



Informatik 11  
Embedded Software

RWTH AACHEN  
UNIVERSITY

## Praktikum Systemprogrammierung

# Versuch 6

### ***Testtaskbeschreibung***

Lehrstuhl Informatik 11 - RWTH Aachen

8. Dezember 2023

Commit: 48145a2f

# Inhaltsverzeichnis

<b>6 Testtaskbeschreibung</b>	<b>3</b>
6.1 Color Rendering . . . . .	3
6.2 Color Retrieval . . . . .	4
6.3 Joystick . . . . .	6

---

Dieses Dokument ist Teil der begleitenden Unterlagen zum *Praktikum Systemprogrammierung*. Alle zu diesem Praktikum benötigten Unterlagen stehen im Moodle-Lernraum unter <https://moodle.rwth-aachen.de> zum Download bereit. Folgende E-Mail-Adresse ist für Kritik, Anregungen oder Verbesserungsvorschläge verfügbar:

support.psp@embedded.rwth-aachen.de

# 6 Testtaskbeschreibung

## 6.1 Color Rendering

Um diesen Testtask ausführen zu können, müssen der Displaytreiber und die Methode `draw_setPixel` wie im Dokument beschrieben implementiert sein. Der Testtask gliedert sich in drei Phasen:

1. Phase: Test, ob die Farbe jedes Pixels gesetzt werden kann
2. Phase: Darstellung von Weiß in 4 Intensitätsstufen
3. Phase: Spaltenweise Ausgabe von verschiedenen Farben

Beachten Sie, dass dieser Testtask bei Fehlern Ihrer Ausgabe nicht stoppt. Um eine der Phasen gesondert zu betrachten, kann das `#define SHOW_AND_HALT_PHASE` gesetzt werden. Wird das Makro als die Zahl  $i$  definiert, zeigt der Testtask nur die  $i$ -te Phase dauerhaft an, damit Fehler besser erkannt werden können. Achten Sie darauf, dass die angezeigten Bilder mit der Beschreibung der jeweiligen Phase übereinstimmen. Beachten Sie weiterhin, dass die im Testtask verwendete Funktion `delayMs` eventuell nicht immer korrekt funktioniert, da Nebeneffekte des Displaytreibers die verwendete Art der Zeitmessung beeinflussen. Falls die Anzeige der Phasen aufgrund falscher Wartezeiten variiert, alle Phasen einzeln jedoch korrekt durchlaufen werden, ist der Testtask bestanden.

**Phase 1** In dieser Phase wird die Methode `draw_setPixel` verwendet, um jedes Pixel auf die Farbe weiß zu setzen. Phase 1 ist erfolgreich, wenn das gesamte Panel weiß leuchtet. Nach einer kurzen Wartezeit wird Phase 2 gestartet.

**Phase 2** In dieser Phase wird das LED-Panel in vier Quadranten aufgeteilt. In allen Quadranten wird die Farbe Weiß angezeigt, allerdings in verschiedenen Helligkeitsstufen (Achtung: Der Bereich oben links erhält die Helligkeit 0 und bleibt damit komplett schwarz). Die Phase ist erfolgreich, wenn sich alle vier Quadranten voneinander unterscheiden lassen. Abbildung 6.1 zeigt die erwartete Ausgabe. Nach einer kurzen Wartezeit wird Phase 3 gestartet.

**Phase 3** In dieser Phase werden einige Farben spaltenweise dargestellt. Jede Farbe wird in einem Block von 2 Spalten dargestellt, zwischen den Farben wird jeweils eine Spalte schwarz gelassen, damit die Farben besser erkennbar sind. Der Rest der LED-Matrix wird schwarz aufgefüllt. Phase 3 ist erfolgreich, wenn die folgenden Farben korrekt

## 6 Testtaskbeschreibung

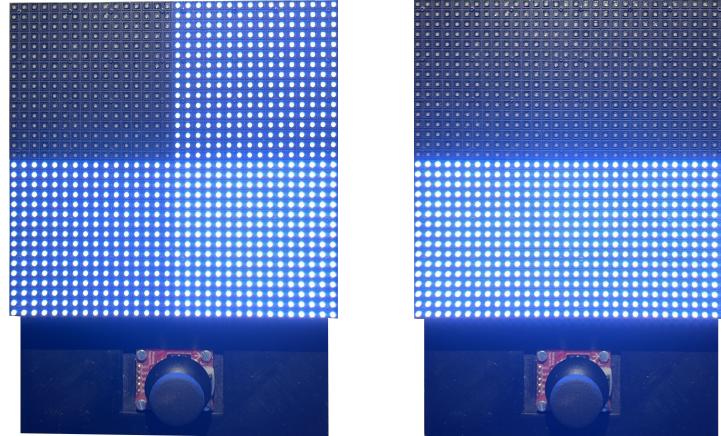


Abbildung 6.1: Ausgabe auf der LED-Matrix. Links: korrekte Implementierung, Rechts: Farbtiefe zu gering

dargestellt werden: Weiß, Rot, Dunkelrot, Grün, Dunkelgrün, Blau, Dunkelblau, Violett, Gelb, Türkis und Schwarz. Abbildung 6.2 zeigt die erwartete Ausgabe.

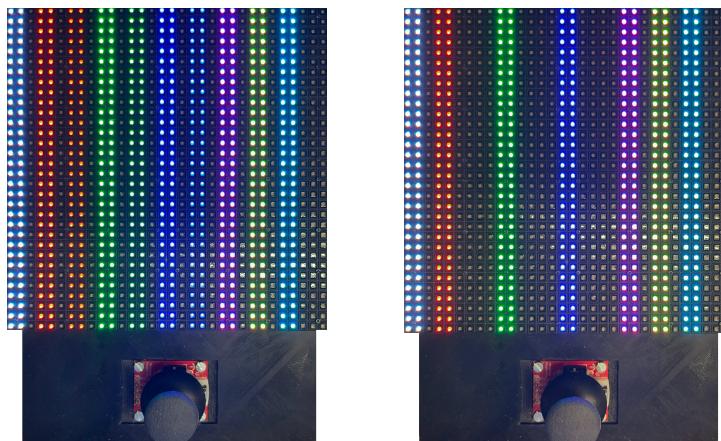


Abbildung 6.2: Ausgabe auf der LED-Matrix. Links: korrekte Implementierung, Rechts: Farbtiefe zu gering

## 6.2 Color Retrieval

Um diesen Testtask ausführen zu können, müssen wie im Testtask *Color Rendering* der Displaytreiber und die Methode `draw_setPixel` wie im Dokument beschrieben implementiert sein. Zusätzlich wird hier die Methode `draw_getPixel` getestet.

## 6 Testtaskbeschreibung

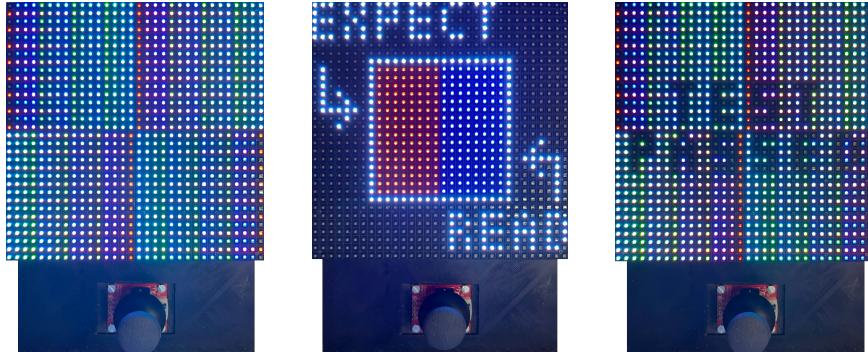


Abbildung 6.3: Von links nach rechts: Das Testmuster, die Ausgabe der Farben im Fehlerfall und die Anzeige “TEST PASSED”

**Ablauf** In diesem Testtask wird ein spezielles Farbmuster auf der Matrix angezeigt, welches die folgenden Eigenschaften hat:

- Jede mögliche Farbe kommt für  $y < 16$  und  $y > 16$  jeweils mindestens einmal vor
- Die Farben an  $(x, y)$  und  $(x, y + 16)$  sind immer unterschiedlich

Damit soll speziell getestet werden, dass das Extrahieren von Farben aus den “Doppel-Pixeln”, die im Framebuffer gespeichert sind, korrekt gehandhabt wird.

Nach dem Zeichnen werden die einzelnen Pixelwerte des Musters mit `draw_getPixel` ausgelesen und mit den ursprünglichen verglichen. Weichen die Werte für ein Pixel voneinander ab, so wird auf dem LCD eine Fehlermeldung ausgegeben, die die Koordinaten, den korrekten Farbwert und den abweichenden gelesenen Farbwert enthält. Auf der LED-Matrix wird ebenfalls in der linken Hälfte die korrekte Farbe und in der rechten Hälfte die gelesene Farbe dargestellt. Wenn kein Fehler auftritt, wird “TEST PASSED” ausgegeben.

F	A	I	L	X	:	0	0	Y	:	0	1		
E	8	0	0	0	0	0		R	0	0	0	0	8

Abbildung 6.4: Ausgabe der Koordinaten und Farbwerte im Fehlerfall (Hier: Vertauschen der Farbwerte für Rot und Blau). Unten links: Erwarteter (**Expected**) Farbwert, rechts: Gelesener (**Read**) Farbwert. Es werden die drei Bytes für Rot, Grün und Blau jeweils in 2 Hex-Digits dargestellt

## 6.3 Joystick

Dieser Testtask prüft, ob die beiden Achsen des Joysticks sowie der eingebaute Button korrekt ausgelesen werden. Dazu muss der Joysticktreiber komplett implementiert sein. Dies umfasst die Funktionen `void js_init()`, `uint16_t js_getVertical()`, `uint16_t js_getHorizontal()`, `Direction js_getDirection()` und `bool js_getButton()`. Der Test fragt kontinuierlich die Werte der oben genannten Funktionen ab und gibt diese entsprechend auf dem Panel aus.

**Ablauf** Die obere Zahl X (in weiß) repräsentiert den Rückgabewert von `js_getHorizontal()`. Die untere Zahl Y (in grün) repräsentiert den Rückgabewert von `js_getVertical()`. Beide Werte sollten ca. einen Bereich von 0 bis 1023 durchschreiten können. Das Symbol im unteren Panelbereich stellt die durch `js_getDirection()` gelesene Richtung durch einen Pfeil dar. Ist diese nicht `JS_NEUTRAL`, wird zusätzlich ein roter Punkt dargestellt, welcher vom Pfeilmittelpunkt aus um eine konstante Entfernung in Richtung des Joysticks verschoben ist. Die Position des Punktes wird mittels `js_getHorizontal` und `js_getVertical` errechnet um eine Möglichkeit zu bieten, die Joystick-Position feiner abzulesen und mit der Pfeilrichtung abzugleichen. Damit der Testtask als bestanden gilt, muss folgendes Verhalten bei Bedienung des Joysticks erkennbar sein.

**Neutrale Joystickstellung** Alle Pfeile werden gleichzeitig angezeigt, optisch entspricht dies einer Diamantform. Die ausgelesenen Zahlenrepräsentationen der Achsen sollten beide um ca. 500 herum liegen. Es wird kein roter Punkt angezeigt.

**Volle Auslenkung nach links** Die Zahlenrepräsentation der horizontalen Achse soll ca. 0 sein. Der Pfeil zeigt nach links direkt auf den roten Punkt.

**Volle Auslenkung nach rechts** Die Zahlenrepräsentation der horizontalen Achse soll ca. 1023 sein. Der Pfeil zeigt nach rechts direkt auf den roten Punkt.

**Volle Auslenkung nach oben** Die Zahlenrepräsentation der vertikalen Achse soll ca. 1023 sein. Der Pfeil zeigt nach oben direkt auf den roten Punkt.

**Volle Auslenkung nach unten** Die Zahlenrepräsentation der vertikalen Achse soll ca. 0 sein. Der Pfeil zeigt nach unten direkt auf den roten Punkt.

**Button** Bei Druck auf den Joystickbutton wird auf dem LCD *Memory cleared!* ausgegeben. Des Weiteren färbt sich der Pfeil blau, solange der Button gedrückt ist.