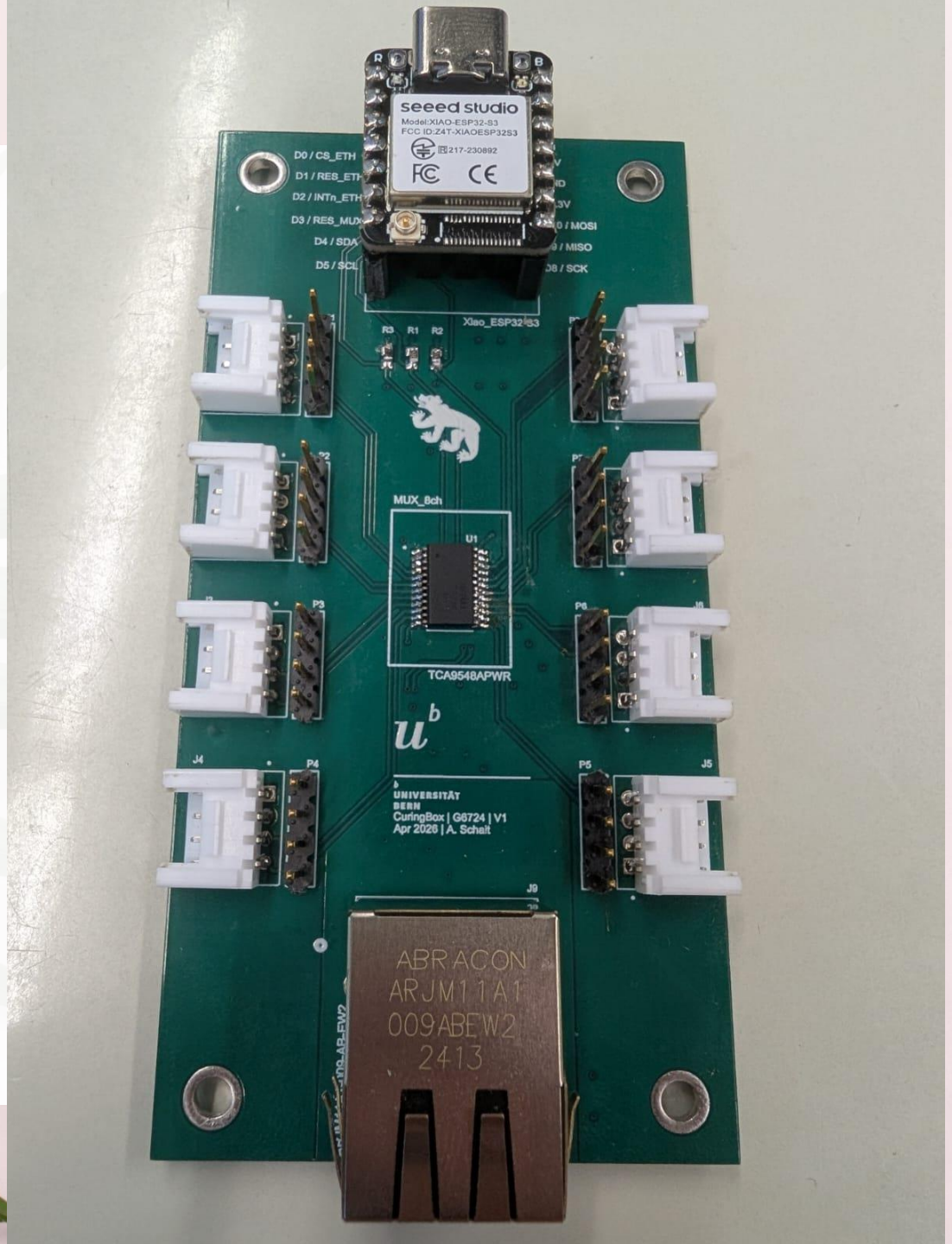
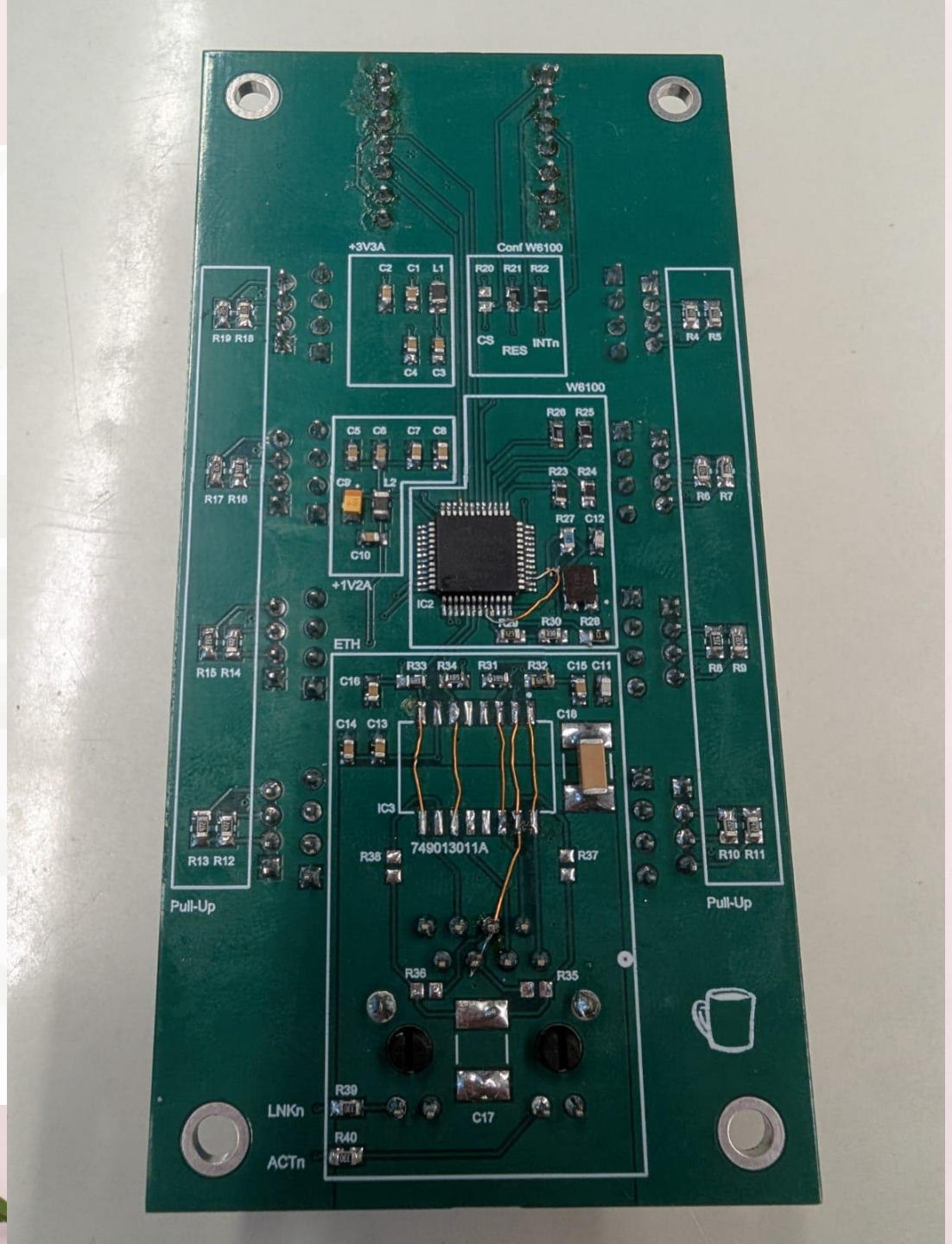
CuringBoxPCB_V0

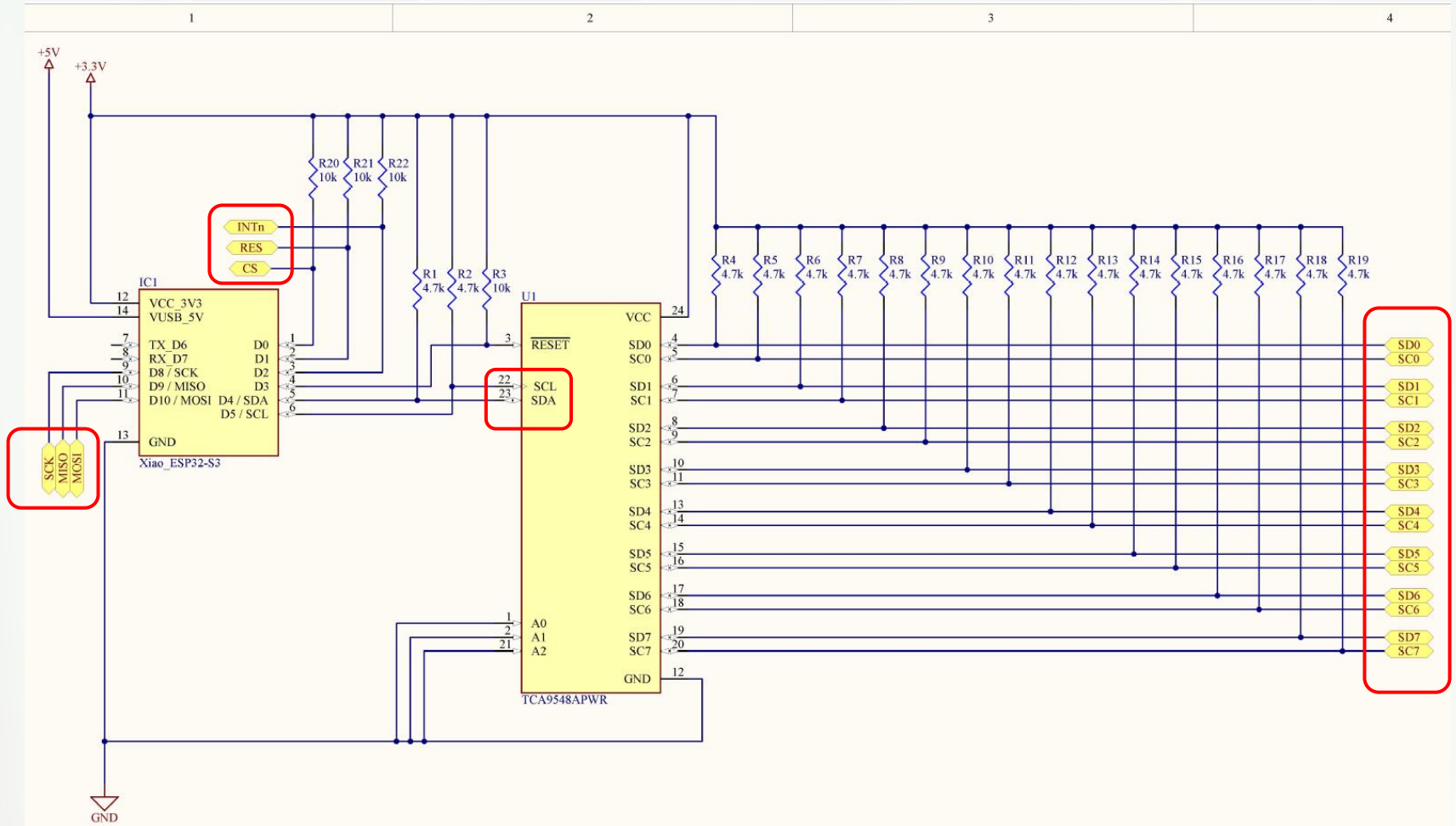
Table of contents

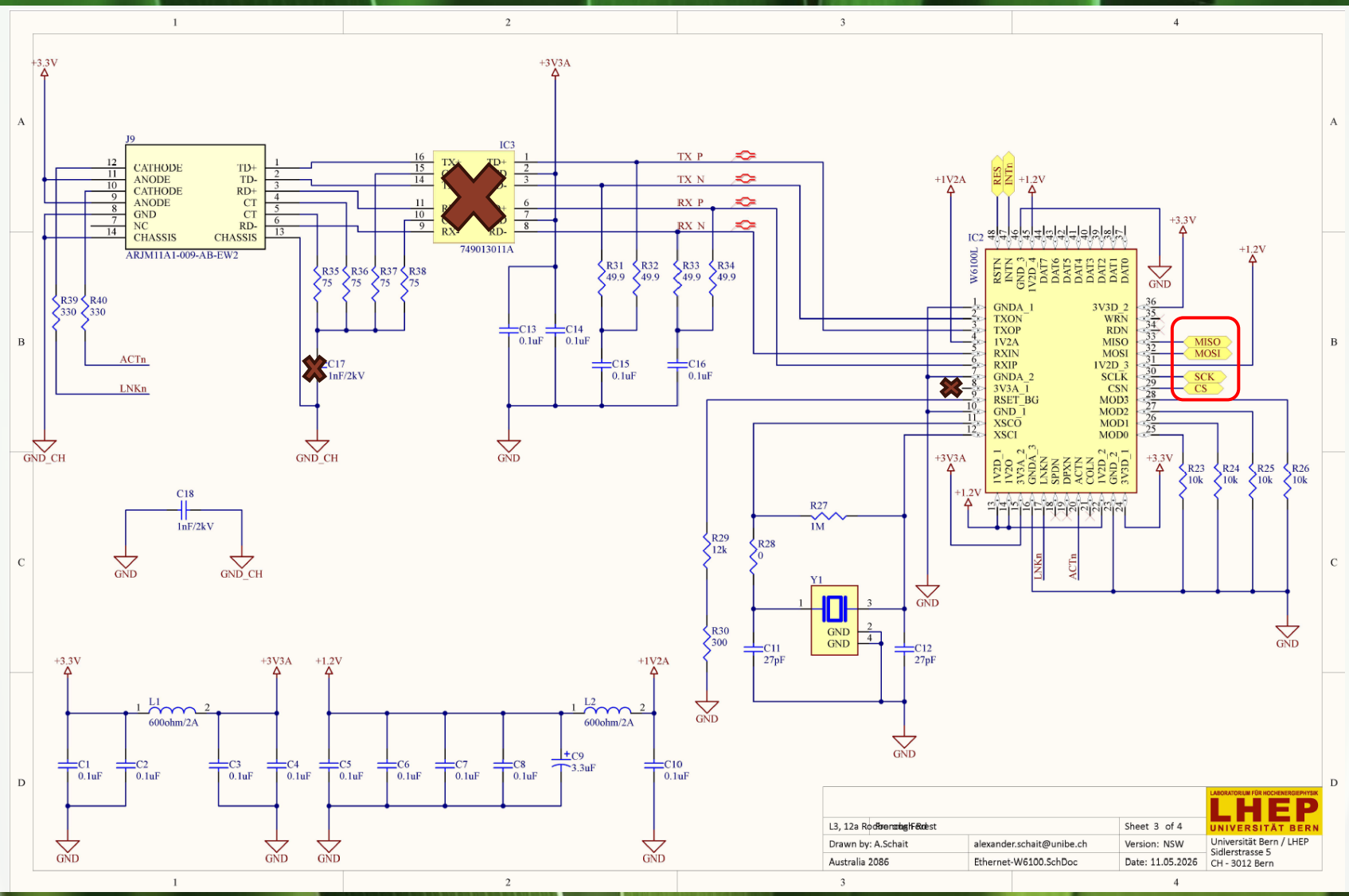
PCB_Top


seeed studio
Model:XIAO-ESP32-S3
FCC ID:Z4T-XIAOESP32S3
D0 / CS_ETH
D1 / RES_ETH
D2 / INTn_ETH
D3 / RES_MUX
D4 / SDA
D5 / SCL
MOSI
MISO
SCK
Xiao_ESP32S3
R3 R1 R2
MUX_8ch
U1
TCA9548APWR
P3
P4
P5
P6
J4
J5
J9
UNIVERSITÄT
BERN
CuringBox | G6724 | V1
Apr 2026 | A. Schalt
ABRACON
ARJM11A1
009ABEW2
2413

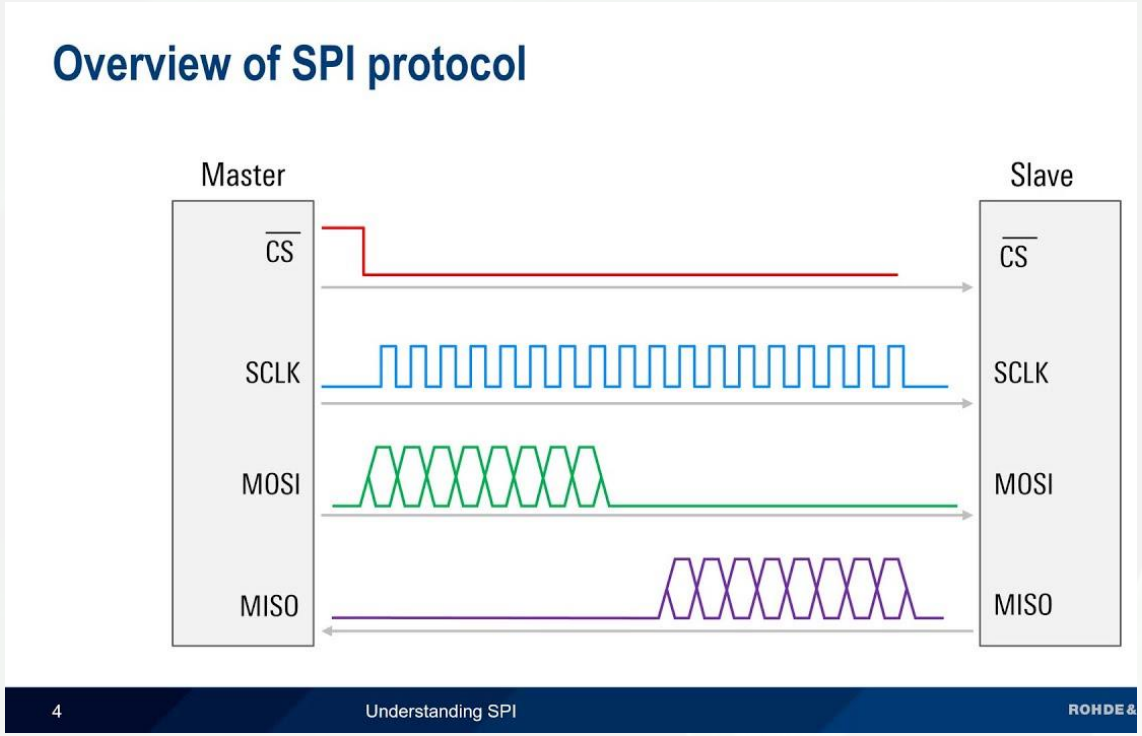
PCB_Bottom


+3V3A
Conf W6100
R20 R21 R22
CS
RES
INTn
C2 C1 L1
C4 C3
R19 R18
R4 R5
W6100
R26 R25
C5 C6 C7 C8
C9
L2
R23 R24
R17 R16
R6 R7
R27 C12
C10
+1V2A
IC2
R30 R28
ETH
R15 R14
R8 R9
C16
R33 R34 R31 R32 C15 C11
C18
C14 C13
IC3
749013011A
R38
R37
R13 R12
R10 R11
Pull-Up
Pull-Up
R36
R35
R39
C17
LNKn
R40
ACTn


1
2
3
4
+5V
+3.3V
R20 10k
R21 10k
R22 10k
INTn
RES
CS
R1 4.7k
R2 4.7k
R3 10k
R4 4.7k
R5 4.7k
R6 4.7k
R7 4.7k
R8 4.7k
R9 4.7k
R10 4.7k
R11 4.7k
R12 4.7k
R13 4.7k
R14 4.7k
R15 4.7k
R16 4.7k
R17 4.7k
R18 4.7k
R19 4.7k
IC1
VCC_3V3
VUSB_5V
TX_D6
RX_D7
D8 / SCK
D9 / MISO
D10 / MOSI
D0
D1
D2
D3
D4 / SDA
D5 / SCL
GND
Xiao_ESP32-S3
SCK
MISO
MOSI
U1
VCC
RESET
SCL
SDA
SD0
SC0
SD1
SC1
SD2
SC2
SD3
SC3
SD4
SC4
SD5
SC5
SD6
SC6
SD7
SC7
A0
A1
A2
GND
TCA9548APWR
GND


J9
ARJM11A1-009-AB-EW2
IC3
749013011A
IC2
W6100L
TX_P
TX_N
RX_P
RX_N
ACTn
LNKn
MISO
MOSI
SCK
CS
RES
INTn
Y1
L1
600ohm/2A
L2
600ohm/2A
+3.3V
+3V3A
+1.2V
+1V2A
GND
GND_CH
Sheet 3 of 4
Drawn by: A.Schait
alexander.schait@unibe.ch
Version: NSW
Australia 2086
Ethernet-W6100.SchDoc
Date: 11.05.2026
LHEP
UNIVERSITÄT BERN
Universität Bern / LHEP
Sidlerstrasse 5
CH - 3012 Bern

SPI – Serial Peripheral Interface



CS -> Chip Select / Slave Select

SCLK -> Serial Clock

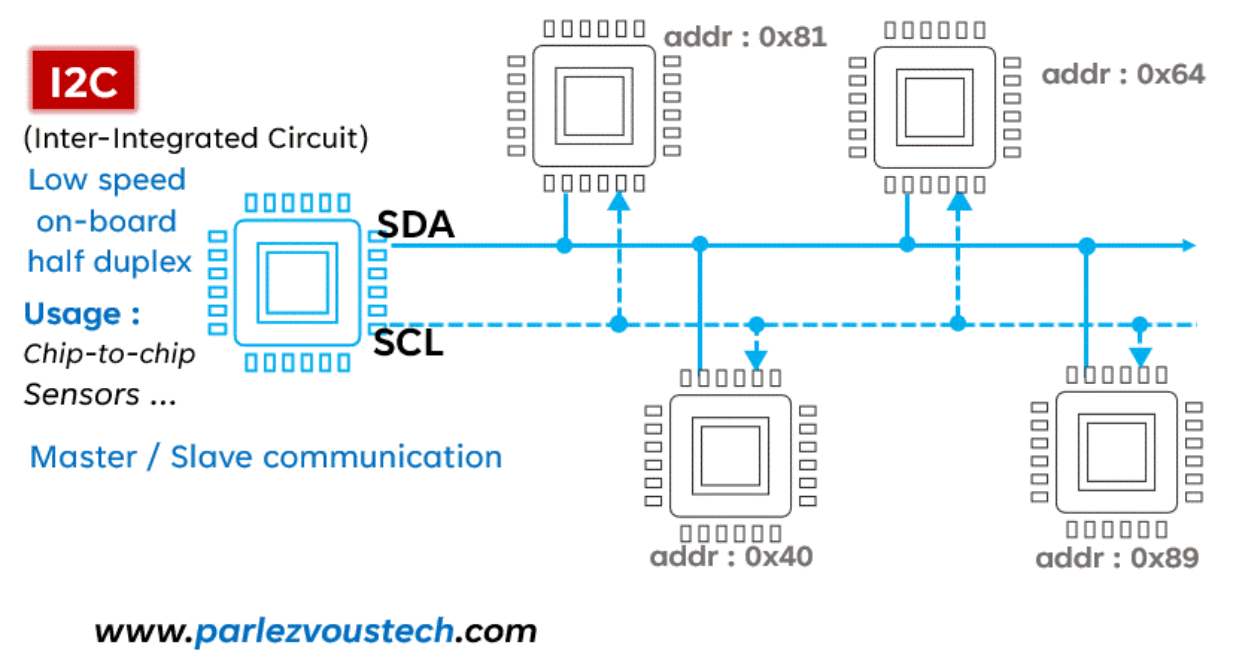
MOSI -> Master Out Slave In

MISO -> Master In Slave Out

High data transfer rates [MHz] on short distance [PCB]

Full Duplex (simultaneous send & receive)

I2C – Inter Integrated Circuit



SCL -> Serial Clock

SDA -> Serial Data

Low data transfer rates [kHz] on longer distance than SPI

BUT distance of cable is restricted by physical capacitance

MUX: isolates the capacitance of each sensor branch

```arduino
#include "config.h"
#include "types.h"
#include "sensors.h"
#include "network.h"
#include "utils.h"

Measurements data;

void setup()
{
    initSerial();
    initEthernet();
    initSensors();
    printHeader("SYSTEM READY");
}

void loop()
{
    readAllSensors(data);
    printMeasurements(data);
    sendToInfluxDB(data);
    delay(10000);
}
```

Arduino-Code

What to change to add a new sensor?

- initSensors()
- All 3 functions in loop()

```
/*
==========================================================
  Sensor Implementation
==========================================================
*/

#include "sensors.h"
#include "config.h"

#include <Arduino.h>
#include <Wire.h>

#include "TCA9548.h"
#include "DFRobot_MultiGasSensor.h"
#include "Seeed_BME280.h"
#include "MODULE_4_20MA.h"

// ==========================================================
// SENSOR OBJECTS
// ==========================================================
// I2C multiplexer
TCA9548 tca(0x70);

// Gas sensor
DFRobot_GAS_I2C gasSensor(&Wire, 0x74);

// Environmental sensor
BME280 bme280;

// M5Stack U162 4-20mA receiver
MODULE_4_20MA ain20ma;
```

Define newSensor

- Install & add library (library manager/GitHub)
- Define & find I2C address
- Initialize communication with sensor

Send data to InfluxDB

```
void sendToInfluxDB(const Measurements& data)
{
    Serial.println("[InfluxDB] Connecting...");

    if (!influxClient.connect(Config::INFLUX_HOST, Config::INFLUX_PORT))
    {
        Serial.println("[InfluxDB] Connection failed!");
        return;
    }

    String payload = createPayload(data);

    String url = "/api/v2/write?org=" + String(Config::INFLUX_ORG) + "&bucket=" +
String(Config::INFLUX_BUCKET) + "&precision=s";

    influxClient.println("POST " + url + " HTTP/1.1");
    influxClient.println("Host: " + String(Config::INFLUX_HOST));
    influxClient.println("Authorization: Token " + String(Config::INFLUX_TOKEN));
    influxClient.println("Content-Type: text/plain");
    influxClient.print("Content-Length: ");
    influxClient.println(payload.length());
    influxClient.println();
    influxClient.println(payload);

    influxClient.stop();
    Serial.println("[InfluxDB] Done.");
}
```

```
String createPayload(const Measurements& data)
{
    String payload = "environment";
    payload += ",device=xiao_esp32s3";
    payload += ",location=curing_box";
    payload += " gas_ppm=" + String(data.gasPPM);
    payload += ",temperature=" + String(data.temperature);
    payload += ",humidity=" + String(data.humidity);
    payload += ",pressure=" + String(data.pressure);
    payload += ",oxygen_percent=" + String(data.oxygenPercent);
    payload += ",oxygen_current=" + String(data.oxygenCurrent);

    return payload;
}
```