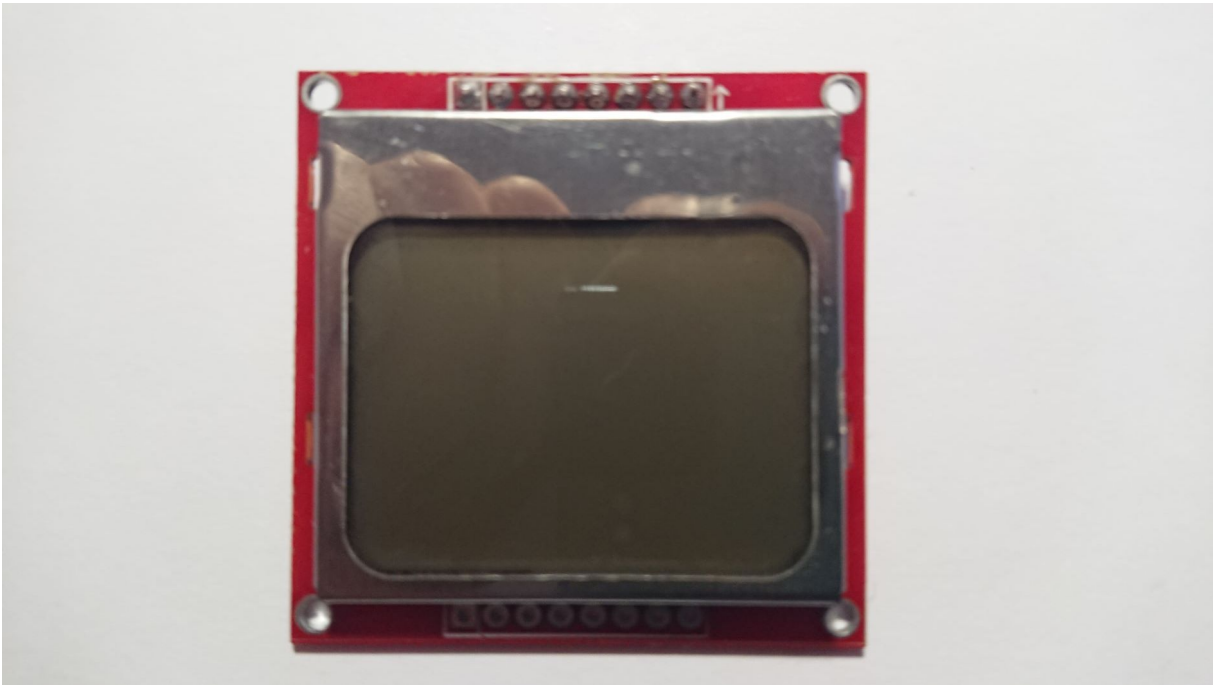
Willkommen!

Und herzlichen Dank für den Kauf unseres Nokia 5110 LCD-Displays. Auf den folgenden Seiten gehen wir mit dir gemeinsam die Anbindung an einen Arduino UNO und ersten Programmierschritte durch.

Viel Spaß!

Das Nokia 5110 Display ist ein monochromes LCD-Display mit 84x48 darstellbaren Pixeln, das über eine LED Hintergrundbeleuchtung verfügt.



(Bild 1)

Da das Display ursprünglich hauptsächlich im Handybereich Verwendung fand, ist die Arbeitsspannung des Displays 3,3 Volt. Dies bedeutet, dass die Versorgungsspannung und die Datenleitungsspannung 3,3 Volt beträgt.

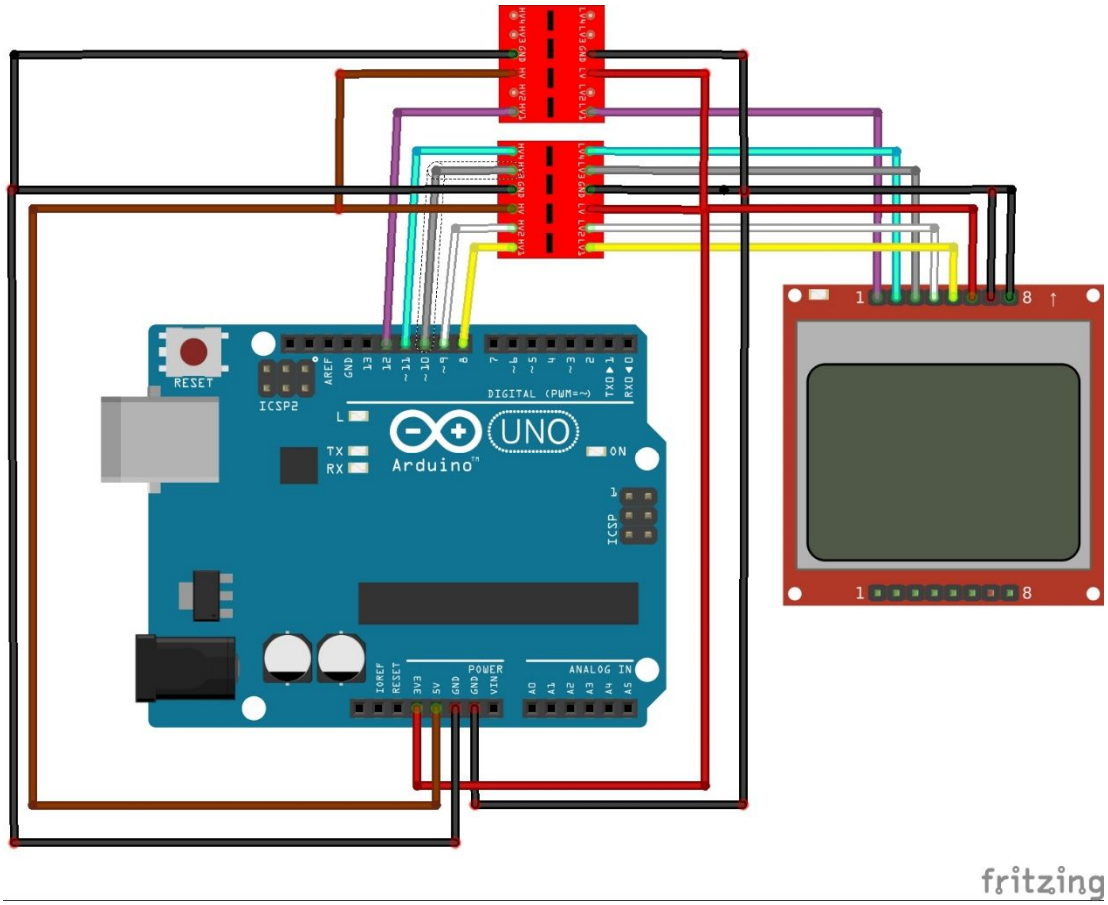
Höhere Spannungen als 3,3 Volt können zur Zerstörung des Displays führen!

Um nun mit dem Display mit unserem Arduino arbeiten zu können, ist es wichtig, die Datenleitungen das Arduinos, die mit 5 Volt arbeiten, auf Displaykompatible 3,3 Volt zu bringen. Diese Spannungskonversation können von Levelshiftern übernommen werden.

Das Display hat insgesamt 8 Anschlüsse, die wie folgt belegt sind:

Pin:	Beschreibung:	Zusatzinformationen:
1	RST (Reset) Input	
2	CE (Chip Enable) Input	
3	DC (Data /Control) Input	
4	DIN (Data in) Input	
5	CLK (Clock) Input	
6	VCC	Spannungsversorgung max. 3,3 Volt!
7	LIGHT	Led aktiv bei Verbindung gegen Masse
8	GND (Ground)	Masse

Folgendes Anschlussdiagramm zeigt schematisch die notwendige Verkabelung des Displays mit den Levelshiftern und dem Arduino.

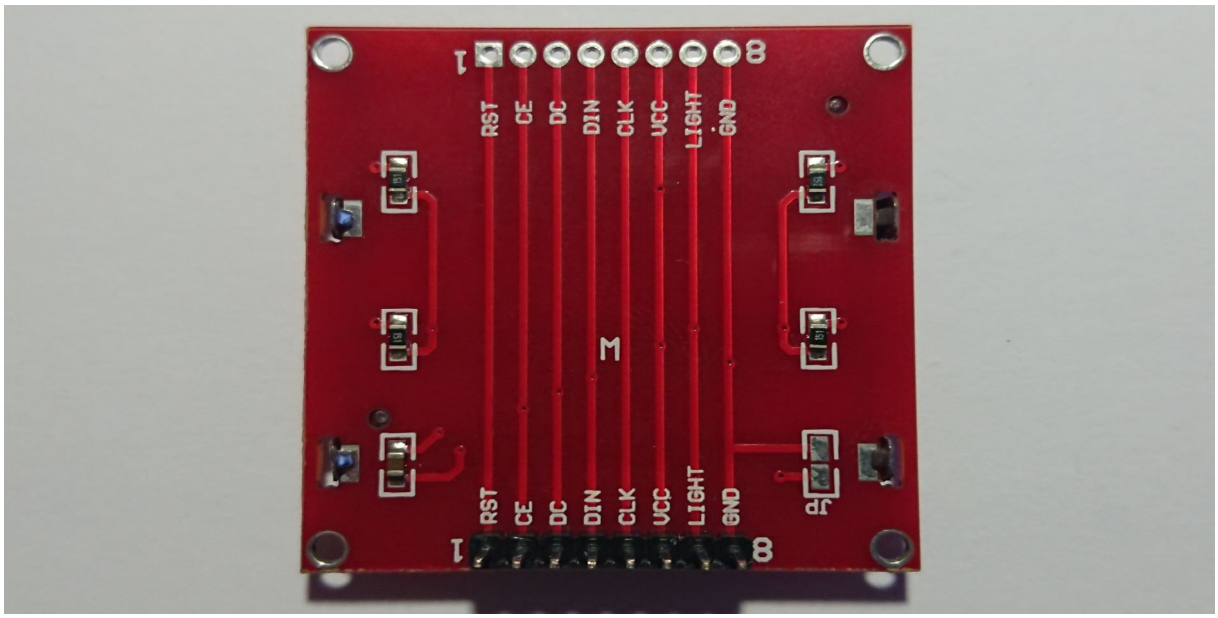


(Bild 2)

Masse (Minus Pol)	Schwarz
+ 5 Volt	Braun
+ 3,3 Volt	Rot
Port 8 Arduino	Gelb
Port 9 Arduino	Weiß
Port 10 Arduino	Grau
Port 11 Arduino	Cyan
Port 12 Arduino	Violett

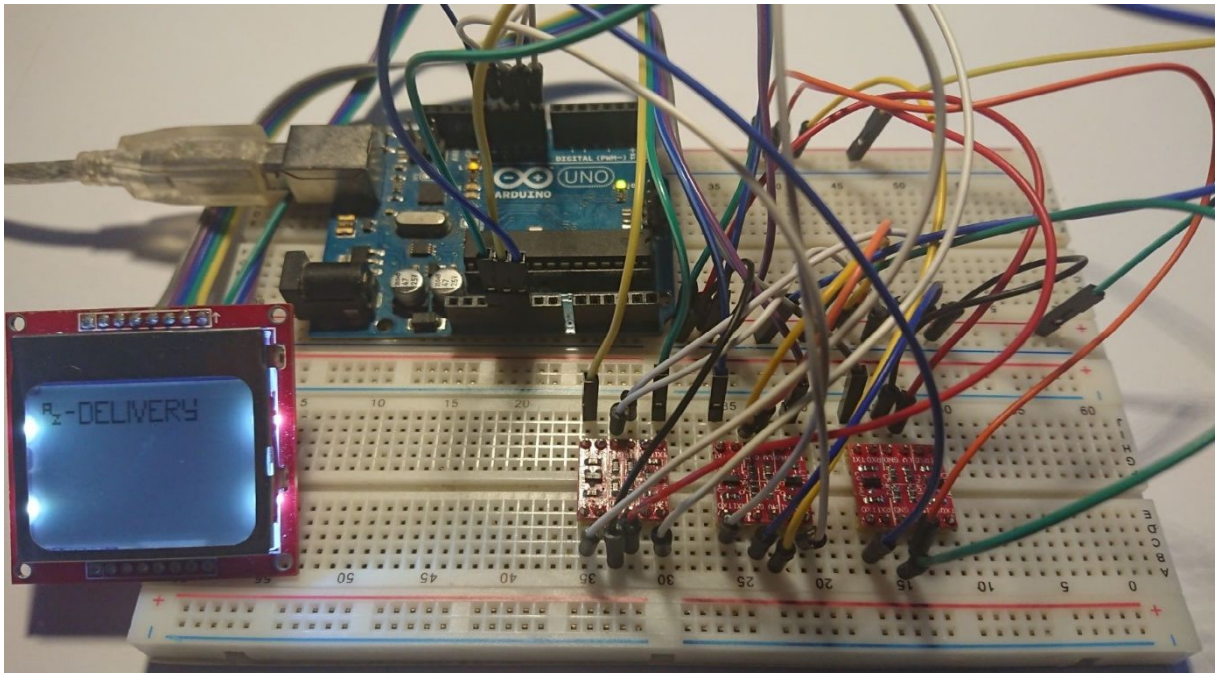
Die Ports sind wie folgt zugeordnet:

Port	Displayanschluss
8	CLK
9	DIN
10	DC
11	CE
12	RST



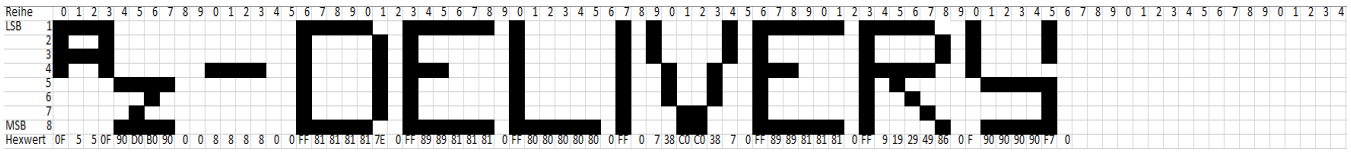
(Bild 3)

Aufgebaut auf einem Breadboard sieht die Schaltung dann in etwa so aus:



(Bild 4)

Kommen wir nun zu der eigentlichen Programmierung des Displays. Wir wollen das folgende Bitmuster an unser Display übertragen und Anzeigen:



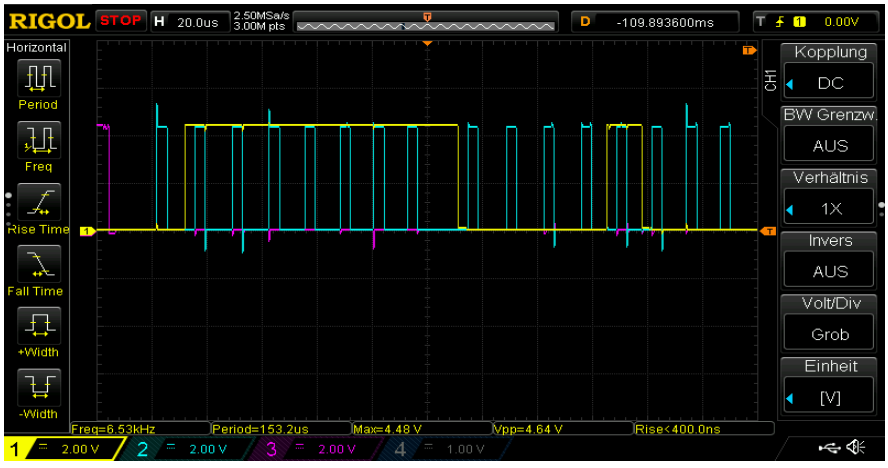
(Bild 5)

Dazu halten wir uns vor Augen, dass das Display eine Auflösung von 84 Spaltenpixeln x 48 Zeilenpixeln hat.

Die Zeilenpixel sind im internen RAM des Displays in jeweils in Bytes organisiert, sodass mit der ersten Ausgabe des ersten Datenbytes 8 Zeilenpixel beschreiben werden. Es folgt ein Sprung in die nächste Spalte, in der dann wieder 8 Bit, immer beginnend mit dem LSB Bit zuerst, als Byte ausgegeben werden. Nach dem Erreichen der Spalte 84 erfolgt ein Wechsel in die nächsten 8 Bit Zeile. So erreichen wir insgesamt 48/8 = 6 Bytezeilen mit je 84 Bytes.

Der Komplette Displayraum im RAM beträgt somit 6 x 84 Bytes = 504 Bytes. Nach erreichen des Limits werden folgende Zeichen wieder am Displayanfang angezeigt. Ein schreiben des Bytes 0 an alle Stellen des Displays bewirkt somit eine Lösung des Displayinhalts.

Um nun unser Logo ins Display zu bekommen, berechnen wir von jeden 8 Bit der Spalte 0 bis 65 den korrespondierenden Hex Wert oder Dezimalwert. Die dabei entstehenden Byte Array Werte sind unsere Daten, die wir zum Display später senden werden. Im Beispiel im Bild 5 wurde in der Zeile „Hexwert“ diese Berechnung schon ausgeführt. Diese Daten dienen uns im Sketch als Displaydaten, die im Array „LOGOTBL“ hinterlegt werden. Die eigentliche Datenübertragung übernimmt die Arduino Funktion „shiftout“ die durch ein in der Funktion selbst generiertes Clock Signal die Daten seriell ausgibt. Die Daten werden durch das Display übernommen, wenn das Clock Signal von LOW auf HIGH wechselt. Im folgendem Oszilloskop Bild ist dies ersichtlich:



(Bild 6)

Pin Clock (CLK) hat hier die Farbe **cyan**, der Datenpin die Farbe **gelb**. Pin Chip Enable (CE) hat die Farbe In **magenta**. Man sieht die Übertragung von 2 Bytes, nachdem CE auf LOW geht.

Der Arduino Code:

```
// Ansteuerung Nokia 5110 LCD Display
#define CLK 8
#define DIN 9
#define DC 10
#define CE 11
#define RST 12

static const byte LOGOTBL[] = {//AZ Delivery Bitmap  Logo
  0x0F,0x5,0x5,0x0F,0x90,0xD0,0xB0,0x90,0x0,0x0,0x8,0x8,0x8,0x8,0x0,0x0,0x0,0xFF,0x81,0x81,0x81,0x81,0x7E,0x0,
  0xFF,0x89,0x89,0x81,0x81,0x81,0x0,0xFF,0x80,0x80,0x80,0x80,0x80,0x0,0xFF,0x0,0x7,0x38,0xC0,0xC0,0x38,0x7,
  0x0,0xFF,0x89,0x89,0x81,0x81,0x81,0x0,0xFF,0x9,0x19,0x29,0x49,0x86,0x0,0xF,0x90,0x90,0x90,0x90,0xF7
 };
// Funktionen
void LcdWriteCmd(byte order)
{
  // Kommando an Display senden
  digitalWrite(DC, LOW); //DC Pin ist LOW für Befehle an das Display
  shiftOut(DIN, CLK, MSBFIRST, order); //Übertrage Datenbyte sweriell mit MSB (höchstwertiges Byte) zuerst.
}

void LcdWriteData(byte data)
{
  // Daten an Display senden
  digitalWrite(DC, HIGH); //DC Pin ist HIGH für Daten an das Display
  shiftOut(DIN, CLK, MSBFIRST, data); //Übertrage Datenbyte seriell mit MSB (höchstwertiges Byte) zuerst.
}

void LcdWriteLogo()
{
  digitalWrite(CE, LOW);   // Chip Enable Pin ist LOW, Datentransfer ist aktiviert.
  // In Zeile 0 Spalte 0 springen
  LcdWriteCmd(0x80); // Sprung Spalte
  LcdWriteCmd(0x40); // Sprung Zeile
  // Bildschirm komplett löschen
  for(int i=0; i < 504; i++)
  {
    LcdWriteData(0x00);  // Lösche komplettes Display
  }
  // AZ Logo ausgeben..
  for(int i=0; i < 66; i++)
  {
    LcdWriteData(LOGOTBL[i]); // Schreibe Logo Daten ins Display
  }
  digitalWrite(CE, HIGH); // Chip Enable Pin ist HIGH, Datentransfer ist dektiviert.
}

void setup()
{
  pinMode(CLK,OUTPUT); // Pin auf Ausgang schalten
  pinMode(DIN,OUTPUT); // Pin auf Ausgang schalten
  pinMode(DC,OUTPUT);  // Pin auf Ausgang schalten
  pinMode(CE,OUTPUT);  // Pin auf Ausgang schalten
  pinMode(RST,OUTPUT); // Pin auf Ausgang schalten
  digitalWrite(RST, HIGH);  //Resetsignal an das Nokia Display
  delay(100);               //senden
  digitalWrite(RST, LOW);
  delay(100);
  digitalWrite(RST, HIGH);
  delay(100);
  //Initalisierung des Displays gemäß Datenblatt
  digitalWrite(CE, LOW);   // Chip Enable Pin ist LOW, Datentransfer ist aktiviert.
  LcdWriteCmd(0x21); // Erweitertes Befehlset LCD
  LcdWriteCmd(0x90); // set LCD Vop
  LcdWriteCmd(0x20); // Normales Befehlset LCD
  LcdWriteCmd(0x0C); // LCD Normal Mode
  digitalWrite(CE, HIGH); // Chip Enable Pin ist HIGH, Datentransfer ist dektiviert.
}

void loop()
{
  LcdWriteLogo();
  delay(20000);
}
```

Nach Verdrahtung des Displays und nach Hochladen des Sketches erhalten wir folgende Ausgabe:

(Bild 7)

Es sind im Internet noch weitere Informationen über das Display in der Form von Datenblätter vorhanden. Einfach mal nach „PCD8544 Datenblatt“ googlen.

Ab jetzt heißt es Experimentieren.

Und für mehr Hardware sorgt natürlich dein Online-Shop auf:

https://az-delivery.de

Viel Spaß!
Impressum
https://az-delivery.de/pages/about-us