64-211 Übung Eingebettete Systeme

Aufgabenblatt 5 Termine: 09.06. + 16.06./12.06. + 19.06.

Gruppe	
Name(n)	Matrikelnummer(n)

LC Displays sind ein häufig verwendetes Ausgabegerät in einem eingebetteten System. Bei der Bearbeitung der Aufgaben werden Sie ein Nokia 5110-kompatibles LC-Display einsetzen, um grundlegende Ausgabemöglichkeiten zu implementieren. Die Schnittstelle (`SPI`) zur Ansteuerung des LC-Displays wird durch den integrierten IC (Philips PCD8544) bereitgestellt. Das entsprechende Datenblatt kann unter `tams.informatik.uni-hamburg.de/lectures/2015ss/vorlesung/es/doc/pcd8544.pdf` eingesehen werden. Betrachten Sie darüber hinaus `en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus` und machen Sie sich mit den Grundbegriffen vertraut. Die Arduino Dokumentation zu `SPI` finden Sie unter `arduino.cc/en/Reference/SPI`.

Achtung: Führen Sie die notwendige Verdrahtung auf dem Steckbrett durch. Abbildung 1 stellt die Anschlüsse der LC-Display Platine dar, die mit dem Arduino Due Board zu verbinden sind.

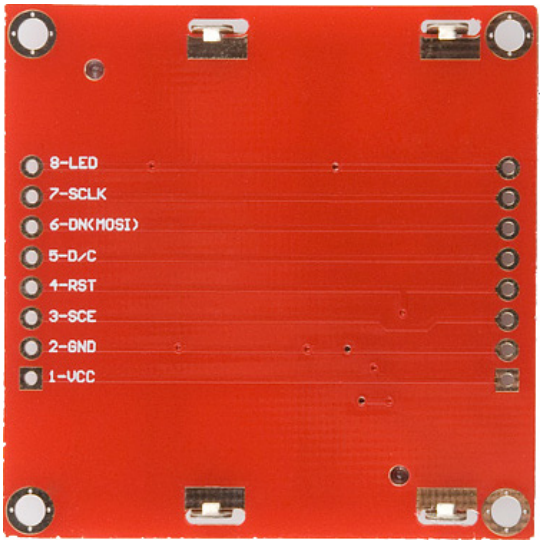

Abbildung 1: Anschlusspins der Nokia 5110-kompatiblen LC-Display Platine.

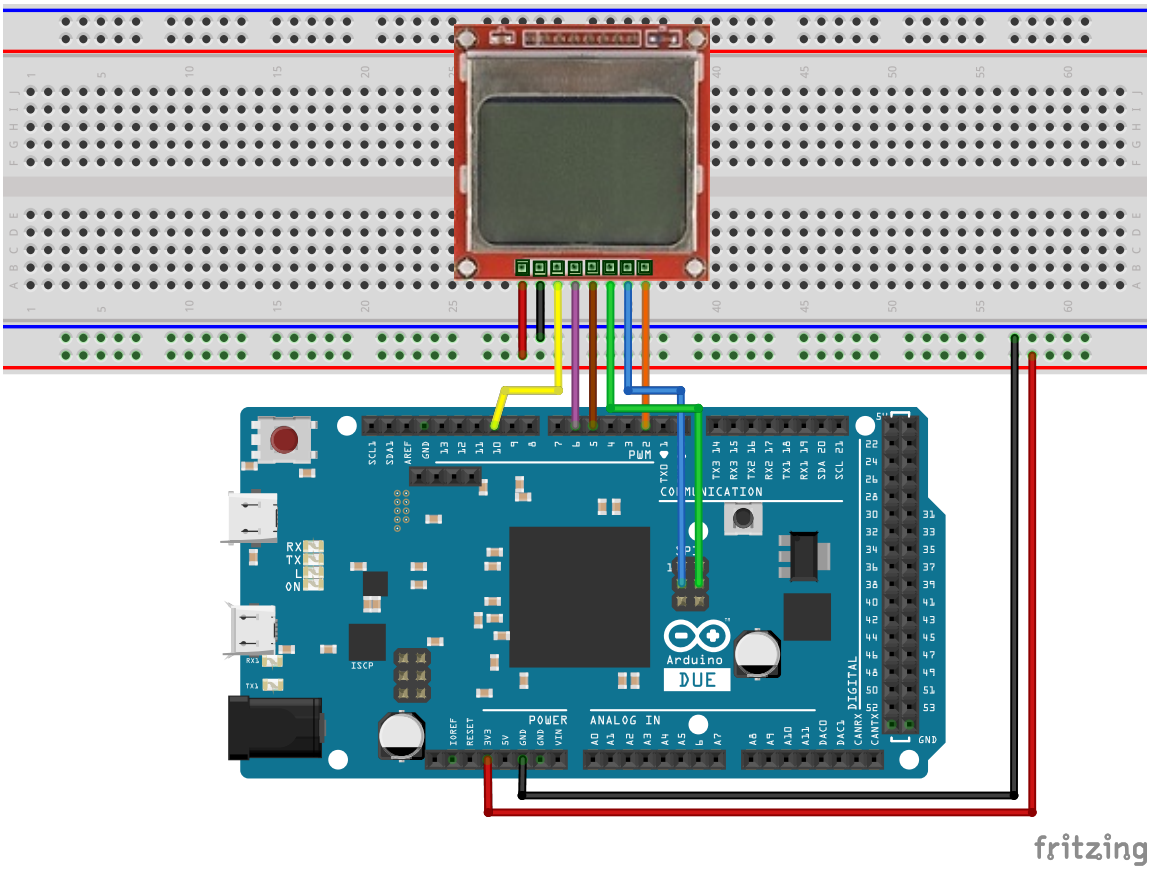

Abbildung 2: Vorschlag für die Verdrahtung des Versuchsaufbaus.

Für die Lösung der Aufgabe wird empfohlen den Versuchsaufbau gemäß Abbildung 2 zu verdrahten. Beachten Sie bei der Verdrahtung folgende Hinweise:

- Verbinden Sie `VCC` mit **3.3V** und stellen Sie zusätzlich die Masseverbindung her.
- Verbinden Sie `SCE` mit einem der `Slave-Select` Pins des Arduino Due - **4**, **10** oder **52**.
- Verwenden Sie für die Verbindung der Pins `RST` und `D/C` beliebige digitale Pins des Arduino Due.
- Verbinden Sie `SCLK` und `DN(MOSI)` mit den entsprechenden Anschlüssen am **SPI**-Block (siehe Arduino Due Pinbelegung).
- Der `LED` Anschluss muss nicht angeschlossen werden (es dient der Spannungsversorgung für die Hintergrundbeleuchtung). Sollten Sie es dennoch tun wollen, schliessen Sie diesen an einen beliebigen digitalen Pin des Arduino Due an.

Aufgabe 5.1

Entwerfen Sie ein Programm, das folgende Funktionalität zur Verfügung stellt:

1. Initialisierung des Displays bei Inbetriebnahme des Systems (`setup()`).
2. Funktionalität zur Puffer-basierten, Pixel-weisen Manipulation des Displayinhalts.
3. Test-Code, der ein wiederkehrendes Muster auf dem Display darstellt.

Machen Sie sich mit den Funktionen der Arduino `SPI` Bibliothek vertraut (`#include <SPI.h>`). Im Folgenden sind die Funktionen aufgelistet, die für die Bearbeitung der Aufgabe benötigt werden:

```
* SPI.begin()                                    arduino.cc/en/Reference/SPIBegin
* SPI.setClockDivider()                arduino.cc/en/Reference/SPISetClockDivider
* SPI.transfer()                             arduino.cc/en/Reference/SPITransfer
```

Hinweise zu Punkt 1:

- Zunächst ist es notwendig ein `RESET` des Displays durchzuführen. Beachten Sie bitte, dass die `RST` Leitung **low-active** ist, d.h. ein `RESET` nur dann erfolgt, wenn der Pegel `LOW` gesetzt wird. **Wichtig:** Es wird eine 500ms lange `RESET`-Phase empfohlen.
- Initialisieren Sie die SPI-Schnittstelle mit `SPI.begin(pin)`, wobei `pin` die Nummer des von Ihnen gewählten `Slave-Select` Anschlusses sein sollte.
- Stellen Sie mit `SPI.setClockDivider(pin, value)` den Taktteiler für den Takt der SPI-Schnittstelle ein. Definieren Sie bitte einen Takt von **1 MHz**.
- Die Daten werden Byte-weise mittels `SPI.transfer(pin, data)` an das Display abgesetzt. `pin` ist dabei der von Ihnen gewählte `Slave-Select` Anschluss.
- Beachten Sie bitte, dass das Display zwischen **LCD-COMMAND** (`D/C` = `LOW`) und **LCD-DATA** (`D/C` = `HIGH`) unterscheidet. Bevor Sie das Display tatsächlich benutzen können, ist folgende Initializierungssequenz als **LCD-COMMAND**s an das Display abzusetzen (siehe auch Seiten 14/22 des Datenblatts):
  1. `0x21` : FUNCTION SET (extended instruction set)
  2. `0x14` : SET BIAS
  3. `0xE0` : SET CONTRAST
  4. `0x20` : FUNCTION SET (normal instruction set)
  5. `0x0C` : SET DISPLAY MODE (normal)

Hinweise zu Punkt 2:

- Das Display besitzt eine Auflösung von **48x84** Pixeln (Zeilen x Spalten). Die Adressierung (siehe auch Seiten 8+9 des Datenblatts) findet jedoch nicht Pixel-weise statt. Die 48 Zeilen sind in 6 `BANK`s unterteilt. Dieses ermöglicht das komfortable Setzen jeder der 84 Spalten einer `BANK` mittels der Übertragung eines einzigen Bytes.

- Um den Kommunikationsaufwand mit dem Display dabei gering zu halten und grössere grafische Operationen schneller erledigen zu können, soll ein Display-Puffer definiert werden. Dabei ist der Inhalt dieses Puffers nach Abschluss der von Ihnen vorgenommenen Operationen jeweils komplett an das Display zu übertragen.
- Um möglichst flexibel zu bleiben (für spätere Aufgaben), soll der Puffer Pixel-weise addressierbar sein. D.h. es ist eine Funktion `setPixel(int x, int y, int value)`, mit `value` $\in (0, 1)$, zu definieren.

Hinweise zu Punkt 3:

- Um die entwickelte Funktionalität zu überprüfen, soll folgender Test implementiert werden:
  1. Beginnen Sie mit der ersten Spalte des Displays und **aktivieren** Sie jeden Pixel dieser Spalte.
  2. Aktualisieren Sie das Display.
  3. Warten Sie 20ms ab, fahren Sie mit der nächsten Spalte des Displays fort und aktivieren Sie auch hier jeden Pixel.
  4. Wiederholen Sie die Schritte 1-3, bis Sie jede Spalte aktiviert haben.
  5. Beginnen Sie erneut mit der ersten Spalte des Displays und **deaktivieren** Sie jeden Pixel dieser Spalte.
  6. Aktualisieren Sie das Display.
  7. Warten Sie 20ms ab, fahren Sie mit der nächsten Spalte des Displays fort und deaktivieren Sie auch hier jeden Pixel.
  8. Wiederholen Sie die Schritte 5-7, bis Sie jede Spalte deaktiviert haben.
  9. Fahren Sie mit Schritt 1. fort.

Aufgabe 5.2

Verwenden Sie den auf der Webseite zur Verfügung gestellten ASCII-Datensatz `tams.informatik.uni-hamburg.de/lectures/2015ss/vorlesung/es/doc/6x8_ascii_data.txt` und implementieren Sie mithilfe der existierenden Funktionalität aus der vorherigen Aufgabe eine Funktion `printChar(int x, int y, char value)`, die es Ihnen ermöglicht ein im Datensatz kodiertes Zeichen an einer beliebigen Stelle des Displays zu positionieren. **Wichtig:** Vergessen Sie dabei nicht die Grenzen des Displaypuffers zu prüfen. Ihre Funktion soll also einen Rückgabewert liefern (z. B. -1) falls die Angabe der Koordinaten es nicht ermöglicht das Zeichen vollständig darzustellen.

Entwerfen Sie als Test-Code eine Funktion, die das Zeichen @ möglichst mittig in der Darstellung positioniert.

Hinweis zum ASCII-Datensatz: Der Datensatz ist ein zweidimensionales Array, das die gängigsten (nicht alle) ASCII-Zeichen im **6x8** Format enthält. Jede Zeile des Arrays definiert ein Zeichen durch die Angabe von **6 Bytes**. Jedes dieser Bytes spezifiziert eine Spalte des 6x8 Blocks. Die letzte Spalte ist immer leer, Sie müssen also nicht für den Abstand zwischen den Zeichen sorgen.

Aufgabe 5.3

Bauen Sie auf der vorherigen Aufgabe auf und enwerfen Sie ein Programm, das Ihren Vornamen, Nachnamen und Matrikelnummer, jeweils untereinander, zentriert im LCD anzeigt. Wechseln Sie die Daten für jeden Teilnehmer Ihrer Gruppe im Abstand von 5 Sekunden aus.