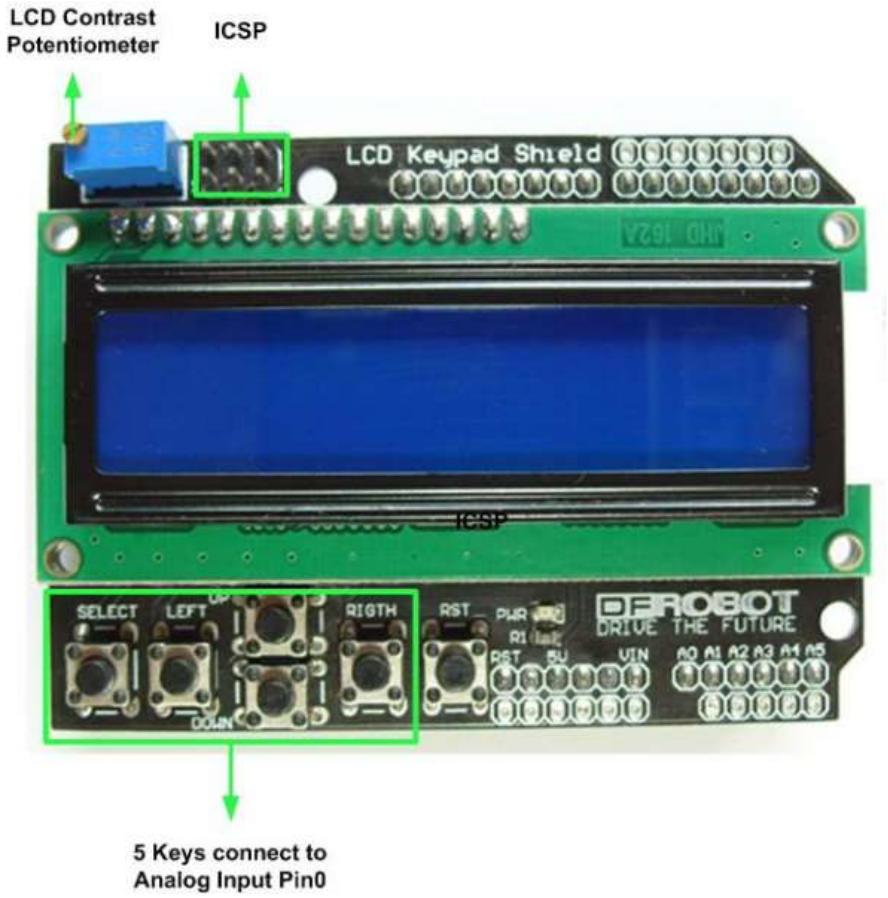
# Arduino LCD Shield




| Pin | Function |
|---|---|
| Analog 0 | Button (select, up, right, down and left) |
| Digital 4 | DB4 |
| Digital 5 | DB5 |
| Digital 6 | DB6 |
| Digital 7 | DB7 |
| Digital 8 | RS (Data or Signal Display Selection) |
| Digital 9 | Enable |
| Digital 10 | Backlit Control |

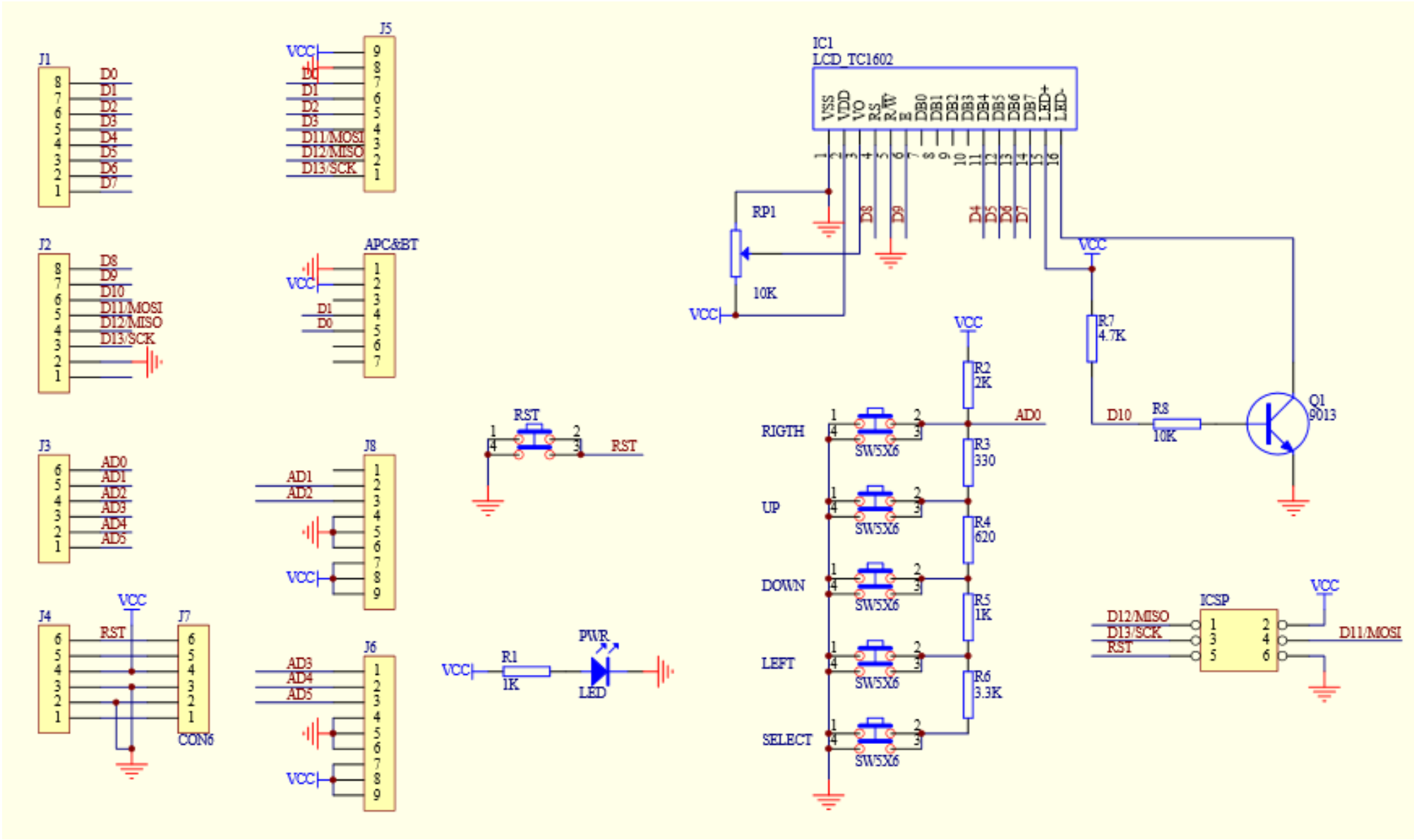
Quelle: http://www.dfrobot.com/wiki/index.php?title=Arduino_LCD_KeyPad_Shield_%28SKU:_DFR0009%29

# Arduino LCD Shield Codebeispiel

```
/*
DFRobot LCD Shield for Arduino
Key Grab v0.2
Written by Glendon Klassen
gjklassen@gmail.com
http://www.sourceforge.net/users/ecefixer
http://ecefixer.tumblr.com

Displays the currently pressed key on the LCD screen.

Key Codes (in left-to-right order):

None   - 0
Select - 1
Left   - 2
Up     - 3
Down   - 4
Right  - 5

*/

#include <LiquidCrystal.h>
#include <DFR_Key.h>

//  Pin assignments for DFRobot LCD Keypad Shield
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
//-------------------------------------------

DFR_Key keypad;

int localKey = 0;
String keyString = "";

void setup()
{
 lcd.begin(16, 2);
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("Key Grab v0.2");
 delay(2500);

 /*
 OPTIONAL
 keypad.setRate(x);
 Sets the sample rate at once every x milliseconds.
 Default: 10ms
 */
 keypad.setRate(10);

}

void loop()
{
 /*
 keypad.getKey();
 Grabs the current key.
 Returns a non-zero integer corresponding to the pressed key,
 OR
 Returns 0 for no keys pressed,
 OR
 Returns -1 (sample wait) when no key is available to be sampled.
 */
 localKey = keypad.getKey();

 if (localKey != SAMPLE_WAIT)
 {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Current Key:");
  lcd.setCursor(0, 1);
  lcd.print(localKey);
 }
}
```

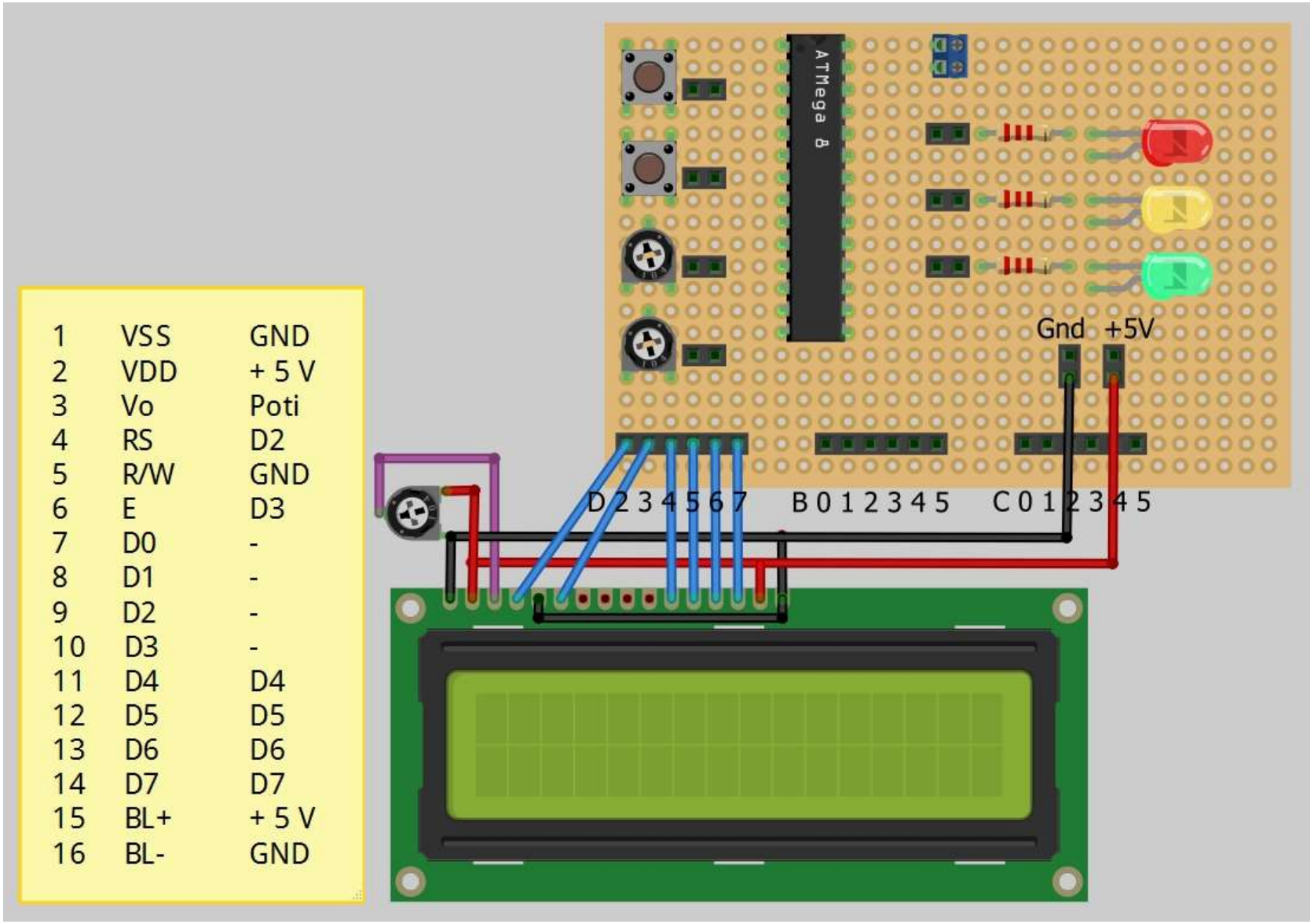
Quelle: http://www.dfrobot.com/wiki/index.php?title=Arduino_LCD_KeyPad_Shield_%28SKU:_DFR0009%29

# Arduino LCD Shield




Quelle: http://www.dfrobot.com/wiki/index.php?title=Arduino_LCD_KeyPad_Shield_%28SKU:_DFR0009%29

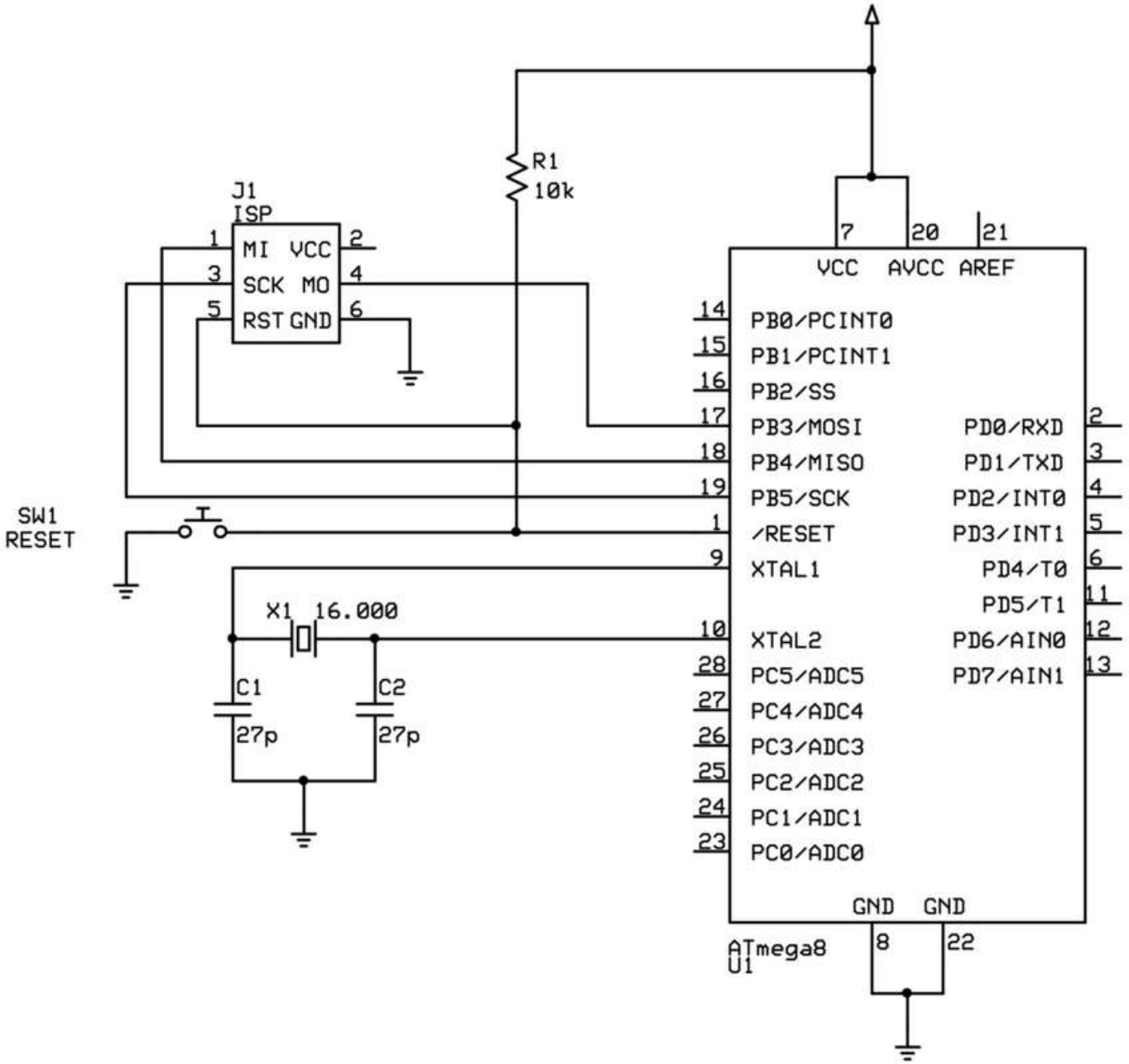
# Anschluss - LCD Anzeige




| | | |
|---|---|---|
| 1 | VSS | GND |
| 2 | VDD | + 5 V |
| 3 | Vo | Poti |
| 4 | RS | D2 |
| 5 | R/W | GND |
| 6 | E | D3 |
| 7 | D0 | - |
| 8 | D1 | - |
| 9 | D2 | - |
| 10 | D3 | - |
| 11 | D4 | D4 |
| 12 | D5 | D5 |
| 13 | D6 | D6 |
| 14 | D7 | D7 |
| 15 | BL+ | + 5 V |
| 16 | BL- | GND |

Pins D2, D3, D4,D5, D6, D7 sind identisch mit Arduino

# ATmega8 Minimal Beschaltung



Standard LCD - Displays haben 16 Anschlüsse. Hierfür werden beim MyAvr Board die folgende Anschlüsse verwendet:

| | | |
|---|---|---|
| **LCD RS Pin** | **digital Pin 2** | **Port B0** |
| **LCD Enable** | **digital Pin 3** | **Port B1** |
| | | |
| **LCD Data4** | **digital Pin 4** | **Port B2** |
| **LCD Data5** | **digital Pin 5** | **Port B3** |
| **LCD Data6** | **digital Pin 6** | **Port B4** |
| **LCD Data7** | **digital Pin 7** | **Port B5** |

Es werden nur 4 Datenleitungen genutzt - dies nennt man auch 4 Bit - Modus.

Unter Arduino gibt es eine LCD Bibliothek - die uns viel Programmierarbeit abnimmt. Diese wird eingebunden mit:

**#include <LiquidCrystal.h>**

Wir können die LCD - Anzeige an beliebigen Pins anschließen. Die verwendeten Pins werden mit folgendem Befehl deklariert.

**LiquidCrystal lcd(2, 3, 4, 5, 6, 7**);

Mit **lcd.begin(16, 2)** wird die Displaygröße angegeben. Dies ist meist 16 Zeichen - 2 Zeilen.

Die Ausgabe von Zeichen auf dem Display erfolgt mit:

**lcd.print("Lima 05")** Text strings werden in **"** … **"** eingebunden.

Variablen können ausgeben werden mit:

```
int val = 99;
lcd.print(val);
```

Mit **lcd.setCursor(5, 1);** wird der Cursur in Spalte 6, Zeile 2 gesetzt. Achtung die **Zählung beginnt bei diesem Befehl mit 0**.

Weitere Befehle sind:

| | |
|---|---|
| **lcd.clear();** | **LCD - Anzeige löschen** |
| **lcd.home();** | **Cursur auf oben links setzen** |
| **lcd.write();** | **einen Buchstaben ausgeben** |
| **lcd.noCursor();** | **Cursor verstecken** |

**// Beispielprogramm LCD Ausgabe**

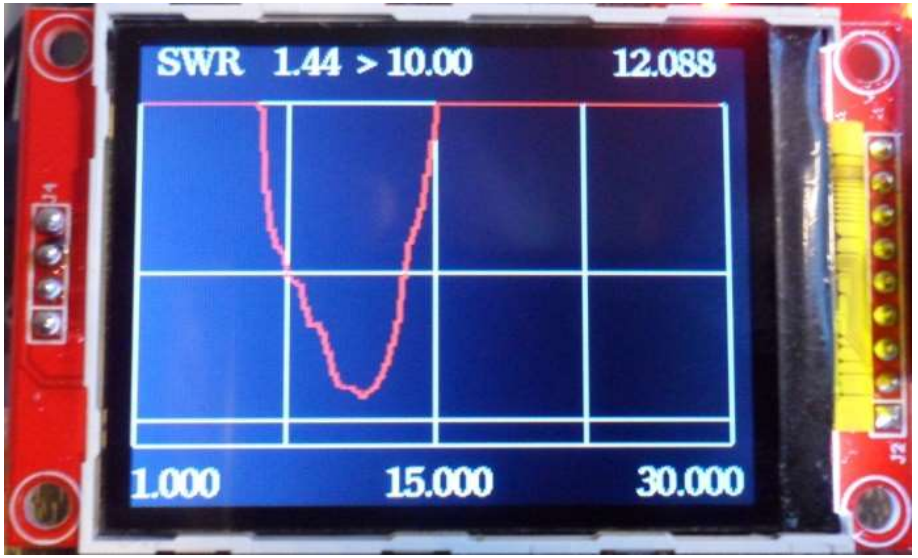
```
#include <LiquidCrystal.h>                  //LCD Bibliothek einbinden

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);        // Initialize LCD Pins

void setup()
{
lcd.begin(16, 2);                           // Setup für 2 x 16 Display
}

void loop()
{
lcd.print("Lima 05");                        // Zeile 1
lcd.setCursor(0, 1);                        // Setze Cursor auf Spalte 0, Zeile 2
                                            // Achtung: Die Zählung beginnt mit 0
lcd.print("DL0ER");
}
```

**// Input und Output mit LCD**

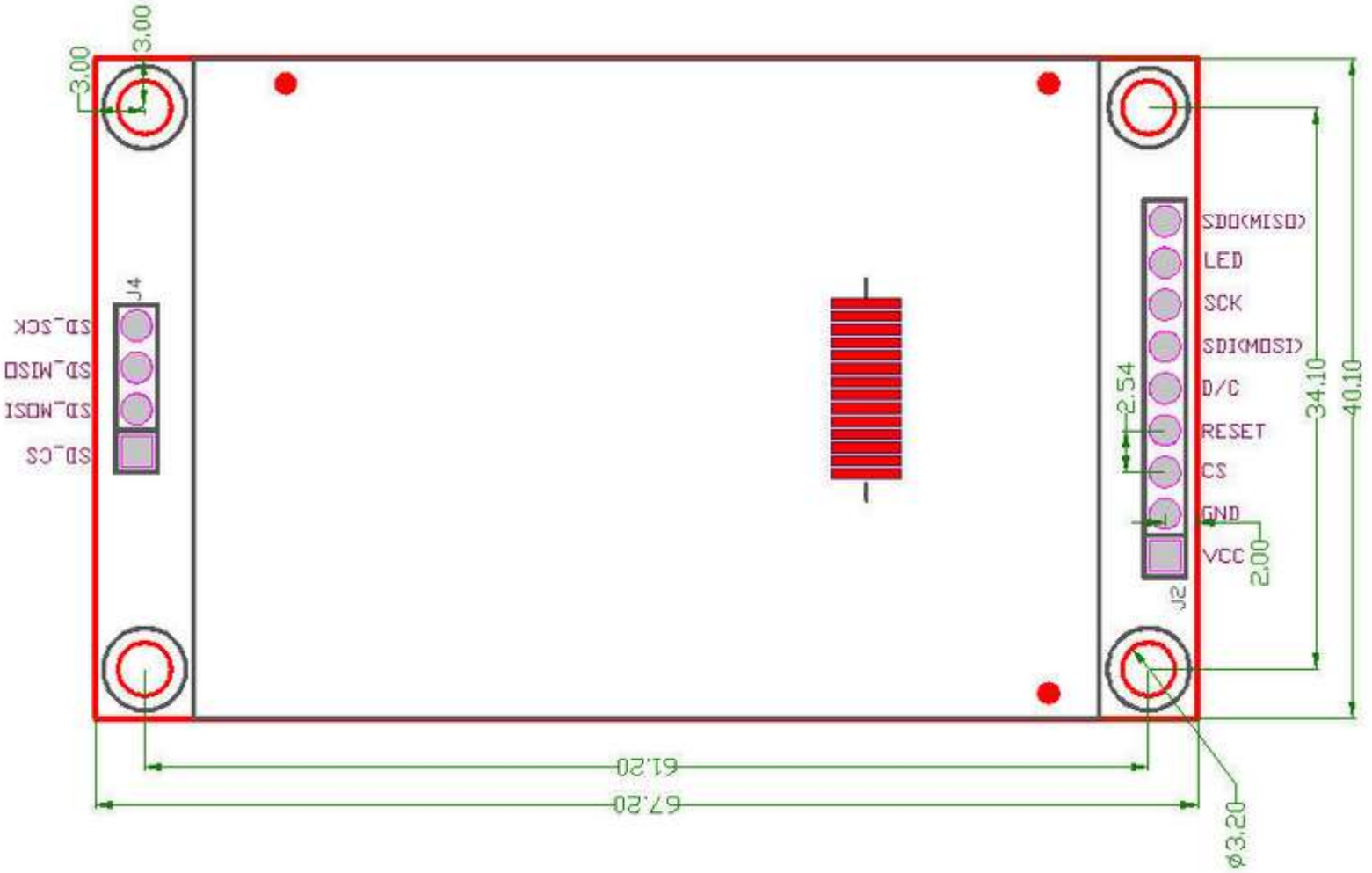
```
#include <LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

int ledPin1 = 8;                          // LED Pin8 – PB0
int inPin = 13;                           // Taster Pin13 - PB5
int val = 0;                              // Variable für Eingangswert

void setup()
{
  pinMode(ledPin1, OUTPUT);               // setze Pin8 als Ausgang
  pinMode(inPin, INPUT);                  // setze Pin13 als Eingang
  digitalWrite(inPin, HIGH);              // Pull Up Eingang auf High
  lcd.begin(16, 2);
  lcd.print("Lima 05");
 }

void loop()
{
  val = digitalRead(inPin);               // Lese Eingangs - Pin

if (val == 0)                             // Schalte LED
{
   digitalWrite(ledPin1, HIGH);
   lcd.setCursor(0, 1);
   lcd.print("Taste       ");
}
else
{
   digitalWrite(ledPin1, LOW);
   lcd.setCursor(0, 1);
   lcd.print("keine Taste");
}    }
```

# Graphik Display - IL9341 - 2,2“ - 240x320

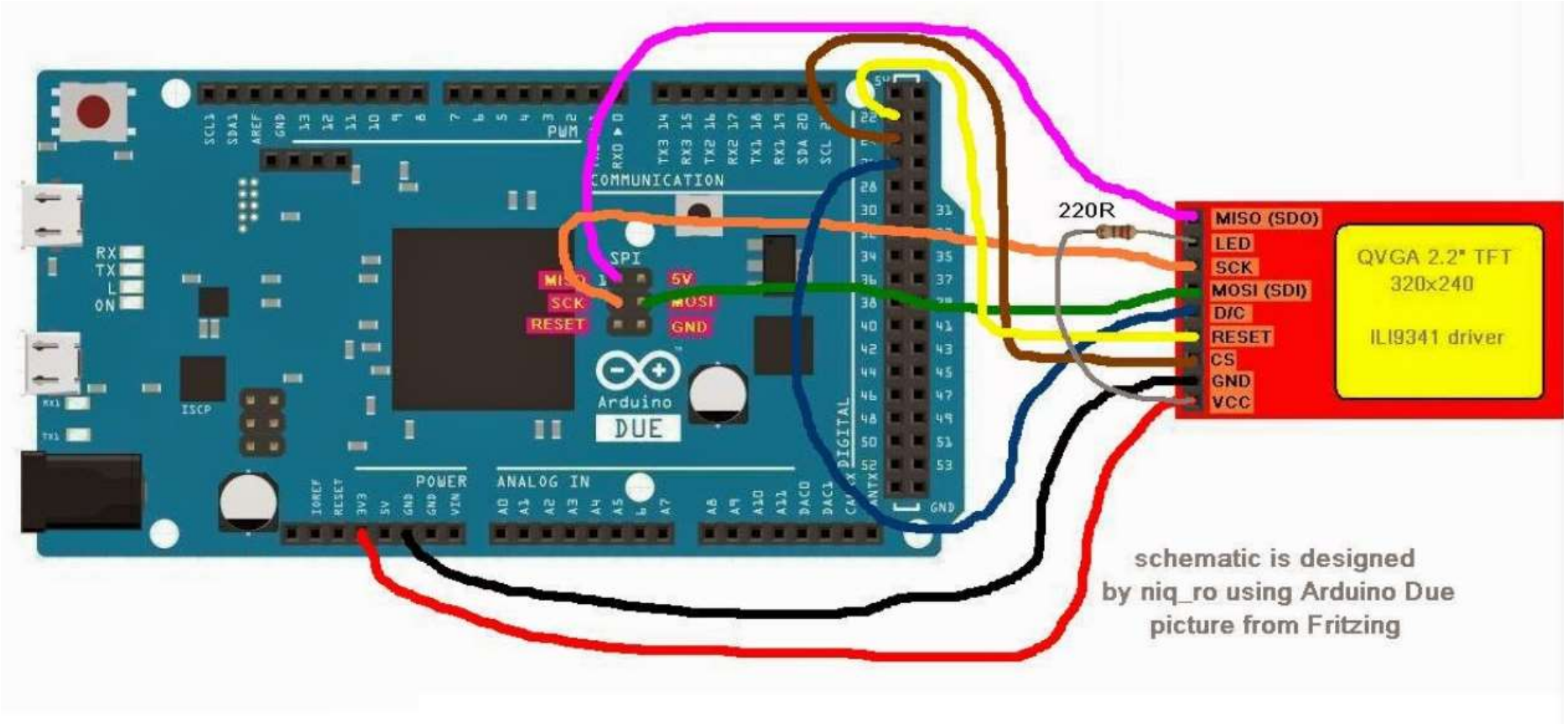

- Ebay 7 .. 15 €
- Hilfsspannung 3.3 / 5 V
- Pegel 3.3 V
- 2.2“ Display
- 240 x 320 px
- Serielle Ansteuerung




# Graphik Display - IL9341 - 2,2“ - 240x320



# Graphik Display - IL9341 - 2,2“ - 240x320



# Kommunikation über den SPI - Bus

**SPI library**

This library allows you to communicate with SPI devices, with the Arduino as the master device.

A Brief Introduction to the Serial Peripheral Interface (SPI)

Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. It can also be used for communication between two microcontrollers.

With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices. Typically there are three lines common to all the devices:

**MISO** **(Master In Slave Out) - The Slave line for sending data to the master,**
**MOSI** **(Master Out Slave In) - The Master line for sending data to the peripherals,**
**SCK** **(Serial Clock) - The clock pulses which synchronize data transmission generated by the master**

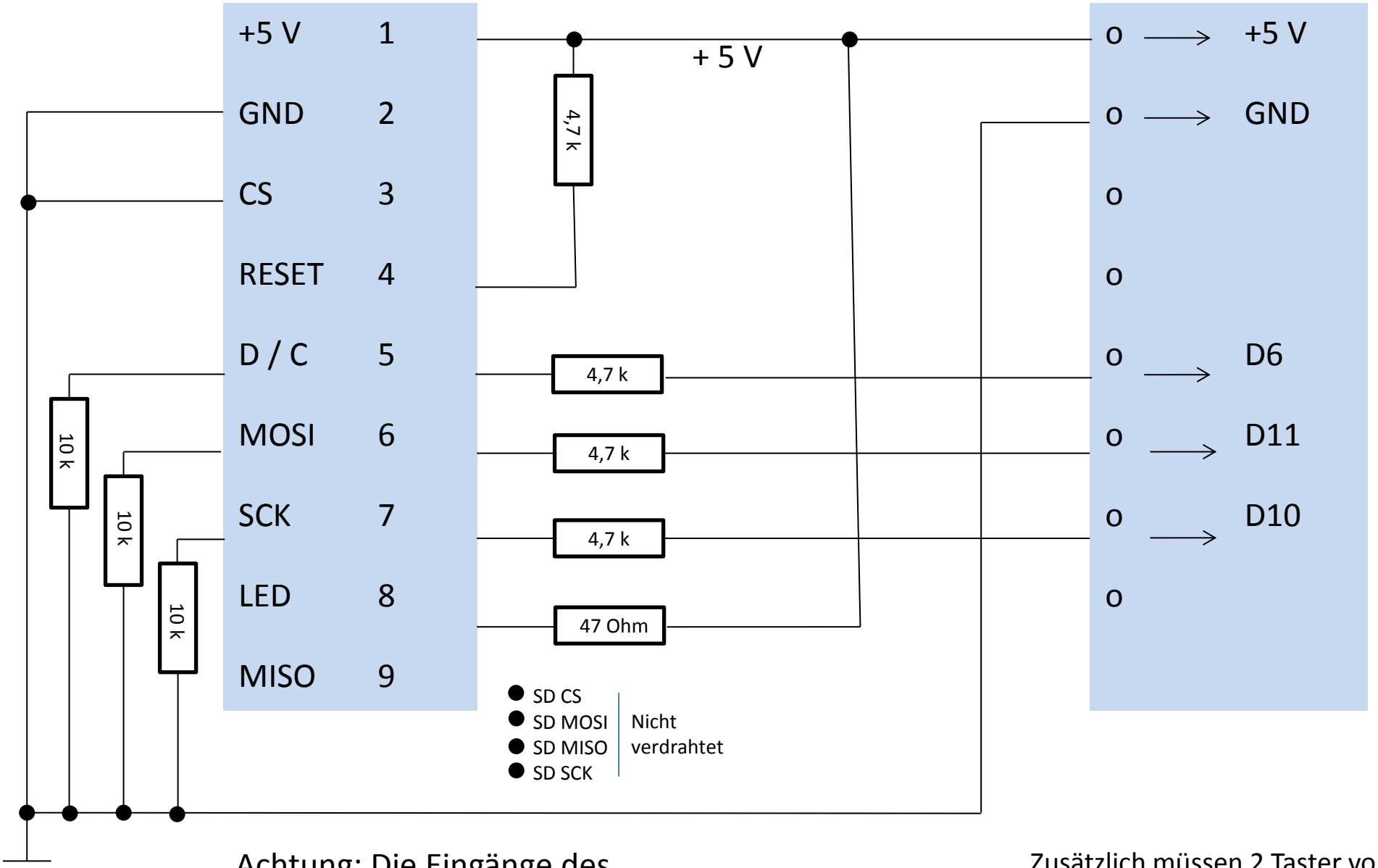
and one line specific for every device:

**SS** **(Slave Select) - the pin on each device that the master can use to enable and disable specific devices.**

TFT 2.2“ SPI
240 x 320, ILI 9341

# Verdrahtung TFT Display

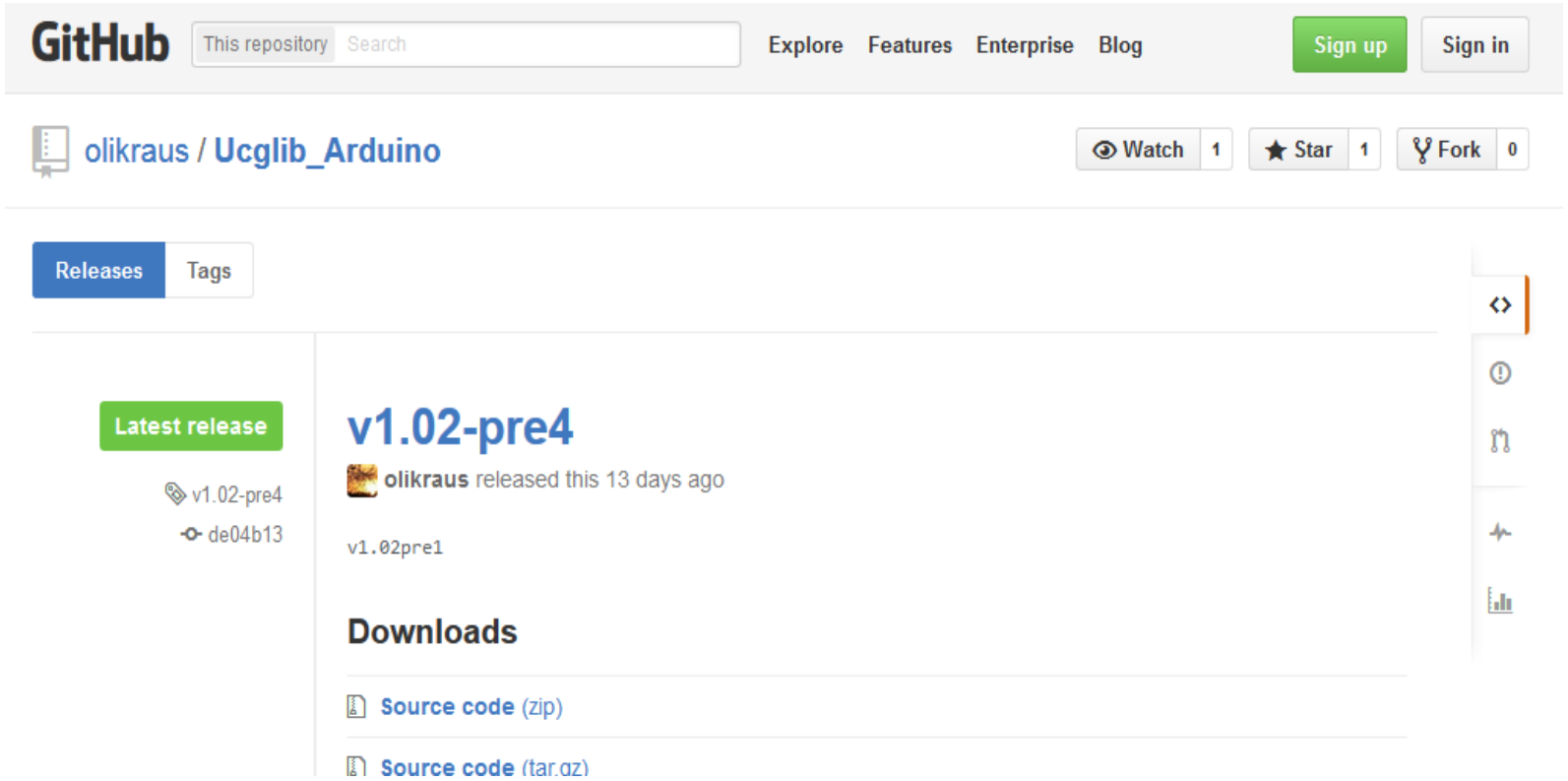
Arduino
Mini Pro




Achtung: Die Eingänge des Displays vertragen nur 3,3 V !

Zusätzlich müssen 2 Taster von Pin 2 + 3 des Arduino gegen Masse geschaltet werden.

# Arduino Ucglib - Library




**Library und Beispiele:**

https://github.com/olikraus/Ucglib_Arduino/archive/v1.02-pre4.zip

**Beschreibung und Tutorial**

https://github.com/olikraus/ucglib/wiki

# Hallo World Beispiel

```
#include <SPI.h>
#include "Ucglib.h"

// Hardware SPI Pins:
// Arduino Uno                  sclk=13, data=11
// Arduino Due                  sclk=76, data=75
// Arduino Mega                 sclk=52, data=51


Ucglib_ILI9341_18x240x320_SWSPI ucg(/*sclk=*/ 10, /*data=*/ 11, /*cd=*/ 6 , /*cs=*/ 5, /*reset=*/ 4);

void setup(void)
{
  delay(1000);
  ucg.begin(UCG_FONT_MODE_TRANSPARENT);
  //ucg.begin(UCG_FONT_MODE_SOLID);
  ucg.clearScreen();
}

void loop(void)
{
  //ucg.setRotate90();
  ucg.setFont(ucg_font_ncenR14r);
  ucg.setColor(255, 255, 255);
  ucg.setColor(255, 0, 0);
  //ucg.setColor(0, 255, 0);
  ucg.setColor(1, 255, 0,0);

  ucg.setPrintPos(0,25);
  ucg.print("Hello World!");

  ucg.setPrintPos(1,50);
  ucg.print("Hello World!");

  delay(500);
}
```

# UCG Library Beispiele

**Bildschirm löschen**

```
ucg.clearScreen();
```

**Horizontale Linie zeichnen - drawHLine**

void Ucglib::drawHLine(ucg_int_t x, ucg_int_t y, ucg_int_t len)

```
ucg.setColor(255, 255, 255);
ucg.drawHLine(ucg, 50, 40, 45);
```

**Linie zeichnen - drawLine**

void Ucglib::drawLine(ucg_int_t x1, ucg_int_t y1, ucg_int_t x2, ucg_int_t y2)

```
Ucg.SetColor(255, 255, 255);
ucg.drawHLine(40, 45, 100, 145);
```

**Text ausgeben im Transperent Modus**

```
ucg.begin(UCG_FONT_MODE_TRANSPARENT);
ucg.setFont(ucg_font_ncenR14r);
ucg.setPrintPos(0,25);
ucg.setColor(255, 255, 255);
ucg.print("Hello World!");
```

**Text ausgeben im überdeckenden Modus**

```
ucg.setFont(ucg_font_ncenR14r);                  // do not use _tf for UCG_FONT_MODE_SOLID
ucg.setFontMode(UCG_FONT_MODE_SOLID);
ucg.setPrintPos(42, 40); ucg.setColor(0, 0, 0, 0);   // black color for the text
ucg.setColor(1, 150, 220, 255);                  // light blue for the background
ucg.print("Ucg");
```

**Referenz (Beschreibung) unter:**

https://github.com/olikraus/ucglib/wiki/reference