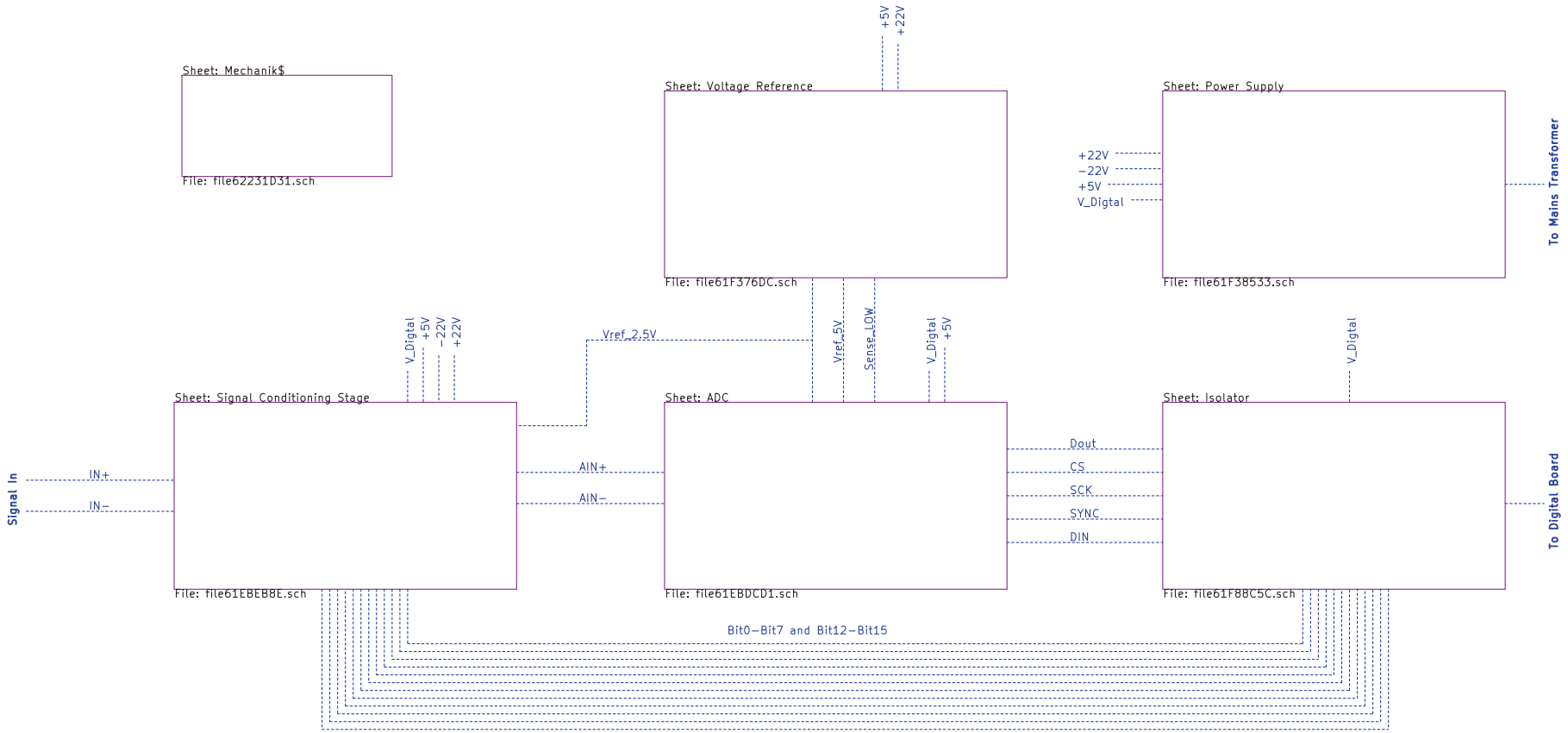
Appendix

Appendix A

ANALOG-BOARD REV. A

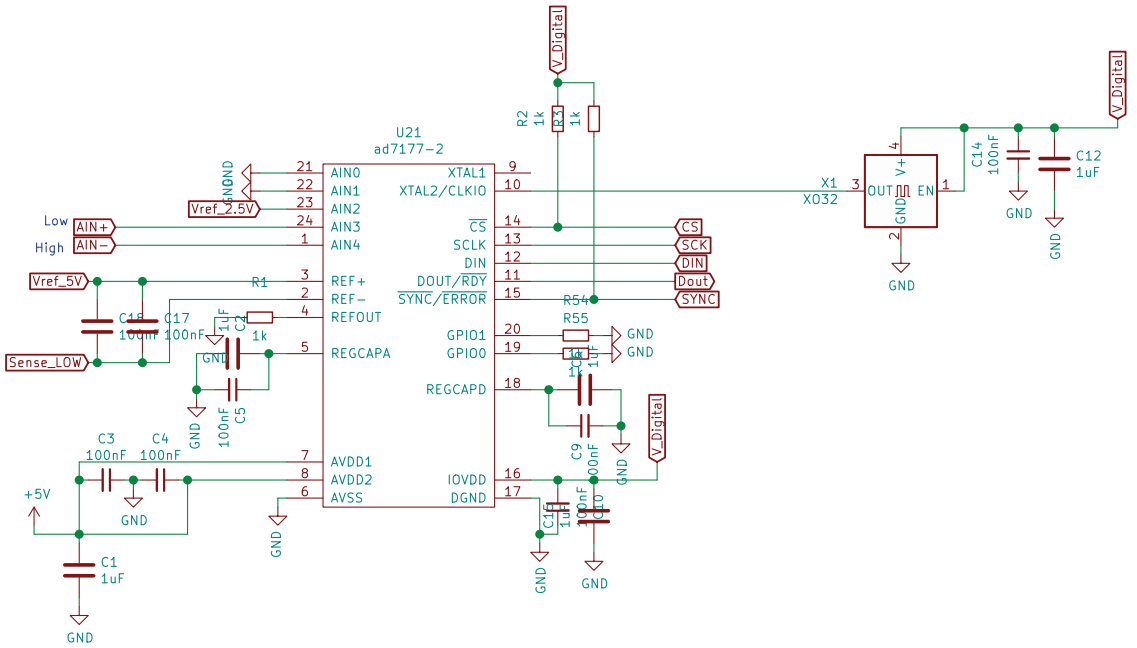


Simon Widmer

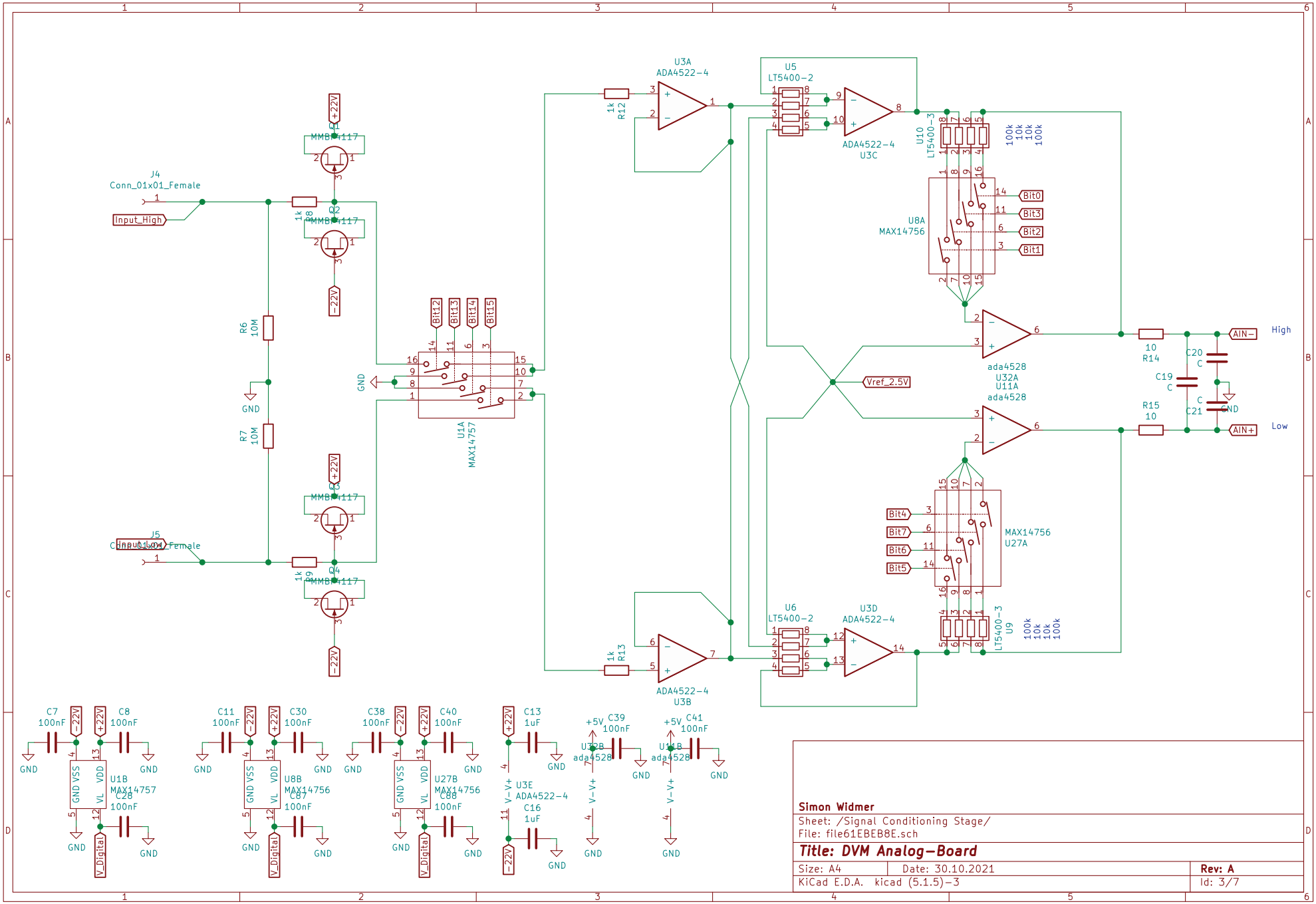
Sheet: /
File: DVM.sch

Title: DVM Analog−Board

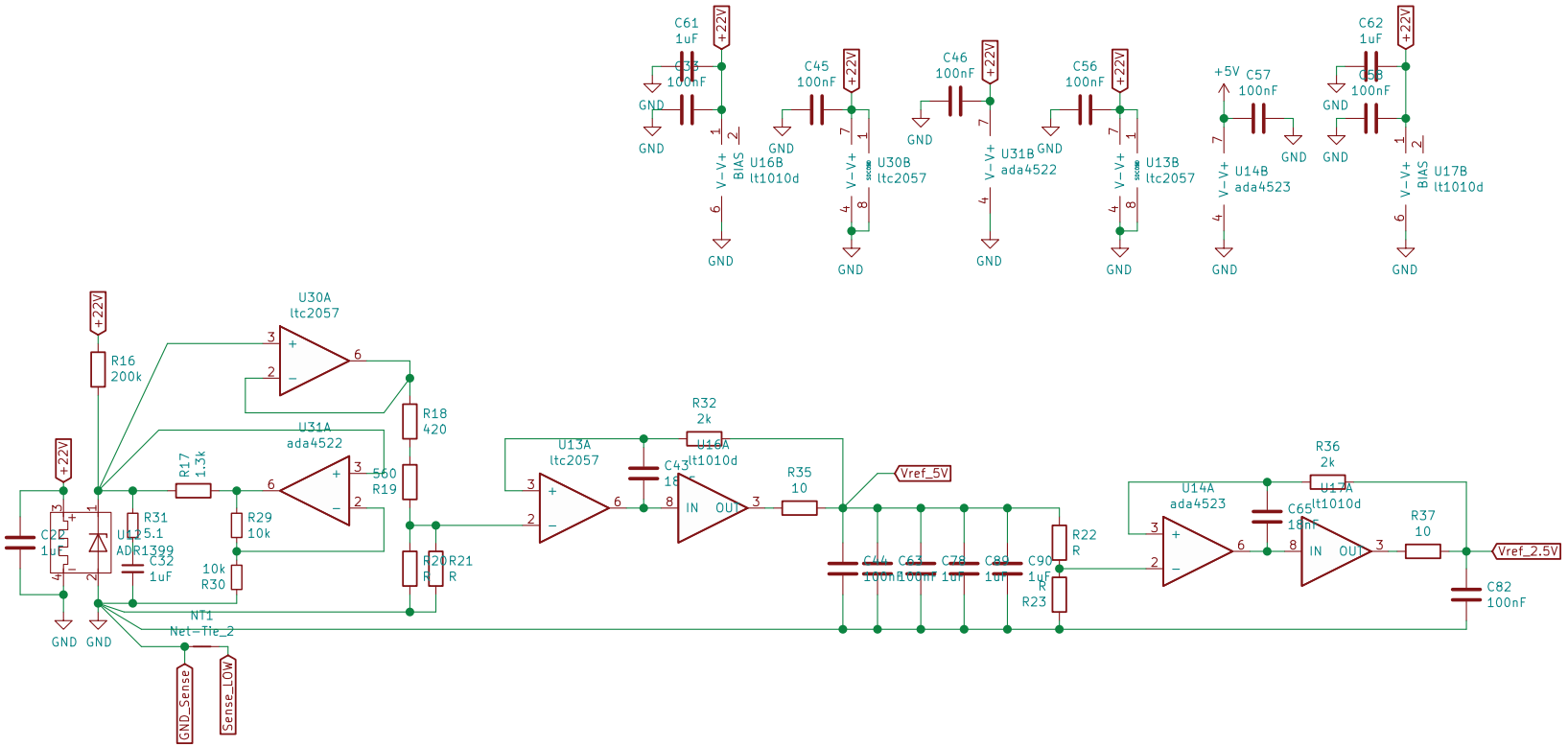
Size: A4	Date: 30.10.2021	**Rev: A**
KiCad E.D.A. kicad (5.1.5)−3		Id: 1/7



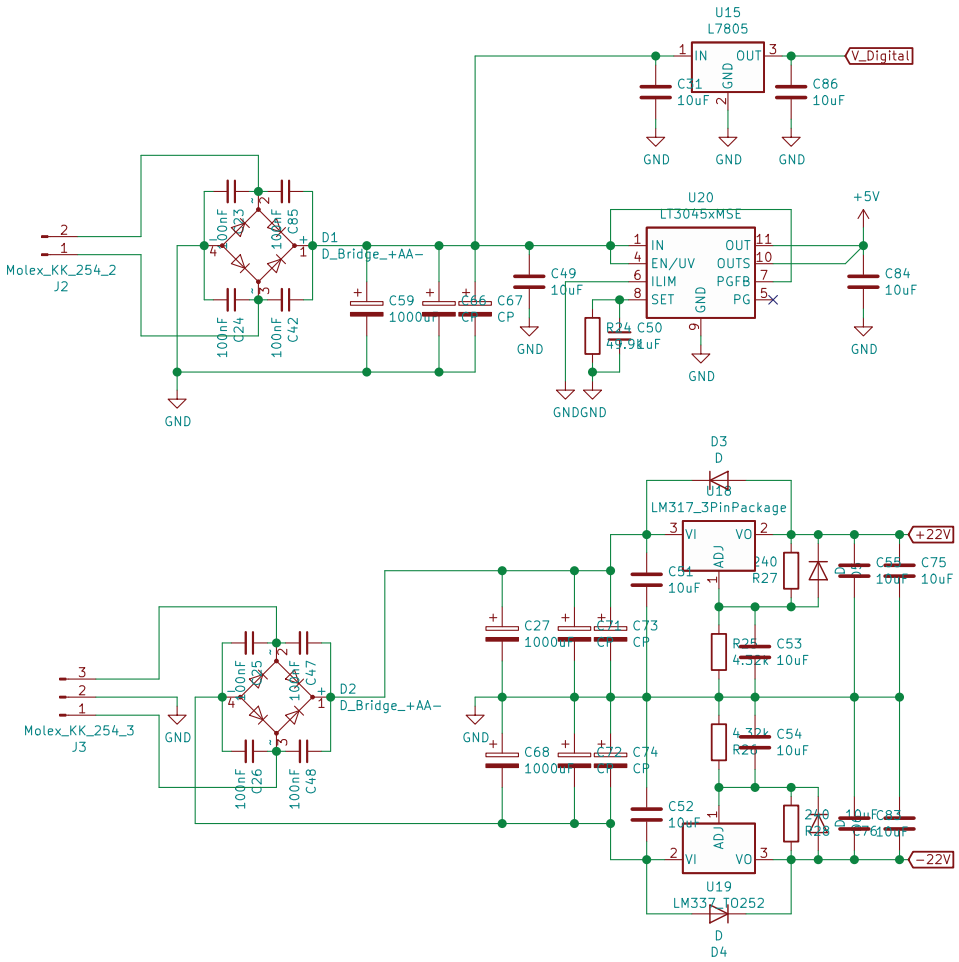
Simon Widmer		
Sheet: /ADC/		
File: file61EBDCD1.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: A**
KiCad E.D.A. kicad (5.1.5)-3		Id: 2/7



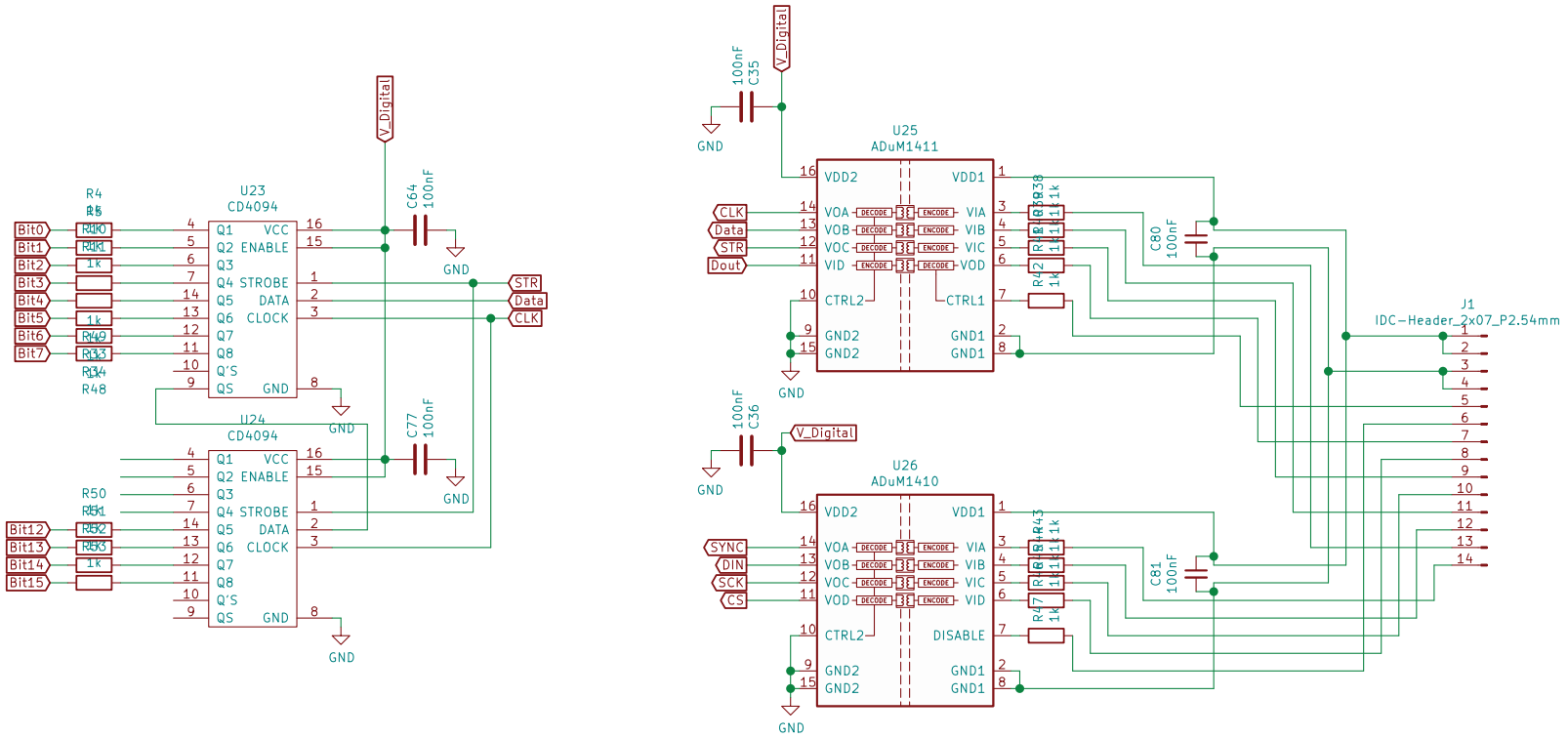
Simon Widmer	
Sheet: /Signal Conditioning Stage/	
File: file61EBEB8E.sch	
Title: DVM Analog-Board	
Size: A4	Date: 30.10.2021
KiCad E.D.A. kicad (5.1.5)-3	Id: 3/7
	Rev: A



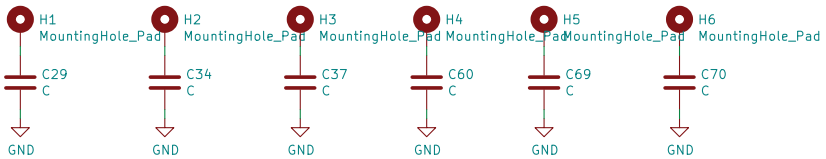
Simon Widmer		
Sheet: /Voltage Reference/		
File: file61F376DC.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: A**
KiCad E.D.A. kicad (5.1.5)-3		Id: 4/7



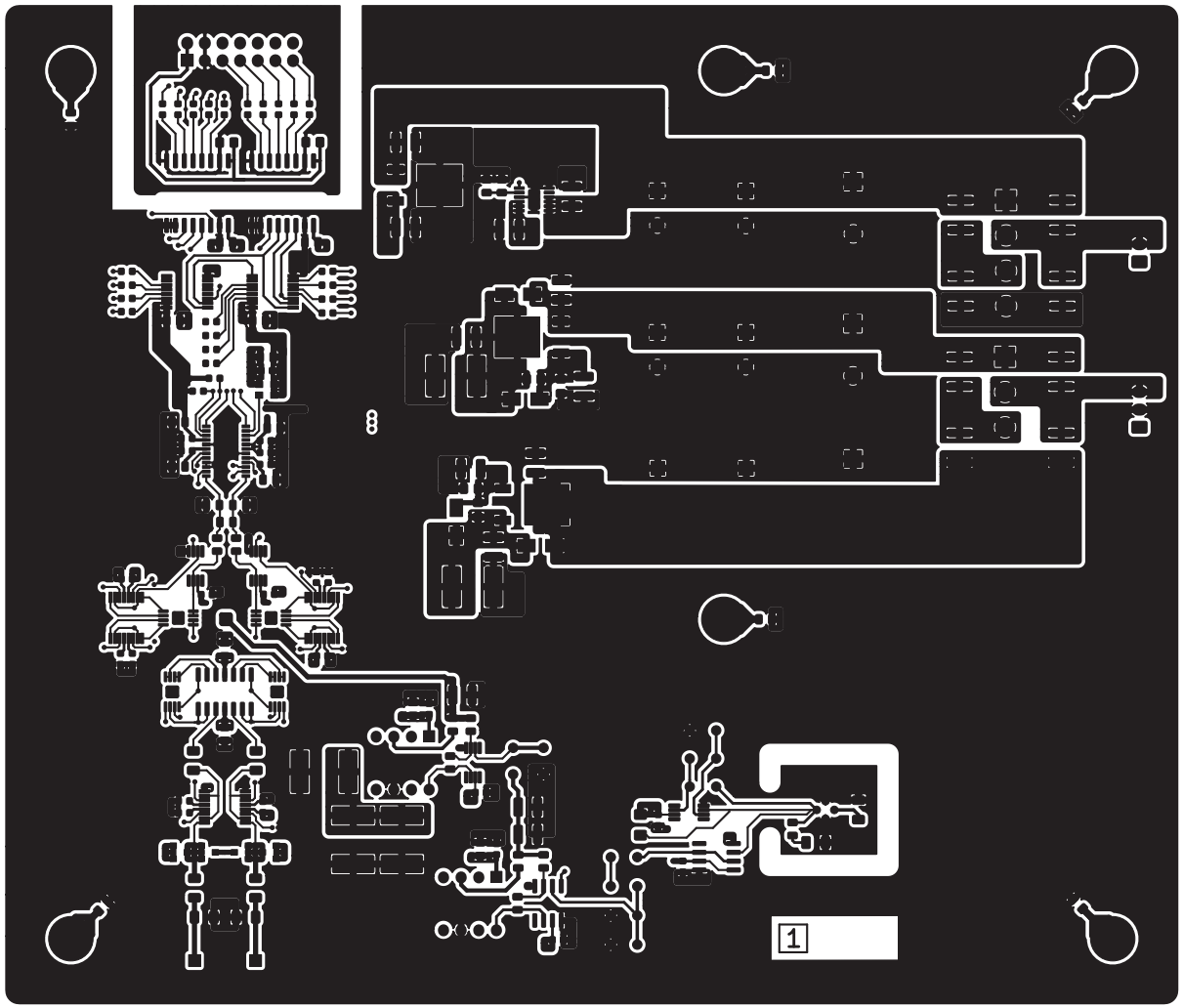
Simon Widmer		
Sheet: /Power Supply/		
File: file61F38533.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: A**
KiCad E.D.A. kicad (5.1.5)-3		Id: 5/7

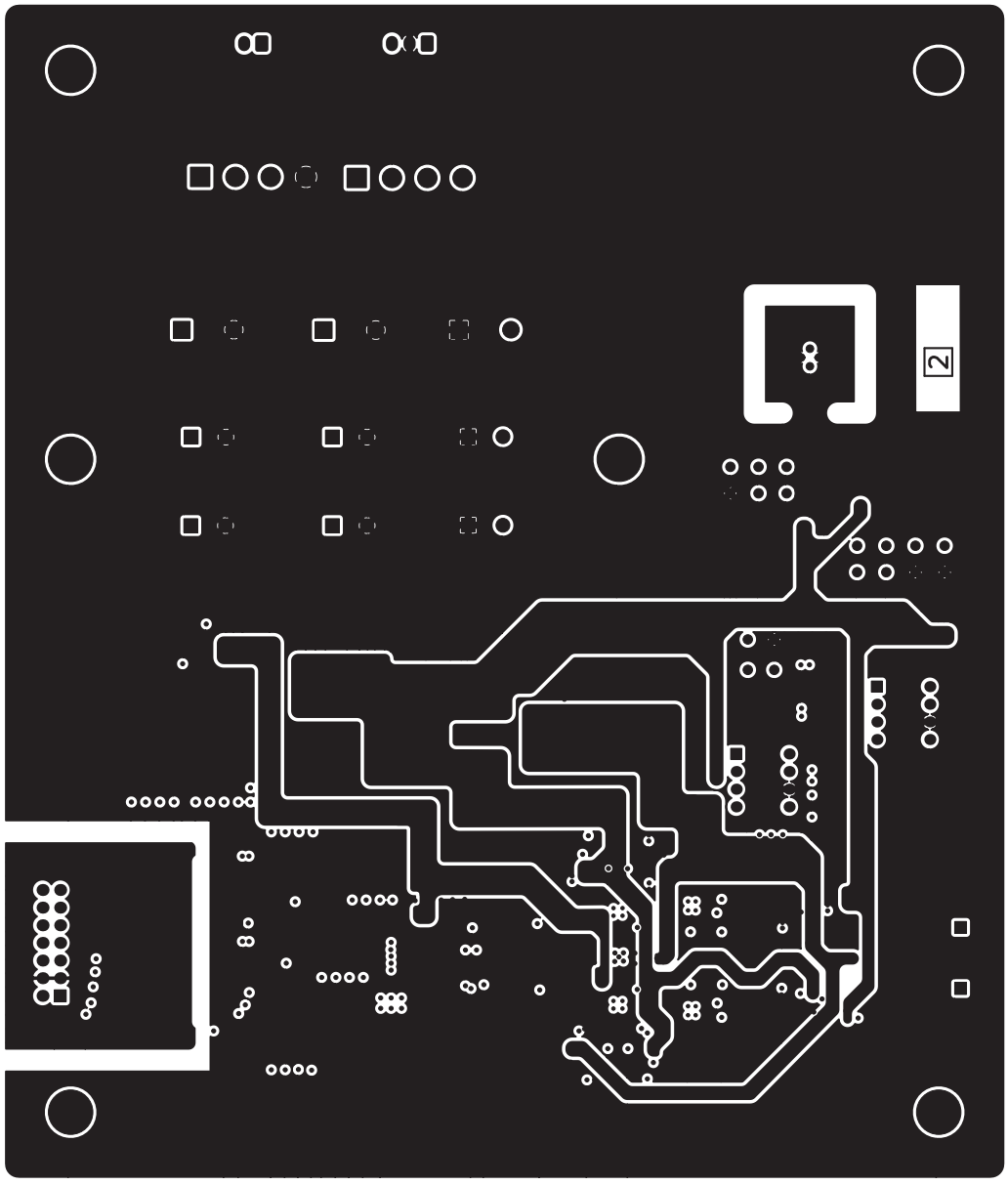


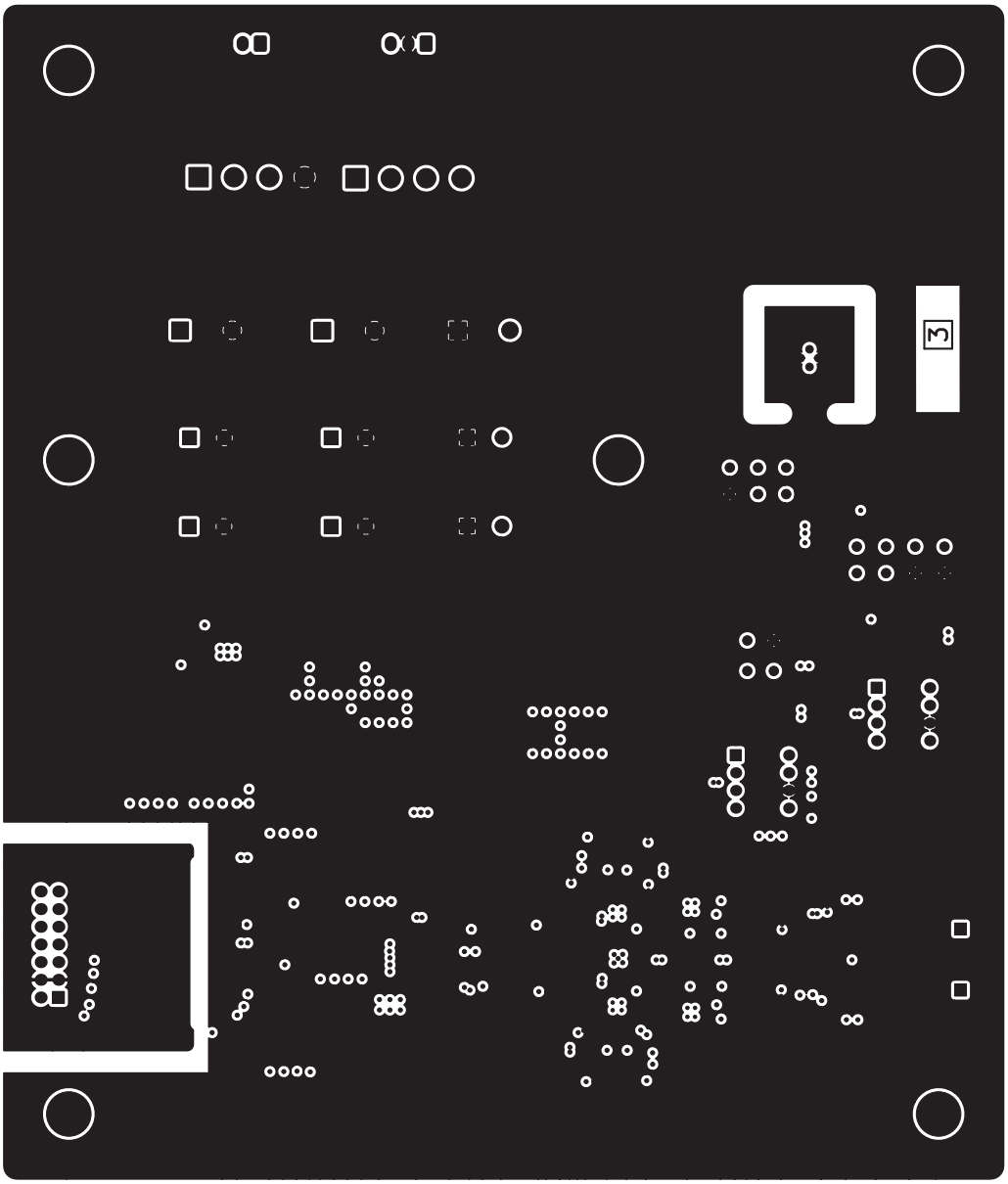
Simon Widmer		
Sheet: /Isolator/		
File: file61F88C5C.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: A**
KiCad E.D.A. kicad (5.1.5)-3		Id: 6/7

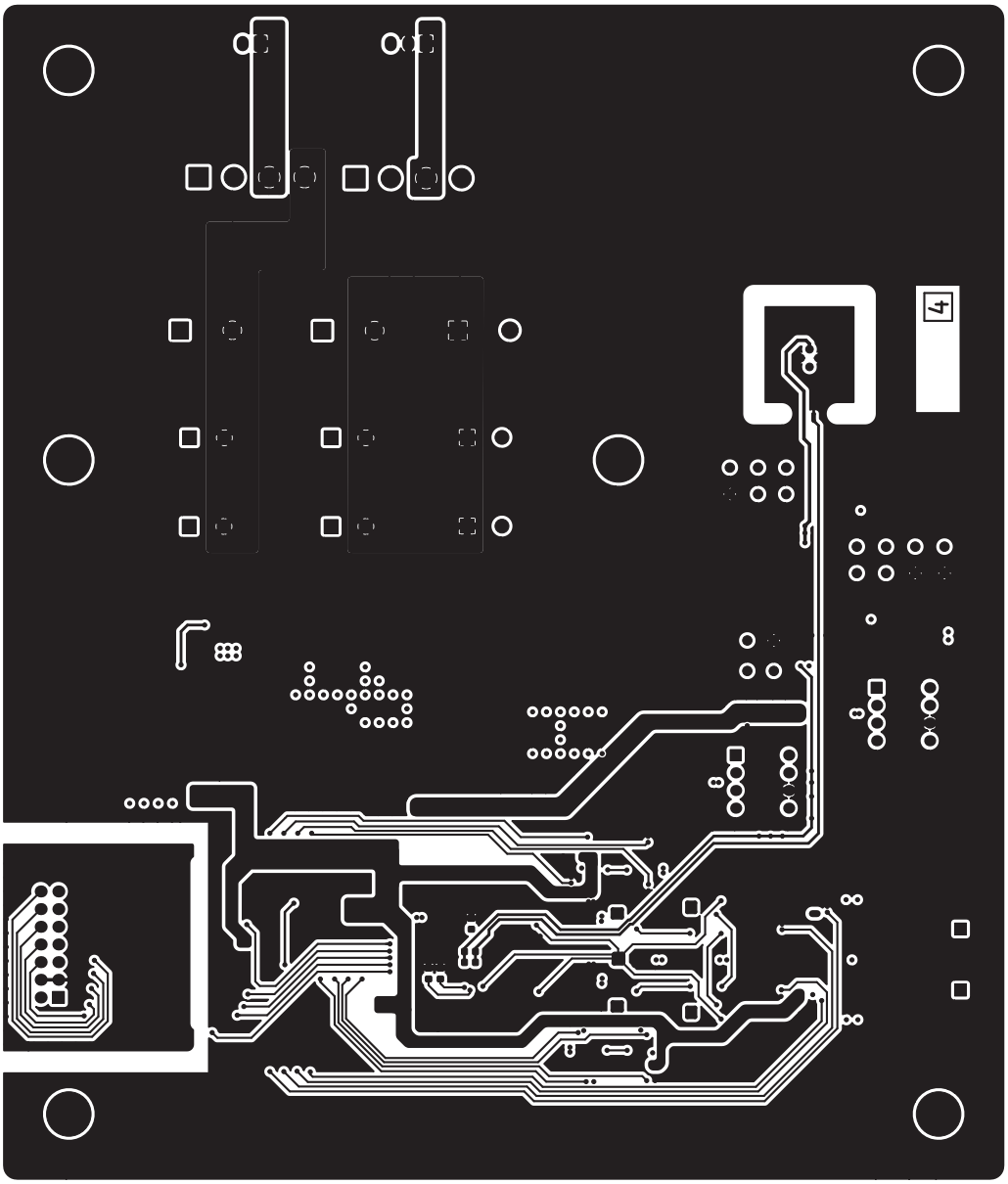


Simon Widmer		
Sheet: /Mechanik$/		
File: file62231D31.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: A**
KiCad E.D.A. kicad (5.1.5)-3		Id: 7/7


1


2


3


4

Analog PCB
Rev. A
Designed by Simon Widmer

NT1

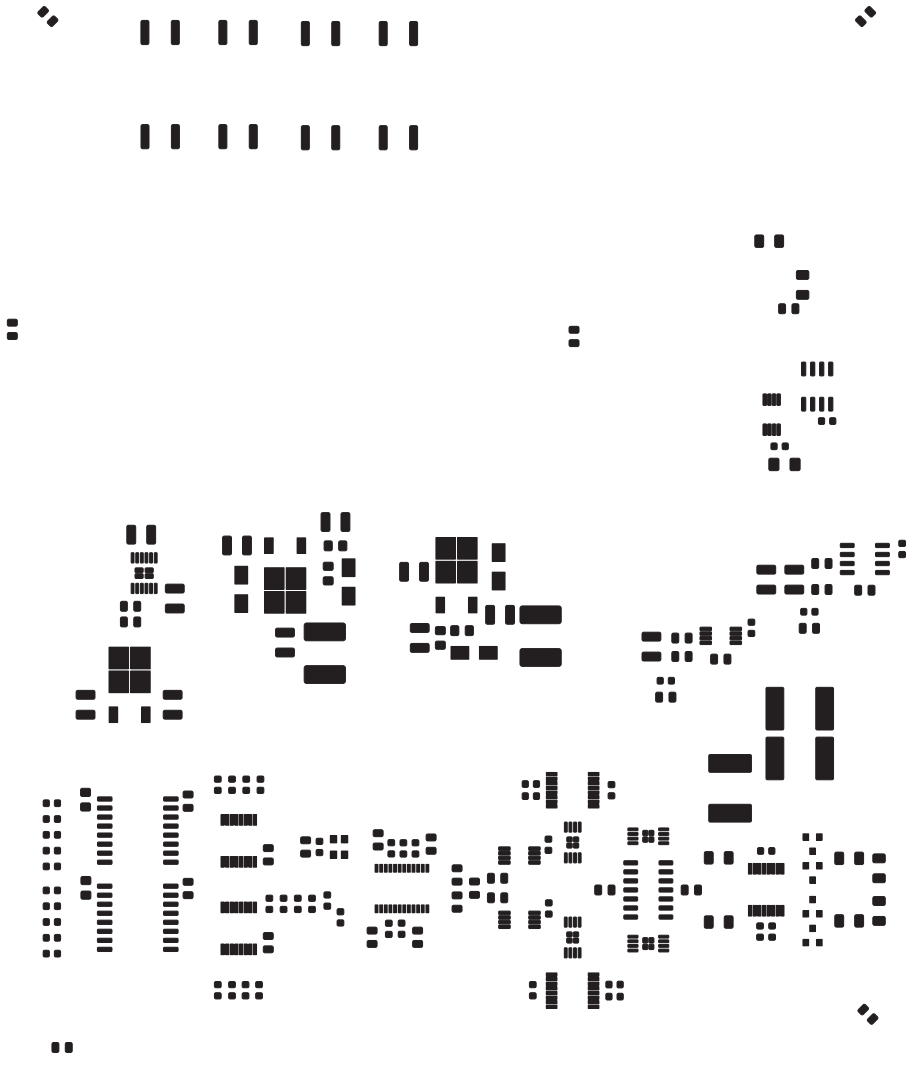
R1

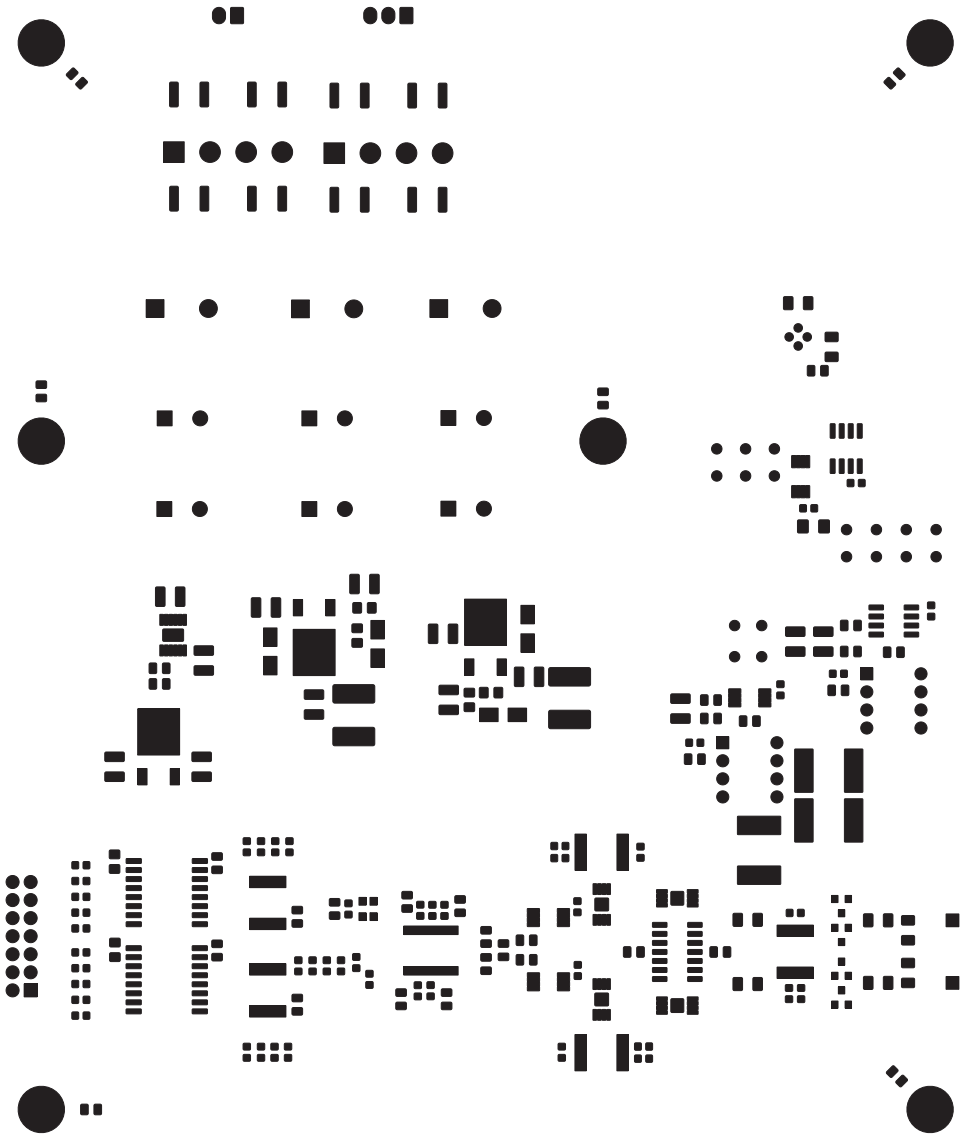
C18

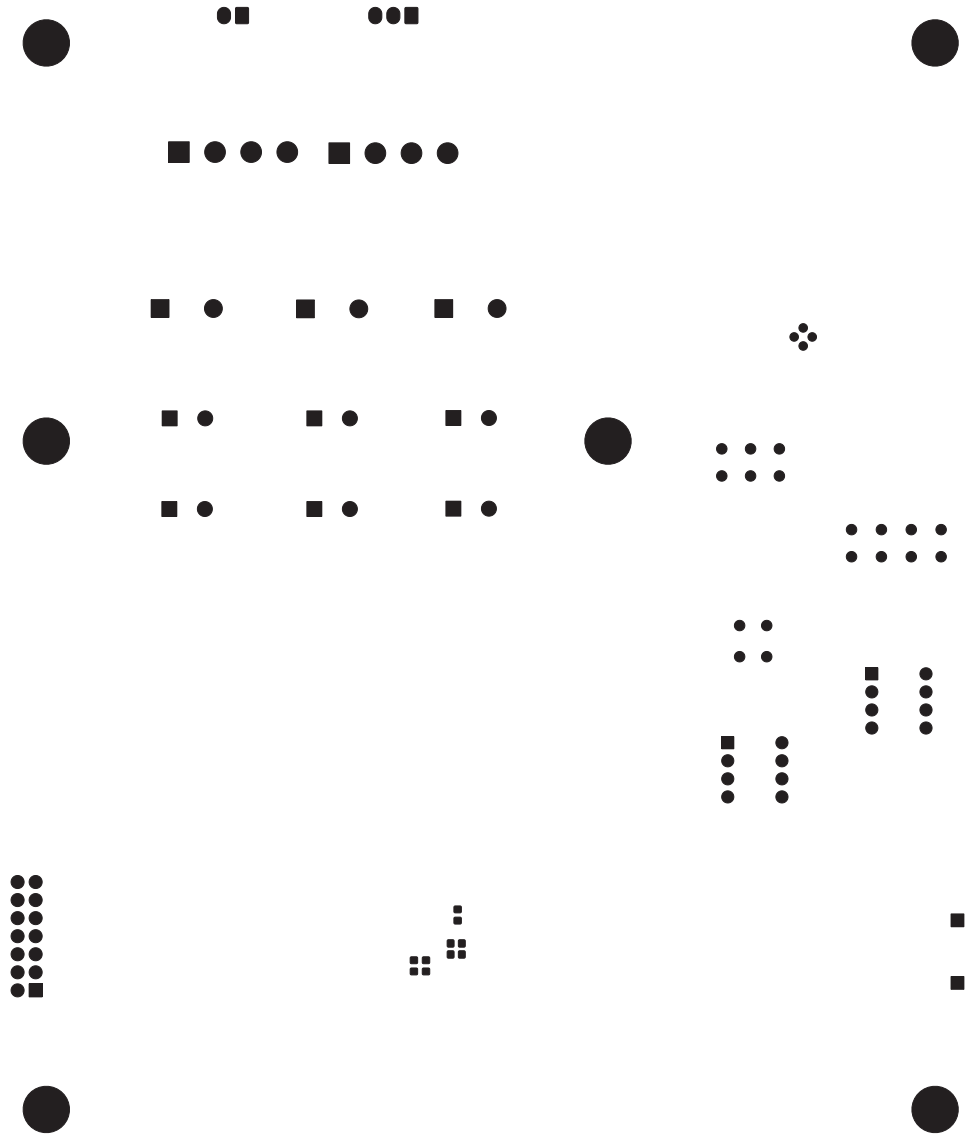
C17

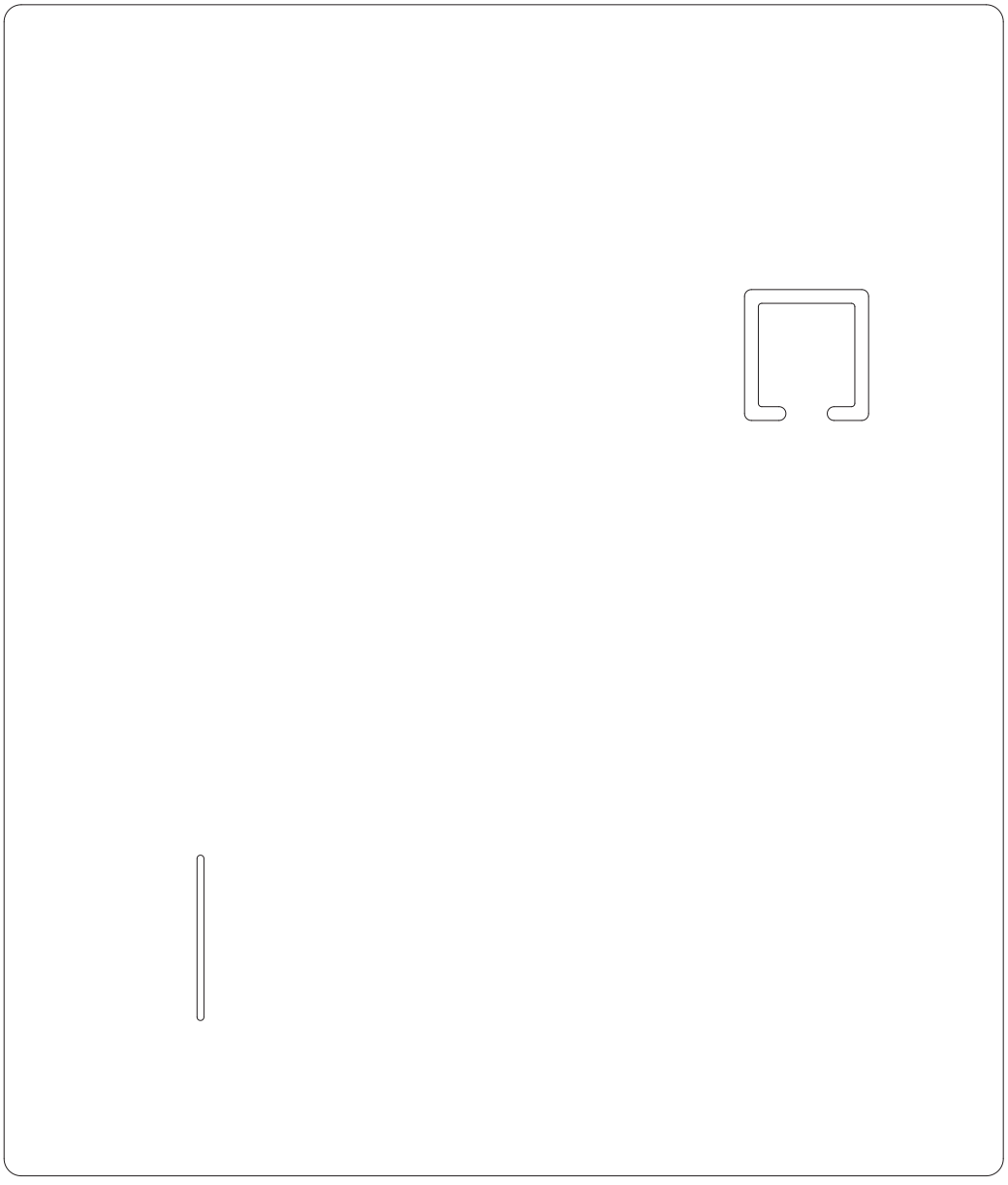
R54

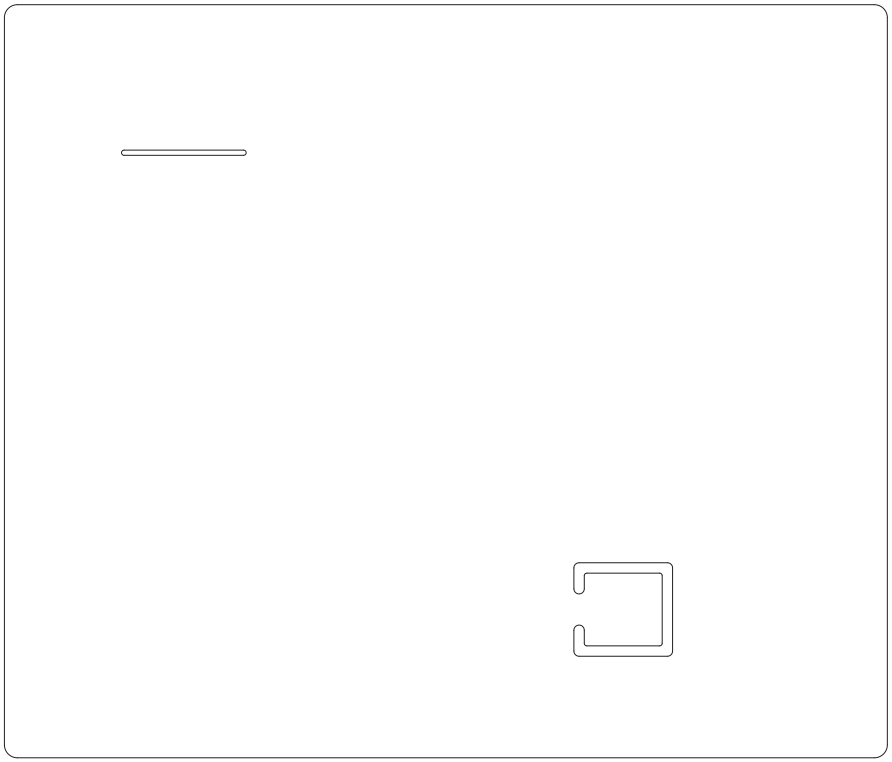
R55



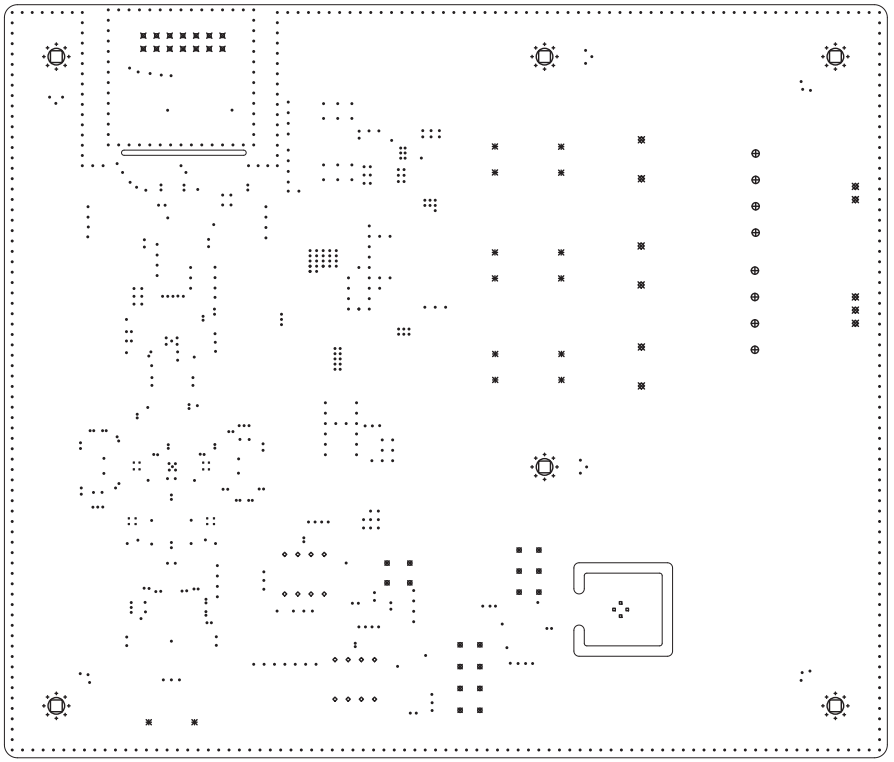








Drill Map:

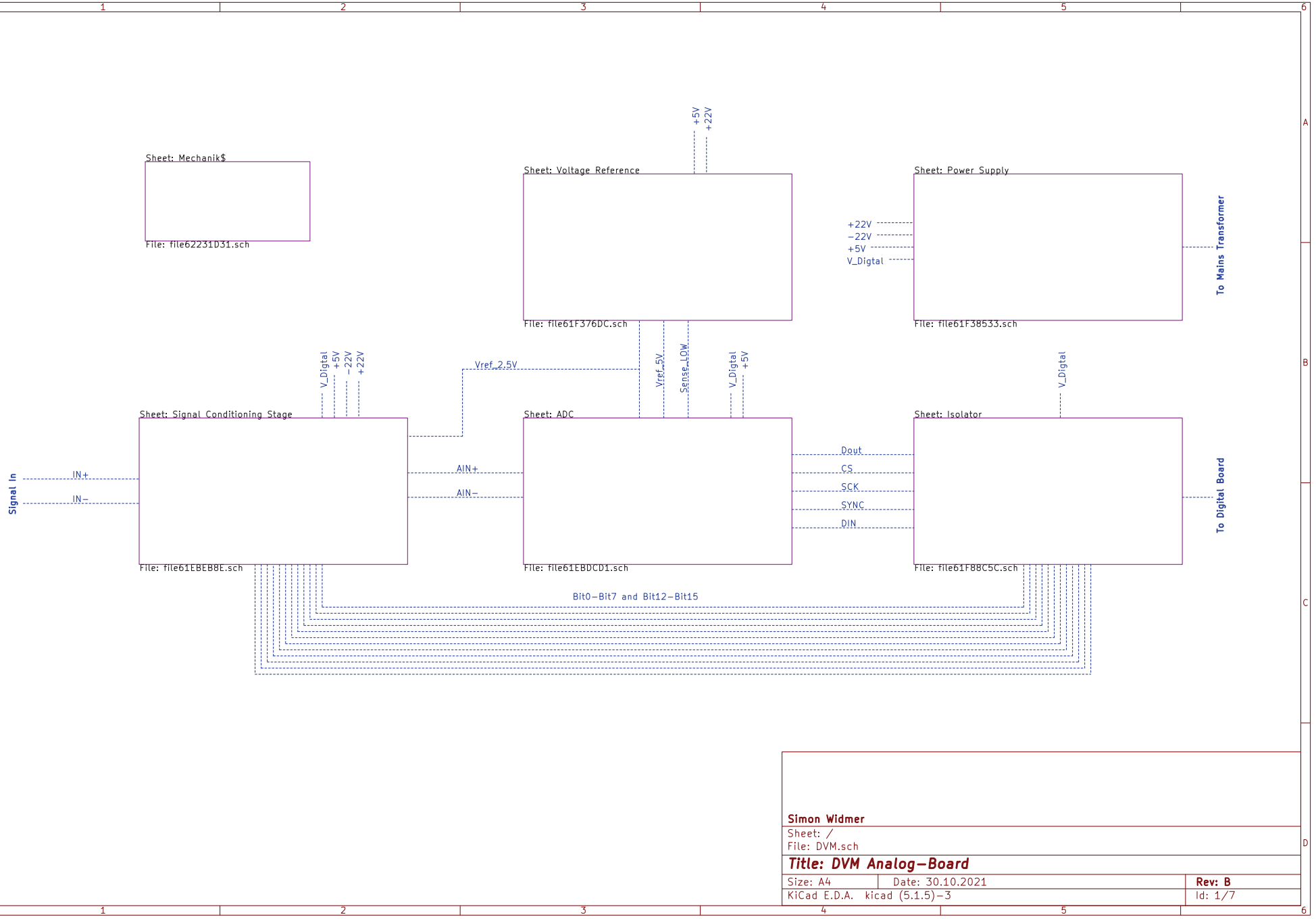


Drill Map:

- 0.20mm / 0.008" (16 holes)
- 0.30mm / 0.012" (812 holes)
- 0.50mm / 0.020" (48 holes)
- 0.68mm / 0.027" (4 holes)
- 0.80mm / 0.031" (16 holes)
- 0.84mm / 0.033" (18 holes)
- 1.00mm / 0.039" (14 holes)
- 1.02mm / 0.040" (14 holes)
- 1.20mm / 0.047" (11 holes)
- 1.40mm / 0.055" (8 holes)
- 3.20mm / 0.126" (6 holes)

Appendix B

ANALOG-BOARD REV. B

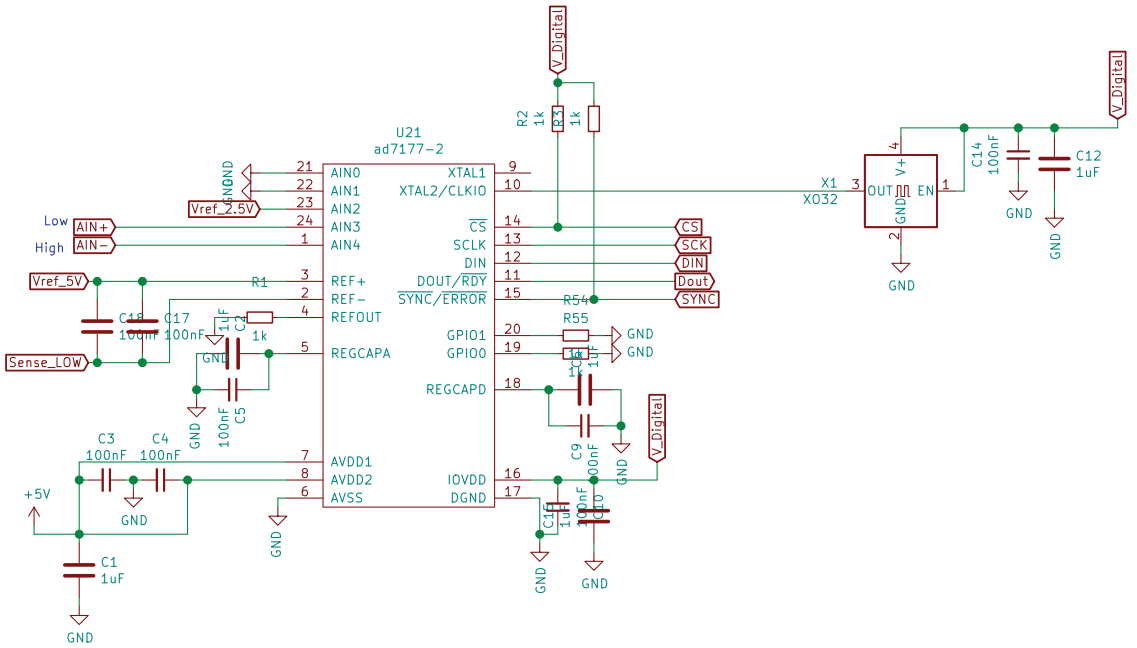


Simon Widmer

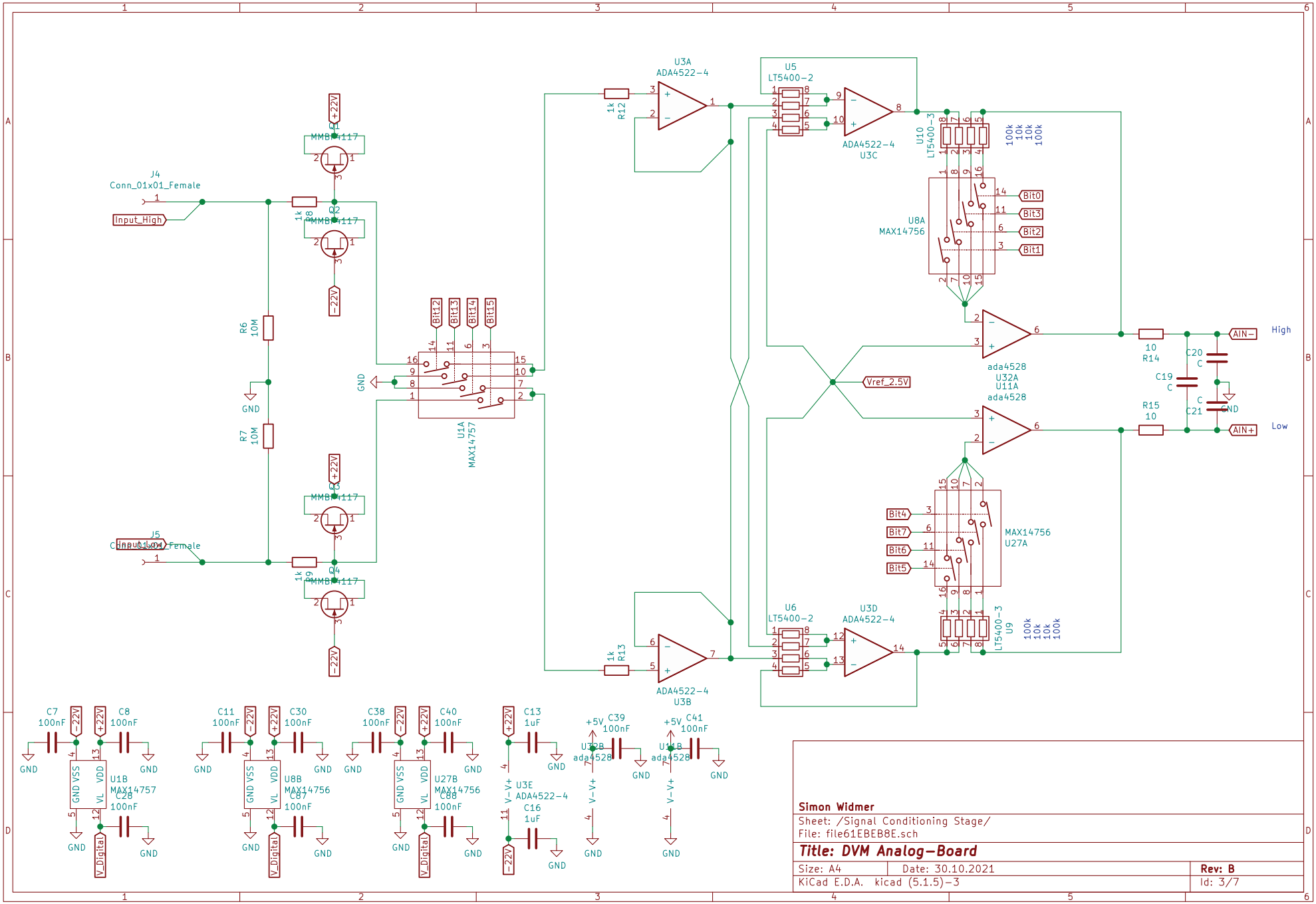
Sheet: /
File: DVM.sch

Title: DVM Analog−Board

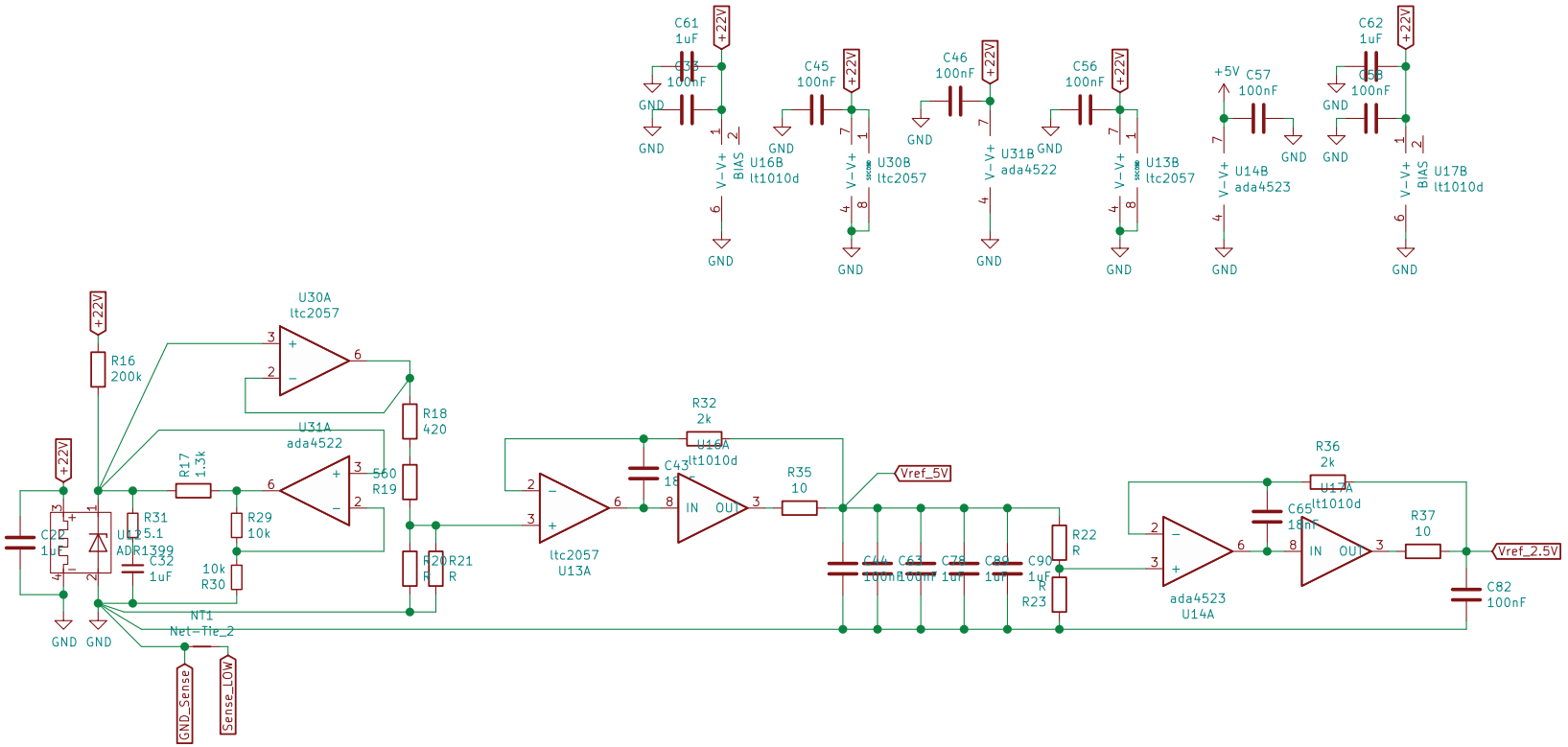
Size: A4	Date: 30.10.2021	Rev: B
KiCad E.D.A. kicad (5.1.5)−3		Id: 1/7



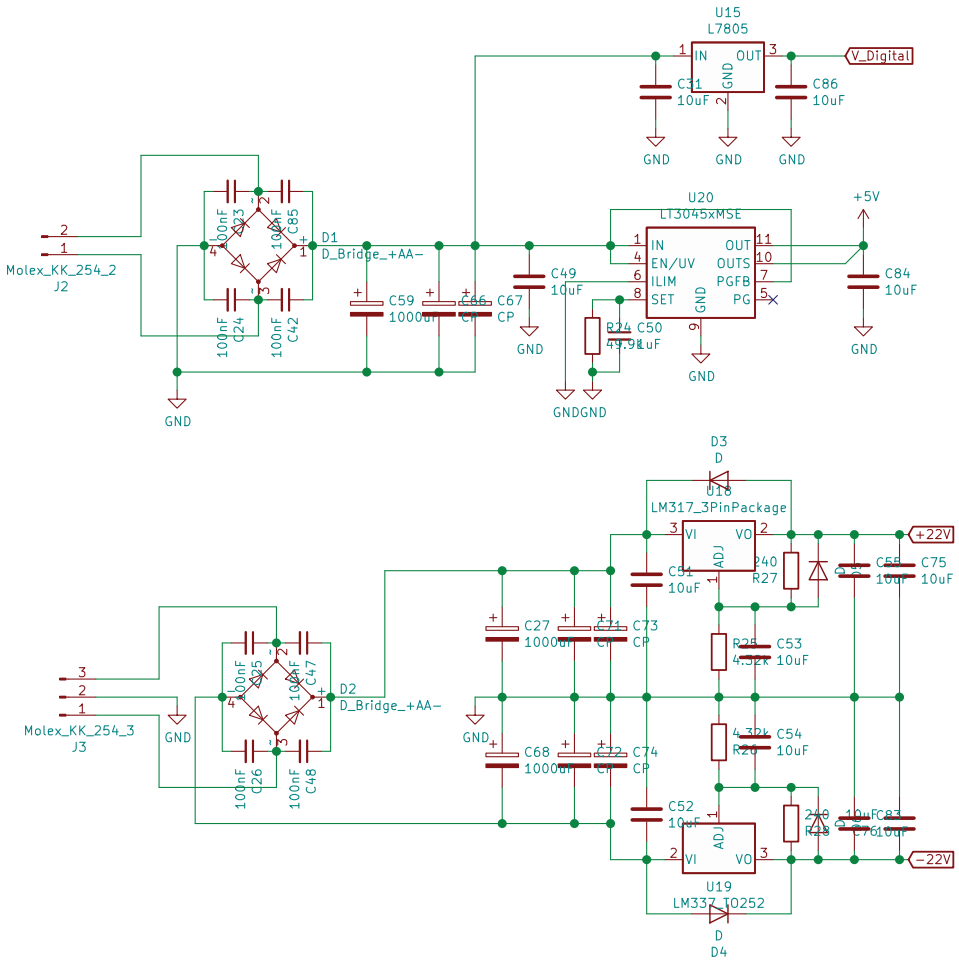
Simon Widmer		
Sheet: /ADC/		
File: file61EBDCD1.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: B**
KiCad E.D.A. kicad (5.1.5)-3		Id: 2/7



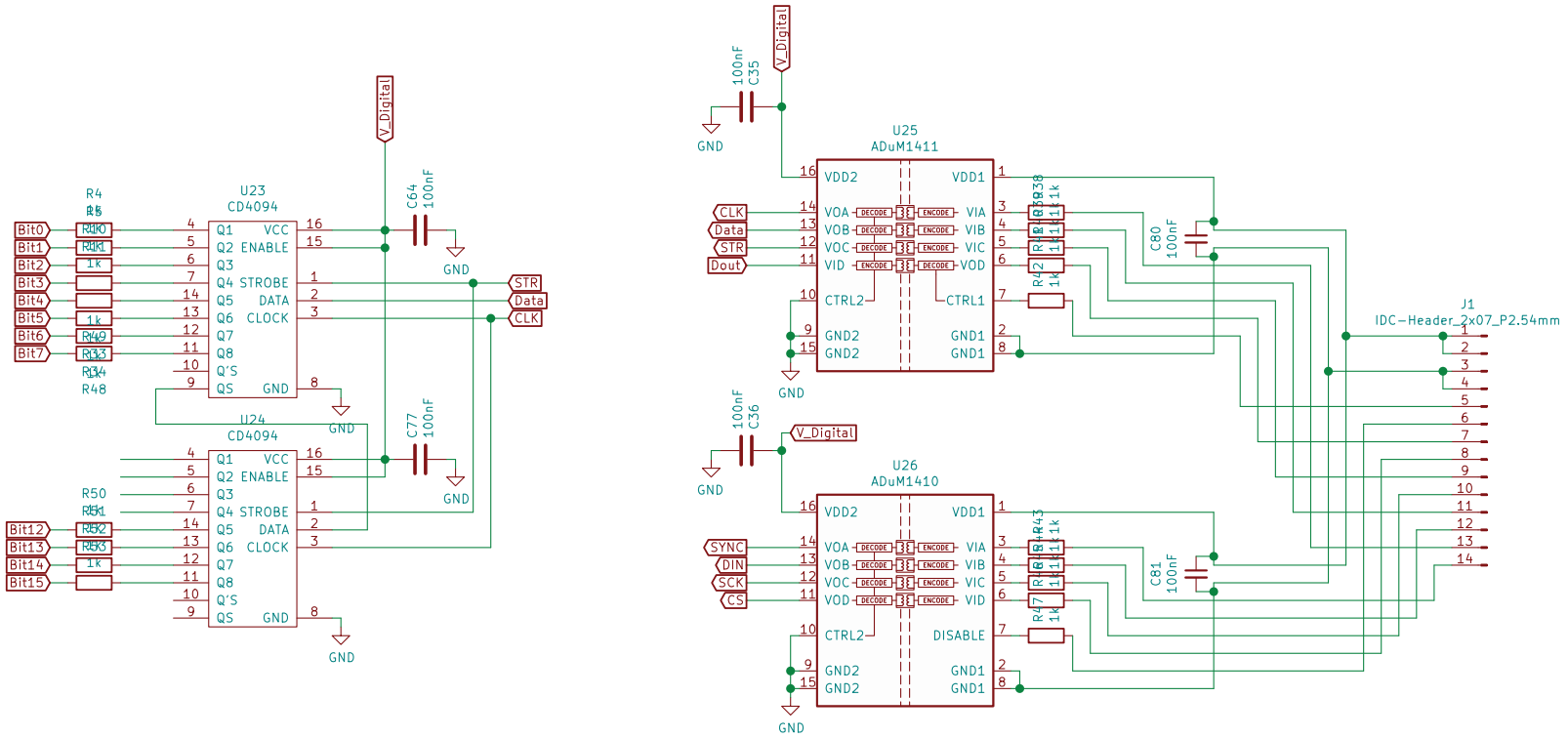
Simon Widmer	
Sheet: /Signal Conditioning Stage/	
File: file61EBEB8E.sch	
Title: DVM Analog-Board	
Size: A4	Date: 30.10.2021
KiCad E.D.A. kicad (5.1.5)-3	Id: 3/7
	Rev: B



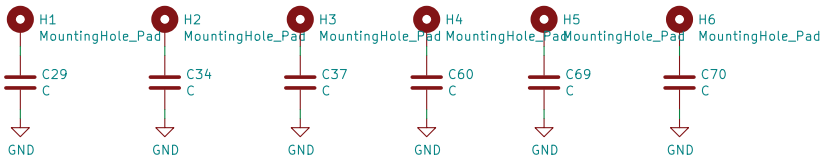
Simon Widmer		
Sheet: /Voltage Reference/		
File: file61F376DC.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: B**
KiCad E.D.A. kicad (5.1.5)-3		Id: 4/7



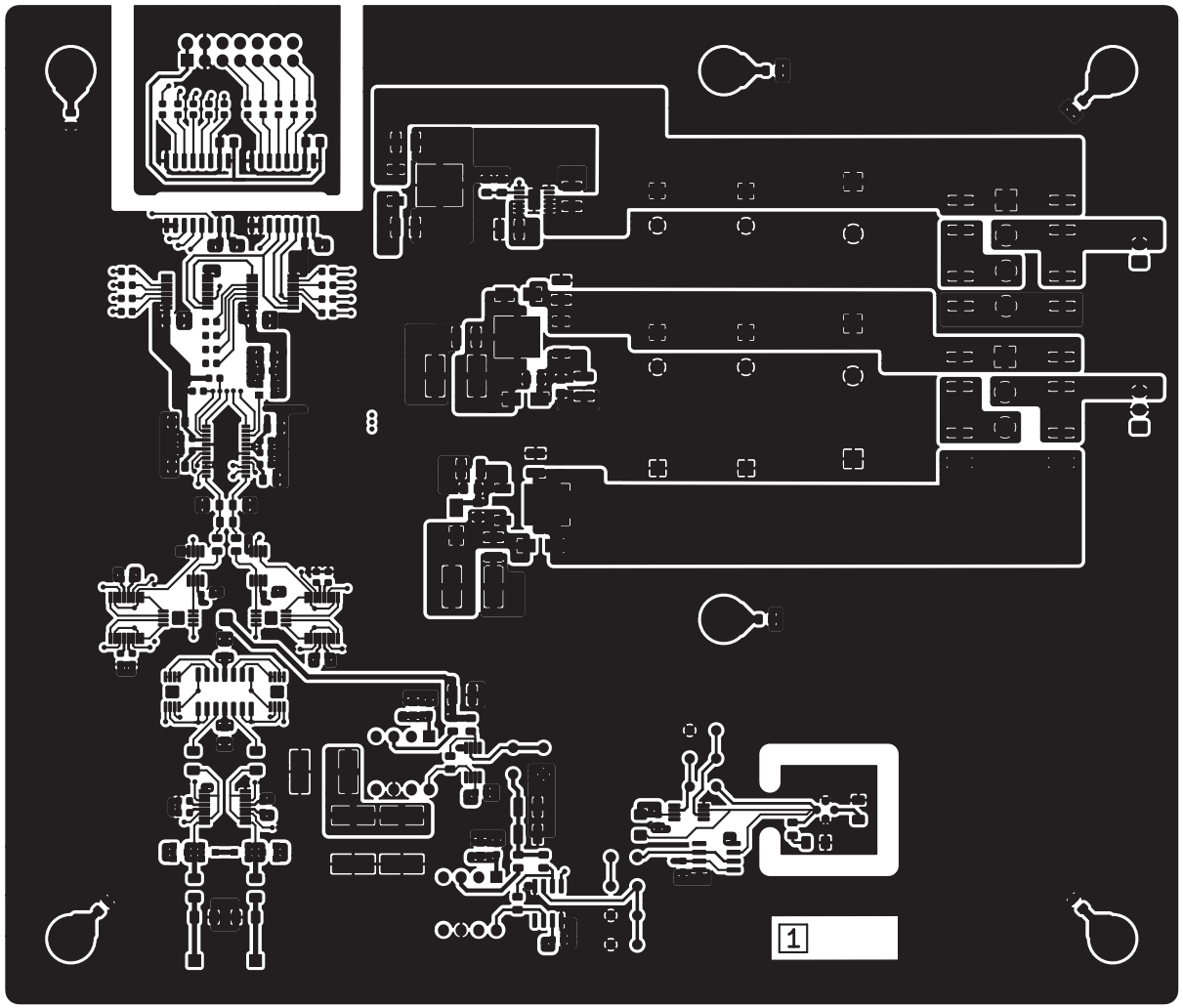
Simon Widmer		
Sheet: /Power Supply/		
File: file61F38533.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: B**
KiCad E.D.A. kicad (5.1.5)-3		Id: 5/7

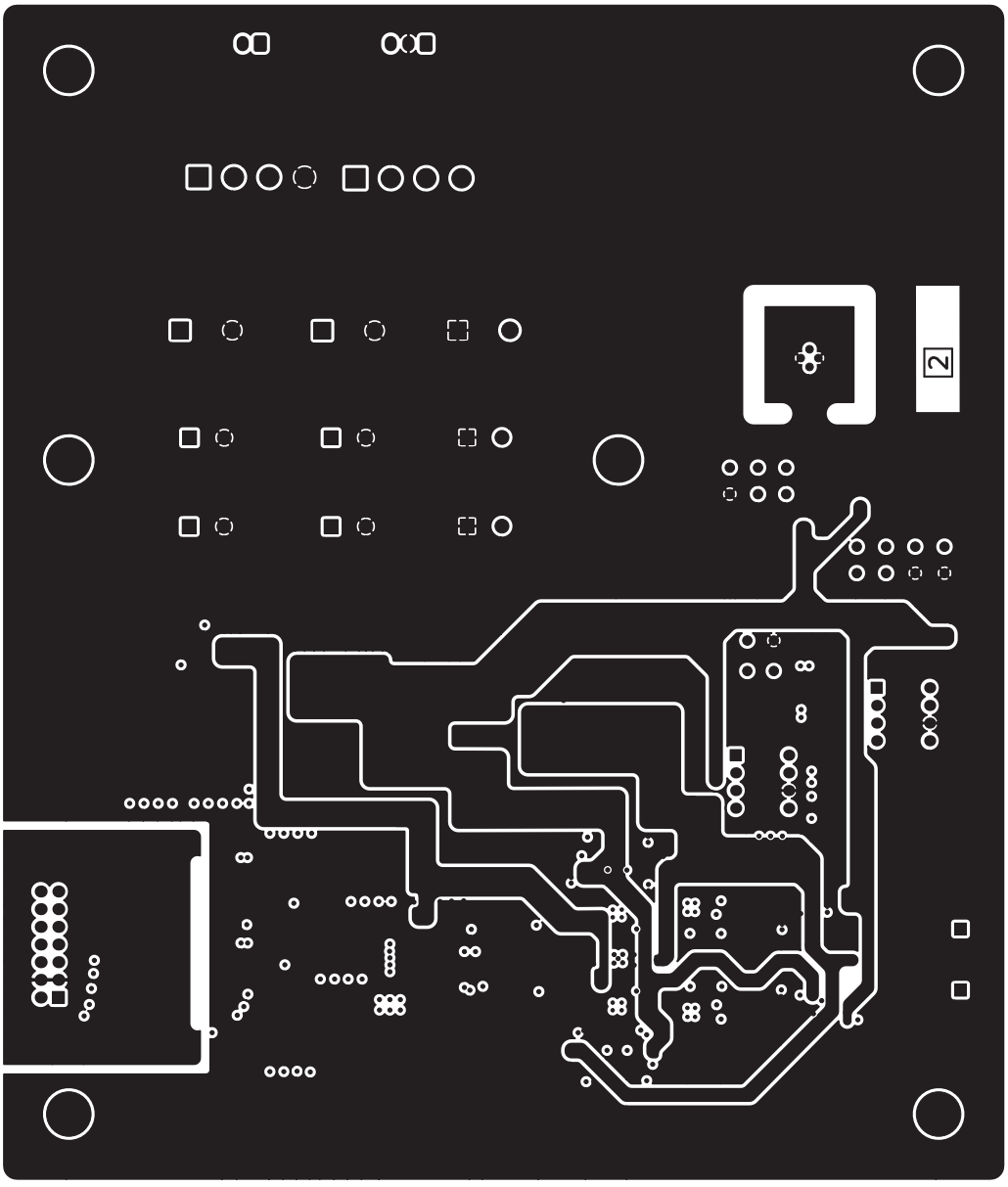


Simon Widmer		
Sheet: /Isolator/		
File: file61F88C5C.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: B**
KiCad E.D.A. kicad (5.1.5)-3		Id: 6/7

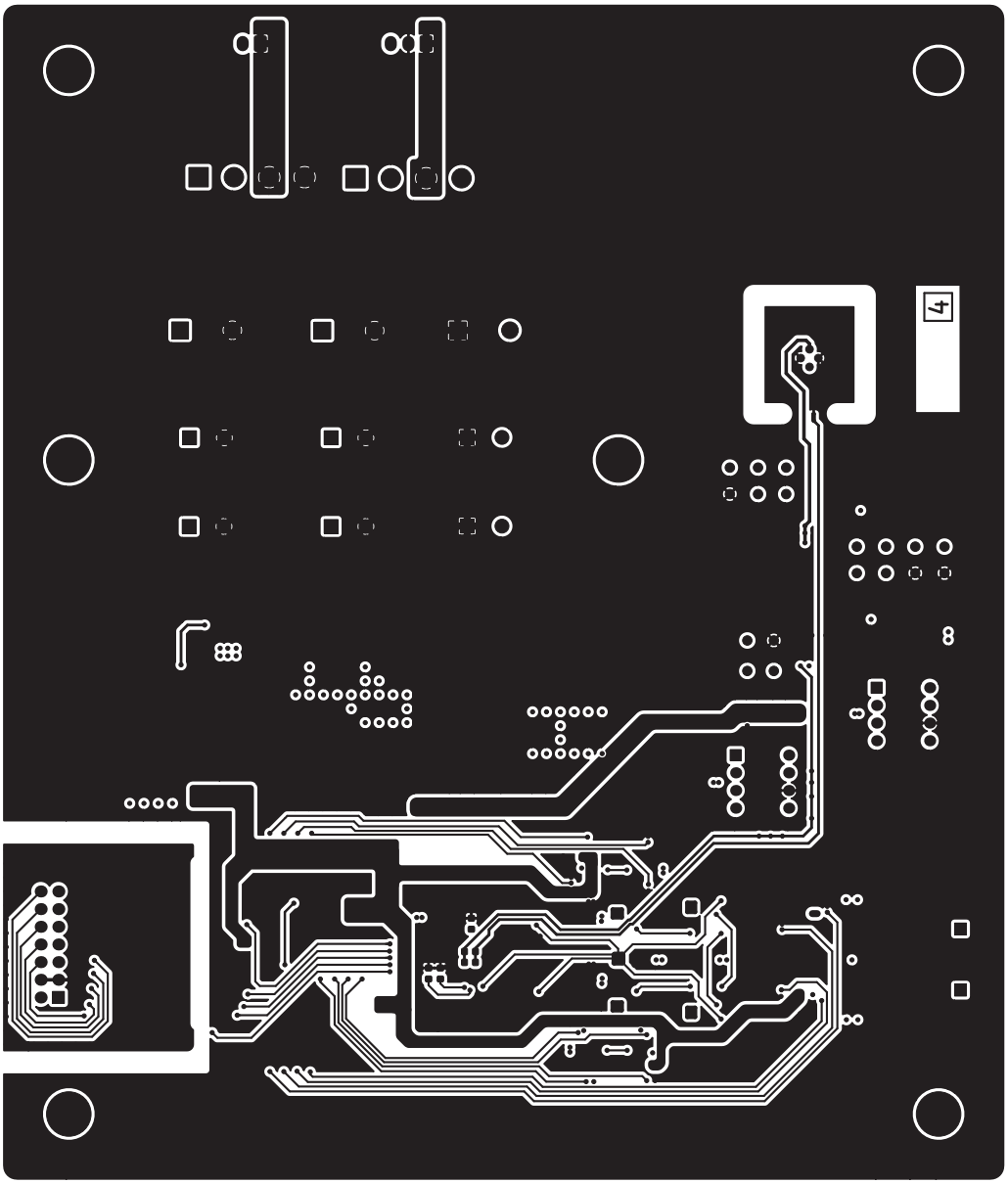


Simon Widmer		
Sheet: /Mechanik$/		
File: file62231D31.sch		
Title: DVM Analog-Board		
Size: A4	Date: 30.10.2021	**Rev: B**
KiCad E.D.A. kicad (5.1.5)-3		Id: 7/7


1


2

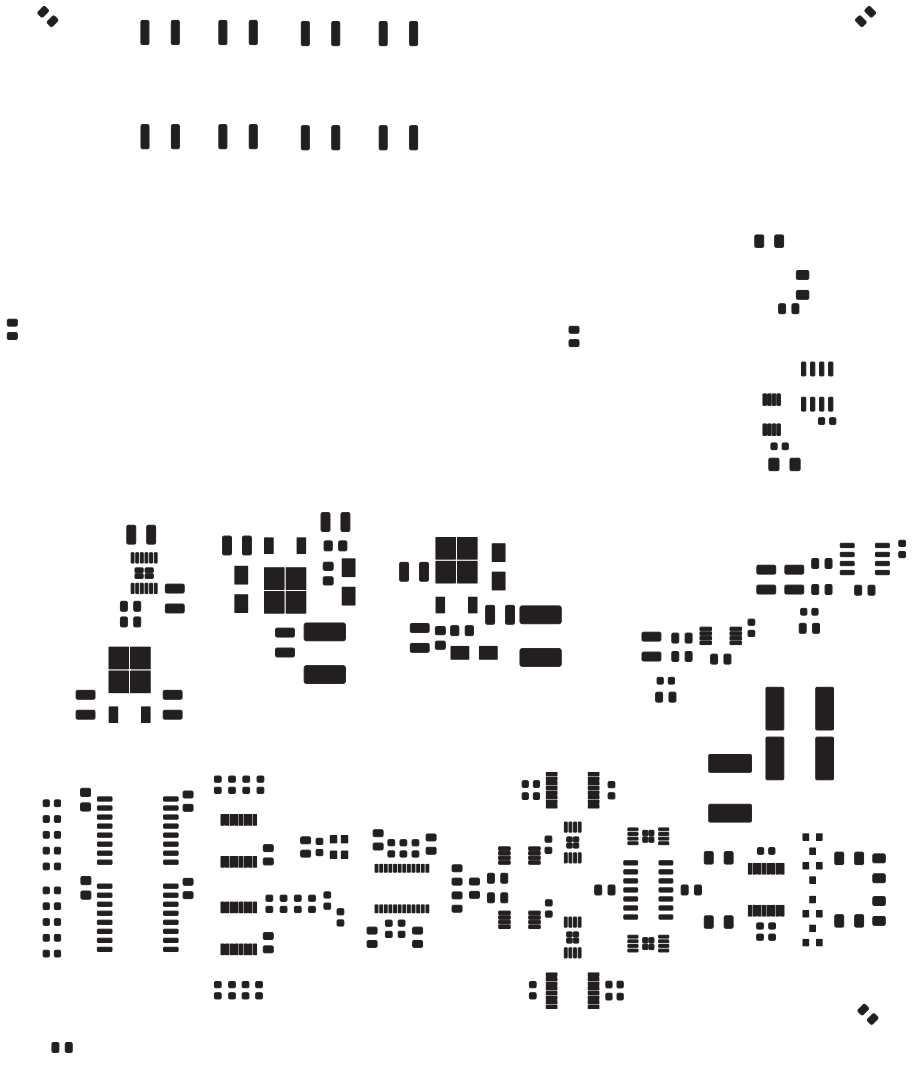
3

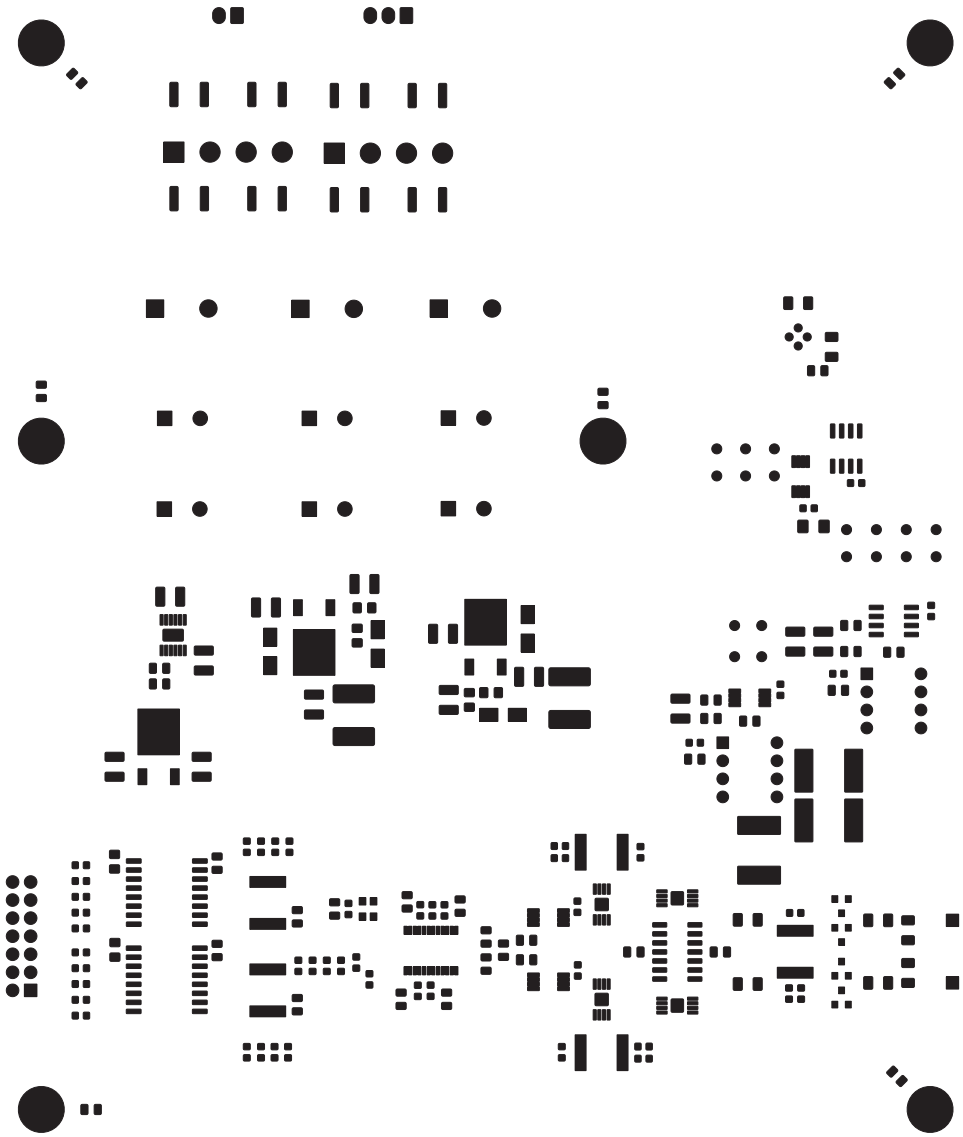

4

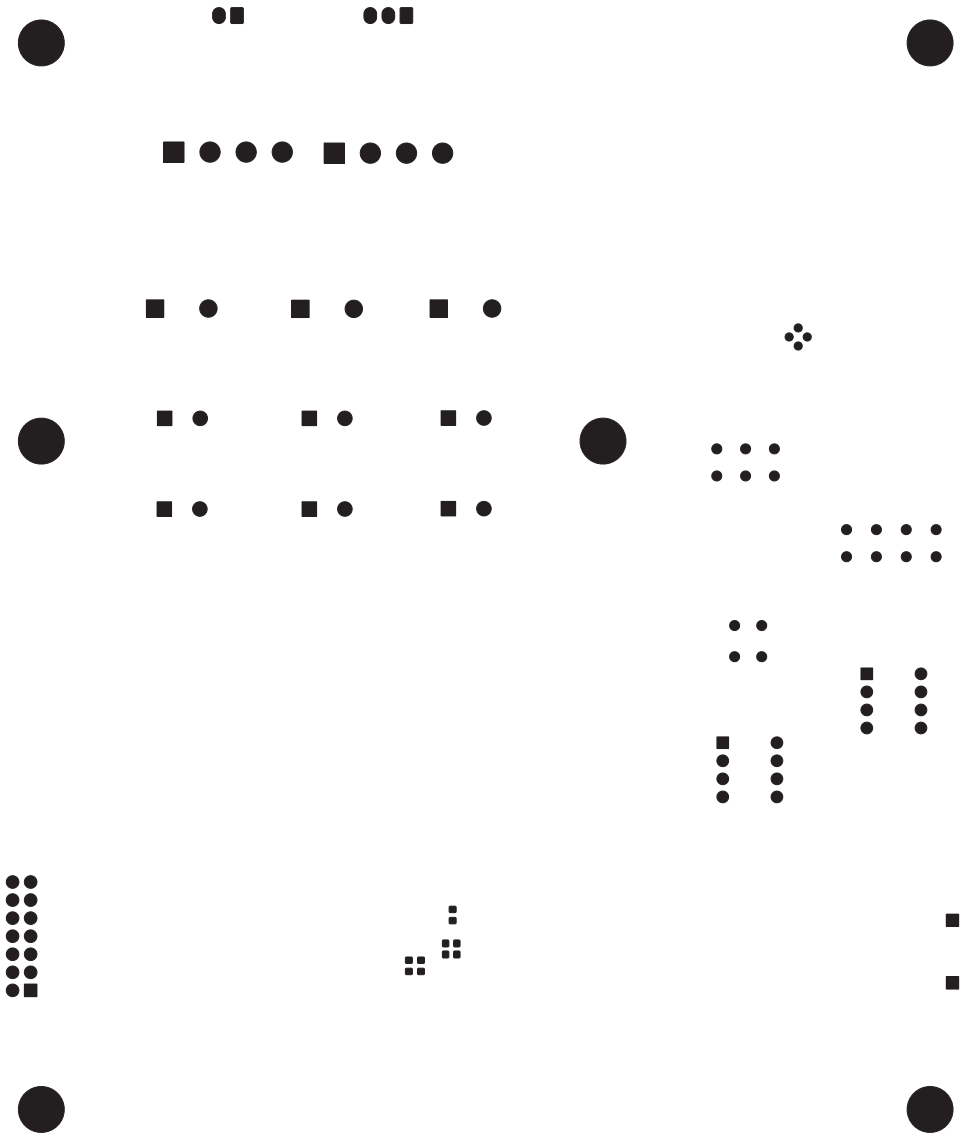
Analog PCB
Rev. B
Designed by Simon Widmer

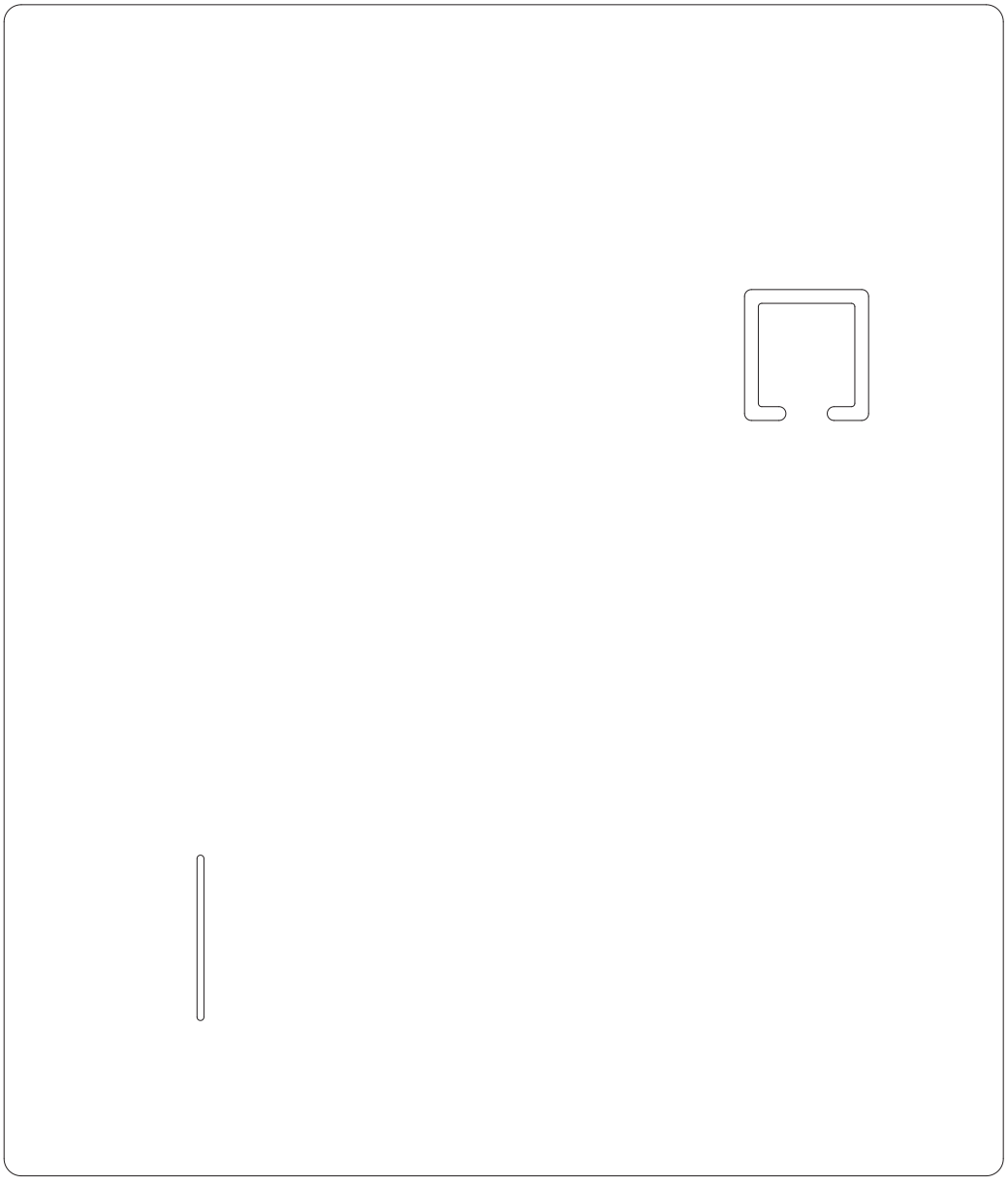
NT1

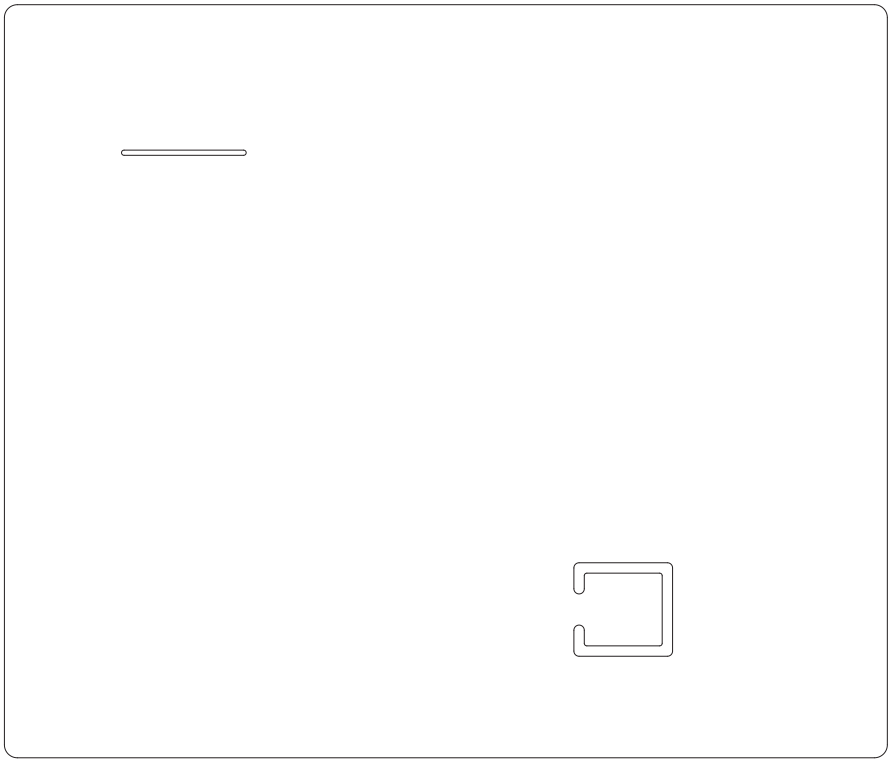
R1
C18
C17
R54
R55



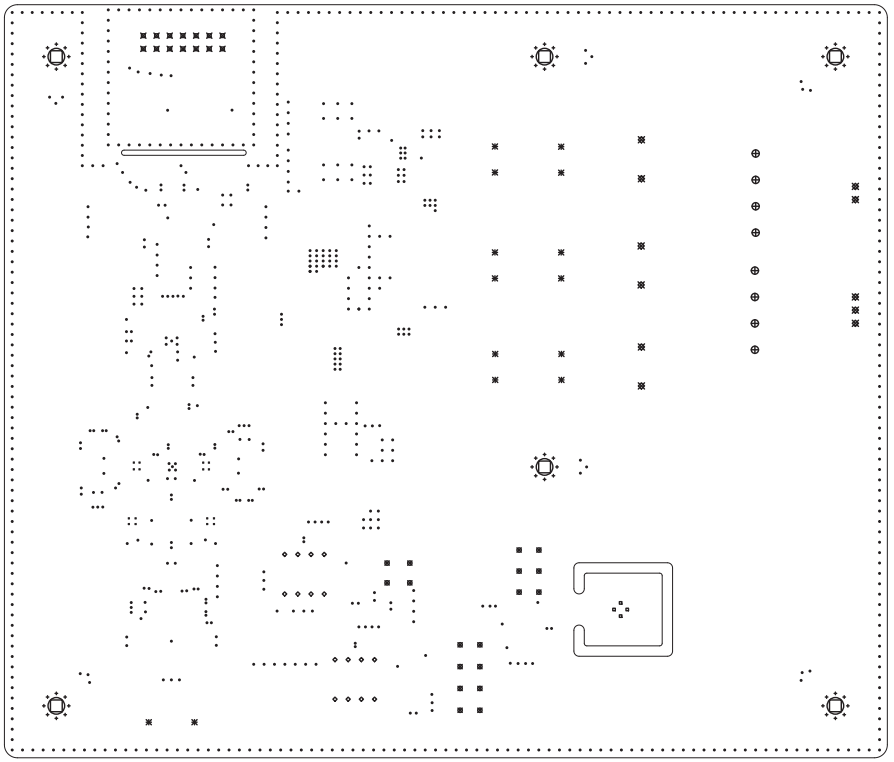








Drill Map:

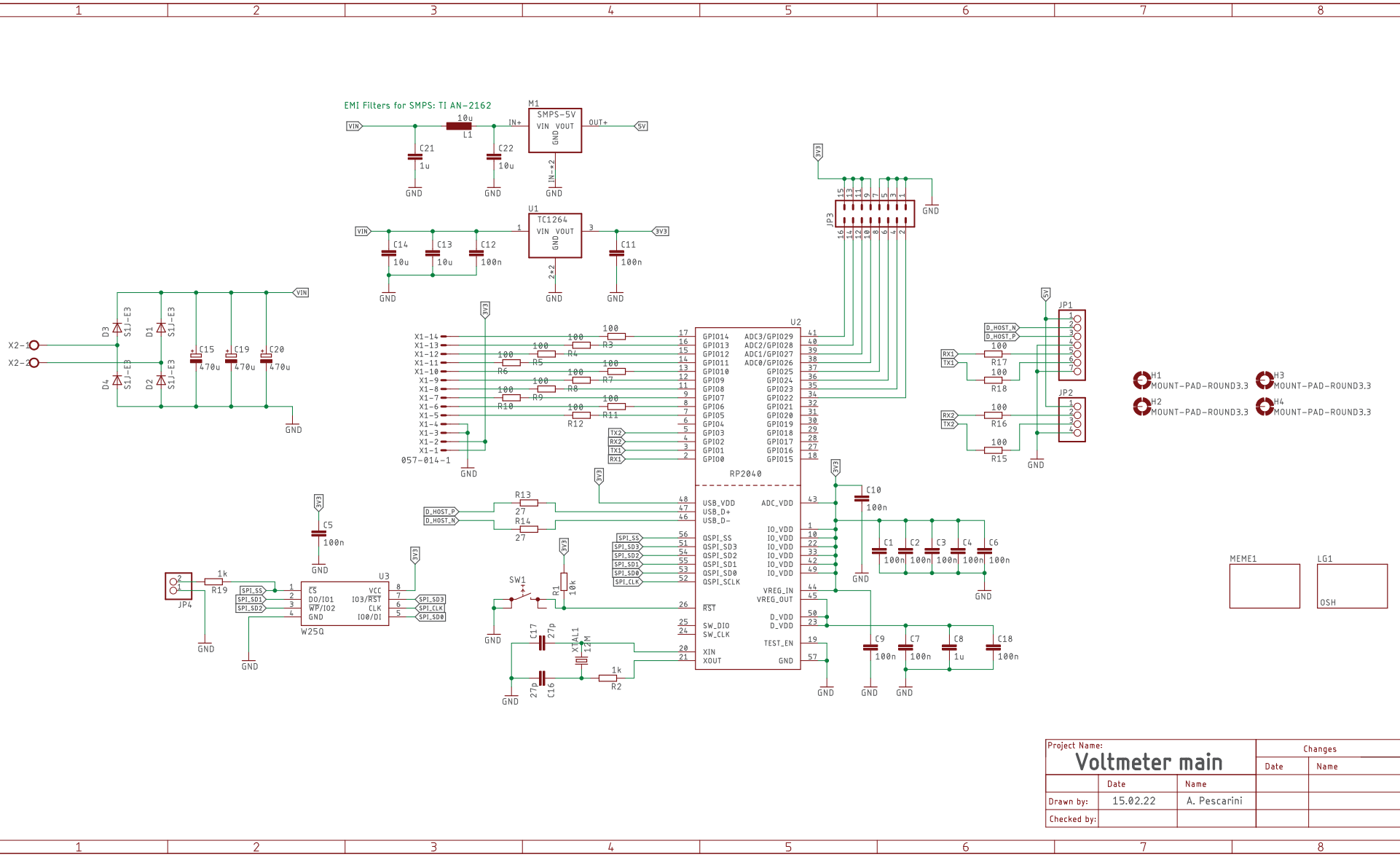


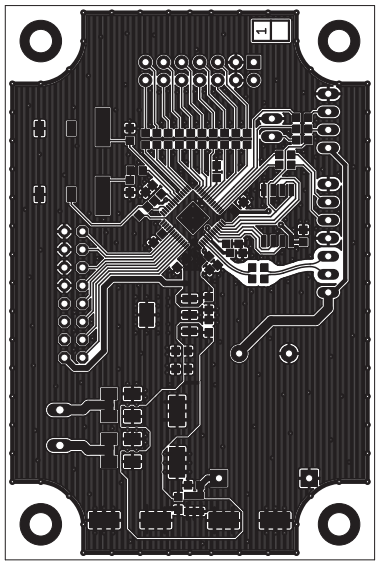
Drill Map:

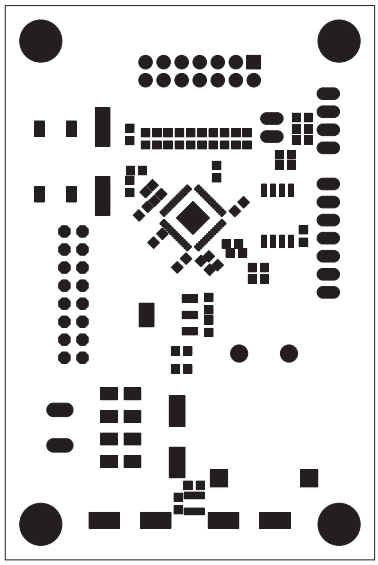
- 0.20mm / 0.008" (16 holes)
- 0.30mm / 0.012" (812 holes)
- 0.50mm / 0.020" (48 holes)
- 0.68mm / 0.027" (4 holes)
- 0.80mm / 0.031" (16 holes)
- 0.84mm / 0.033" (18 holes)
- 1.00mm / 0.039" (14 holes)
- 1.02mm / 0.040" (14 holes)
- 1.20mm / 0.047" (11 holes)
- 1.40mm / 0.055" (8 holes)
- 3.20mm / 0.126" (6 holes)

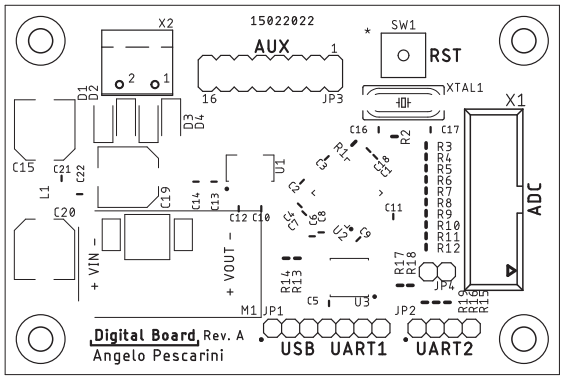
Appendix C

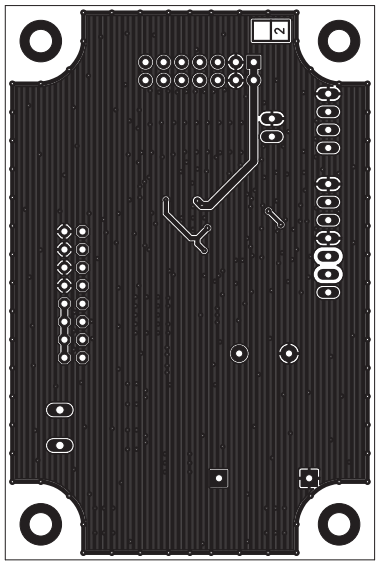
DIGITAL-BOARD REV. A

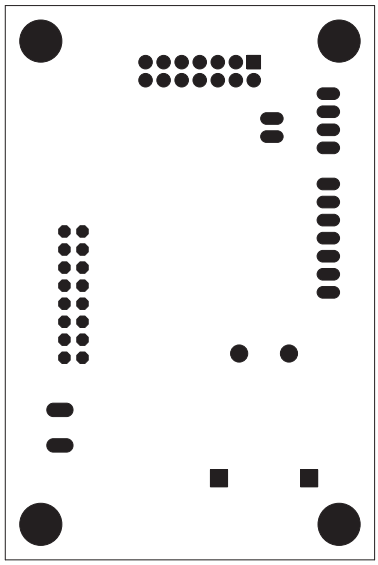

EMI Filters for SMPS: TI AN-2162
M1
SMPS-5V
VIN VOUT
GND
OUT+
IN+
IN-+2
L1
10u
C21
1u
C22
10u
U1
TC1264
VIN VOUT
GND
C14
10u
C13
10u
C12
100n
C11
100n
2+2
D3
D1
D4
D2
S1J-E3
C15
470u
C19
470u
C20
470u
X2-1
X2-2
U2
RP2040
GPIO14
GPIO13
GPIO12
GPIO11
GPIO10
GPIO9
GPIO8
GPIO7
GPIO6
GPIO5
GPIO4
GPIO3
GPIO2
GPIO1
GPIO0
ADC3/GPIO29
ADC2/GPIO28
ADC1/GPIO27
ADC0/GPIO26
GPIO25
GPIO24
GPIO23
GPIO22
GPIO21
GPIO20
GPIO19
GPIO18
GPIO17
GPIO16
GPIO15
USB_VDD
USB_D+
USB_D-
ADC_VDD
IO_VDD
QSPI_SS
QSPI_SD3
QSPI_SD2
QSPI_SD1
QSPI_SD0
QSPI_SCLK
VREG_IN
VREG_OUT
D_VDD
RST
SW_DIO
SW_CLK
TEST_EN
XIN
XOUT
057-014-1
JP1
JP2
JP3
JP4
D_HOST_N
D_HOST_P
RX1
TX1
RX2
TX2
R17
R18
R16
R15
R13
R14
27
R1
10k
R2
1k
R19
SW1
C17
27p
C16
XTAL1
12M
C5
100n
C10
C1
C2
C3
C4
C6
C9
C7
C8
1u
C18
U3
W25Q
CS
DO/IO1
WP/IO2
GND
VCC
IO3/RST
CLK
IO0/DI
H1
MOUNT-PAD-ROUND3.3
H2
MOUNT-PAD-ROUND3.3
H3
MOUNT-PAD-ROUND3.3
H4
MOUNT-PAD-ROUND3.3
MEME1
LG1
OSH
Project Name:
Voltmeter main
Changes
Date
Name
Drawn by:
15.02.22
A. Pescarini
Checked by:

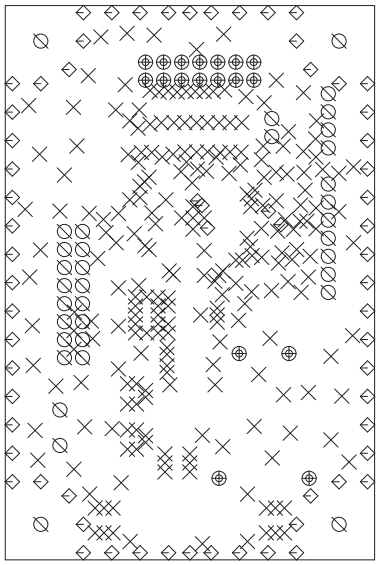

1




15022022
X2
AUX
SW1
RST
JP3
XTAL1
X1
ADC
C15
C21
C22
L1
C20
C19
C14
C13
C12
C10
U1
C11
C16
C17
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R14
R13
R17
R18
JP4
C5
U3
JP1
JP2
M1
+ VIN –
+ VOUT –
Digital Board Rev. A
Angelo Pescarini
USB
UART1
UART2


2





Appendix D

FIRMWARE

```cpp
/* Author: Yoris Kucera
     Date: 05.04.2022
File name: main.cpp
  Purpose: Firmware for the Digital Board of the Digital Voltmeter
*/
#include <pico/stdlib.h>

#include "DigitalVoltmeter.h"

int main() {
    auto& dvm = DigitalVoltmeter::get();

    dvm.initialize();

    while (true) {
        dvm.update();
    }
}

/* Author: Yoris Kucera
     Date: 05.04.2022
File name: DigitalVoltmeter.h
  Purpose: Defines the main DigitalVoltmeter class
*/
#ifndef INCLUDE_DIGITAL_VOLTMETER
#define INCLUDE_DIGITAL_VOLTMETER

#include <cstdint>
#include <queue>

#include <hardware/sync.h>

#include "ShiftRegister.h"
#include "AD7177.h"
#include "UART.h"

class DigitalVoltmeter {
    union RangeMuxes {
        struct {
            uint8_t bit0 : 1;
            uint8_t bit1 : 1;
            uint8_t bit2 : 1;
            uint8_t bit3 : 1;
            uint8_t bit4 : 1;
            uint8_t bit5 : 1;
            uint8_t bit6 : 1;
            uint8_t bit7 : 1;
        };
```

```
        uint8_t value; // Zugriff auf alle Bits
    };

public:
    enum class Range {
        Measure_200mV,
        Measure_2V,
        Measure_20V
    };
    struct IsolatorPins {
        uint8_t control;
        uint8_t disable;
    };

    DigitalVoltmeter();

    static DigitalVoltmeter& get();

    bool initialize();
    void update();

    void set_range(Range range);
    void set_autozero(bool enabled);

private:
    void apply_range() const;
    float convert_to_volts(uint32_t adc_value) const;

    static void communication();

    AD7177 m_adc;

    ShiftRegister<uint16_t> m_muxes;
    RangeMuxes m_range_state;

    UART m_uart;

    Range m_range;
    bool m_autozero;
    bool m_has_autozero_code;
    uint32_t m_autozero_code;

    int32_t m_spinlock_num;
    spin_lock_t* m_spinlock;

    std::queue<float> m_values;

    float m_vref = 2.5f;
```

```
    float m_gains[3] = {
        0.2f,
        2.0f,
        20.0f
    };

    // Pins
    static constexpr ShiftRegister<uint16_t>::Pins MuxPins = {
        .strobe = 9,
        .data = 11,
        .clock = 13
    };
    static constexpr AD7177::Pins ADCPins = {
        .cs = 8,
        .din = 12,
        .dout = 7,
        .sclk = 10
    }; // .error = ...
    static constexpr UART::Pins UartPins = {
        .rx = 25,
        .tx = 24
    };
    static constexpr IsolatorPins IsolatorPins = {
        .control = 5,
        .disable = 6
    };
    static constexpr uint32_t UartBaudrate = 115200;
};

#endif



/* Author: Yoris Kucera
     Date: 05.04.2022
File name: DigitalVoltmeter.cpp
  Purpose: Implementation of the DigitalVoltmeter class
*/
#include "DigitalVoltmeter.h"

#include <pico/stdlib.h>
#include <pico/multicore.h>

DigitalVoltmeter::DigitalVoltmeter()
    : m_muxes(MuxPins)
    , m_autozero(false)
    , m_range(Range::Measure_200mV)
    , m_adc(ADCPins)
```

```
    , m_autozero_code(0)
    , m_has_autozero_code(false)
    , m_uart(UartPins) {

    m_spinlock_num = spin_lock_claim_unused(true);
    m_spinlock = spin_lock_init(m_spinlock_num);
}

DigitalVoltmeter& DigitalVoltmeter::get() {
    static DigitalVoltmeter dvm;
    return dvm;
}

bool DigitalVoltmeter::initialize() {
    m_adc.initialize();
    m_uart.initialize(UartBaudrate);

    multicore_launch_core1(DigitalVoltmeter::communication);

    return true;
}


void DigitalVoltmeter::update() {
    // Read ADC and transmit data
    // Adjust range based on voltage read

    if (m_adc.data_ready()) {
        // Read value from ADC
        const auto val = m_adc.read_data();

        // Convert to Volts
        float voltage = convert_to_volts(val);

        if (m_autozero) {
            m_autozero_code = val;
            m_autozero = false;
        }

        spin_lock_unsafe_blocking(m_spinlock);
        m_values.push(voltage);
        spin_unlock_unsafe(m_spinlock);
    }
}

void DigitalVoltmeter::set_range(Range range) {
    if (range != m_range) {
        m_range = range;
```

```
        switch (m_range) {
        case Range::Measure_200mV:
            m_range_state.bit0 = 0;
            m_range_state.bit1 = 1;
            m_range_state.bit2 = 0;
            m_range_state.bit3 = 1;
            m_range_state.bit4 = 0;
            m_range_state.bit5 = 1;
            m_range_state.bit6 = 0;
            m_range_state.bit7 = 1;
            break;
        case Range::Measure_2V:
            m_range_state.bit0 = 1;
            m_range_state.bit1 = 1;
            m_range_state.bit2 = 0;
            m_range_state.bit3 = 0;
            m_range_state.bit4 = 1;
            m_range_state.bit5 = 1;
            m_range_state.bit6 = 0;
            m_range_state.bit7 = 0;
            break;
        case Range::Measure_20V:
            m_range_state.bit0 = 1;
            m_range_state.bit1 = 0;
            m_range_state.bit2 = 1;
            m_range_state.bit3 = 0;
            m_range_state.bit4 = 1;
            m_range_state.bit5 = 0;
            m_range_state.bit6 = 1;
            m_range_state.bit7 = 0;
            break;
        }
    }

    this->apply_range();
}

void DigitalVoltmeter::set_autozero(bool enabled) {
    if (m_autozero != enabled) {
        m_autozero = enabled;

        uint16_t autozero_value = m_autozero ? 0b0110 : 0b1001;
        uint16_t value = (autozero_value << 8) |
static_cast<uint16_t>(m_range_state.value);

        m_muxes.shift_out(static_cast<uint16_t>(m_range_state.value),
ShiftOrder::MSBFirst);
        m_muxes.set_data();
```

```
        sleep_ms(10); // Artificial delay to allow muxes to switch
        m_muxes.shift_out(value, ShiftOrder::MSBFirst);
        m_muxes.set_data();
    }
}

void DigitalVoltmeter::apply_range() const {
    if (!m_autozero) {
        uint16_t autozero_value = m_autozero ? 0b0110 : 0b1001;
        uint16_t value = (autozero_value << 8) |
static_cast<uint16_t>(m_range_state.value);

        m_muxes.shift_out(static_cast<uint16_t>(m_range_state.value),
ShiftOrder::MSBFirst);
        m_muxes.set_data();

        sleep_ms(10); // Artificial delay to allow muxes to switch
        m_muxes.shift_out(value, ShiftOrder::MSBFirst);
        m_muxes.set_data();
    }
}

float DigitalVoltmeter::convert_to_volts(uint32_t adc_value) const {
    // Code = 2^(N - 1) * ((AIN / VREF) + 1)
    // AIN = VREF((Code / 2^31) - 1)
    // AIN = VREF((Code - 2^31) / 2^31)

    bool negative = !(adc_value & 0x80000000);
    constexpr uint32_t two_pow_31 = 0x80000000; // 2^(32 - 1)

    if (m_has_autozero_code) {
        adc_value = adc_value - m_autozero_code + two_pow_31 - 1;
    }

    auto voltage = ((static_cast<float>(adc_value) - two_pow_31) /
two_pow_31) * m_vref;
    voltage *= m_gains[static_cast<int>(m_range)];

    return negative ? -voltage : voltage;
}

/* Author: Yoris Kucera
     Date: 05.04.2022
File name: ShiftRegister.h
  Purpose: Defines the ShiftRegister class
*/
#ifndef INCLUDE_SHIFT_REGISTER
```

```cpp
#define INCLUDE_SHIFT_REGISTER

#include <cstdint>

#include "ShiftRegister.h"
#include "pico/stdlib.h"

enum class ShiftOrder {
    MSBFirst,
    LSBFirst
};

template<class T>
class ShiftRegister {
public:
    struct Pins {
        uint8_t strobe;
        uint8_t data;
        uint8_t clock;
    };

    ShiftRegister(Pins pins);

    void shift_out(T data, ShiftOrder order) const;
    void set_data() const;

private:
    Pins m_pins;
};

template<class T>
ShiftRegister<T>::ShiftRegister(Pins pins)
    : m_pins(pins) {

    gpio_init(m_pins.strobe);
    gpio_set_dir(m_pins.strobe, GPIO_OUT);

    gpio_init(m_pins.data);
    gpio_set_dir(m_pins.data, GPIO_OUT);

    gpio_init(m_pins.clock);
    gpio_set_dir(m_pins.clock, GPIO_OUT);
}

template<class T>
void ShiftRegister<T>::shift_out(T data, ShiftOrder order) const {
    auto clk_pin = m_pins.clock;
    auto data_pin = m_pins.data;
```

```
    const uint8_t bit_count = sizeof(T) * 8;

    if (order == ShiftOrder::MSBFirst) {
        for (auto i = 0; i < bit_count; i++, data >>= 1) {
            gpio_put(data_pin, data & 1);

            gpio_put(clk_pin, true);
            gpio_put(clk_pin, false);
        }
    } else if (order == ShiftOrder::LSBFirst) {
        for (auto i = 0; i < bit_count; i++, data <<= 1) {
            gpio_put(data_pin, data & 0x80);

            gpio_put(clk_pin, true);
            gpio_put(clk_pin, false);
        }
    }
}

template<class T>
void ShiftRegister<T>::set_data() const {
    gpio_put(m_pins.strobe, true);
    gpio_put(m_pins.strobe, false);
}

#endif // INCLUDE_SHIFT_REGISTER

/* Author: Yoris Kucera
     Date: 05.04.2022
File name: ShiftRegister.h
  Purpose: Defines values for interfacing with the AD7177-2 ADC
*/
#ifndef INCLUDE_AD7177_DEFINE
#define INCLUDE_AD7177_DEFINE

/* AD7177 Register Map */
#define AD7177_COMM_REG       0x00
#define AD7177_STATUS_REG     0x00
#define AD7177_ADCMODE_REG    0x01
#define AD7177_IFMODE_REG     0x02
#define AD7177_REGCHECK_REG   0x03
#define AD7177_DATA_REG       0x04
#define AD7177_GPIOCON_REG    0x06
#define AD7177_ID_REG         0x07
#define AD7177_CHMAP0_REG     0x10
#define AD7177_CHMAP1_REG     0x11
#define AD7177_CHMAP2_REG     0x12
```

```
#define AD7177_CHMAP3_REG     0x13
#define AD7177_CHMAP4_REG     0x14
#define AD7177_CHMAP5_REG     0x15
#define AD7177_CHMAP6_REG     0x16
#define AD7177_CHMAP7_REG     0x17
#define AD7177_CHMAP8_REG     0x18
#define AD7177_CHMAP9_REG     0x19
#define AD7177_CHMAP10_REG    0x1A
#define AD7177_CHMAP11_REG    0x1B
#define AD7177_CHMAP12_REG    0x1C
#define AD7177_CHMAP13_REG    0x1D
#define AD7177_CHMAP14_REG    0x1E
#define AD7177_CHMAP15_REG    0x1F
#define AD7177_SETUPCON0_REG  0x20
#define AD7177_SETUPCON1_REG  0x21
#define AD7177_SETUPCON2_REG  0x22
#define AD7177_SETUPCON3_REG  0x23
#define AD7177_SETUPCON4_REG  0x24
#define AD7177_SETUPCON5_REG  0x25
#define AD7177_SETUPCON6_REG  0x26
#define AD7177_SETUPCON7_REG  0x27
#define AD7177_FILTCON0_REG   0x28
#define AD7177_FILTCON1_REG   0x29
#define AD7177_FILTCON2_REG   0x2A
#define AD7177_FILTCON3_REG   0x2B
#define AD7177_FILTCON4_REG   0x2C
#define AD7177_FILTCON5_REG   0x2D
#define AD7177_FILTCON6_REG   0x2E
#define AD7177_FILTCON7_REG   0x2F
#define AD7177_OFFSET0_REG    0x30
#define AD7177_OFFSET1_REG    0x31
#define AD7177_OFFSET2_REG    0x32
#define AD7177_OFFSET3_REG    0x33
#define AD7177_OFFSET4_REG    0x34
#define AD7177_OFFSET5_REG    0x35
#define AD7177_OFFSET6_REG    0x36
#define AD7177_OFFSET7_REG    0x37
#define AD7177_GAIN0_REG      0x38
#define AD7177_GAIN1_REG      0x39
#define AD7177_GAIN2_REG      0x3A
#define AD7177_GAIN3_REG      0x3B
#define AD7177_GAIN4_REG      0x3C
#define AD7177_GAIN5_REG      0x3D
#define AD7177_GAIN6_REG      0x3E
#define AD7177_GAIN7_REG      0x3F

/* Communication Register bits */
#define AD7177_COMM_REG_WEN    (0 << 7)
```

```
#define AD7177_COMM_REG_WR     (0 << 6)
#define AD7177_COMM_REG_RD     (1 << 6)
#define AD7177_COMM_REG_RA(x)  ((x) & 0x3F)

/* Status Register bits */
#define AD7177_STATUS_REG_RDY      (1 << 7)
#define AD7177_STATUS_REG_ADC_ERR  (1 << 6)
#define AD7177_STATUS_REG_CRC_ERR  (1 << 5)
#define AD7177_STATUS_REG_REG_ERR  (1 << 4)
#define AD7177_STATUS_REG_CH(x)    ((x) & 0x0F)

/* ADC Mode Register bits */
#define AD7177_ADCMODE_REG_REF_EN     (1 << 15)
#define AD7177_ADCMODE_SING_CYC       (1 << 13)
#define AD7177_ADCMODE_REG_DELAY(x)   (((x) & 0x7) << 8)
#define AD7177_ADCMODE_REG_MODE(x)    (((x) & 0x7) << 4)
#define AD7177_ADCMODE_REG_CLKSEL(x)  (((x) & 0x3) << 2)

/* ADC Mode Register additional bits for AD7172-2 */
#define AD7177_ADCMODE_REG_HIDE_DELAY   (1 << 14)

/* Interface Mode Register bits */
#define AD7177_IFMODE_REG_ALT_SYNC      (1 << 12)
#define AD7177_IFMODE_REG_IOSTRENGTH    (1 << 11)
#define AD7177_IFMODE_REG_HIDE_DELAY    (1 << 10)
#define AD7177_IFMODE_REG_DOUT_RESET    (1 << 8)
#define AD7177_IFMODE_REG_CONT_READ     (1 << 7)
#define AD7177_IFMODE_REG_DATA_STAT     (1 << 6)
#define AD7177_IFMODE_REG_REG_CHECK     (1 << 5)
#define AD7177_IFMODE_REG_XOR_EN        (0x01 << 2)
#define AD7177_IFMODE_REG_CRC_EN        (0x02 << 2)
#define AD7177_IFMODE_REG_XOR_STAT(x)   (((x) & AD7177_IFMODE_REG_XOR_EN)
== AD7177_IFMODE_REG_XOR_EN)
#define AD7177_IFMODE_REG_CRC_STAT(x)   (((x) & AD7177_IFMODE_REG_CRC_EN)
== AD7177_IFMODE_REG_CRC_EN)
#define AD7177_IFMODE_REG_DATA_WL16     (1 << 0)
#define AD7177_IFMODE_REG_DATA_WL32     (1 << 1)

/* GPIO Configuration Register bits */
#define AD7177_GPIOCON_REG_MUX_IO      (1 << 12)
#define AD7177_GPIOCON_REG_SYNC_EN     (1 << 11)
#define AD7177_GPIOCON_REG_ERR_EN(x)   (((x) & 0x3) << 9)
#define AD7177_GPIOCON_REG_ERR_DAT     (1 << 8)
#define AD7177_GPIOCON_REG_IP_EN1      (1 << 5)
#define AD7177_GPIOCON_REG_IP_EN0      (1 << 4)
#define AD7177_GPIOCON_REG_OP_EN1      (1 << 3)
#define AD7177_GPIOCON_REG_OP_EN0      (1 << 2)
#define AD7177_GPIOCON_REG_DATA1       (1 << 1)
```

```
#define AD7177_GPIOCON_REG_DATA0       (1 << 0)

/* GPIO Configuration Register additional bits for AD7172-4, AD7173-8 */
#define AD7177_GPIOCON_REG_GP_DATA3    (1 << 7)
#define AD7177_GPIOCON_REG_GP_DATA2    (1 << 6)
#define AD7177_GPIOCON_REG_GP_DATA1    (1 << 1)
#define AD7177_GPIOCON_REG_GP_DATA0    (1 << 0)

/* GPIO Configuration Register additional bits for AD7173-8 */
#define AD7177_GPIOCON_REG_PDSW        (1 << 14)
#define AD7177_GPIOCON_REG_OP_EN2_3    (1 << 13)

/* Channel Map Register 0-3 bits */
#define AD7177_CHMAP_REG_CH_EN         (1 << 15)
#define AD7177_CHMAP_REG_SETUP_SEL(x)  (((x) & 0x7) << 12)
#define AD7177_CHMAP_REG_AINPOS(x)     (((x) & 0x1F) << 5)
#define AD7177_CHMAP_REG_AINNEG(x)     (((x) & 0x1F) << 0)

/* Channel Map Register additional bits for AD4111 */
#define AD4111_CHMAP_REG_INPUT(x)      (((x) & 0x3FF) << 0)

/* Setup Configuration Register 0-3 bits */
#define AD7177_SETUP_CONF_REG_BI_UNIPOLAR  (1 << 12)
#define AD7177_SETUP_CONF_REG_REF_SEL(x)   (((x) & 0x3) << 4)

/* Setup Configuration Register additional bits for AD7173-8 */
#define AD7177_SETUP_CONF_REG_REF_BUF(x)  (((x) & 0x3) << 10)
#define AD7177_SETUP_CONF_REG_AIN_BUF(x)  (((x) & 0x3) << 8)
#define AD7177_SETUP_CONF_REG_BURNOUT_EN  (1 << 7)
#define AD7177_SETUP_CONF_REG_BUFCHOPMAX  (1 << 6)

/* Setup Configuration Register additional bits for AD7172-2, AD7172-4,
AD7175-2 */
#define AD7177_SETUP_CONF_REG_REFBUF_P    (1 << 11)
#define AD7177_SETUP_CONF_REG_REFBUF_N    (1 << 10)
#define AD7177_SETUP_CONF_REG_AINBUF_P    (1 << 9)
#define AD7177_SETUP_CONF_REG_AINBUF_N    (1 << 8)

/* Filter Configuration Register 0-3 bits */
#define AD7177_FILT_CONF_REG_SINC3_MAP    (1 << 15)
#define AD7177_FILT_CONF_REG_ENHFILTEN    (1 << 11)
#define AD7177_FILT_CONF_REG_ENHFILT(x)   (((x) & 0x7) << 8)
#define AD7177_FILT_CONF_REG_ORDER(x)     (((x) & 0x3) << 5)
#define AD7177_FILT_CONF_REG_ODR(x)       (((x) & 0x1F) << 0)

/* ID register mask for relevant bits */
#define AD7177_ID_REG_MASK    0xFFF0
/* AD7177-2 ID */
```

```
#define AD7177_2_ID_REG_VALUE 0x4FD0

#endif

/* Author: Yoris Kucera
     Date: 05.04.2022
File name: AD7177.h
  Purpose: Defines the AD7177 class
*/
#ifndef INCLUDE_AD7177
#define INCLUDE_AD7177

#include "ShiftRegister.h"

#include <cstdint>

class AD7177 {
public:
    struct Pins {
        uint8_t cs;
        uint8_t din;
        uint8_t dout;
        uint8_t sclk;
    };

    AD7177(Pins pins);

    void initialize();
    void reset();

    bool data_ready() const;
    uint32_t read_data();

private:
    uint8_t write_byte(uint8_t data, ShiftOrder order =
ShiftOrder::MSBFirst);
    uint8_t read_byte();

    uint16_t read_register(uint8_t addr);
    void write_register(uint8_t addr, uint16_t data);

private:
    Pins m_pins;
};

#endif
```

```
/* Author: Yoris Kucera
     Date: 05.04.2022
File name: AD7177.cpp
  Purpose: Implementation of the AD7177 class
*/
#include "AD7177.h"
#include "AD7177Define.h"

#include <pico/stdlib.h>

#define STALL() asm volatile("nop \n nop \n nop")

AD7177::AD7177(Pins pins)
    : m_pins(pins) {
}

void AD7177::initialize() {
    gpio_init(m_pins.cs);
    gpio_init(m_pins.din);
    gpio_init(m_pins.dout);
    gpio_init(m_pins.sclk);

    gpio_set_dir(m_pins.cs, GPIO_OUT);
    gpio_set_dir(m_pins.din, GPIO_OUT);
    gpio_set_dir(m_pins.dout, GPIO_IN);
    gpio_set_dir(m_pins.sclk, GPIO_OUT);

    gpio_put(m_pins.cs, false);

    write_register(
        AD7177_GPIOCON_REG,
        AD7177_GPIOCON_REG_DATA0 | AD7177_GPIOCON_REG_OP_EN0 |
AD7177_GPIOCON_REG_OP_EN1
    );

    write_register(
        AD7177_ADCMODE_REG,
        AD7177_ADCMODE_REG_CLKSEL(0x2)
    );

    write_register(
        AD7177_CHMAP0_REG,
        AD7177_CHMAP_REG_AINPOS(0x4) | AD7177_CHMAP_REG_AINNEG(0x3) |
AD7177_CHMAP_REG_CH_EN
    );

    write_register(
        AD7177_SETUPCON0_REG,
```

```
        AD7177_SETUP_CONF_REG_AIN_BUF(0x3) |
AD7177_SETUP_CONF_REG_REF_BUF(0x3) | AD7177_SETUP_CONF_REG_BI_UNIPOLAR
    );

    write_register(
        AD7177_FILTCON0_REG,
        AD7177_FILT_CONF_REG_ODR(0x14) | AD7177_FILT_CONF_REG_ORDER(0x3)
    );

    write_register(
        AD7177_IFMODE_REG,
        AD7177_IFMODE_REG_ALT_SYNC | AD7177_IFMODE_REG_CONT_READ |
AD7177_IFMODE_REG_DATA_WL32
    );
}

void AD7177::reset() {
    gpio_put(m_pins.sclk, true);
    sleep_ms(10);
    gpio_put(m_pins.cs, false);

    for (auto i = 0; i < 10; i++) {
        write_byte(0xFF, ShiftOrder::LSBFirst);
    }

    gpio_put(m_pins.cs, true);
    sleep_ms(10);
    gpio_put(m_pins.cs, false);
}

bool AD7177::data_ready() const {
    return !gpio_get(m_pins.dout);
}

uint8_t AD7177::write_byte(uint8_t data, ShiftOrder order) {
    if (order == ShiftOrder::MSBFirst) {
        for (auto i = 0; i < 8; i++, data <<= 1) {
            gpio_put(m_pins.din, data & 0x80);

            gpio_put(m_pins.sclk, false);
            STALL();
            gpio_put(m_pins.sclk, true);
        }
    } else if (order == ShiftOrder::LSBFirst) {
        for (auto i = 0; i < 8; i++, data >>= 1) {
            gpio_put(m_pins.din, data & 0x01);

            gpio_put(m_pins.sclk, false);
```

```
            STALL();
            gpio_put(m_pins.sclk, true);
        }
    }

    return 0;
}

uint8_t AD7177::read_byte() {
    uint8_t val = 0;

    for (auto i = 0u; i < 8; i++) {
        gpio_put(m_pins.sclk, false);
        val |= (static_cast<uint8_t>(gpio_get(m_pins.dout)) << i);
        gpio_put(m_pins.sclk, true);
    }

    return val;
}

uint32_t AD7177::read_data() {
    uint32_t val = 0;

    for (auto i = 0u; i < 32; i++) {
        // SCLK Pin auf Low setzen
        gpio_put(m_pins.sclk, false);
        // DOUT Pin lesen, und Bit setzen/löschen
        val |= (static_cast<uint32_t>(gpio_get(m_pins.dout)) << i);
        // SCLK Pin auf High setzen
        gpio_put(m_pins.sclk, true);
    }

    return val;
}

uint16_t AD7177::read_register(uint8_t addr) {
    uint16_t register_value = 0;

    gpio_put(m_pins.cs, false);

    register_value = static_cast<uint16_t>(read_byte()) << 8;
    register_value |= read_byte();

    gpio_put(m_pins.cs, true);

    return register_value;
}
```

```
void AD7177::write_register(uint8_t addr, uint16_t data) {
    write_byte(addr);
    write_byte(data >> 8);
    write_byte(data & 0xFF);
}

/* Author: Yoris Kucera
     Date: 05.04.2022
File name: UART.h
  Purpose: Defines the UART class
*/
#ifndef INCLUDE_UART_H
#define INCLUDE_UART_H

#include <cstdint>
#include <cstddef>

#include <hardware/uart.h>

class UART {
public:
    struct Pins {
        uint8_t rx;
        uint8_t tx;
    };

    enum class TxCommand : uint8_t {
        AdcValue = 0x00
    };

    enum class RxCommand : uint8_t {
        SetRangeMux = 0x00,
        SetAutozero = 0x01,
        SetRange = 0x02,
        SetRefVoltage = 0x03
    };

    UART(Pins pins);

    void initialize(uint32_t baudrate);

    bool has_tx_data() const;

    // Read Functions
    char read_char() const;
    RxCommand read_command() const;
    size_t read_str(char* buffer, size_t size) const;
```

```
    bool read_str_blocking(char* buffer, size_t count, uint64_t timout_us =
-1) const;
    float read_float() const;

    // Write Functions
    void write_command(TxCommand cmd) const;
    void write_str(const char* buffer, size_t size) const;
    void write_float(float val) const;

private:
    uart_inst_t* m_inst;
    Pins m_pins;
};

#endif // INCLUDE_UART_H

/* Author: Yoris Kucera
     Date: 05.04.2022
File name: UART.cpp
  Purpose: Implementation of the UART class
*/
#include "UART.h"

#include <pico/stdlib.h>
#include <hardware/uart.h>

#include <cmath>

UART::UART(Pins pins) : m_pins(pins), m_inst(uart1) {
}

void UART::initialize(uint32_t baudrate) {
    uart_init(m_inst, baudrate);

    gpio_set_function(m_pins.rx, GPIO_FUNC_UART);
    gpio_set_function(m_pins.tx, GPIO_FUNC_UART);
}

bool UART::has_tx_data() const {
    return uart_is_readable(m_inst);
}

char UART::read_char() const {
    return has_tx_data() ? uart_getc(m_inst) : 0;
}

UART::RxCommand UART::read_command() const {
    return static_cast<RxCommand>(read_char());
```

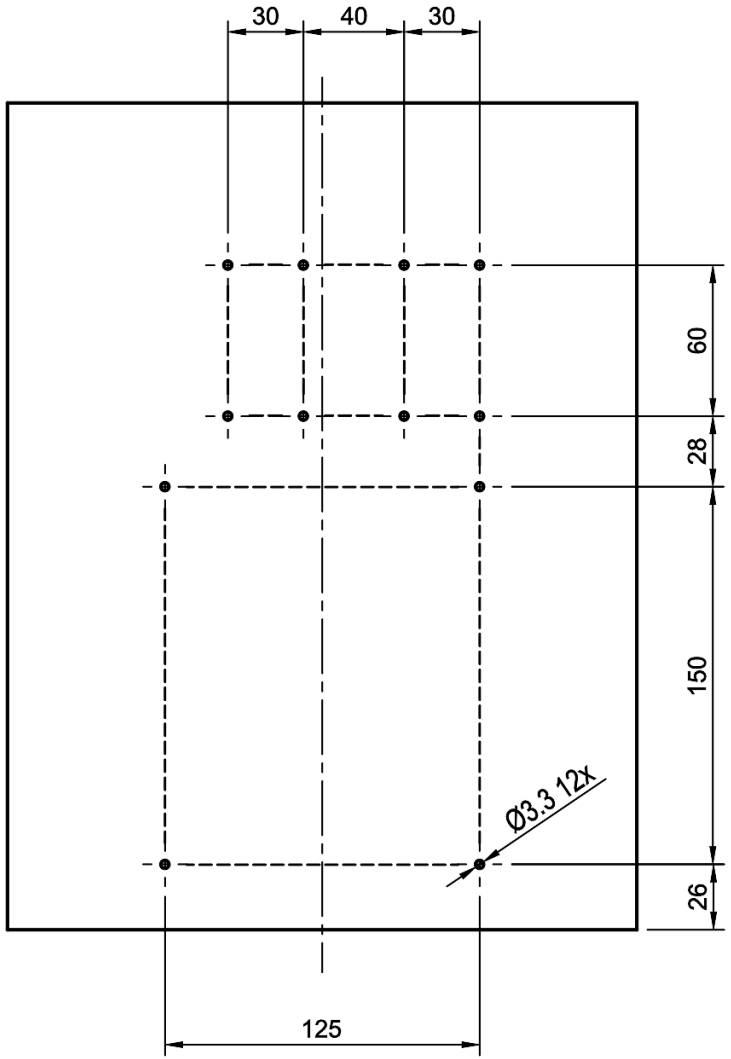
```
}

size_t UART::read_str(char* buffer, size_t size) const {
    size_t count = 0;
    while (has_tx_data()) {
        buffer[count] = uart_getc(m_inst);
        count++;
    }

    return count;
}

bool UART::read_str_blocking(char* buffer, size_t count, uint64_t
timeout_us) const {
    if (timeout_us != -1) {
        absolute_time_t t = time_us_64() + timeout_us;

        for (size_t i = 0; i < count; ++i) {
            while (!uart_is_readable(m_inst)) {
                if (time_reached(t))
                    return false;

                tight_loop_contents();
            }

            *buffer++ = static_cast<uint8_t>(uart_get_hw(m_inst)->dr);
        }
    } else {
        uart_read_blocking(m_inst, reinterpret_cast<uint8_t*>(buffer),
count);
    }

    return true;
}

float UART::read_float() const {
    float val = 0;
    const auto n = read_str(reinterpret_cast<char*>(&val), sizeof(val));

    return n == sizeof(val) ? val : NAN;
}

void UART::write_command(TxCommand cmd) const {
    if (uart_is_writable(m_inst)) {
        uart_putc_raw(m_inst, static_cast<char>(cmd));
    }
}
```

```
void UART::write_str(const char* buffer, size_t size) const {
    if (uart_is_writable(m_inst)) {
        for (auto i = 0u; i < size; i++) {
            uart_putc_raw(m_inst, buffer[i]);
        }
    }
}

void UART::write_float(float val) const {
    write_str(reinterpret_cast<char*>(&val), sizeof(val));
}


/* Author: Yoris Kucera
     Date: 05.04.2022
File name: Communication.cpp
  Purpose: Implementation of the USB Functionality of the firmware
*/
#include "DigitalVoltmeter.h"

#include <pico/stdlib.h>
#include <hardware/uart.h>

#define UART_ID uart0
#define UART_BAUDRATE 115200

void DigitalVoltmeter::communication() {
    auto& dvm = get();

    char command_buffer[8] = { 0 };

    while (true) {
        if (dvm.m_uart.has_tx_data()) {
            const UART::RxCommand cmd = dvm.m_uart.read_command();

            switch (cmd) {
                case UART::RxCommand::SetRangeMux:
                    if (dvm.m_uart.read_str_blocking(command_buffer, 1, 100
* 1000)) {
                        dvm.m_range_state.value = command_buffer[0];
                        dvm.apply_range();
                    }
                    break;
                case UART::RxCommand::SetAutozero:
                    if (dvm.m_uart.read_str_blocking(command_buffer, 1, 100
* 1000)) {
                        dvm.set_autozero(command_buffer[0]);
                    }
                    break;
```

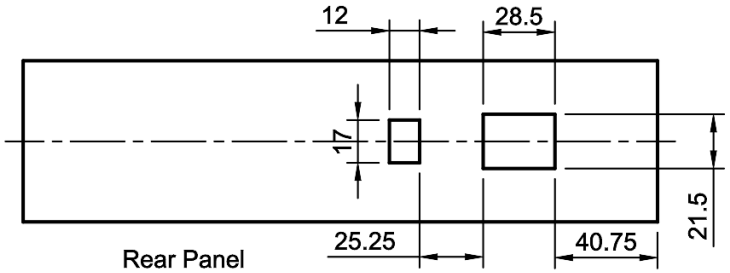
```
                case UART::RxCommand::SetRange:
                    if (dvm.m_uart.read_str_blocking(command_buffer, 1, 100
* 1000)) {
                        dvm.set_range(static_cast<Range>(command_buffer[0])
);
                    }
                    break;
                case UART::RxCommand::SetRefVoltage:
                    if (dvm.m_uart.read_str_blocking(command_buffer, 4, 100
* 1000)) {
                        dvm.m_vref =
*reinterpret_cast<float*>(command_buffer);
                    }
                    break;
            }
        }

        if (!dvm.m_values.empty()) {
            spin_lock_unsafe_blocking(dvm.m_spinlock);

            float value = dvm.m_values.front();
            dvm.m_values.pop();
            spin_unlock_unsafe(dvm.m_spinlock);

            dvm.m_uart.write_command(UART::TxCommand::AdcValue);
            dvm.m_uart.write_float(value);
        }
    }
}
```

Appendix E

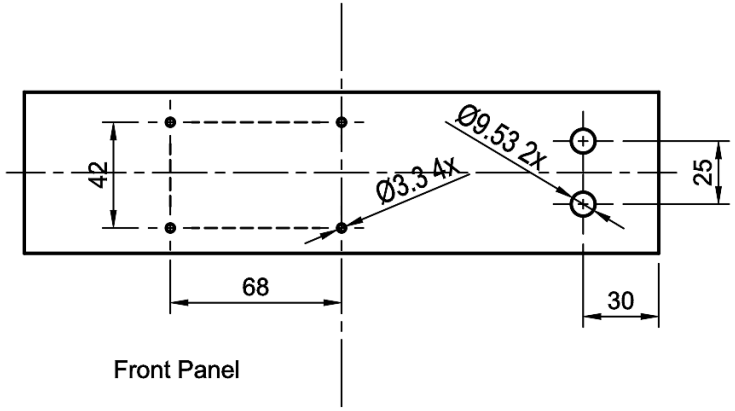
MECHANIK



Platte Boden



Rear Panel



Front Panel

Dept.	Technical reference	Created by	Approved by
		Angelo Pescarini 02.04.2022	
		Document type	Document status
		Title: DVM Montage Löcher	DWG No.
M: 1:2			Rev. A / Date of issue / Sheet 1/1

Appendix F

BILL OF MATERIAL

Analog Board

Bauteil	Wert	Footprint	Beschreibung	Hersteller	Hersteller Nr.
C3,C4,C5,C7,C8,C9,C10,C11,C14,C28,C30,C33,C38,C39,C40,C41,C45,C46,C56,C57,C58,C87,C88	100nF	C_0603	0603 100nF 10% 50VDC X7R MLCC	Wurth Elektronik	885382206004
C17,C18	18nF	C_0603	0603 35V 0.018uF C0G 5% MLCC	TDK	C1608C0G1V183J080AC
C1,C2,C6,C12,C13,C15,C16,C50,C61,C62	1uF	C_0805	X7R MLCC 1uF 10% 25VDC	Wurth Elektronik	885382207003
C35,C36,C64,C77,C80,C81	100nF	C_0805	X7R MLCC 100nF 10% 50VDC	Wurth Elektronik	885382207007
C43,C65	18nF	C_0805	C0G MLCC 18nF 35VDC 5%	TDK	C1608C0G1V183J080AC
C19,C20,C21	33nF	C_0805	C0G MLCC 50V 0.033uF 5%	TDK	CGA4J2NP01H333J125AA
C29,C34,C37,C60,C69,C70	DNP	C_0805	N/A	N/A	N/A
C22,C32	1uF	C_1206	X7R MLCC 1uF 10% 50VDC	Wurth Elektronik	885382208005
C44,C63,C88	100nF	C_1210	C0G MLCC 50V 0.1uF 5%	TDK	C3225C0G1H104J250AA
C31,C49,C51,C52,C53,C54,C55,C82,C83,C84,C86	10uF	C_1210	X7R MLCC 4.7uF+/-10% 50V	Samsung Electro-Mechanics	CL32B475KBUYNNE
C23,C24,C25,C26,C42,C47,C48,C85	100nF	C_1812	X7R MLCC 0.1 uF +/ -20% 500 V	Walsin	1812B104M501CT
C75,C76	10uF	C_2320	X7R MLCC 10uF 100 Volt	United Chemi-Con	KTJ101B106M55BFT00
C78,C89,C90	1uF	C_2824	PET Foil 1uF 63 Volts 5%	WIMA	SMDTC04100TA00JS00
C66,C67,C71,C72,C73,C74	100uF	CP_Radial_D10.0mm_P5.00mm	100 µF 50 V Aluminum Electrolytic Capacitors Radial	United Chemi-Con	EGXE500ELL101MH12D
C27,C59,C68	2200uF	CP_Radial_D16.0mm_P7.50mm	LOW IMPEDANCE ELECTROLYTIC CAPACITORS 50VDC	Rubycon	50ZLJ2200M16X31.5
J1					
J2	N/A	Molex_KK_254_1X2	KK 254 Connector 2 Pin, 2.54mm	Molex	22-11-2032
J3	N/A	Molex_KK_254_1X3	KK 254 Connector 3 Pin, 2.54mm	Molex	22-11-2032
D3,D4,D5,D6	N/A	D_SMA	DIODE SCHOTTKY 60V 1A SMA	STMicroelectronics	STPS160A
D1,D2	N/A	Diode_Bridge_Vishay_KBL	Brückengleichrichter 4A 50V	Vishay	KBL005-E4/51
X1	16MHz	XO32-4Pin_3.2x2.5mm	Clock-Standardoszillatoren WE-SPXO 16.0MHz 25ppm	Wurth Elektronik	830205889709
U16,U17	N/A	DIP-8_W7.62mm	Fast ±150mA Power Buffer	Analog Devices	LT1010CN8#PBF
U20	N/A	MSOP-12-1EP_3x4mm_P0.65mm	20V, 500mA, Ultralow Noise, Ultrahigh PSRR Linear Regulator	Analog Devices	LT3045EMSE#PBF
U11,U14,U32	N/A	MSOP-8_3x3mm_P0.65mm	Precision, Ultralow Noise, RRIO, Zero-Drift Single Op Amp	Analog Devices	ADA4528-1ARMZ
U31	N/A	MSOP-8_3x3mm_P0.65mm	55V, EMI Enhanced, Zero Drift, Ultralow Noise Op Amp	Analog Devices	ADA4522-1ARMZ
U5,U6	100k : 100k	MSOP-8-1EP_3x3mm_P0.65mm	Quad Matched Resistor Network, 0.01% Matching, 0.2ppm/°C	Analog Devices	LT5400AHMS8E-2#PBF
U9,U10	10k : 100k	MSOP-8-1EP_3x3mm_P0.65mm	Quad Matched Resistor Network, 0.01% Matching, 0.2ppm/°C	Analog Devices	LT5400AHMS8E-3#PBF
U3	N/A	SOIC-14_3.9x8.7mm_P1.27mm	55V, EMI Enhanced, Zero Drift, Ultralow Noise, Quad Op Amp	Analog Devices	ADA4522-4ARZ
U25	N/A	SOIC-16W_7.5x10.3mm_P1.27mm	Quad-Channel Digital Isolators	Analog Devices	ADuM1411BRWZ
U26	N/A	SOIC-16W_7.5x10.3mm_P1.27mm	Quad-Channel Digital Isolators	Analog Devices	ADuM1410BRWZ
U13,U30	N/A	SOIC-8_3.9x4.9mm_P1.27mm	High Voltage, Low Noise Zero-Drift OpAmp	Analog Devices	LTC2057IS8#PBF
U23,U24	N/A	TSSOP-16_4.4x5mm_P0.65mm	8-stage serial shift register	Texas Instruments	CD4094BPW
U8,U27	N/A	TSSOP-16_4.4x5mm_P0.65mm	Quad SPST +70V Analog Switches	Maxim Integrated	MAX14756EUE+
U1	N/A	TSSOP-16_4.4x5mm_P0.65mm	Quad SPST +70V Analog Switches	Maxim Integrated	MAX14757EUE+
U21	N/A	TSSOP-24_4.4x7.8mm_P0.65mm	32-Bit, 10 kSPS, Sigma-Delta ADC	Analog Devices	AD7177-2BRUZ
Q1,Q2,Q3,Q4	N/A	SOT-23	JFET N-Channel Switch	onsemi	MMBF4117
U15	N/A	TO-252-2	Automotive 500-mA, 40-V, low-dropout linear regulator	Texas Instruments	TPS7B8833QKVURQ1
U19	N/A	TO-252-3	3-Terminal Adjustable Negative Regulator	Texas Instruments	LM337KVURG3
U12	N/A	TO-46	Oven-Compensated, Buried Zener, 7.05 V Voltage Reference	Analog Devices	ADR1399KHZ
U18	N/A	TO-252-4	3-Terminal Adjustable Positive Regulator	STMicroelectronics	LM317MDT
R1,R2,R3,R4,R5,R10,R11,R33,R34,R38,R39,R40,R41,R42,R43,R44,R45,R46,R47,R48,R49,R50,R51,R52,R53,R54,R55	1k	R_0603	0.1W 1Kohm 1% 0603 50ppm Automotive	Vishay / Beyschlag	MCT06030C1001FP500
R14,R15,R35,R37	10	R_0805	0.400W 10ohms 0.5% 25ppm	Vishay / BC Components	MCU0805PD1009DP500
R31	5,1	R_0805	5.1 Ohm 0.5% 1/8W 25ppm	YAGEO	RT0805DRD075R1L
R25,R26,R32,R36	2k	R_0805	4.32Kohms 0.1% 25ppm	Panasonic	ERA-6AEB4321V

R24	49.9k	R_0805	0.2W 49.9Kohms 0.1% 25ppm Auto	Vishay / Beyschlag	MCU0805MD4992BP100
R27,R28	280	R_0805	0805 240ohms 25ppm 0.5%	Panasonic	ERA-6AED241V
R6,R7	10M	R_1206	10M .25W 1% 100ppm	Bourns	CHV1206-FX-1005ELF
R16	200k	R_1206	200Kohms 0.1% 25ppm	Vishay / Dale	TNPW1206200KBEEA
R8,R9,R12,R13	1k	R_1206	1/4W 1Kohms .5% 1206 25ppm Auto	Vishay / Beyschlag	MCA1206MD1001DP500
R29,R30	10k	R_S102C	High Precision Foil Resistor with TCR 2ppm 0.1% 10k	Vishay Precision Group Foil Resistors	Y000710K0000B9L
R17	1.3k	R_S102C	High Precision Foil Resistor with TCR 2ppm 0.1% 1.3k	Vishay Precision Group Foil Resistors	Y00071K3000B9L
R22,R23	2k	R_S102K	High Precision Foil Resistor with TCR 1ppm 1% 2k	Vishay Precision Group Foil Resistors	Y00622K0000F9L
R18	420	R_S102K	High Precision Foil Resistor with TCR 1ppm 1% 420	Vishay Precision Group Foil Resistors	Y0062420R000F9L
R19	560	R_S102K	High Precision Foil Resistor with TCR 1ppm 1% 560	Vishay Precision Group Foil Resistors	Y0062560R000F9L
R20	2k	R_S102K	High Precision Foil Resistor with TCR 1ppm 1% 2k	Vishay Precision Group Foil Resistors	Y00622K0000F9L
R21	Defined at assembly				

Digital Board

Bauteile	Wert	Footprint	Beschreibung	Hersteller	Hersteller Nr.
C1, C2, C3, C4, C5, C6, C7, C9, C10, C11, C12, C18	100n	C0603K	10000 pF ±10% 50V Ceramic Capacitor X7R 0603 (1608 Metric)	Samsung Electro-Mechanics	CL10B103KB8WPNC
C13, C14, C22	10u	C0603K	SMD/SMT Multilayer Ceramic Capacitors MLCC - SMD/SMT 0805 25VDC 10uF 20% X6S	Taiyo Yuden	TMK212BC6106MG-T
C15, C19, C20	470u	CAPAE840X1050N	SMD Aluminium Electrolytic Capacitors - SMD 25VDC 470uF 20% Anti-Vibe AEC-Q200	Panasonic	EEE-FT1E471AV
C16, C17	27p	C0603K	C0805 27pF 5% C0G	muRata	GRM2165C1II270JZ01D
C8, C21	1u	C0603K	1 µF ±10% 25V Ceramic Capacitor X7R 0603 (1608 Metric)	Samsung Electro-Mechanics	CL10B105KA8VPNC
D1, D2, D3, D4	S1J-E3	DIOM4226X229N	Rectifiers Rectifiers 1.0 Amp 600 Volt 40A IFSM @ 8.3ms	Vishay General Semiconductor	S1J-E3/5AT
JP1	N/A	1X07	.1" Pinheader 1x7	Unbekannt	Unbekannt
JP2	N/A	1X04	.1" Pinheader 1x4	Unbekannt	Unbekannt
JP3	N/A	MA08-2	.1" Pinheader 2x8	Unbekannt	Unbekannt
JP4	N/A	1X02	.1" Pinheader 1x2	Unbekannt	Unbekannt
L1	10u	IFSC-1111	Fixed Inductors Fixed Inductors 10uH 20%	Vishay / Dale	IFSC1111ABER100M01
M1	N/A	5V_MODULE_UPSDWN	5V 3A-Peak StepDown DC-DC Module	Unbekannt	Unbekannt
R1	10k	R0603	10k 5%	Yageo	RC0805JR-0710K
R13, R14	27	R0603	27Ohm 5%	Yageo	RC0805JR-0727R
R2, R19	1k	R0603	1k 5%	Yageo	RC0805JR-071K
R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R15, R16, R17, R18	100	R0603	100Ohm 5%	Yageo	RC0805JR-07100R
SW1	N/A	SWITCH_1571563-4-M	Tactile Switch SPST-NO Top Actuated Surface Mount	TE Connectivity	1571563-2
U1	TC1264	SOT223-3	LDO Voltage Regulators LDO Voltage Regulators 800mA Fixed	Microchip	TC1264-3.3VDBTR
U2	RP2040	QFN-56	MCU RP2040 ARM Cortex-M0+	Raspberry Pi	RP2040TR
U3	W25Q	SOIC127P790X216-8N	NOR Flash NOR Flash spiFlash, 3V, 128M-bit, 4Kb Uniform Sector	Winbond	W25Q128JVSIQ
X1	N/A	IDC-Header_2x07_P2.54mm	HDR VERT DOUBLE 14P low profile	TE Connectivity	5103308-2
X2	N/A	W237-102	2 Pole Screw Terminal	Unbekannt	Unbekannt
XTAL1	12M	HC49UP	Crystals Crystals CSM-7X, 12MHz, 20pF, +/-30ppm, +/-50ppm, -40C +85 C	ECS	ECS-120-20-5PX-EN-TR

Mechanik

Bauteile	Wert	Footprint	Beschreibung	Hersteller	Hersteller Nr.
Gehäuse	N/A	N/A	Electronic Aluminium Instrument Case, 10.2 X 2.7 X 12.9 In	Bud Industries	IA-6161
Netztransformator 40V	40V	N/A	Power Transformer, 40Vct 48VA	Triad Magnetics	FP40-1200
Netztransformator 10V	10V	N/A	Power Transformer, 10Vct 48VA	Triad Magnetics	FP10-4800
Messbuchsen High	N/A	N/A	Gold Plated Tellurium Copper, Standard Binding Post, RED	Pomona Electronics	3770-2
Messbuchsen Low	N/A	N/A	Gold Plated Tellurium Copper, Standard Binding Post, BLACK	Pomona Electronics	3770-0

Eigenständigkeitserklärung

Der/die Unterzeichnende bestätigt mit seiner/ihrer Unterschrift, dass die Arbeit selbstständig verfasst und in schriftliche Form gebracht worden ist, dass sich die Mitwirkung anderer Personen auf Beratung und Korrekturlesen beschränkt hat und dass alle verwendeten Unterlagen und Gewährspersonen aufgeführt sind.

Der/die Unterzeichnende erlaubt, dass die Arbeit in anonymisierter Form elektronisch auf Plagiate überprüft wird.

Datum: 12.04.2022

Unterschrift/-en: L. Widmer A. Pascarmin V. Wyss