Praktikumsaufgabe 4 - Pong-Spiel

Aufgabe

In diesem Praktikum implementieren Sie das Pong-Spiel auf dem RED-V Board. Nutzen Sie dafür die im Labor vorhandene Hardware (siehe Abbildung 1) oder die pong-Konfiguration im Simulator. Die Abgabe und die Bewertung besteht aus mehreren Anforderungen, die erfüllt werden müssen. Diese sind im Folgenden aufgelistet. Bitte lesen Sie sich alle Informationen gründlich durch.

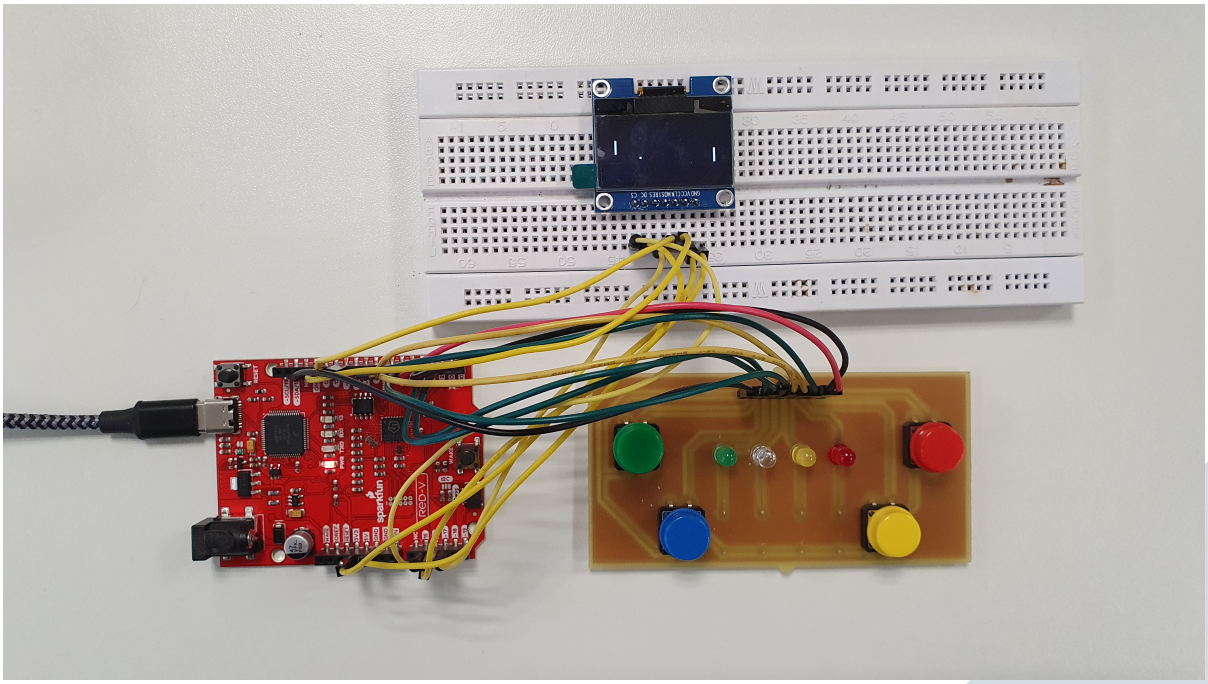

Abbildung 1: Pong Hardware mit vier LEDs, vier Buttons und OLED

Funktionsweise

Grundprinzip

Das Pong-Spiel ist ein Zwei-Spieler Spiel. Es bewegt sich ein Ball auf dem Bildschirm hin und her und jeder Spieler steuert über zwei Knöpfe einen senkrechten Strich (Schläger) auf einer Seite des Bildschirms hoch und runter. Wenn der Ball an dem Spieler vorbeilässt, bekommt der andere Spieler einen Punkt. Wenn der Ball an den Schläger kommt, prallt er abhängig von dem Einfallswinkel des Balls und der Position, an dem er den Schläger trifft, ab. Wenn der Ball die obere bzw. untere Bildschirmgrenze trifft, prallt er an dieser abhängig von Einfallswinkel ab (Einfallswinkel=Ausfallswinkel).

Anforderungen

Um das Spiel zu steuern, wird die Hardware aus Praktikum 1 um ein OLED Display ergänzt. Es handelt sich dabei um das SH1106-Display, das über SPI angesprochen werden soll. Eine Bibliothek, über die das Display angesprochen werden kann und deren Verwendung ist unter https://github.com/U2654/embedded/tree/master/workspace/sh1106-spi/ zu finden.

Beim Entwickeln des Spiels müssen folgende Anforderungen umgesetzt werden:

- Nutzen Sie FreeRTOS (Für weitere Informationen neben der Vorlesung nutzen Sie die öffentlich zugängliche Dokumentation von FreeRTOS). Eine FreeRTOSConfig.h ist im Ilias unter *Extra Dateien für das Praktikum* gegeben.
- Auf Buttons soll über Interrupts reagiert werden. (Hilfe hierzu finden Sie unter Interrupts)
- Für die Synchronisation der Tasks mit den Interrupts sollen binäre Semaphoren genutzt werden.
- Realisieren Sie die Steuerung der Buttons über mindestens einen Task.
- Nutzen Sie einen gemeinsamen Framebuffer, der die aktuelle Visualisierung des Spiels enthält.
- Implementieren Sie mindestens einen Task, um die Position des Balls und der Schläger in den Framebuffer zu schreiben.
- Implementieren Sie einen weiteren Task, der die Visualisierung des aktuellen Punktestandes in den Framebuffer schreibt.
- Machen Sie den Framebuffer und alle weitere gemeinsam genutzten Ressourcen über Mutexe sicher vor Schreib- und Lesekonflikten.
- Setzen Sie die MISRA-Normen bei der Implementierung um und evaluieren Sie Ihren Code, wie in der Vorlesung gezeigt. Verbessern Sie ihren Code so weit wie möglich, um möglichst keine Warnungen zu bekommen.
- Der abzugebende Code für das Spiel muss kompilierbar und in der Simulation sowie auf dem RED-V Board lauffähig sein. Kommentieren Sie notfalls fehlerhafte Codeabschnitte bzw. Funktionalitäten aus. Die Lauffähigkeit hat Vorrang.

Pinbelegung

- Grüner Knopf => 2
- Blauer Knopf => 3
- Gelber Knopf => 4
- Roter Knopf => 5
- Grüne LED => 6
- Blaue LED => 7
- Gelbe LED => 8
- Rote LED => 9

- OLED
  - CLK => SPI1_SCK
  - MOSI => SPI1_MOSI
  - RES => 17
  - DC => SPI1_SS3
  - CS => SPI1_SS2

Hilfen

Interrupts

FreeRTOS nutzt einen eigenen Trap Handler zum Reagieren auf Interrupts, deswegen müssen folgende Sachen beachtet werden, um den eigenen Interrupthandler zu integrieren. Das Assemblermakro -DportasmHANDLE_INTERRUPT=<interrupt_handler> muss in der Platform.ini gesetzt werden. Da es bereits einen Interrupthandler gibt, darf die Adresse des neuen Interrupthandlers nicht in das mtvec Register geschrieben werden. Außerdem muss bei der Deklaration der Interruptfunktion __attribute__((interrupt)) weggelassen werden. Des Weiteren sollten Sie nicht die Interrupts manuell über das Schreiben in das mie und mstatus Register enablen, da FreeRTOS dieses eigenständig macht, um Interrupts währen Kontextwechsel und der Initialisierung zu verhindern. Nutzen Sie die Informationen zu dem Platform Level Interrupt Controller (PLIC) in der Vorlesung und dem Datenblatt, um Interrupts für Buttons zu aktiviern.

Abgabe für die Bewertung (Abgabe im Ilias als ZIP)

1. Der implementierte, lauffähige Code, der sowohl die Funktionsweise, als auch die technische Umsetzung erfüllt.