

# Workshop

# Arduino-Programmierung #8

SPI, TFT-Display, Adafruit\_GFX, EEPROM

Joachim Baur

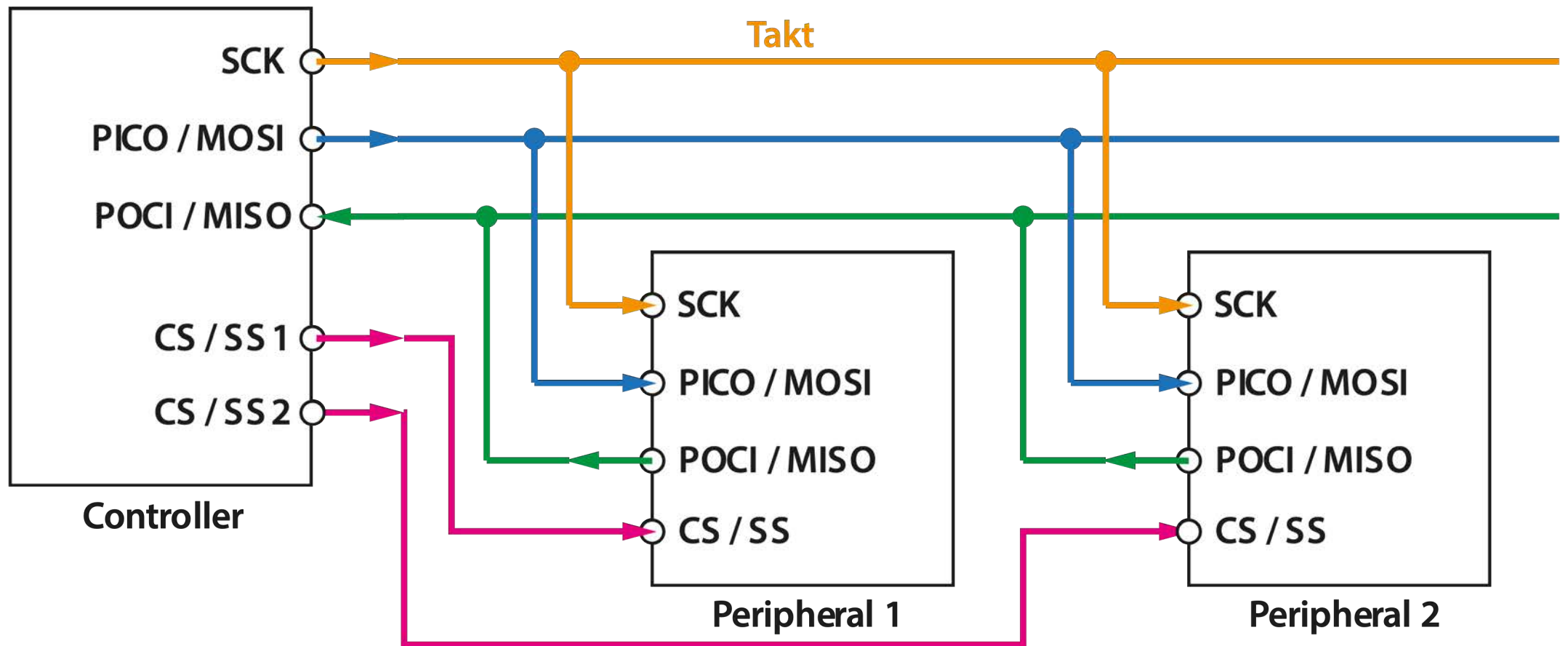
E-Mail: [post@joachimbaur.de](mailto:post@joachimbaur.de)

ZTL-Alias: [@joachimbaur](https://twitter.com/joachimbaur)

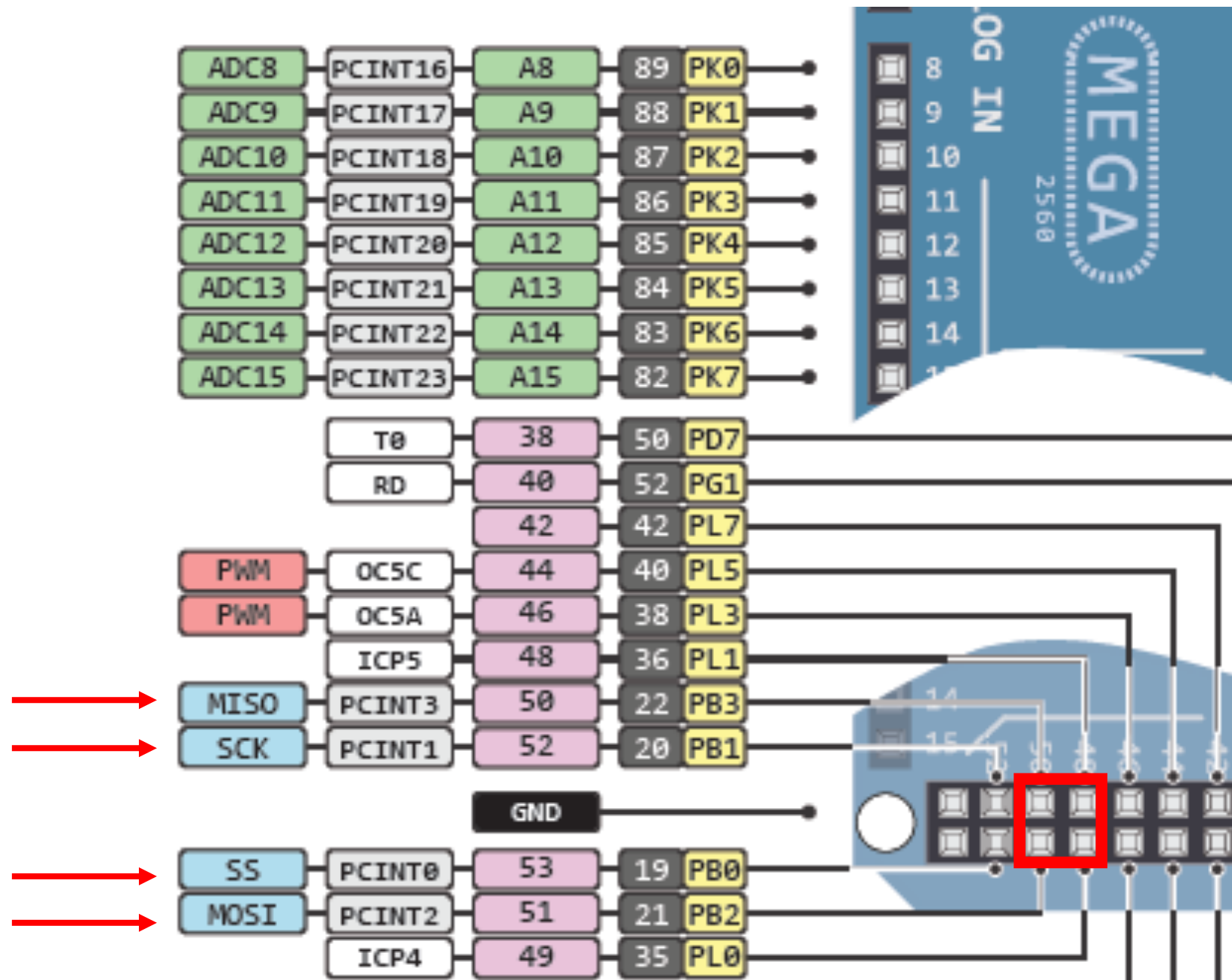
Download für diesen Workshop: [www.joachimbaur.de/WS8.zip](http://www.joachimbaur.de/WS8.zip)

# SPI-Datenbus

- SPI („Serial Peripheral Interface“) ist ebenfalls ein serieller Datenbus, im Gegensatz zu I2C werden aber separate Sende- und Empfangsleitungen benutzt. Auch die Übertragungs-Geschwindigkeit kann viel höher eingestellt werden.
- Es gibt ebenfalls (nur) einen Controller im SPI-Verbund, aber im Gegensatz zu I2C keine Adressen für die Peripherals
- Über eine separate Leitung kann jedes Peripheral aktiviert oder deaktiviert werden, so ist es trotzdem möglich, mehrere Peripherals am selben SPI-Bus zu betreiben (bei gleichem Takt!)
- Leitung SCK = Serial Clock, Systemtakt vom Controller
- Leitung POCI / MISO = Peri Out, Controller In
- Leitung PICO / MOSI = Peri In, Controller Out
- Leitung CS / SS = Chip Select, Peri (de-)aktivieren
- Senden und empfangen erfolgt in 1 Vorgang (zB 4 Byte senden -> 4 Byte empfangen), falls das Gerät das Senden unterstützt (POCI-Leitung)



# Mega Hardware-SPI Pins




## 2.4" TFT Display

- Wir verwenden ein 2,4 Inch 240x320 SPI TFT, dieses hat als Displaytreiber einen **ST7789V** Chip verbaut
- Das **Datenblatt** zu dem Treiberchip beschreibt die genaue Ansteuerung des Displays (ein Treiber kann mehrere Displaybauarten ansteuern, zB mit verschiedenen Auflösungen / Farbtiefen / Schnittstellen )
- Am einfachsten ist der Einsatz einer fertigen Bibliothek: "**Adafruit ST7735 und ST7789 Library**"
- Bitte installieren! Zudem "**Adafruit GFX Library**"
- Dann im Menü Beispiele -> "Adafruit ST7735 und ST7789 Library" -> "**graphicstest\_7789**" öffnen

# Sketch "graphicstest\_7789" anpassen

```
74
75 // Use this initializer (uncomment) if using a 1.3" or 1.54" 240x240 TFT:
76 tft.init(240, 240); // Init ST7789 240x240
77
78 // OR use this initializer (uncomment) if using a 1.69" 280x240 TFT:
79 //tft.init(240, 280); // Init ST7789 280x240
80
81 // OR use this initializer (uncomment) if using a 2.0" 320x240 TFT:
82 //tft.init(240, 320); // Init ST7789 320x240
83
```



---

```
74
75 // Use this initializer (uncomment) if using a 1.3" or 1.54" 240x240 TFT:
76 //tft.init(240, 240); // Init ST7789 240x240
77
78 // OR use this initializer (uncomment) if using a 1.69" 280x240 TFT:
79 //tft.init(240, 280); // Init ST7789 280x240
80
81 // OR use this initializer (uncomment) if using a 2.0" 320x240 TFT:
82 tft.init(240, 320); // Init ST7789 320x240
83
```

# Display am Mega anschließen

\*Pin Nr. im Sketch definiert

```

48 // these pins will also w
49 #define TFT_CS      10
50 #define TFT_RST      9
51 #define TFT_DC      8
--

```

	-	T_IRQ
	-	T_DO
	-	T_DIN
	-	T_CS
	-	T_CLK
SPI MISO	50	SDO (MISO)
Hintergrund-Beleuchtung	3V3	LED
SPI CLOCK	52	SCK
SPI MOSI	51	SDI (MOSI)
Data/Command*	8	DC
Reset*	9	RESET
SPI Chip Select*	10	CS
GND	GND	GND
Versorgungs-Spannung	3V3	VCC



# Logic Level Shifter

- Arduino Mega GPIO: 5V
- TFT pins: 3,3V (= „3V3“)
- Bi-Direktionaler Logic Level Shifter TXS0108E

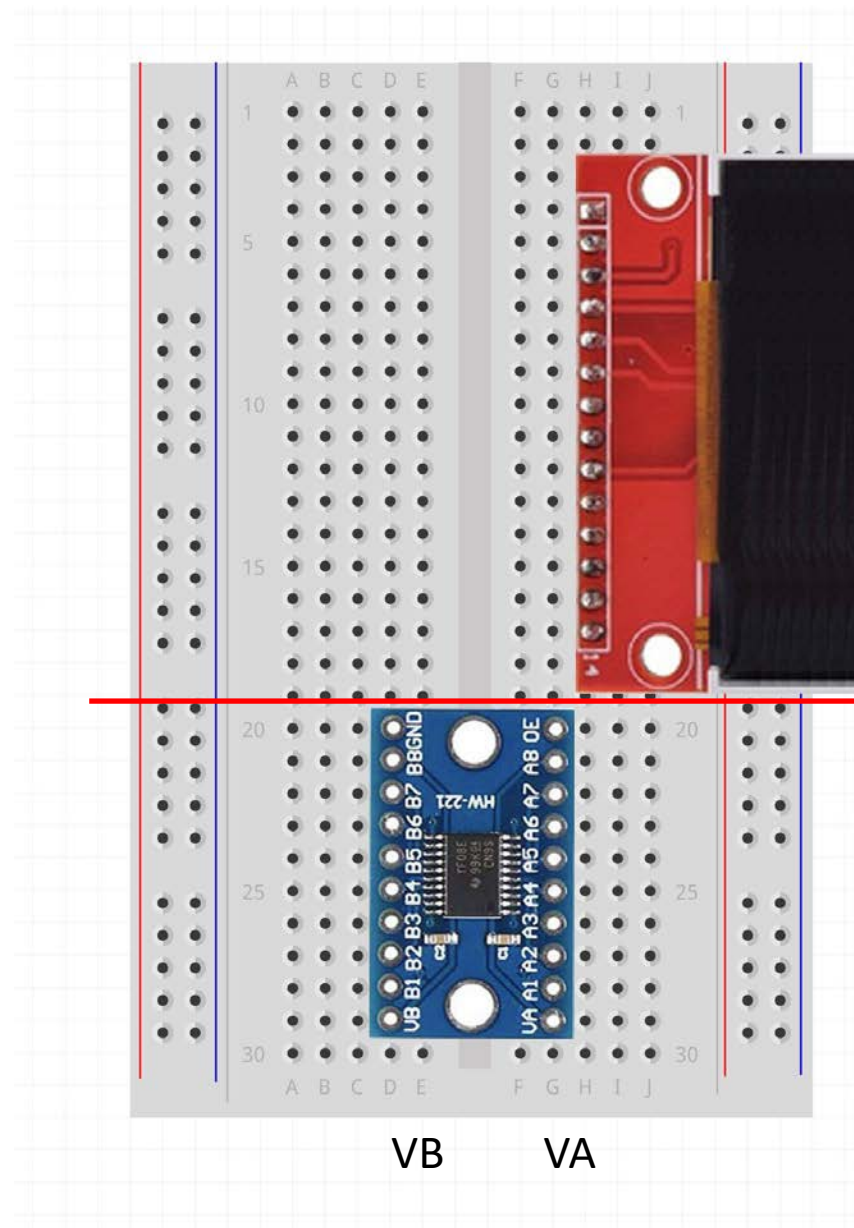
**VB: 5V**      B1...B8 = Leitungen      GND



**VA: 3V3**      A1...A8 = Leitungen      OE: Output Enable bei +5V



# TFT mit Level Shifter anschließen - 1



## TFT rechts einstecken

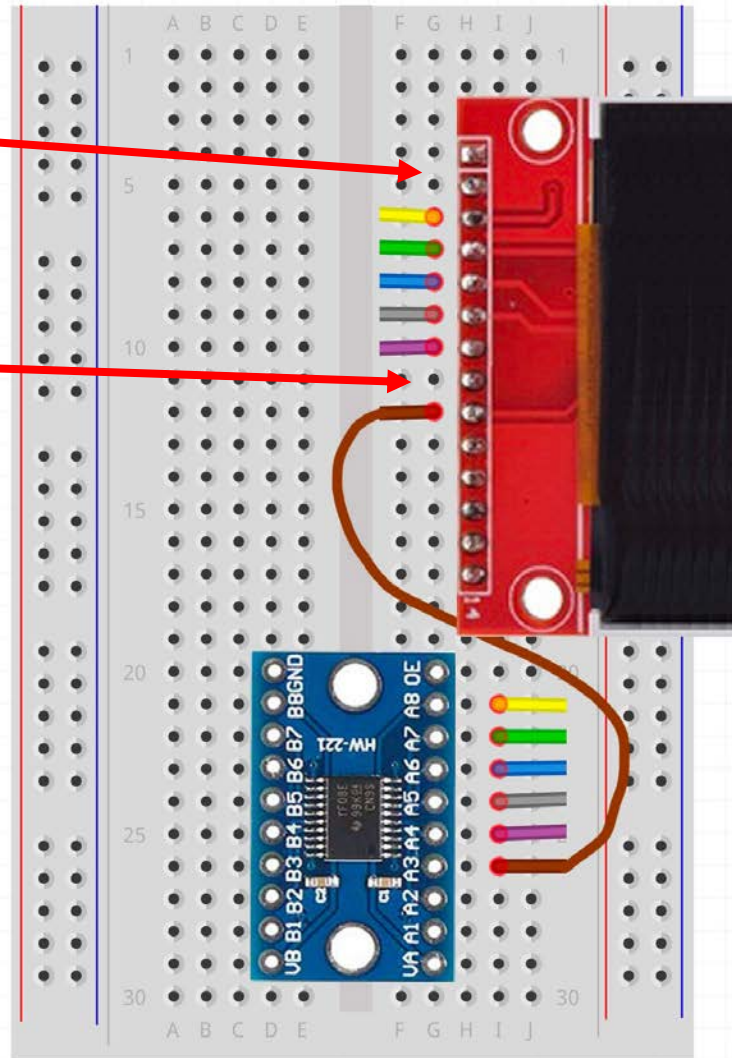
Keine Überlappung  
der Pins!

Level Shifter  
In der Mitte einstecken  
VB nach links

# TFT mit Level Shifter anschließen - 2

Erste 2 Pins leer lassen

1 Pin auslassen



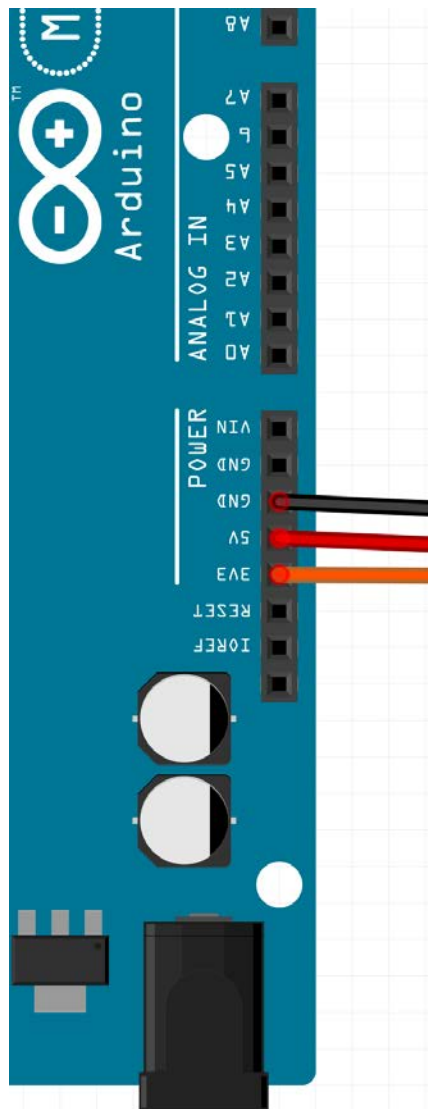
6x TFT Pins  
mit Level Shifter (Seite VA)  
verbinden

VB

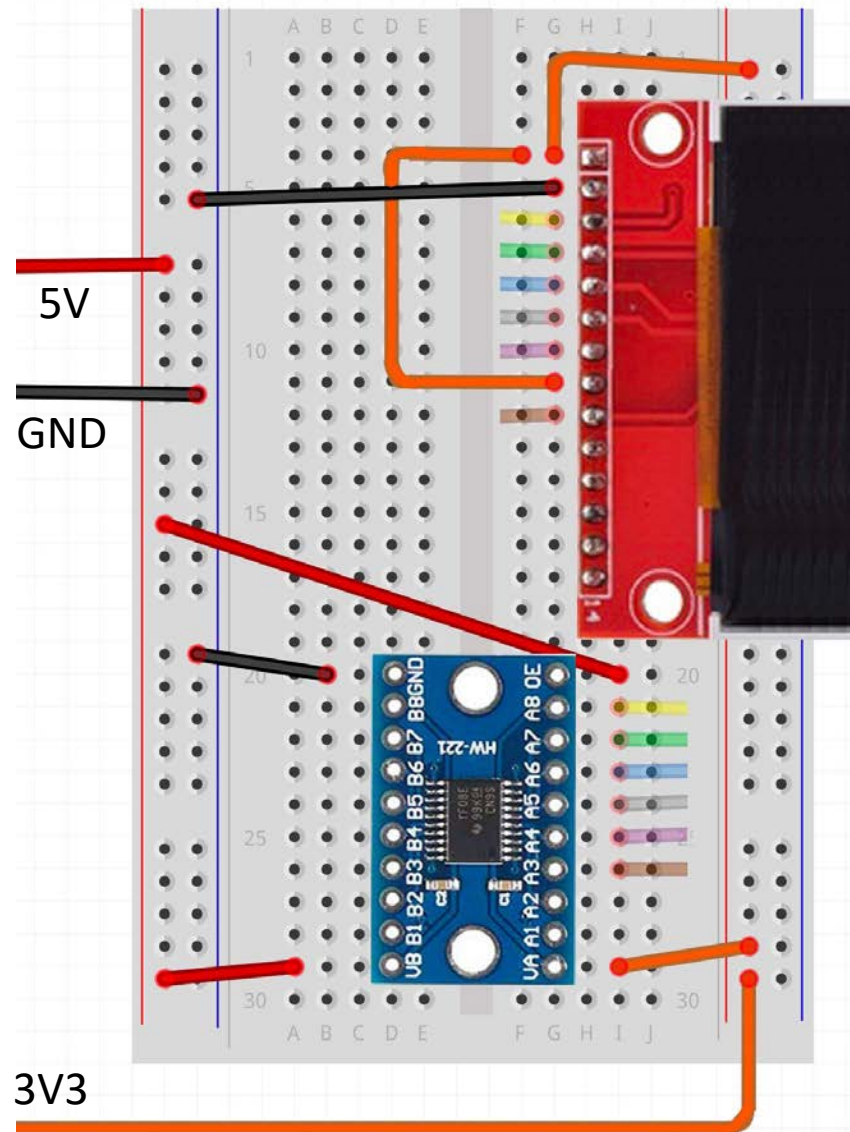
VA

# TFT mit Level Shifter anschließen - 3

## Stromversorgung verbinden



GND  
5V  
3V3



3V3 zu  
TFT VCC  
TFT LED  
Level Shifter VA

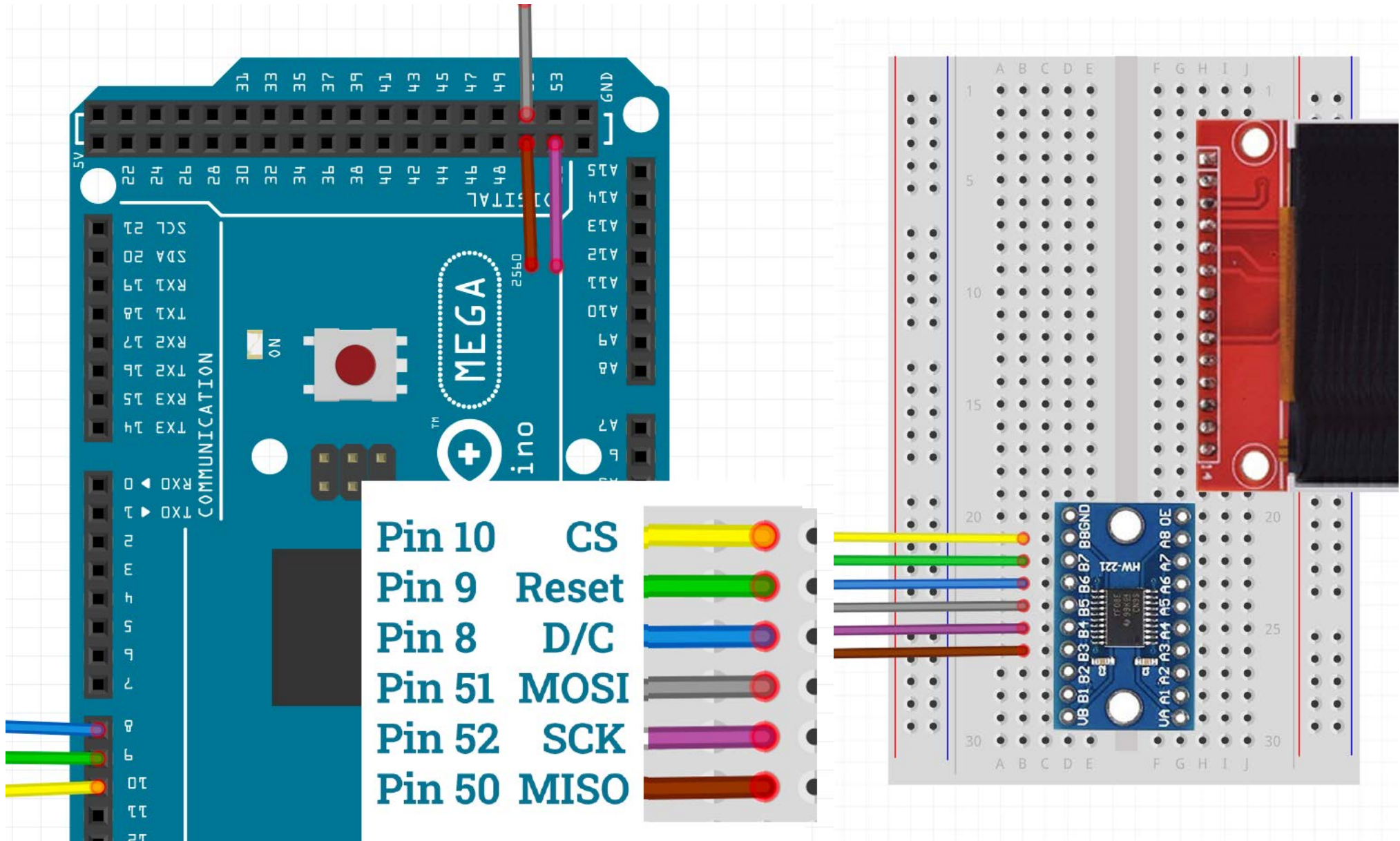
5V zu  
Level Shifter VB  
Level Shifter OE

GND zu  
TFT GND  
Level Shifter GND



# TFT mit Level Shifter anschließen - 4

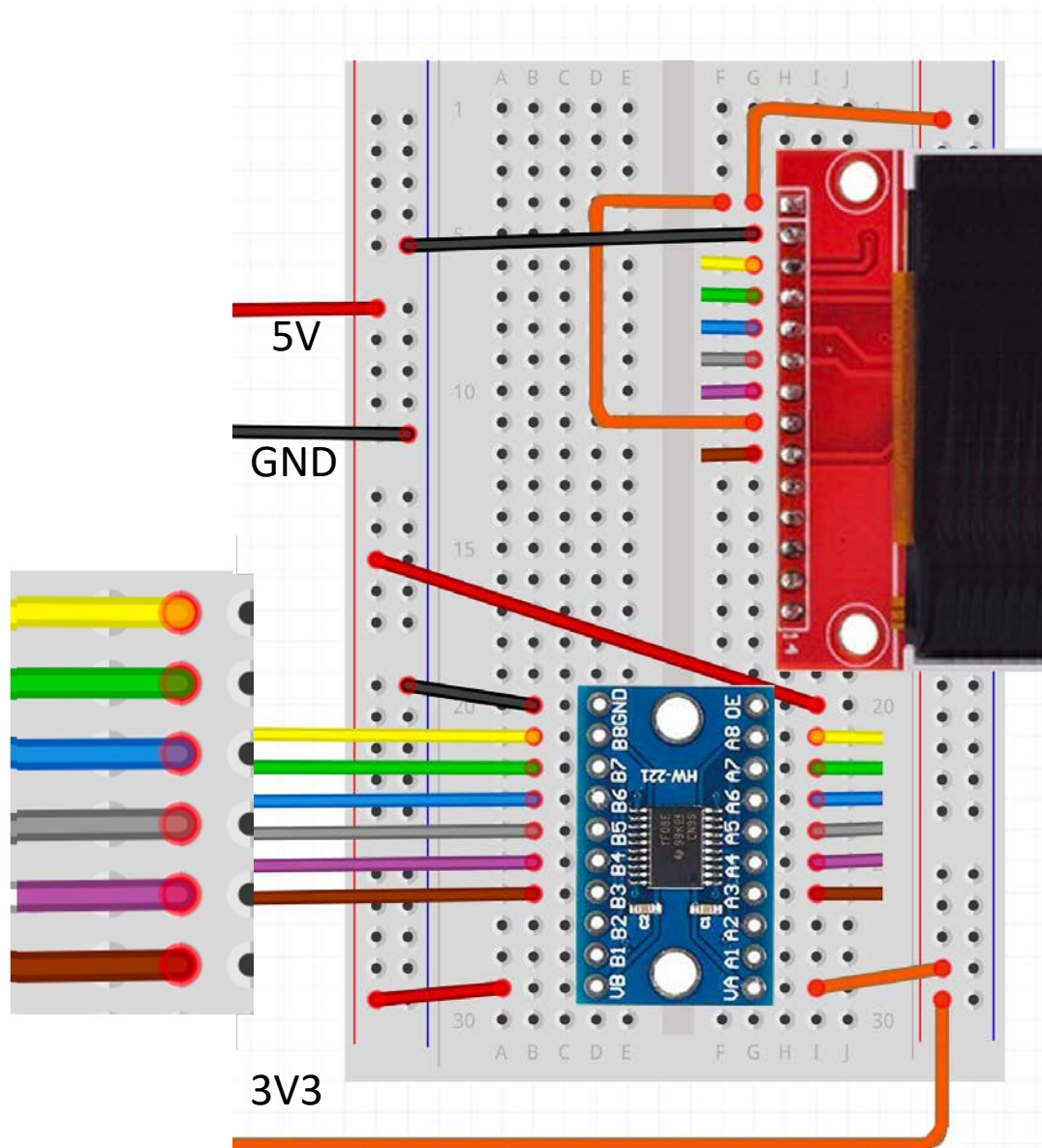
6x Level Shifter Pins (Seite VB) mit Mega verbinden



# TFT mit Level Shifter anschließen - 5

## Gesamtansicht Verkabelung

Pin 10 CS  
Pin 9 Reset  
Pin 8 D/C  
Pin 51 MOSI  
Pin 52 SCK  
Pin 50 MISO



# Adafruit Graphics Library

```
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_ST7789.h> // Hardware-specific library for ST7789
#include <SPI.h>
```

<https://learn.adafruit.com/adafruit-gfx-graphics-library/overview>

```
#define TFT_CS      10
#define TFT_RST      9
#define TFT_DC      8
```

```
Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_RST);
```

```
tft.init(breite, hoehe);
// was Breite und was Hoehe ist, ist vom Display abhängig
```

```
tft.fillScreen(farbe); // ab hier GFX Befehle
```

```
tft.drawPixel(x, y, farbe);
tft.drawLine(x1, y1, x2, y2, farbe);
tft.drawFastHLine(x, y, laenge, farbe);
tft.drawFastVLine(x, y, laenge, farbe);
```

```
tft.drawRect(x, y, breite, hoehe, farbe); // auch fillRect
tft.drawRoundRect(x, y, breite, hoehe, eckenradius, farbe); // fillRoundRect
tft.drawCircle(x, y, radius, farbe); // fillCircle
tft.drawTriangle(x1, y1, x2, y2, x3, y3, farbe); // fillTriangle
```

```
tft.setCursor(x, y);
tft.setTextColor(farbe);
tft.setTextSize(schriftgroesse); // multiplikator
tft.setTextWrap(true); // Automatischer Zeilenumbruch
tft.print(text);
tft.println(text);
```

```
#define ST77XX_BLACK    0x0000
#define ST77XX_WHITE    0xFFFF
#define ST77XX_RED      0xF800
#define ST77XX_GREEN    0x07E0
#define ST77XX_BLUE     0x001F
#define ST77XX_CYAN     0x07FF
#define ST77XX_MAGENTA  0xF81F
#define ST77XX_YELLOW   0xFFE0
#define ST77XX_ORANGE   0xFC00
```

**tft.invertDisplay(false); WICHTIG für Aliexpress TFT! Sonst falsche Farben...**

# RGB565

- Um Speicherplatz zu sparen und die Übertragungsgeschwindigkeit zu erhöhen, werden die Farben für das TFT-Display nicht mit 8 Bit je Farbkomponente (= 24 Bit, RGB24 bzw. RGB888) übertragen, sondern gekürzt und zusammengesetzt zu einer insgesamt 16 Bit langen Zahl (= RGB565)



ROT = 8 Bit (256 Werte 0...255, in 1er-Schritten)



ROT = 5 Bit (32 Werte 0...248, in 8er-Schritten)



```
int rgb565( int r, int g, int b ) {  
    return ((r / 8) << 11) + ((g / 4) << 5) + (b / 8);  
}
```



Aufgabe: Badge zeichnen



## Sketch: 41\_Badge.ino

```
#include <Adafruit_GFX.h>
#include <Adafruit_ST7789.h>
#include <SPI.h>

#define TFT_CS 10
#define TFT_RST 9
#define TFT_DC 8

Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_RST);

void setup() {

  tft.init(240, 320); // Init ST7789 320x240
  tft.invertDisplay(false);

  tft.fillScreen(ST77XX_BLACK);
  tft.fillCircle(120, 120, 100, ST77XX_YELLOW ); // Kopf
  tft.drawCircle(120, 120, 60, ST77XX_BLACK ); // Smile
  tft.fillRect(45, 55, 150, 100, ST77XX_YELLOW ); // Smile
  tft.fillCircle(70, 80, 10, ST77XX_BLACK ); // Auge links
  tft.fillCircle(170, 80, 10, ST77XX_BLACK ); // Auge rechts
  // NAME
  tft.setCursor( 40, 270 );
  tft.setTextSize(4);
  tft.setTextColor( ST77XX_WHITE );
  tft.println( "JOACHIM" );

}
```

# EEPROM

- Ein EEPROM ist ein Speicher auf dem Mega, der vom Programmcode selbst beschrieben und wieder ausgelesen werden kann
- Beim Mega ist das EEPROM 4 KB groß (4096 Byte)
- Die EEPROM-Speicherzellen sind für ca. 100.000 **Schreib**vorgänge ausgelegt (Auslesen beliebig oft)

```
#include <EEPROM.h>
```

```
int adresse = 0; // 0...4095 beim Mega
```

```
char wert = 'A'; // oder int bzw. byte
```

```
// Wert in EEPROM schreiben
```

```
EEPROM.update( adresse, wert ); // nicht EEPROM.write()!
```

```
// Wert aus EEPROM lesen
```

```
byte gelesen = EEPROM.read( adresse ); // 0x65 ('A')
```

# Name in EEPROM speichern

- Name für Badge soll per Serieller Konsole dauerhaft geändert werden können
- Beim Programmstart EEPROM auslesen und gespeicherten Namen anzeigen
- Bei Eingabe in Serieller Konsole die Eingabe im EEPROM speichern und anzeigen
- **PROBLEM:** Beim ersten Programmlauf steht noch kein Name im EEPROM
- Deshalb eine Kennung + den eigentlichen Namen speichern



## Sketch: 42\_EEPROM.ino

```
#include <EEPROM.h>

String defaultName = "Joachim";
String name = ""; // wird aus EEPROM gelesen

int adresse = 0;
char kennung[] = "NAME";

String eingabe = "";

void setup() {
    Serial.begin(115200);

    name = leseNameEEPROM();
    Serial.println("Name aus EEPROM: " + name);
}

void loop() {
    eingabe = "";
    while (Serial.available() > 0) {
        eingabe += char(Serial.read());
        delay(1);
    }
    if (eingabe != "") {
        speichereNameEEPROM(eingabe);
        // zur Kontrolle: lesen und ausgeben
        name = leseNameEEPROM();
        Serial.println("Name aus EEPROM: " + name);
    }
}
```

```
int adresse = 0;
char kennung[] = "NAME";

void speichereNameEEPROM( String eingabe ) {

    int eepromIndex = adresse; // ab Speicherstelle adresse schreiben

    // zuerst die Kennung schreiben
    for (int i=0; i<4; i++) {
        EEPROM.update( eepromIndex++, (byte)kennung[i] );
    }

    // dann den Eingabe-String
    for (int i=0; i<eingabe.length(); i++) {
        EEPROM.update( eepromIndex++, eingabe.charAt(i) );
    }

    // zuletzt eine Null als Zeichen für das String-Ende
    EEPROM.update( eepromIndex++, 0 );
}
```

```
int adresse = 0;
char kennung[] = "NAME";
```

```
String leseNameEEPROM() {
    String gelesen = "";

    int eepromIndex = adresse; // ab Speicherstelle adresse lesen

    // Stimmt die Kennung (= haben wir bereits einen Namen gespeichert)?
    for (int i=0; i<4; i++) {
        if (EEPROM.read( eepromIndex++ ) != kennung[i] ) {
            return defaultName; // kennung stimmt nicht, default name zurückgeben
        }
    }

    // Kennung stimmt, dann jetzt Name lesen
    // wir wissen die Länge nicht, also lesen wir bis eine 0 auftaucht
    while( EEPROM.read( eepromIndex ) != 0 ) {
        gelesen += char(EEPROM.read( eepromIndex ));
        eepromIndex++;
    }

    return gelesen;
}
```

# EEPROM-Code in Badge-Code integrieren

Sketch: 43\_Badge\_EEPROM.ino



# Wie geht's weiter?

## Empfehlung Hardware:

- Arduino-Kits von **Elegoo** (mit Anleitungen auf CD, 20-70 €)  
<https://www.elegoo.com/en-de/collections/arduino-learning-sets>

Arduino-Kits von **Funduino** (40-70€)

<https://funduinoshop.com/funduino-starter-kits/>

- Calliope Mini Rev2 (für Kids, 30-40 €) Touch, Sound, LEDs onboard  
<https://calliope.cc/en>

## Empfehlung Buch:

- Pollux-Labs Verlag: "Vom Anfänger zum Maker" (9,90 €)  
<https://polluxlabs.net/e-book-vom-anfaenger-zum-maker-mit-dem-arduino-esp8266/>

## Online-Simulator für Arduino, ESP32 etc:

- <https://wokwi.com/>

## Fritzing (8 €) für (einfache) Diagramme, Schaltpläne, Platinen:

- <https://fritzing.org/>

## PlatformIO Online-IDE

- <https://platformio.org/platformio-ide>

