Arduino und die Displays

Ursprünglich war die digitale Welt auf die Pegel +5V und 0V eingestellt. Und so hatte man auch den Arduino entwickelt. Bald kamen aber hoch integrierte ICs auf. Man konnte die Verlustwärme vermindern, wenn man die Schaltungen nur mit 3V betrieb. Und so stehen wir heute vor einem Problem:

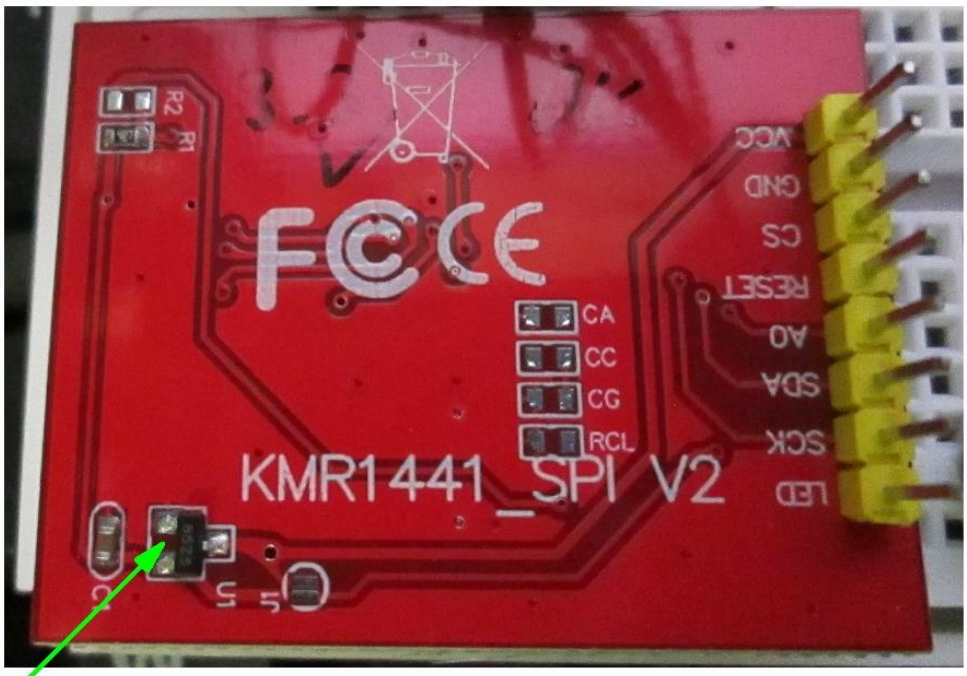
Schaltet man die modernen Displays direkt an die Datenausgänge des Arduinos, dann kann es passieren, dass die sofort defekt werden. Denn sie vertragen die hohen Pegel von 5V nicht. Man muss also eine Wandlung vornehmen. Das kann man mit Spannungsteilern mit Widerständen machen. Um aber auch Signale einer Rückmeldung von einem Modul aufzunehmen und dabei auf 5V/0V angewiesen ist, muss man auch „rückwärts“ wandeln. Es gibt solche undirektionalen Pegelwandler zu kaufen. Sie arbeiten mit MOS-Fets und einigen Widerständen. In China kosten sie einen Bruchteil im Vergleich zu unseren Preisen.

Warnung

Wer unbeabsichtigt den Arduino ohne Pegelwandler an ein Display anschließt, muss mit der Zerstörung des Displays rechnen. Der Händler freut sich.

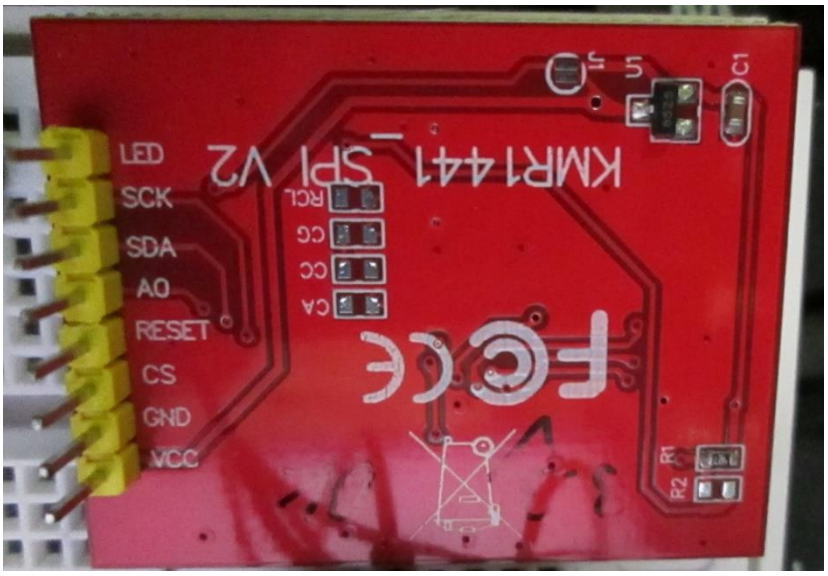
Es gibt aber auch noch Displays, die 5V vertragen. Woran erkennt man denn diejenigen für 3V-Betrieb?

Das folgende Bild zeigt den winzigen Baustein, der die +5V Betriebsspannung vom Arduino auf +3.3V m. Das Ansteuer-IC vom Display wird hier versorgt.



hier ist der Spannungswandler für den Betrieb mit 3.3v

Da nun das IC mit 3.3V betrieben wird, verlangt es auch diese Pegel an den Steuerleitungen. Das nächste Bild zeigt die Signaleingänge.



Dazu müssen wir eine Pegelwandler für die wichtigsten Logigpegel vorschalten.

Oben schließen wir den Arduino an, unten das Display. Die Converter haben jeweils eine eigene Stromversorgung. Oben +5V und GND, unten +3.3V und GND. Die 3.3V kann man vom PIN 3.3Ref des Arduino abnehmen. Ebenso die Betriebsspannung für die Hintergrundbeleuchtung: LED. Sie ist in den meisten Fällen ausreichend für die Helligkeit der Anzeige.

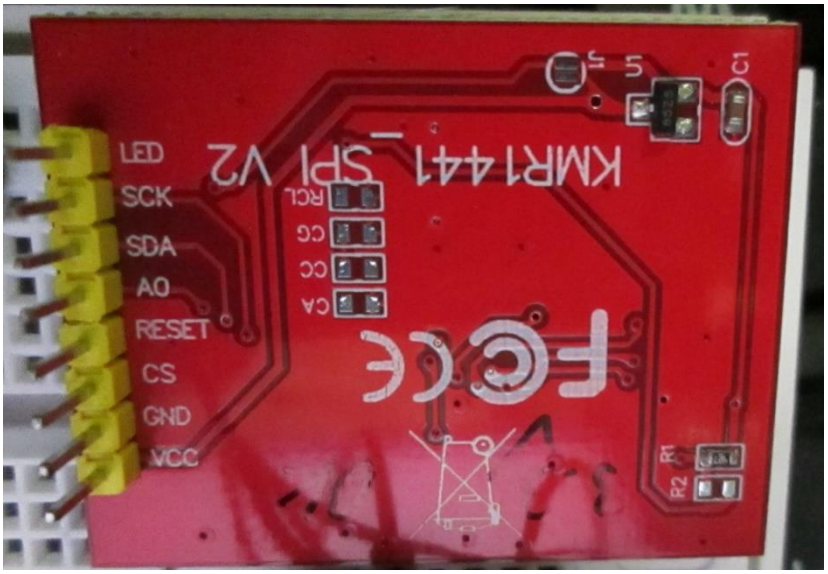
Ohne die beiden Versorgungsspannungen funktioniert es nicht!

HV4 z.B. kommt vom Signalausgang des Arduinos, LV4 geht dann in den Signaleingang des Displays. In den meisten Fällen kommt man mit vier Wandlern aus. Will man den Betrieb mit

RESET des Displays machen, dann braucht man einen 5. Wandler. Notfalls kann man aber hier auch einen Spannungsteiler mit 510R/1k aufbauen. Für das Display gehen ja die Signale nur in Richtung Arduino zum Display, nicht rückwärts. Der Spannungsteiler funktioniert, da hier die beiden GNDs miteinander verbunden sind.

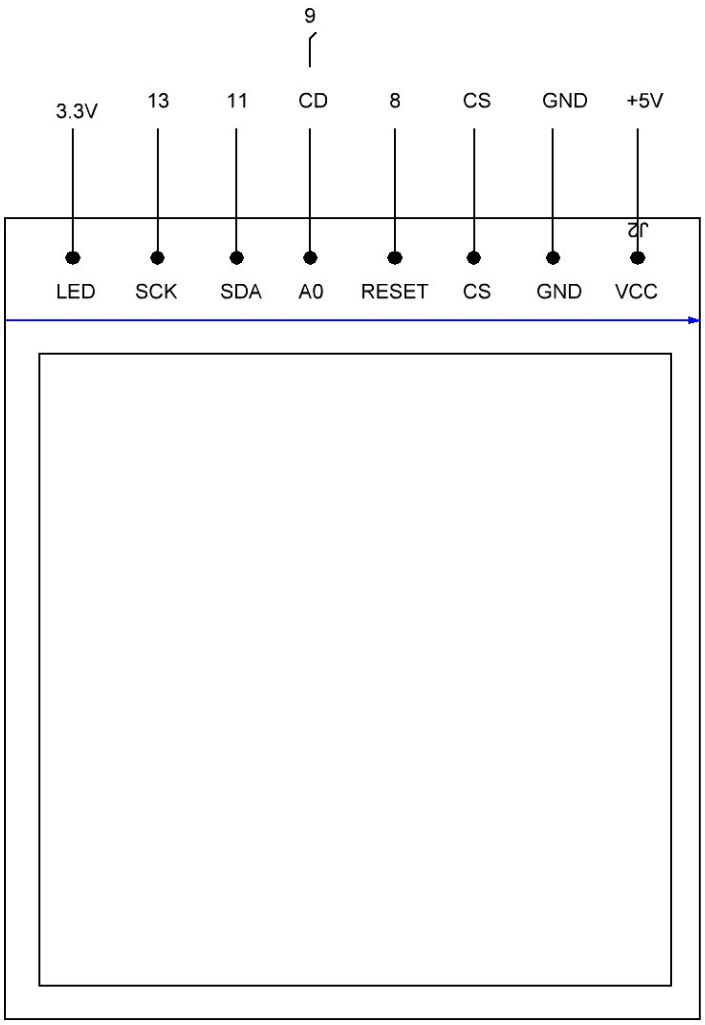
Verwirrend

Leider haben sich die Hersteller nicht auf verbindliche namen der Signale geeinigt. Da kann man schon Problem bekommen und ratlos vor dem teil sitzen. Hier nochmal die Signale, die dieses Display möchte.



VCC erhält +5V vom Arduino, ebenso den GND. LED kann man an +5V(sehr hell) oder +3.3V anschließen. Bei den übrigen Signalen hilft vielleicht die folgende Darstellung:

In unserem Display sieht die Sache so aus:



Das ist die Ansicht auf die Anschlusspins von oben. Dazu die PIN-Nummern des Arduinos. Es ist die Standardauswahl der Ausgabepins. Kann man bei Bedarf auch ändern. Der Anschluss A0 entspricht dem sonst üblichen CD.

Mein Display hat die Bezeichnung:

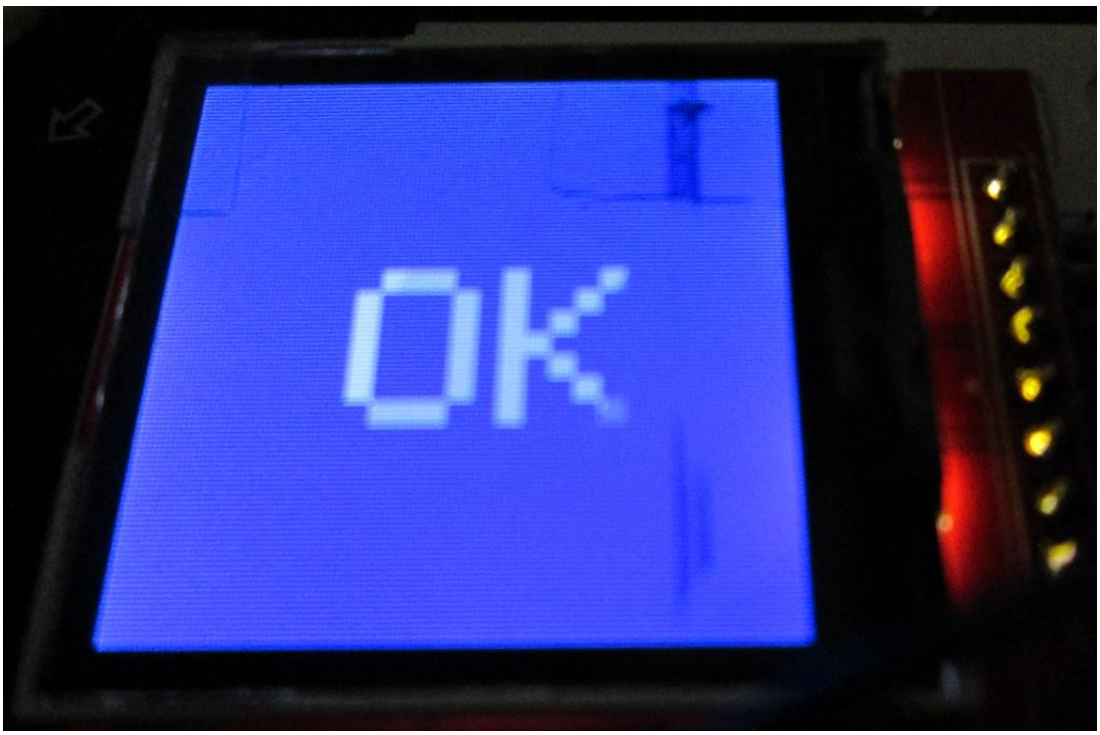
KMR1441_SPI V2

Nachfolgend eine kurzer Sketch(Software) für einen Test.

```
/*
  habe diesen Code für den Test einiger STK 1.44" geschrieben
 Die Displays müssen mit einem Pegelwandler 5V/3V betrieben werden, denn sonst
gehen sie kaputt!!
 Die LED kann man gut mit 3.3V betreiben; 3.3_Referenzausgang von Arduino Nano

 */

#include <TFT.h>  // Arduino LCD library
#include <SPI.h>


// pin definition
#define cs   10
#define dc   9
#define rst  8 // LOW setzt das Display in den Ruhezustand = hell
//erst wenn rst wieder HIGH wird und TFT mit TFTscreen.begin(); neu gestartet
wird, funktioniert das Display
//rst kann HIGH bleiben, wenn man die Funktion Reset nicht braucht; also ist
dann auch kein Pegelwandler nötig
//man kommt dann mit einem 4-er Pegelwandler aus; könnte den 5. aber mit einem
widerstandsteiler 510/1k machen



// create an instance of the library
TFT TFTscreen = TFT(cs, dc, rst);



void setup() {


  TFTscreen.begin();

  // clear the screen with a black background
  TFTscreen.background(152, 190, 100);

  // write the static text to the screen
  // set the font color to white
  //TFTscreen.stroke(255, 255, 255);
  TFTscreen.stroke(152, 190, 100);
  // set the font size
  TFTscreen.setTextSize(5);
  // write the text to the screen
  TFTscreen.text("OK\n ", 60,60);

}

void loop() {
```

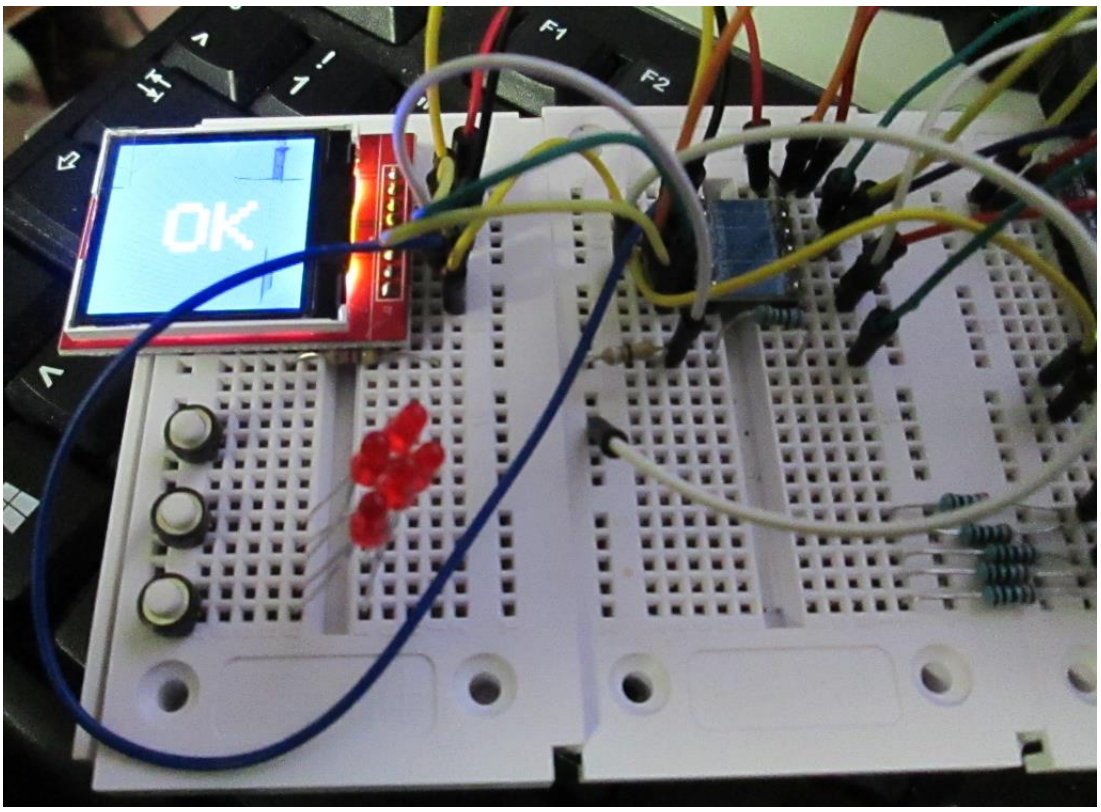
```
   // set the font color
  TFTscreen.stroke(255, 255, 255);
  //TFTscreen.background(182, 120, 100);//hellblau
  //TFTscreen.background(0,0,0);//black
  TFTscreen.background(ST7735_RED);//wird aber blau angezeigt!?
  TFTscreen.text("OK", 40, 50);

  delay(250);
  // erase the text you just wrote
  //TFTscreen.stroke(0, 0, 0);//black überscheiben
  //TFTscreen.text("  ", 40, 50);
  //weil die Schrift blinkt, kann man mit dem Oszillografen
  //die Impulse an den Anschlüssen messen; ist im Fehlerfall nützlich
}
```

Das Ergebnis bei einem ordentlich funktionierendem Display sieht so aus:



Und hier noch der Aufbau auf einem Breadboard. Man erkennt den Pegelwandler in der Mitte. Das Bild flackert, denn es wird immer wieder neu geschrieben. Dadurch kann man aber auch die Signale an den Pins mit einem

Oszillografen beobachten, was manchmal bei der Fehlersuche nützlich ist.



Viel Spaß beim Basteln!

DF8ZR; im Mai 2024