchronous buck converter that can deliver 15W of continuos output power. [18]

The system power inlet is protected against over-current conditions by a fast blowing fuse. Unlimited reverse polarity protection is achieved by a P-channel MOSFET[10] circuit. This circuit provides a good tradeoff between part count and power loss and does not suffer from a significant voltage drop. [11, 26]
Red and green LED indicators show if the board supply voltage is applied correctly.

Heat Sink, Fan and Heating Circuit

The control path is constructed around an extruded aluminum heat sink (Figure 4.3b).



(a) Cooling Fan (b) Heat Sink (c) Heating Resistor

Figure 4.3: Control Path Components

[9]In-System Programming
[10]Metal Oxide Semiconductor Field Effect Transistor

Two resistors in the TO-220 package (Figure 4.3c) are used to dissipate power into the heat sink to raise its temperature under direct control of NODE 1.

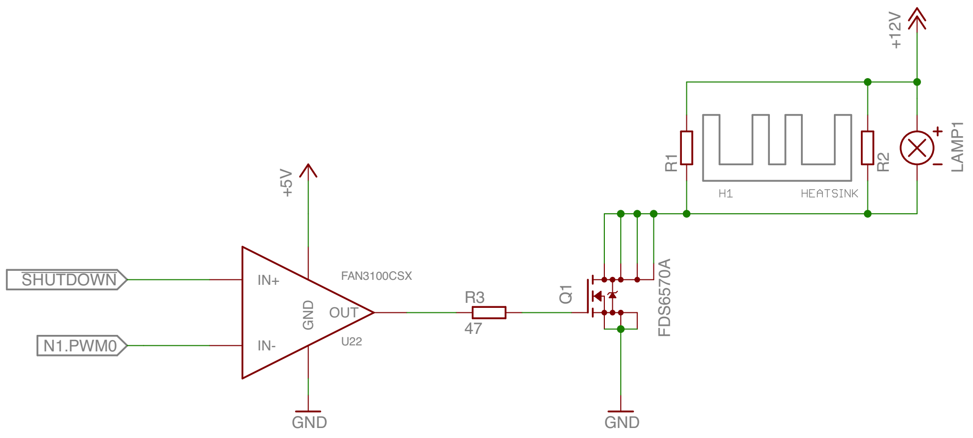


Figure 4.4: Heating Driver Circuit

Figure 4.4 shows the drive circuit for the heating resistors using NODE 1's PWM0 signal. A discrete low side MOSFET gate driver [13] ensures fast switching performance which is desirable to avoid nonlinear behavior of the circuit at higher switching frequencies. An incandescent light bulb provides direct visual feedback of the resistor dropout voltage. When driven with a sufficiently high PWM frequency, the brightness of the light bulb will reflect the average power dissipation in the resistors.

A PWM-controlled radial blower (Figure 4.3a) is mounted close to the heat sink to create a forced airflow over the heat sink's fins, thereby reducing its temperature. The selected 50mm radial fan provides a 4-wire interface that accepts a 21 kHz to 28 kHz PWM control signal to limit the rotor speed. This concept makes use of the fan's integrated power switching stage which reduces the overall component count for fan control. [16]
A tachometer signal that generates two pulses per fan rotor rotation is fed back to the micro-controller's capture input.

Companion Controller

The LPC1760[11] micro-controller by NXP Semiconductors is used as companion controller. This 32-bit ARM Cortex M3-based micro-controller can be clocked at up to 120Mhz using an integrated PLL[12]. The on-chip USB 2.0 device is used for host PC communication. The devices offer up to 64KiB of SRAM[13] and 512KiB of flash memory. [23]

The 32-bit logic analyzer bus is connected to the companion controller in groups of 8 bits. While it has not been possible to map out an entire 32-bit I/O port for capturing logic analyzer

[11] Any of the pin-compatible LPC1769/68/66/65/64 can be used.
[12] Phase Locked Loop
[13] Static Random Access Memory

data, care has been taken to place each group on a byte boundary in the controller's ports. This makes it possible to use DMA[14] transfers to capture the data with very little computational overhead.

The JTAG programmer's CLOCK, DATA IN and DATA OUT lines are connected to an SPI peripheral core inside the LPC1760 to allow fast and efficient data transfers at up to 50 MHz, depending on the configured internal bus clock frequency. [23]

Martin Schmölzer's XML[15]-based programmer software [28] was designed to run on the AVR micro-controller family. The software intended to run on the companion controller contains a port of Martin Schmölzer's device software as well as several feature enhancements including the logic analyzer and bus taps. [14]
Table 4.3 shows the pin mapping for all major function blocks of the companion controller.

TDO	P0.0	JTAG Data Out coming from all programmable devices
TDI	P0.9	JTAG Data In going to all programmable devices
TCK	P1.31	JTAG clock output
TMS0	P1.24	JTAG TMS for NODE 0
TMS1	P1.25	JTAG TMS for NODE 1
TMS2	P1.26	JTAG TMS for NODE 2
TMS3	P1.27	JTAG TMS for NODE 3
TMS4	P1.28	JTAG TMS for the Zigbee Bridge
TMS5	P1.29	JTAG TMS for the external scan chain
RST0	P1.0	NODE 0 Reset
RST1	P1.1	NODE 1 Reset
RST2	P1.4	NODE 2 Reset
RST3	P1.8	NODE 3 Reset
RST4	P1.10	Zigbee Bridge Reset
RST5	P1.9	external scan chain reset
LA Node 0	P0.0..7	JTAG Logic Analyzer bus for NODE 0
LA Node 1	P0.16..23	JTAG Logic Analyzer bus for NODE 1
LA Node 2	P1.16..23	JTAG Logic Analyzer bus for NODE 2
LA Node 3	P2.0..7	JTAG Logic Analyzer bus for NODE 3
LASEL	P1.14	Logic analyzer special function selector (active high)
PROGLED0	P2.11	General Purpose Programmer LED (green)
PROGLED1	P2.12	General Purpose Programmer LED (green)
PROGLED2	P2.13	General Purpose Programmer LED (green)
PROGLED3	P0.11	General Purpose Programmer LED (red)

Table 4.3: Companion Controller Pin Assignment

Two kinds of programming interfaces are provided to program the companion controller

[14]Direct Memory Access
[15]eXtensible Markup Language

itself. A serial ISP header can be used in "production" environments to allow easy firmware uploading. The interface for flash programming uses a simple text-based serial protocol. [24] Full symbolic debugging can be performed by using the 20-pin standard ARM JTAG header.

Bus Components

The communication busses are driven by dual-gate little logic 74LVC2G07 open drain drivers. Bus isolation is performed by 74LVC2G17 Schmitt triggers. [21, 22]

Passive termination pull-up resistors matched to the trace impedance are located at both ends of the buses. This should allow to operate the bus at frequencies way beyond the maximum usable baud rate of the targeted micro-controllers. [5, 15]

LED Matrix

A 6-digit LED matrix is connected to NODE 3. The matrix is organized in 7 rows by 30 columns. Rows are selected by a 3-to-8 line decoder in conjunction with one high side P-channel MOSFET for each row.

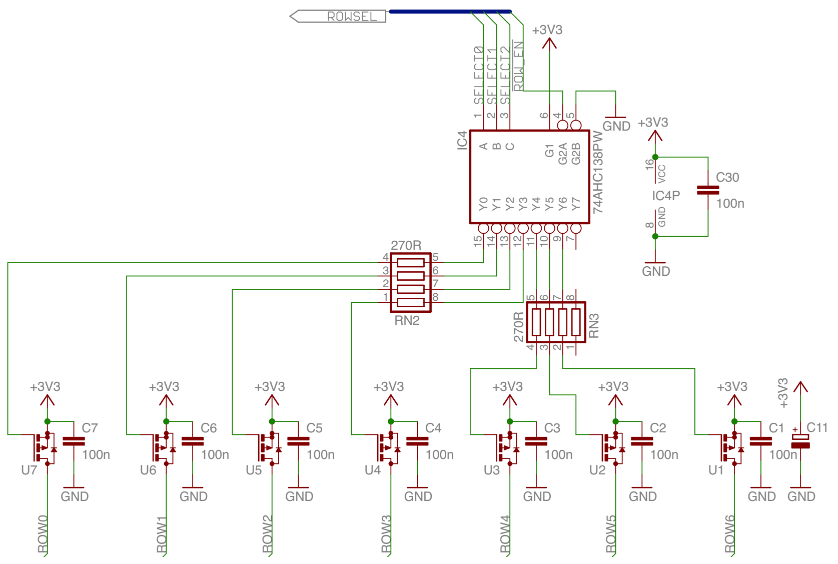


Figure 4.5: LED Matrix High Side

The columns are controlled by constant current low-side LED drivers. The CAT4016 by ON Semiconductor is a cascadable 16-channel constant current LED driver that is internally organized as a shift register. [25]

A column of 30 LEDs is driven by two 16-bit LED drivers, leaving each LSB[16] unused. The cascaded digital shift register ports are connected to NODE 3's SPI interface.

A multiplexing sequence is required to display all 7 rows in a round-robin fashion. If this sequence is driven fast enough, the rows will appear continuously lit to a human observer.

[16]Least Significant Bit

Wireless Bridge

A drop-in solution has been adopted to integrate a wireless bridge into the *mainboard*. The Atmel Zigbit module ATZB-24-A2 includes an ATMega1281 micro-controller, a Zigbee compliant radio chip and a chip antenna. [7]
This hardware is capable of running the TinyOS operating system. Its hardware layout is similar to the supported IRIS mote. [4, 6]

A patch adding support for this module against TinyOS version 2.1 is supplied with this thesis (see section 6).

Digital Temperature Sensors

The heat sink temperature is measured at four points at the bottom of the heat sink by integrated digital temperature sensors. Three of these sensors are connected to NODE 1's I^2C[17] bus, while the fourth is wired directly to the companion controller. The fully integrated SE95 digital temperature sensors by NXP Semiconductors offer very high accuracy sensing without the need for analog calibration or correction. This sensor reports its temperature in degrees centigrade with a resolution of 0.03125 °C/bit. [19]

Battery-backed Real-Time Clock

NODE 0 is equipped with a Real-Time clock that is supplied with backup power provided by a battery. The ST M41T81S is an I^2C compliant low-power clock IC that can also be used to generate timed interrupt requests. [30]

Extensibility

The *mainboard* features a 10-pin dual-row 2.54mm connection header that allows interfacing and programming custom add-on devices. This connector offers a JTAG programming chain and connects to `BUS1`.

+5V	external power supply	1	2	JTAG data in	TDI
$\overline{RST}$	external device reset	3	4	JTAG clock	TCK
TMS5	JTAG TMS for external chain	5	6	JTAG data out	TDO
GND	optional Ground or keying	7	8	device presence sense	EXT_SENSE
BUS1	serial communication `BUS1`	9	10	Ground (0V)	GND

Table 4.4: Extension Port Pins

4.4 Simulations

LTSpice by Linear Technology was used to create pre-layout analog electrical simulations. These simulations were run on equivalent schematics shown in Figure 4.6 and Figure 4.8.

[17] Inter-Integrated Circuit

LED Matrix High Side

The LED Matrix high-side driver was simulated to verify its correctness. A P-channel MOSFET array is used to select one out of seven rows which are driven at their low side by a shift register chain. The gate signals are generated by a 3-to-8 line demultiplexer with inverted outputs. Since

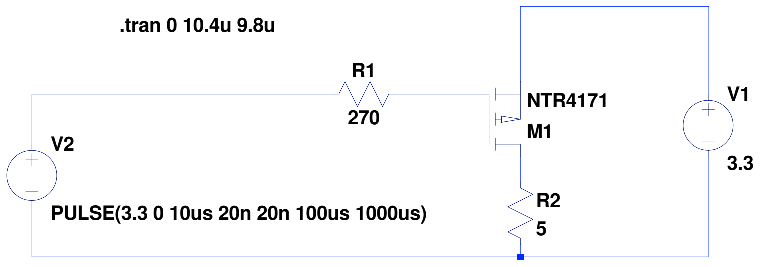


Figure 4.6: Equivalent Schematics for Simulation

the high side needs to supply all 30 LEDs in parallel, a total current draw of up to $600mA$ can be expected. The expected switching waveform of a single high-side row driver is shown in Figure 4.7, which shows an estimated timing of the drain-source current (blue) in response to a 1-to-0 transition at the MOSFET's gate input (green).

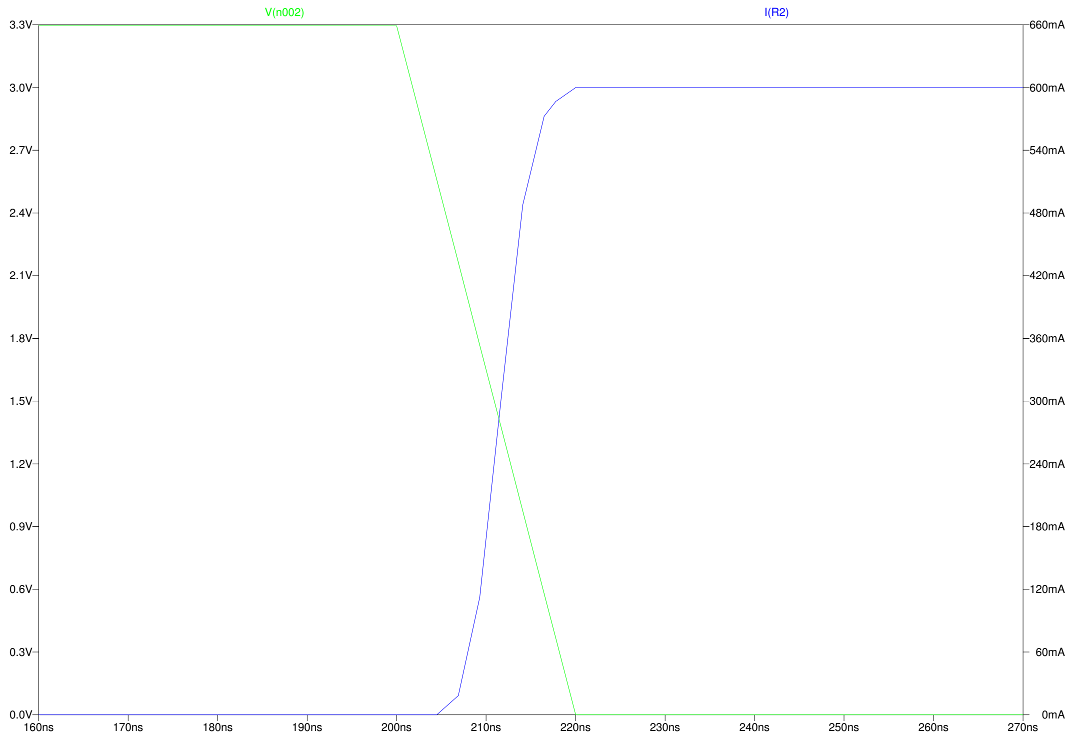


Figure 4.7: LED matrix row control voltag vs. LED current

Heating Driver

The heating resistors are driven by an N-channel MOSFET circuit. A simplified equivalent circuit shown in Figure 4.8 was used for this simulation. The simulation in Figure 4.9 provides a

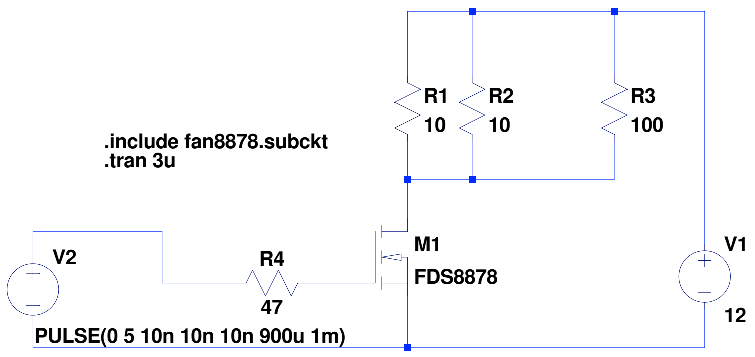


Figure 4.8: Equivalent schematics for heating simulation

timing estimate of the gate drive current (blue), MOSFET power loss (red) and resistor heating current (turquoise) in response to a step input (green).

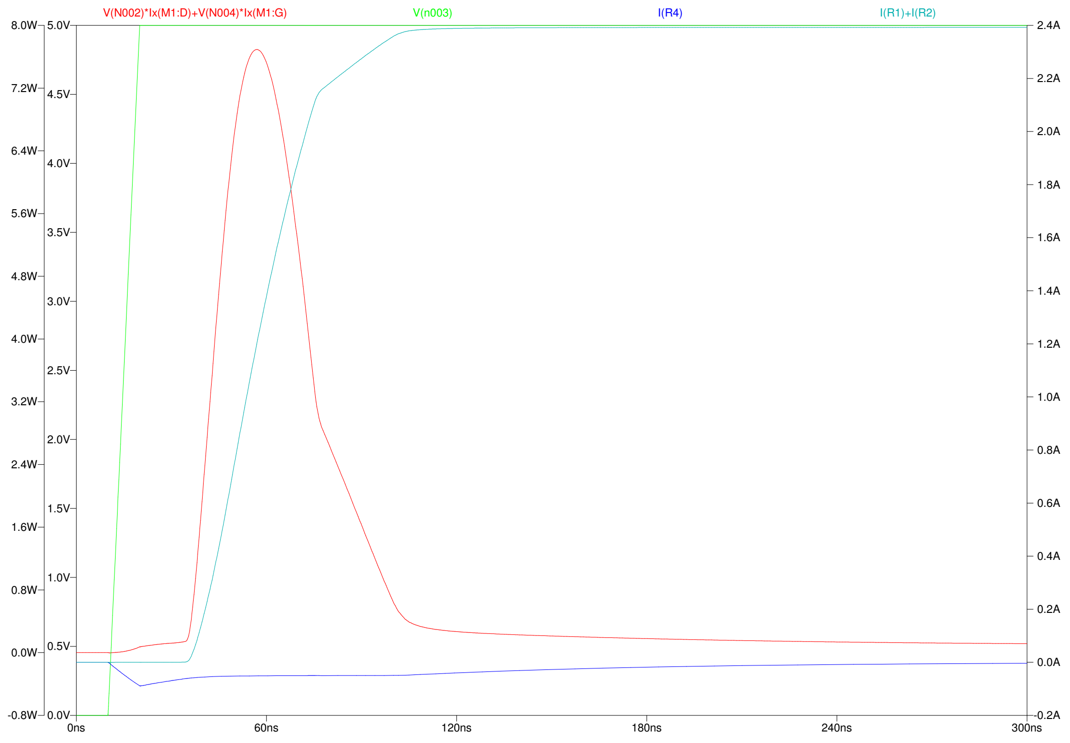


Figure 4.9: Heating control voltag vs. resistor current and MOSFET power dissipation

4.5 Schematics and Parts Libraries

All PCBs were designed using the CadSoft EAGLE schematic capture and PCB design software. This program allows consistent schematic editing and layouting by using back-annotation to synchronize the schematic while editing the layout and vice versa. [9]

Most of the definitions for electronic components used in this design are not included in EAGLE's parts libraries. The missing libraries were created from the vendor-supplied data sheets.

4.6 PCB Design

An initial mechanical system layout plan (Figure 4.10) was created using an open-source CAD[18] software.

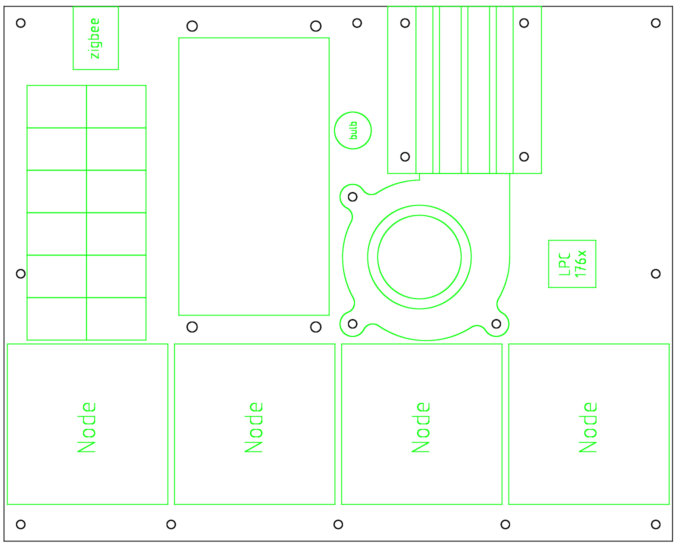


Figure 4.10: Preliminary Mechanical Mainboard Layout

This preliminary mechanical layout was then conveyed to the EAGLE EDA[19] software. In order to keep the manufacturing effort to a minimum, surface mounted electronic components were chosen in favor of through-hole parts whenever possible. Furthermore, all electronic components are placed on the top side of the *mainboard*. Boards with single-side components can even be assembled "manually", that is by placing the components by hand and then using a reflow process for soldering.

All PCBs were routed on two copper layers which allows very cost-effective manufacturing. The EAGLE autorouter did not produce satisfactory results. Figure 4.12 shows a section from the bottom copper layer on the *mainboard*, routed automatically and by hand. The autorouter

[18]Computer-Aided Design
[19]Electronic Design Automation

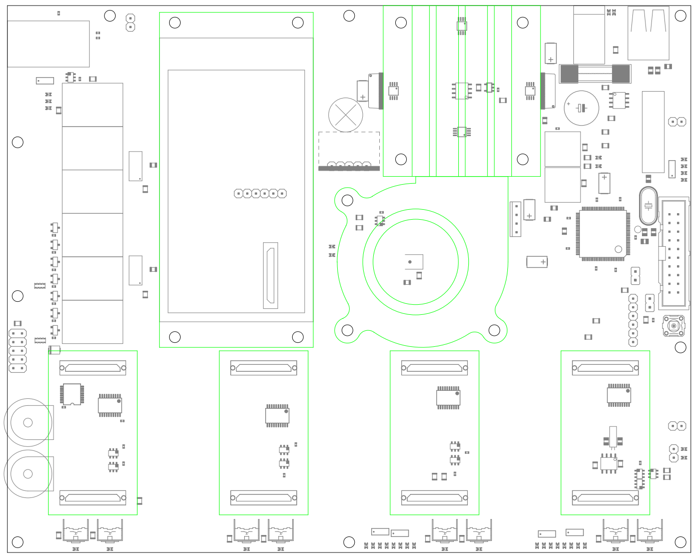


Figure 4.11: Final Mechanical Mainboard Layout

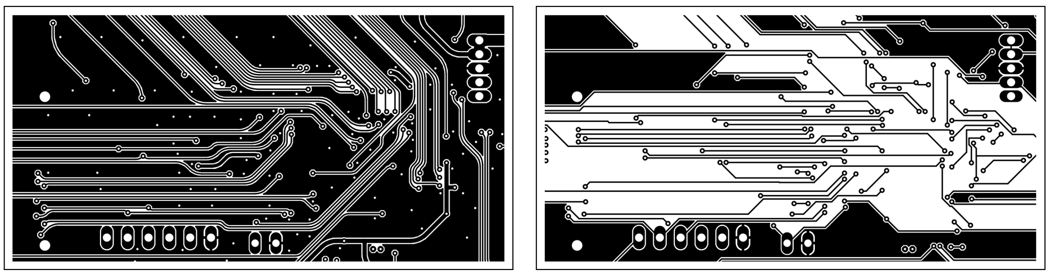

(a) Manual Routing (b) EAGLE Autorouter

Figure 4.12: Manual Routing vs. Autorouter Results

failed to create a continuous ground plane from the defined copper pour area (see Figure 4.12b). It also ignored the analog ground areas and left behind 40 unrouted connections which would have required a final manual routing pass.

In order to create boards with improved electrical characteristics, the autorouter result was discarded and a complete manual routing was performed. The FreeRoute push & shove routing program [31] was used to quickly create a first routing pass. Subsequent refinements were done in the EAGLE layout editor, since exporting from EAGLE to FreeRoute is not a lossless process when certain advanced layout features are used.

CHAPTER 5

Results

5.1 Hardware

The completely assembled prototype *mainboard* is presented in Figure 5.1. This PCB was as-

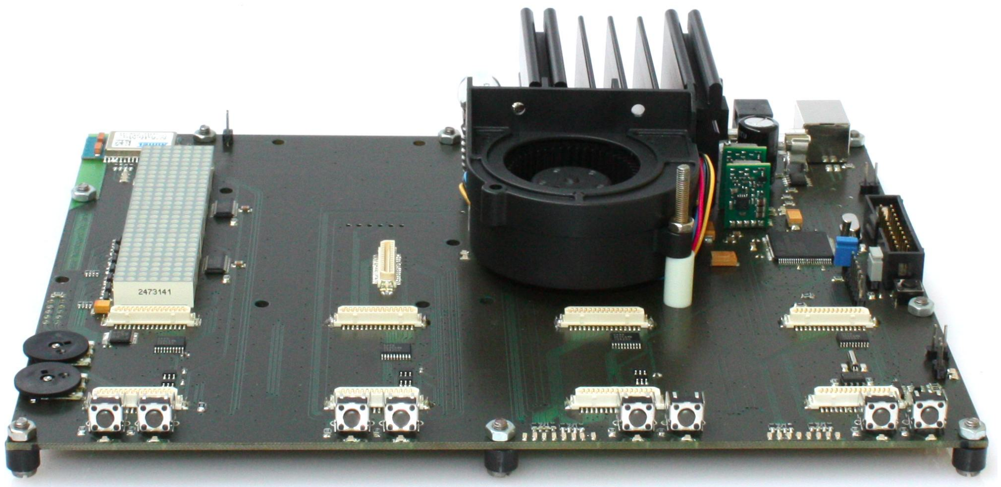


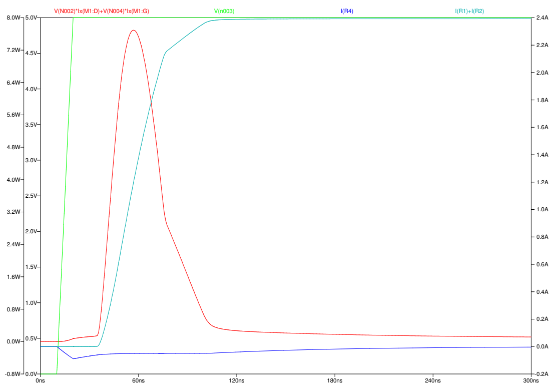
Figure 5.1: The new Embedded Systems Engineering Mainboard

sembled by hand, manual pick-and-place was followed by a reflow cycle and a small number of reworks with a soldering iron. As a last step, the through-hole components were soldered in manually.

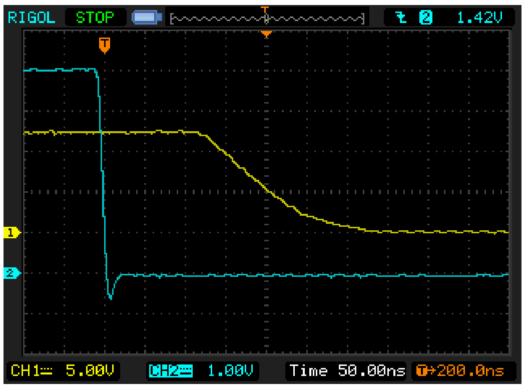
The thermal inertia of the control path is considerable. Even at a heat ingress of $P = \frac{U^2}{R} = \frac{144}{7.5} = 19.2W$, it takes several seconds of heating until the temperature of the heat sink begins to rise noticeably. This can be considered an advantage, since slower control cycles can be used. A later version of the hardware could also be equipped with smaller heating resistors for increased power dissipation.

5.2 Measurements

Since the heating circuit is one of the most important parts of this design, its performance was evaluated using an oscilloscope. A comparison against the simulated characteristics is shown in Figure 5.2. Some of simulated parameters could not be obtained for measurement. The drive signal and the voltage drop across the MOSFET are inverted in the measured waveform.



(a) Simulation



(b) Measurement

Figure 5.2: Heating Driver Circuit: Simulation vs. Measurement

A timing deviation of a factor ≈ 2 can be observed. Still, the response time of $< 300ns$ is excellent. Since the typical PWM frequency will not be higher than 25 kHz, the rise time is very short in comparison to the duty cycle, yielding a sufficiently linear response.

5.3 Change Requests

The prototyping process revealed several improvements and fixes to be implemented in a subsequent version of the hardware.

BOM fixes: Some electronic components were referenced incorrectly in the schematics. This led to wrong part variants being ordered and assembled onto the boards. Affected components are the LED matrix driver ICs and the LED arrays.

Companion controller oscillator: The main oscillator for the companion controller was selected to provide 8 MHz, while Jürgen Galler's software expects a 12 MHz main clock. The faster clock speed allows to use a secondary PLL as clock source for the USB subsystem, which increases the flexibility of main PLL and therefore the choice of CPU clock speeds.

Bus Pull-Up Resistor Values: Incorrect resistance values for the bus pull-ups were selected that exceeded the drive capability of the employed bus drivers.

Temperature Sensor Power Supply: A power supply filter circuit should be introduced to improve the measurement error of the digital temperature sensors. This is suggested by the SE95 data sheet and only requires two extra passive components per sensor.

Temperature Sensor Package: The SE95 temperature sensor was used in the very small TSSOP8 package to save board space. Since the power transistor must contact the same surface, the height difference of $\approx$ 1mm must be compensated by an elastic thermal gap filler. A new hardware revision could use the SE95 in the same SO8 package as the transistor, improving the thermal conductance to the temperature sensors because of the reduced distance to the heat sink.

USB SoftConnect: The implemented self-powered USB device does not correctly terminate the host connection when it powers down. As a result, the host device drivers stay active and produce errors when accessed. The situation can be rectified by un-plugging the USB cable. Although the impact should be minimal in practice, a fix should be implemented by using the LPC1760's SoftConnect feature.

TX/RX swap: An error exists in the schematics for the *CPU module*; the RX/TX pairs of both bus links are swapped. This error prohibits the use of hardware UARTs, but does not affect software UART implementations.

CHAPTER 6

Conclusion

The presented work covers the whole electronic design workflow. A system design was proposed, electronic components have been selected and the circuits using them were designed. Finally, the PCB layouts were constructed, the boards were manufactured and assembled, and basic test software was written. This allowed me to gain insight into some of the challenges at different stages of an electronic product design.

PCB Layout

The ambitious design goals – only two copper layers and single side component placement – proved to be difficult to meet. The EAGLE auto-router was unable produce good results under the given circumstances. The solution was manual push and shove routing using a third party Java program. While the provided user interface was a bit hard to get used to, the software quickly produced usable results. The main advantage over EAGLE's built-in layout editor is the ability to respect design criteria like trace width and minimum distances between copper areas during manual routing. This creates a kind of semi-automatic routing environment which is much more flexible than the auto-router.

Software Performance

The *mainboard* shows some of the limits of the EAGLE software package. The design has over 800 electrical connections on an area of $320cm^2$. Copper pour areas are normally computed within a fraction of a second while the same calculation in this design takes about 30 seconds to complete. This makes working with polygons harder than usual. The autorouter worked for more than 5 hours before finishing up.[1] Since all computationally heavy algorithms in EAGLE seem to execute in a single thread, the program does not benefit from modern multi-core CPUs.

[1]This test was conducted using EAGLE 5.11.0 with default autorouter settings on an Apple MacBook, Intel Core 2 Duo 2GHz, 4GB DDR3 RAM, running Mac OS X 10.6.7

Electronic Component Supply Chain

The presented system design requires about 100 distinct electronic components. Most of those are easy-to-acquire mainstream components like resistors, capacitors or standard logic ICs. Some of the more specialized components include micro-controllers, power supply modules, surface mount connectors or mechanical components such as the fan or the heat sink. Those rarer components are only stocked by a few distributors, which made it necessary to split the BOM[2] into three separate orders at different distributors to come by the required parts. Even so, the Hirose DF9-31 surface mount connector went out of stock with almost all distributors during two months spent on PCB design.[3] As a result of this supply shortage, only the mainboard and two Micro-controller Nodes could be assembled and tested for prototyping.

Simulation Accuracy

All design-time simulations were obtained by a numerical pre-layout simulation process that does not respect design parameters like trace resistance or component placement. The measured result timing deviates by a factor of ≈ 2. This leads me to conclude that pre-layout simulation are a well-suited method to study the expected behavior of a circuit when taken with a grain of salt.

Outlook

This thesis presented a working prototype of an up-to-date *Embedded Systems Engineering* learning platform. I hope that it will spark ideas for a lot of interesting applications in the field of distributed embedded computing and embedded-related HCI once it is in use in actual courses at the university.
A number of future enhancements could enrich the learning experience of this work even more.

Remote workplace configurations can be established by placing a "sandwich" board between the *mainboard* and the CPU modules. This additional board can be equipped with a powerful FPGA[4] to implement data snooping and protocol analysis in hardware.

Upgraded CPU modules with stronger processors or more memory can be designed with minimal effort and manufacturing costs.

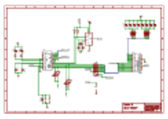
Battery powered remote sensors can deliver measurements wirelessly over the Zigbee radio channel. These modules can be produced cost-effectively and programmed using the *mainboard*'s external programming connector.

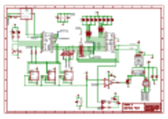
[2]Bill of Materials
[3]The manufacturing facilities of the Hirose Electric Group were hit in the 2011 series of earthquakes in Japan.
[4]Field-Programmable Gate Array

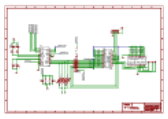
APPENDIX A

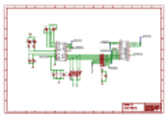
Schematics

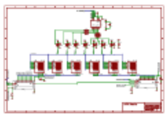
Mainboard

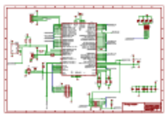

Peripherals for NODE 0

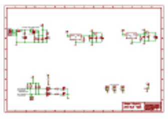

Peripherals for NODE 1

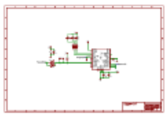

Peripherals for NODE 2


Peripherals for NODE 3

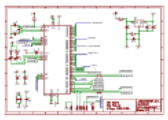

LED Matrix

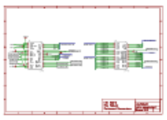

Companion Controller with programming interfaces


Power supply and Bus Taps

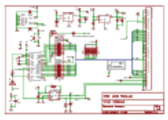

Zigbee Controller

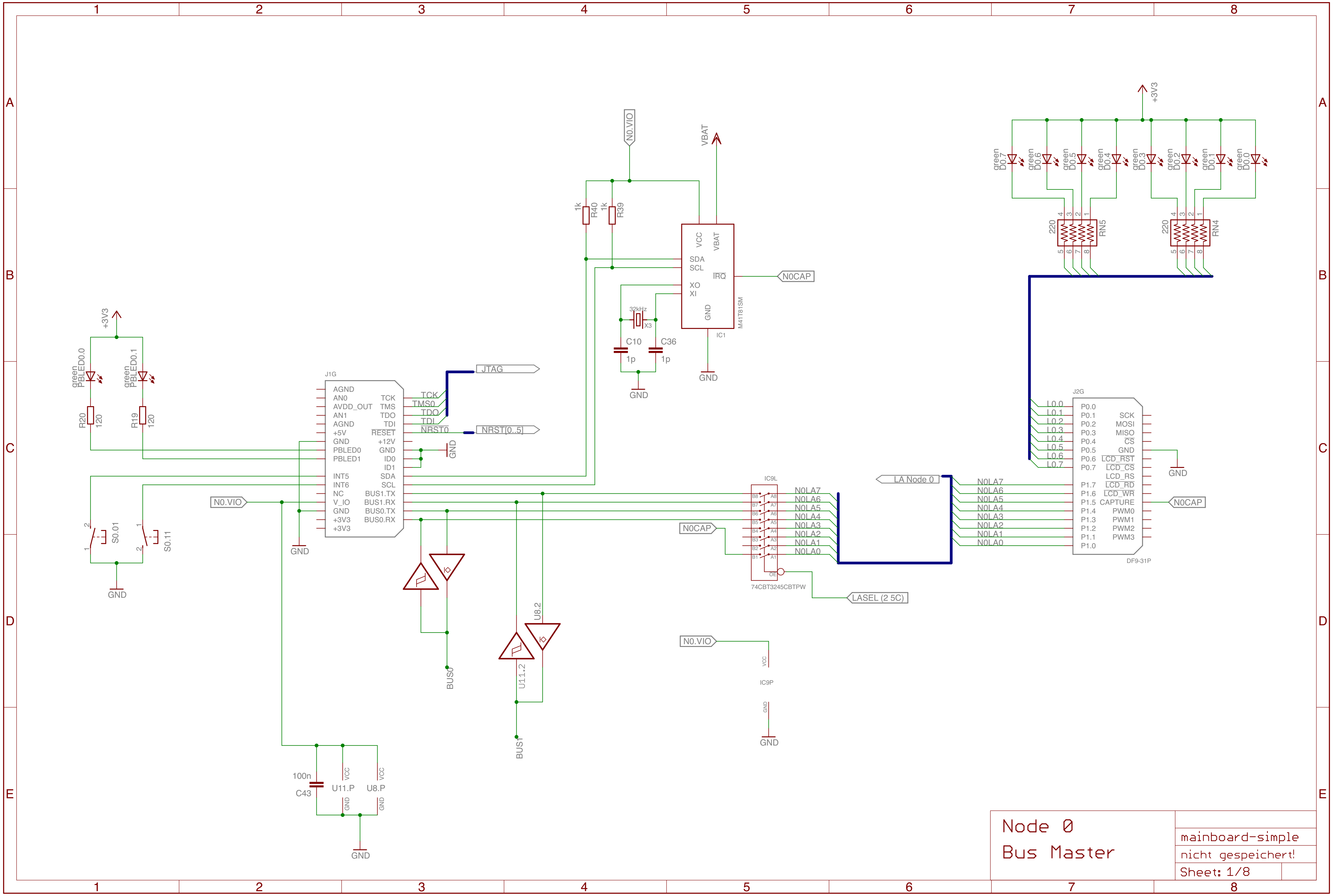
CPU Module


Micro-controller and Analog Power Supply


Node Interface connectors

LCD Module





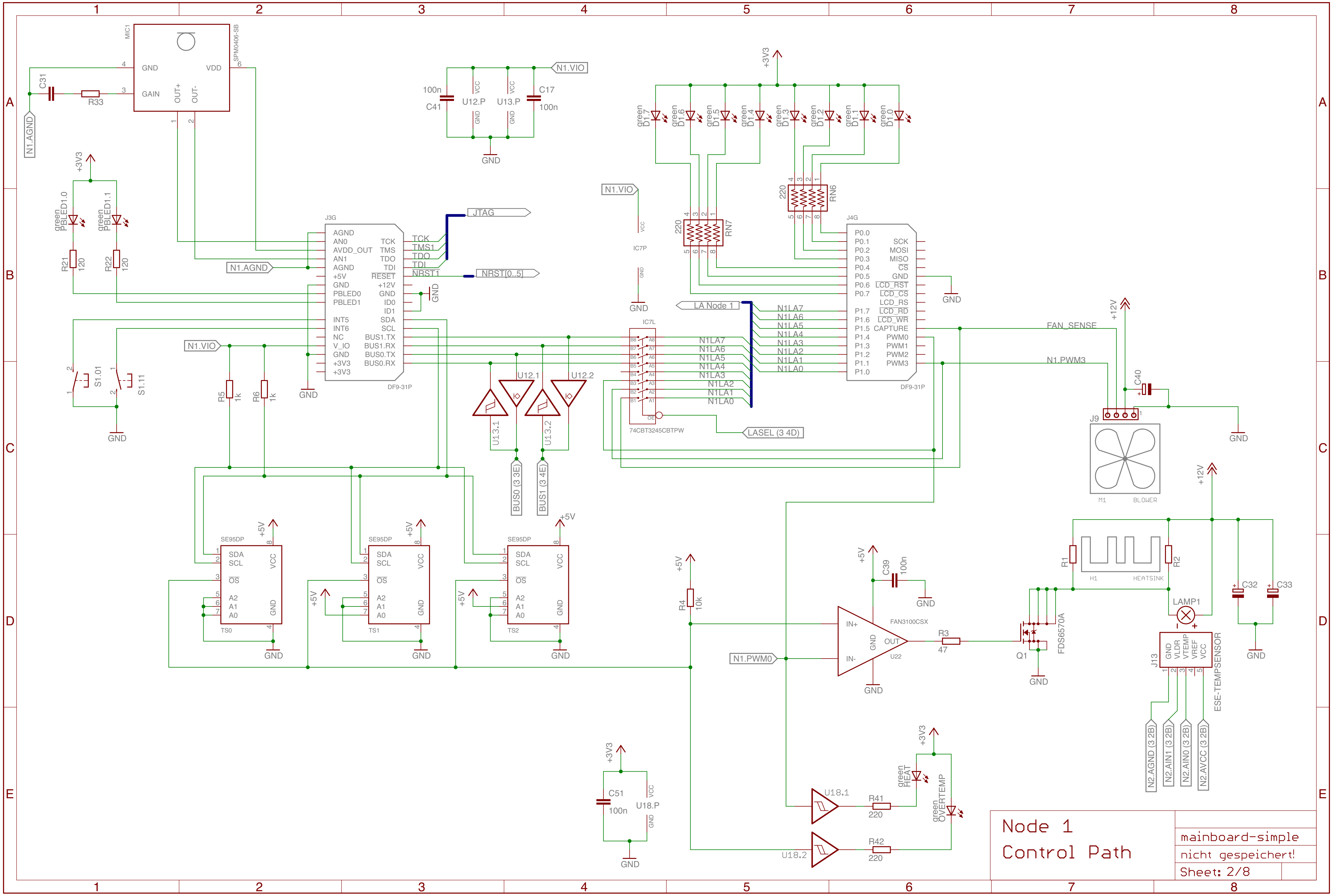
Node 0

Bus Master

mainboard-simple

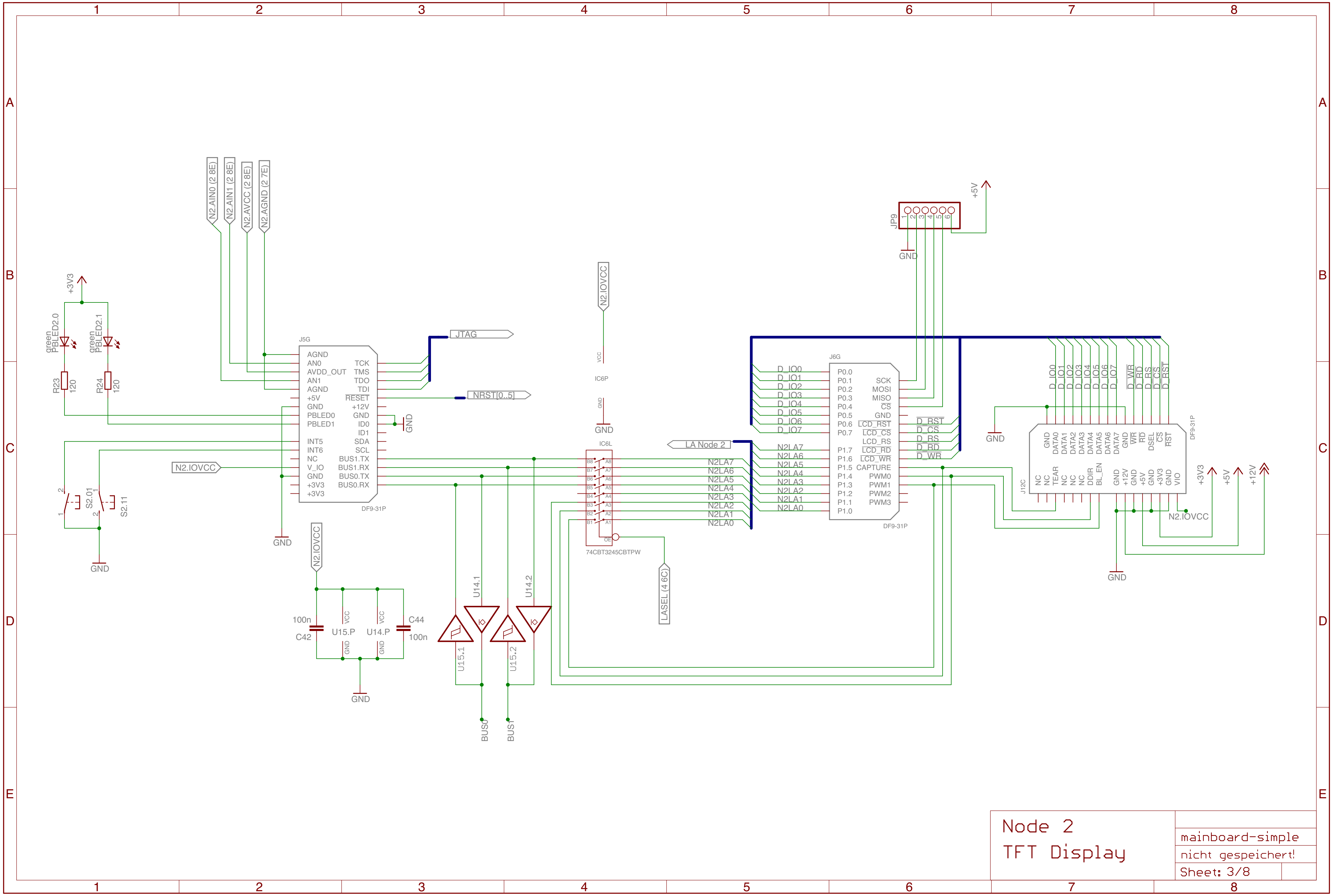
nicht gespeichert!

Sheet: 1/8



Node 1
Control Path

mainboard-simple
nicht gespeichert!
Sheet: 2/8



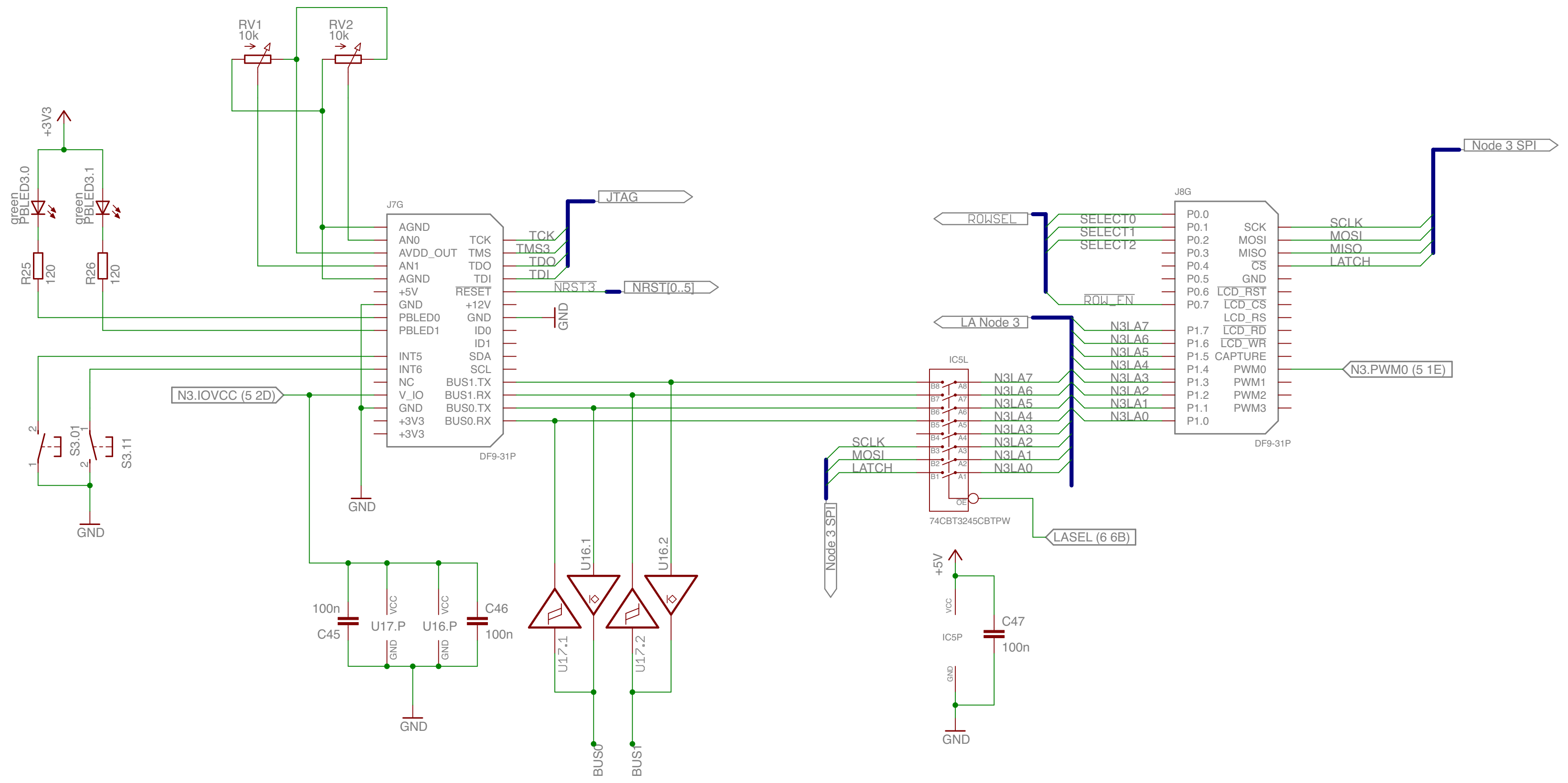
Node 2

TFT Display

mainboard-simple

nicht gespeichert!

Sheet: 3/8



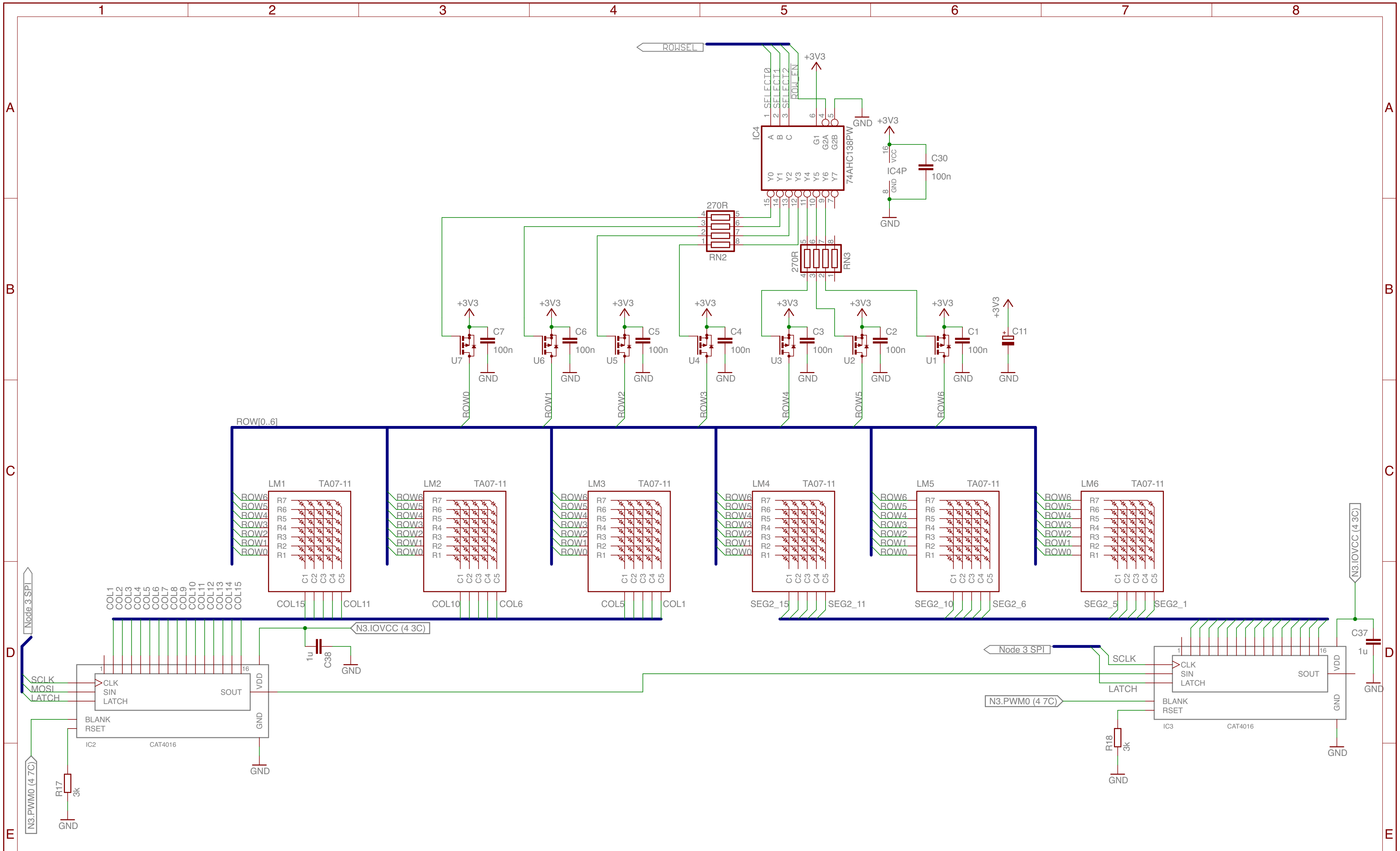
Node 3

LED Matrix

mainboard-simple

nicht gespeichert!

Sheet: 4/8

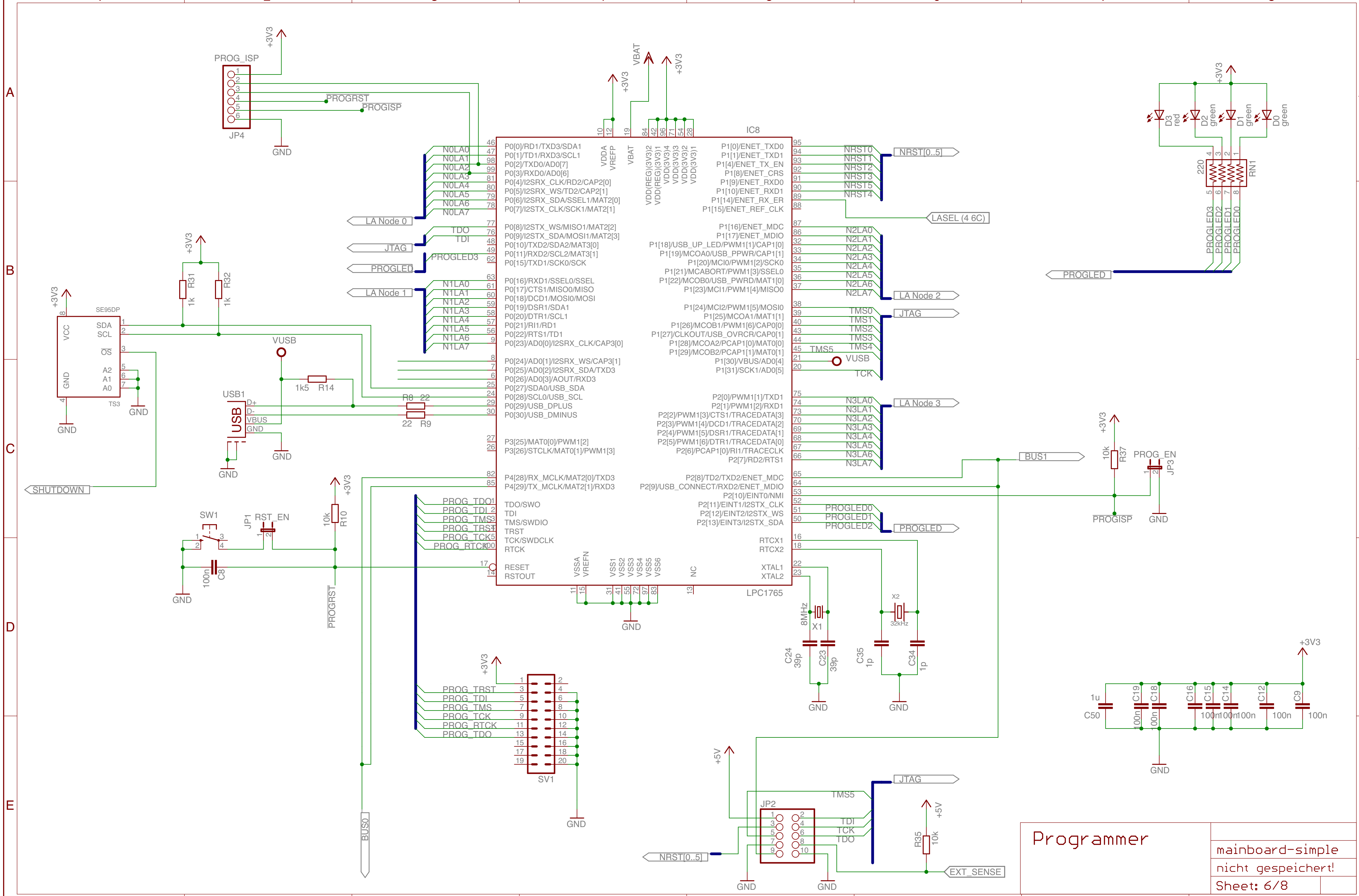


LED Matrix

mainboard-simple

nicht gespeichert!

Sheet: 5/8

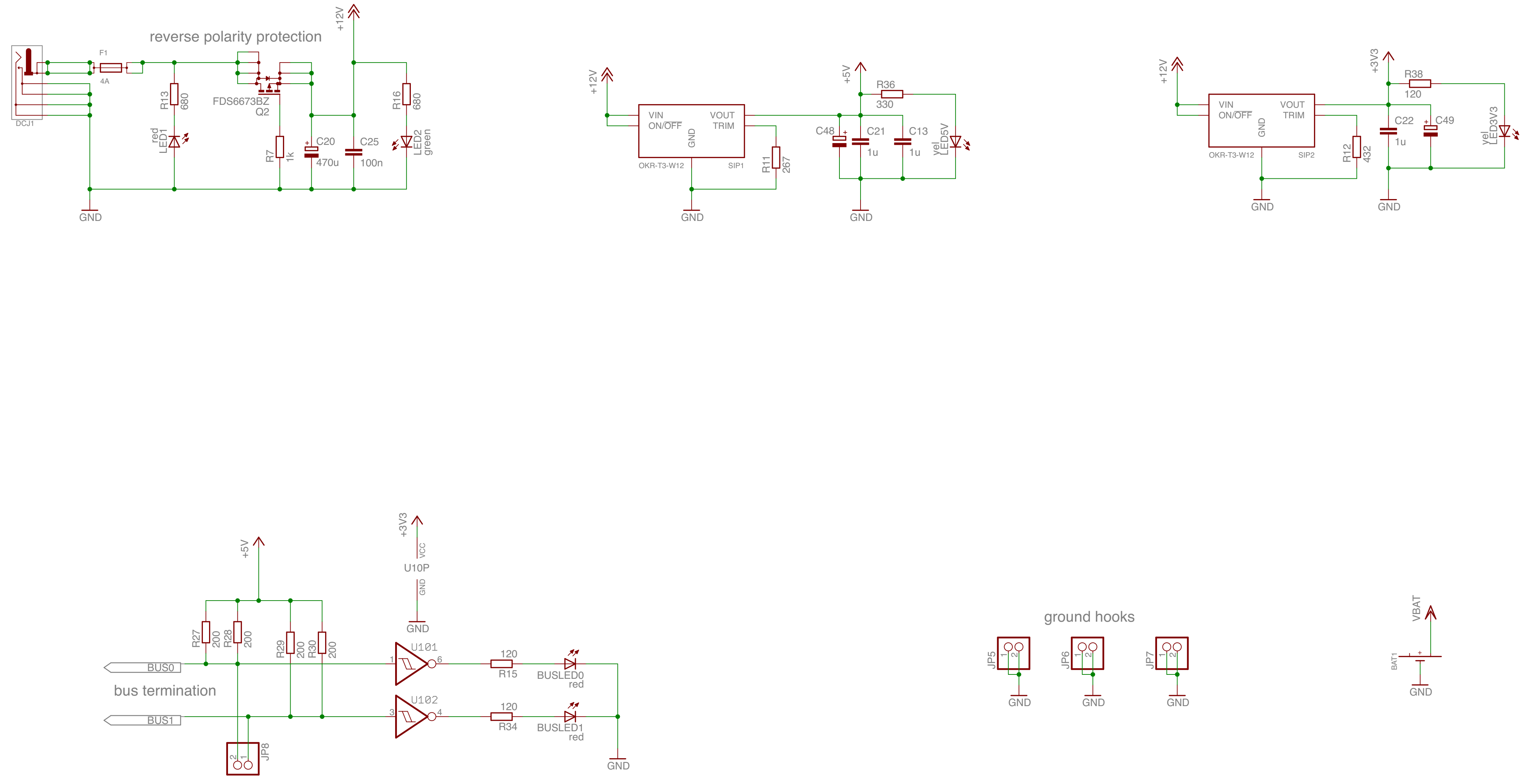


Programmer

mainboard-simple

nicht gespeichert!

Sheet: 6/8

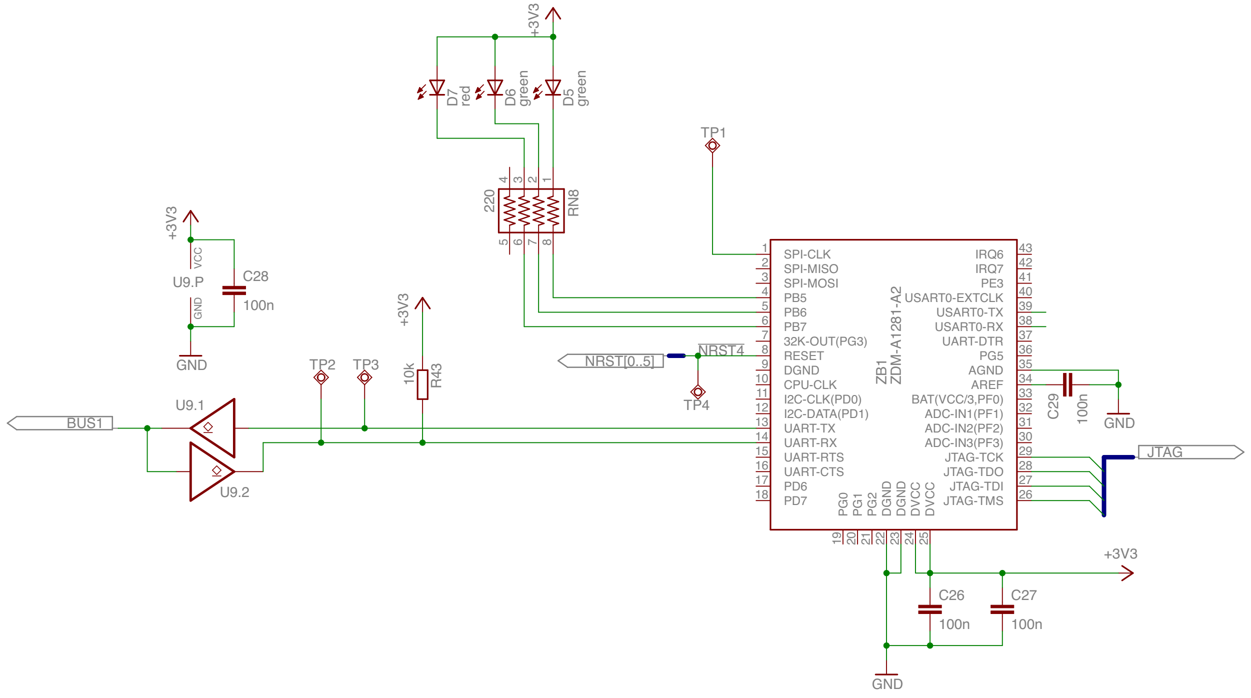


Power Supply
and Bus Taps

mainboard-simple

nicht gespeichert!

Sheet: 7/8

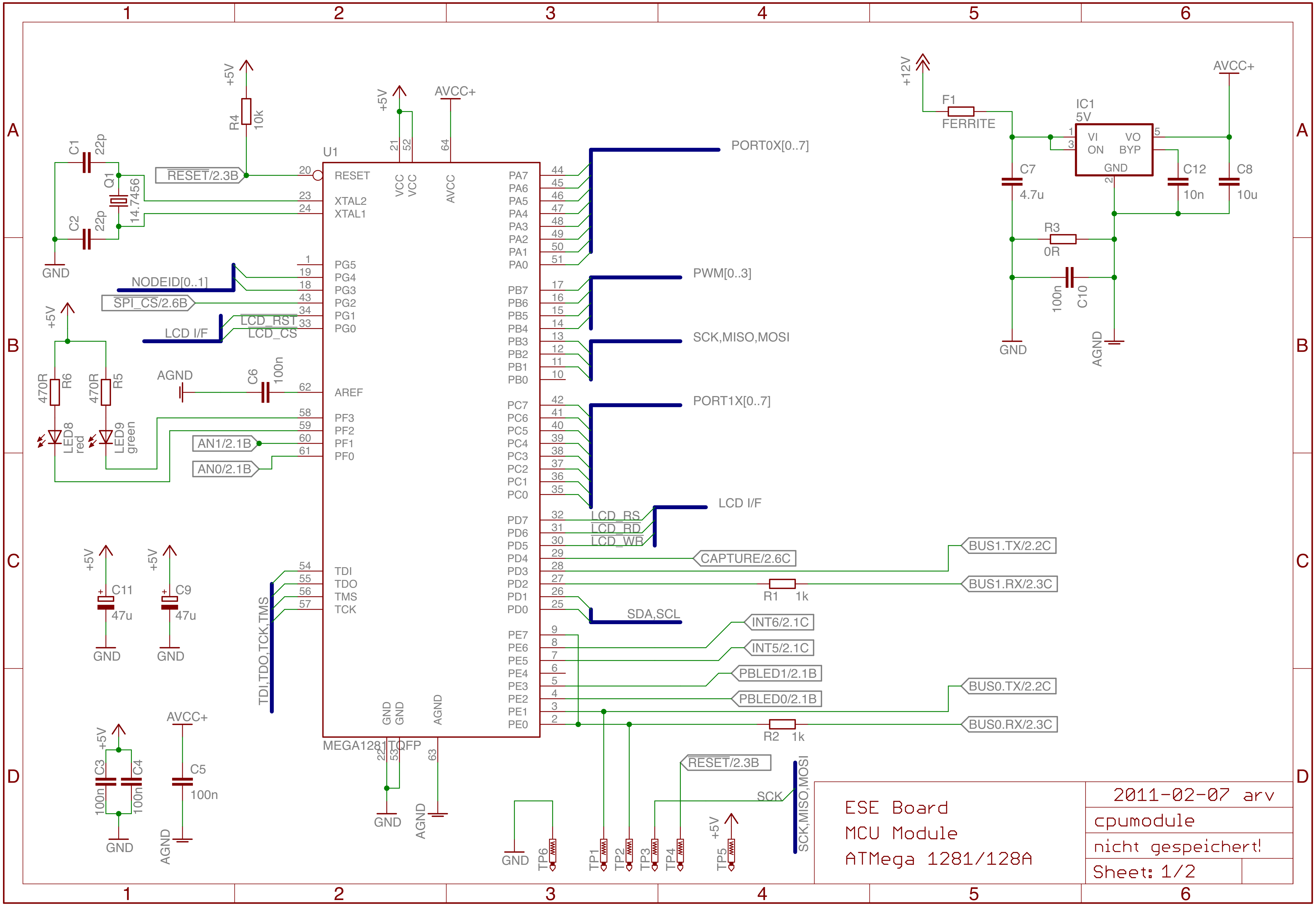


Zigbee I/F

mainboard-simple

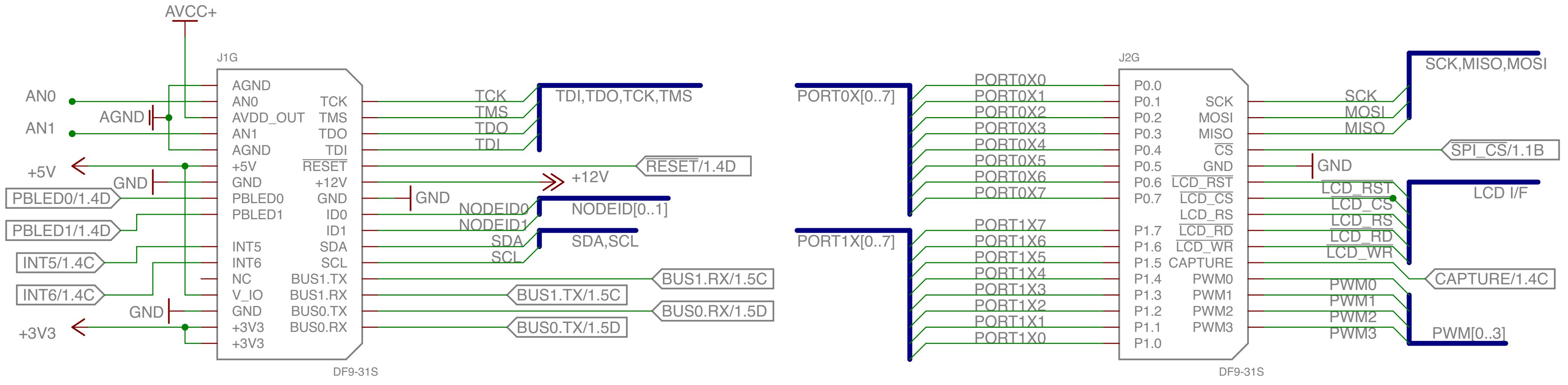
nicht gespeichert!

Sheet: 8/8



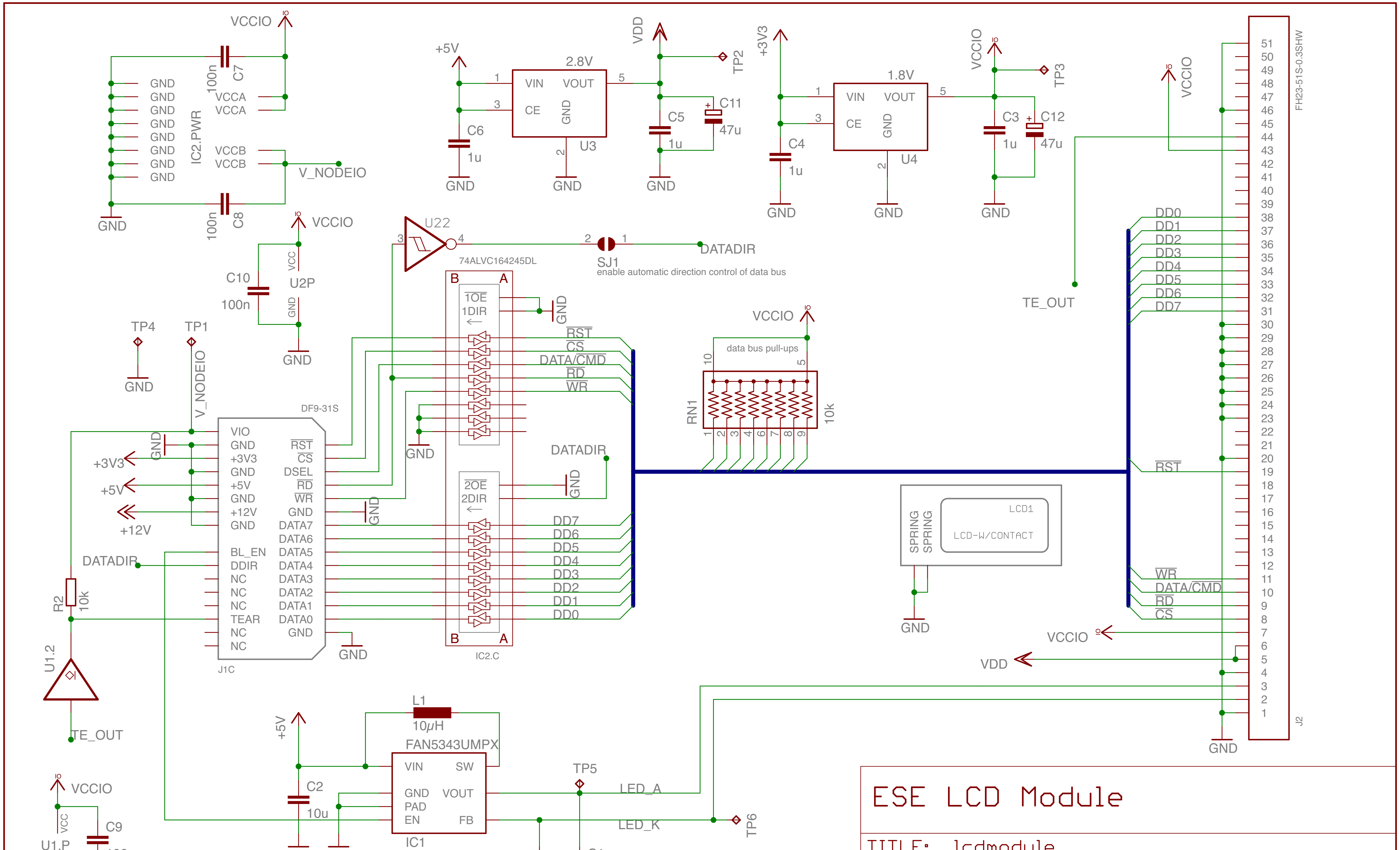
ESE Board
MCU Module
ATMega 1281/128A

2011-02-07 arv
cpumodule
nicht gespeichert!
Sheet: 1/2



ESE Board
MCU Module
Mainboard Connectors

cpumodule
nicht gespeichert!
Sheet: 2/2



ESE LCD Module

TITLE: lcdmodule

Document Number:

REV: 1.0

Date: 10.05.11 17:59

Sheet: 1/1

A.4 Listings

Node 1 Test Program

```
/* ==========================================================================
 *
 *  Networked Embedded Systems
 *  Node 1 Test Program
 *
 * Target:  ATMega128
 * Version: 0.1
 * Author:  Arvid Staub <arvid@innoc.at>
 *
 *
 *   This program is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * ========================================================================== */

#include <avr/io.h>
#include <stdio.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/power.h>

#define F_CPU 14745600UL

#include <util/delay.h>

#include "TWI_Slave.h"
#include "atomic.h"

#define LED_PORT        PORTF
#define LED_DDR         DDRF

#define LED_GREEN       PF3
#define LED_RED         PF2

#define led_off(led) do{ LED_PORT |= _BV(led); }while(0)
#define led_on(led) do{ LED_PORT &=  ~_BV(led); }while(0)

#define BTN_PORT        PORTE
#define BTN_DDR         DDRE
#define BTN_PIN         PINE
#define BTN1            PE5
#define BTN2            PE6
```

```
#define button_down(btn) (0==(BTN_PIN & _BV(btn)))


#define PWM_PORT        PORTB
#define PWM_DDR         DDRB
#define PWM3            PB7
#define PWM0            PB4

#define SE95_CONF       (3<<3)

uint8_t write_se95(uint8_t slave, uint8_t reg, uint8_t val)
{
        TWBR = 255;
        TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);


        while (!(TWCR & (1<<TWINT)));
        if ((TWSR & 0xF8) != TWI_START) return(1);

        TWDR = slave | 0;
        TWCR = (1<<TWINT) | (1<<TWEN);

        while (!(TWCR & (1<<TWINT))) ;
        if ((TWSR & 0xF8) != TWI_MTX_ADR_ACK) return(2);

        TWDR = reg;
        TWCR = (1<<TWINT) | (1<<TWEN);

        while (!(TWCR & (1<<TWINT))) ;
        if ((TWSR & 0xF8) != TWI_MTX_DATA_ACK) return(3);

        TWDR = val;
        TWCR = (1<<TWINT) | (1<<TWEN);

        while (!(TWCR & (1<<TWINT))) ;
        if ((TWSR & 0xF8) != TWI_MTX_DATA_ACK) return(4);

        TWDR = 0;
        TWCR = (1<<TWINT) | (1<<TWEN);

        while (!(TWCR & (1<<TWINT))) ;
        if ((TWSR & 0xF8) != TWI_MTX_DATA_ACK) return(5);

        TWCR = (1<<TWINT)|(1<<TWEN)| (1<<TWSTO);

        while((TWCR & (1<<TWINT))) TWCR = (1<<TWINT) | (1<<TWEN);
        TWCR  = 0;

        return 0;
```

```
}


int main(void)
{
        uint8_t cnt;
        uint8_t ok;

        LED_PORT = 0;
        LED_DDR = _BV(LED_GREEN) | _BV(LED_RED);

        DDRA = 0xFF;
        BTN_PORT = _BV(BTN1) | _BV(BTN2);

        PWM_DDR = _BV(PWM3) | _BV(PWM0);
        PWM_PORT = _BV(PWM0);

        /* enable pull-ups on BUS1 and BUS0 control pins */
        PORTE |= _BV(PE0);
        PORTE |= _BV(PE1);

        PORTD |= _BV(PD2);
        PORTD |= _BV(PD3);


        led_off(LED_GREEN);
        led_on(LED_RED);

        PORTA = 0xFF;

        ok = 0;

        if(0 == write_se95(0x92, 0x03, 35) &&
           0 == write_se95(0x92, 0x04, 25) &&
           0 == write_se95(0x92, 0x01, SE95_CONF))  {
                ok |= _BV(0);
        }

        if(0 == write_se95(0x90, 0x03, 35) &&
           0 == write_se95(0x90, 0x04, 25) &&
           0 == write_se95(0x90, 0x01, SE95_CONF)) {
                ok |= _BV(1);
        }

        if(0 == write_se95(0x94, 0x03, 35) &&
           0 == write_se95(0x94, 0x04, 25) &&
           0 == write_se95(0x94, 0x01, SE95_CONF)) {
                ok |= _BV(2);
        }
        if(ok == 7) {
                led_on(LED_GREEN);
                led_off(LED_RED);
        }

        for(;;) {


                if(button_down(BTN1)) {
                        PWM_PORT &= ~_BV(PWM0);
                } else {
                        PWM_PORT |= _BV(PWM0);
                }

                if(button_down(BTN2)) {
                        PWM_PORT |= _BV(PWM3);
                } else {
                        PWM_PORT &= ~_BV(PWM3);
                }

                cnt++;
                PORTA = ~cnt;

                _delay_ms(10);
        }

        return 0;
}
```

Node 3 Test Program

```
/* ===========================================================================
 *
 *  Networked Embedded Systems
 *  Node 3 Test Program
 *
 * Target:  ATMega128
 * Version: 0.1
 * Author:  Arvid Staub <arvid@innoc.at>
 *
 *
 *   This program is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * =========================================================================== */

#include <avr/io.h>
#include <stdio.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/power.h>

#define F_CPU 14745600UL

#include <util/delay.h>

#include "atomic.h"

#define LED_PORT         PORTF
#define LED_DDR          DDRF

#define LED_GREEN        PF3
#define LED_RED          PF2

#define led_off(led) do{ LED_PORT |= _BV(led); }while(0)
#define led_on(led) do{ LED_PORT &=  ~_BV(led); }while(0)

#define BTN_PORT         PORTE
#define BTN_DDR          DDRE
#define BTN_PIN          PINE
#define BTN1             PE5
#define BTN2             PE6
#define LED_BTN1         PE2
#define LED_BTN2         PE3

#define button_down(btn) (0==(BTN_PIN & _BV(btn)))
```

```
#define button_led(btn, state) do{ \
        if(state) BTN_PORT &= ~_BV(LED_##btn); \
        else BTN_PORT |= _BV(LED_##btn); }while(0)


#define PWM_PORT        PORTB
#define PWM_DDR         DDRB
#define PWM3            PB7
#define PWM0            PB4

#define ROW_PORT        PORTA
#define ROW_DDR         DDRA

#define DBG_PORT        PORTC
#define DBG_DDR         DDRC

#include "spimaster.h"

int main(void)
{
        uint8_t t;
        uint8_t cnt;
        uint8_t bitmask[4];

        LED_PORT = 0;
        LED_DDR = _BV(LED_GREEN) | _BV(LED_RED);

        ROW_DDR = 0xFF;
        BTN_PORT = _BV(BTN1) | _BV(BTN2);
        BTN_DDR = _BV(LED_BTN1) | _BV(LED_BTN2);

        PWM_DDR = _BV(PWM3) | _BV(PWM0);
        PWM_PORT = _BV(PWM0);

        DBG_DDR = 0xFF;
        DBG_PORT = 0;

        /* enable pull-ups on BUS1 and BUS0 control pins */
        PORTE |= _BV(PE0);
        PORTE |= _BV(PE1);

        PORTD |= _BV(PD2);
        PORTD |= _BV(PD3);


        led_off(LED_GREEN);
        led_on(LED_RED);
        spimaster_setup();

        spimaster_set_cs(0);
        spimaster_transfer_sync(4, "aaaa");

        spimaster_set_cs(0);
        _delay_us(1);
        spimaster_set_cs(1);
        _delay_us(1);
        spimaster_set_cs(0);

        ROW_PORT = 0;
        spimaster_set_cs(0);

        led_on(LED_GREEN);
        led_off(LED_RED);

        t = 0;
        for(;;) {


                if(!button_down(BTN1)) {
                        PWM_PORT &= ~_BV(PWM0);
                } else {
                        PWM_PORT |= _BV(PWM0);
                }

                if(button_down(BTN2)) {
                        PWM_PORT |= _BV(PWM3);

                        bitmask[0] = _BV(6-cnt);
                        bitmask[1] = _BV(6-cnt);

                } else {
                        PWM_PORT &= ~_BV(PWM3);

                        bitmask[0] = _BV(cnt);
                        bitmask[1] = _BV(cnt);
                }

                bitmask[2] = ~bitmask[0];
                bitmask[3] = ~bitmask[1];

                if(0 == (cnt & _BV(2))) {
                        button_led(BTN1, 0);
                        button_led(BTN2, 1);
                } else {
                        button_led(BTN2, 0);
                        button_led(BTN1, 1);
                }


                /* transfer bitmask */
                spimaster_set_cs(0);
                spimaster_transfer_sync(4, bitmask);

                /* latch data */
                _delay_us(1);
                spimaster_set_cs(1);
                _delay_us(1);
                spimaster_set_cs(0);

                /* select next row */
                ROW_PORT = cnt;

                _delay_ms(1);

                cnt++;
                if(cnt>6) {
                        cnt=0;
                }


                DBG_PORT++;
        }

        return 0;
}
```

Bibliography

[1] Ieee standard test access port and boundary-scan architecture. *IEEE Std 1149.1-2001* (June 2001).

[2] *ATmega128A DC Characteristics and Speed Grades*. In [8], February 2011, chapters 27.2 and 27.3.

[3] *ATmega128A JTAG errata*. In [8], February 2011, chapters 26.9.5, 26.9.6 and 33.1.5.

[4] Iris - tinyos documentation. `http://docs.tinyos.net/tinywiki/index.php/IRIS`, May 2011.

[5] Pcb microstrip impedance calculator. `http://www.eeweb.com/toolbox/microstrip-impedance`, May 2011.

[6] Tinyos open source operating system for low-power wireless devices. `http://www.tinyos.net/`, May 2011.

[7] ATMEL CORPORATION. *ZigBit 2.4 GHz Wireless Modules*, June 2009.

[8] ATMEL CORPORATION. *ATmega128A Datasheet*, February 2011.

[9] CADSOFT COMPUTER GMBH. *The EAGLE Layout Editor*. `http://www.cadsoft.de/info.htm`, May 2011.

[10] DAVIS, L. Jtag bus description and interface. `http://www.interfacebus.com/Design_Connector_JTAG_Bus.html`, June 2011.

[11] FAIRCHILD SEMICONDUCTOR CORPORATION. *FDS6673BZ P-Channel PowerTrench® MOSFET*, March 2009.

[12] FAIRCHILD SEMICONDUCTOR CORPORATION. *FAN5343: 6-LED Series Boost LED Driver with Integrated Schottky Diode and Single-Wire Digital Interface*, August 2010.

[13] FAIRCHILD SEMICONDUCTOR CORPORATION. *FAN3100: Single 2A High-Speed, Low-Side Gate Driver*, January 2011.

[14] GALLER, J. A modular software package for the embedded systems engineering board. Bachelor's thesis, TU Wien, 2011.

[15] HALL, S. H., AND HECK, H. L. *Advanced Signal Integrity for High-Speed Digital Designs*. John Wiley & Sons, Inc., 2008, p. 477f.

[16] INTEL CORPORATION. *4-Wire Pulse Width Modulation (PWM) Controlled Fans*, July 2004.

[17] KÖSSLER, A. A platform for teaching and researching distributed real-time systems. Master's thesis, Technische Universität Wien, 2009.

[18] MURATA POWER SOLUTIONS. *Adjustable Output 3-Amp SIP-mount DC/DC Converters*, February 2010.

[19] NXP SEMICONDUCTORS. *SE95: Ultra high accuracy digital temperature sensor and thermal watchdog*, September 2009.

[20] NXP SEMICONDUCTORS. *74ALVC164245: 16-bit dual supply translating transceiver; 3-state*, April 2010.

[21] NXP SEMICONDUCTORS. *74LVC2G07: Buffers with open-drain outputs*, August 2010.

[22] NXP SEMICONDUCTORS. *74LVC2G17: Dual non-inverting Schmitt trigger with 5 V tolerant input*, August 2010.

[23] NXP SEMICONDUCTORS. *LPC1769/68/67/66/65/64/63 Product data sheet*, 6^{th} ed., August 2010.

[24] NXP SEMICONDUCTORS. *LPC17xx User manual*, 2^{nd} ed., August 2010.

[25] ON SEMICONDUCTOR. *CAT4016: 16-Channel Constant Current LED Driver*, April 2010.

[26] PÜRSCHEL, M. Reverse battery protection. Infineon Technologies AG, May 2005.

[27] RENESAS ELECTRONICS. *R61523 16,777,216–Color, 360x640-Dot Graphics LCD Controller Driver for α-Si TFT Panel*, December 2009.

[28] SCHMÖLZER, M. A modular, xml-based jtag programmer for embedded devices. Bachelor's thesis, TU Wien, November 2008.

[29] SEIKO INSTRUMENTS INC. *3.2" 360 x RGB x 640 TFT Development Specification*, March 2010.

[30] STMICROELECTRONICS. *M41T81S: Serial access real-time clock with alarms*, September 2010.

[31] WIRTZ, A. Freerouting: The interactive push and shove router. `http://www.freerouting.net/`, September 2010.