Ein kleines Vorwort:

Hier meine Begleit “Dokumentation” zu meinem Projekt.
Meinen Code und seinen Verlauf finden sie auf
https://github.com/TizianoHranov/PottiPong

Wenn Sie sich meine 2 Videos zum Projekt schon angesehen haben werden sie wisse, dass es nicht funktioniert aufgrund meiner LED-Matrix. Die Spiellogik ist nicht fertig implementiert, da ich zuerst meinen C Code für den AD Wandler, und die LED Matrix implementieren und verlässlich testen wollte. Nachdem ich mich solange mit der LED Matrix beschäftigt hatte, und diese dann nicht mehr funktioniert hat, trotz meines erfolgreichen Tests, habe ich dann aufgegeben. Mein Zeitaufwand für dieses Projekt waren zu diesem Zeitpunkt ~30h .

Nichtsdestotrotz möchte ich hier trotzdem erklären aus was sich mein Projekt zusammensetzt und wie ich an eventuelle Probleme und Ihrer Lösung herangegangen bin.

Projektaufbau:

Grundidee ist ein Pongspiel mit Hardwarekomponente nachzubauen.

Gespielt wird mit 2 Potentiometer, welche die Bewegung des eigenen “Players” bzw. des eigenen Balkens, steuern. Um diese Potentiometer auszuwerten, habe ich den MCP3008 via SPI ausgelesen.

Das Spiel wird auf einer 8x32 MAX7219 LED Matrix dargestellt, auf welcher mit geplanten 20 fps das Spiel gezeichnet wird.

Meine Software setzt sich aus folgenden Teilen zusammen:

- Spiellogik und quasi “main” des Projekts
    - Ball.h
    - Player.h
    - PottiPong.h … hier sollte die Spiellogik ausprogrammiert sein, hier werden auch die folgenden .h’s verwendet um das Spiel dann auch zu visualisieren. Dies ist quasi das Herzstück des ganzen Projektes. Jeden Step/Frame sollen Spielerbewegungen ausgewertet, und auf die Matrix übertragen werden. Nach N Spielerbewegungs-auswertungen, bewegt sich der Ball um 1 LED in X, und 1 LED in Y Richtung. Hier wäre

eine “Difficulty” angedacht gewesen, welche das N festlegt – also wie schnell sich der Ball bewegt im Verhältnis zum Spieler. Wieviel FPS sinnvoll sind, wollte ich zuerst mit der Matrix ertesten. Ich habe meine Funktionen und Variablen in den zugehörigen .c Files beschrieben. Falls Sie wollen, sind diese sicher einfach nachvollziehbar, und Sie werden verstehen, wie die eigentliche Logik zum Spiel angedacht war.

- AD Wandler für die Bewegung der Spieler
    - Bcm2835.h … diese Library ist in vielen modernen Librarys zum Raspberry Pi bereits standardmäßig inkludiert und geht auf die Arbeit von “Mike McCauley” zurück. Die derzeit sichtbare “main.c” in meine Repo, enthällt den Testcase zu meiner Pottischaltung, und ist auch der Code der in den Videos zum Einsatz gekommen ist um die Pottis auszulesen.

      die library ist hier erhälltlich:
      https://www.airspayce.com/mikem/bcm2835/
- LED Matrix
    - Luma.LED_Matrix … dies ist die library die ich am Ende verwenden wollte, da sie die erste war, mit der ich einen tatsächlich funktionierenden Test hinbekommen habe (s. .c und .py file in “https://github.com/TizianoHranov/PottiPong/tree/main/test_cases/wokringLedTest_nowXandY_checkersboard”). Ich habe in summe ~4 verschiedene librarys ausprobiert und das hat mir auch bei weitem am meisten Zeit gekostet, dieses Projekt umzusetzen. Die library für die ich mich am Ende entschieden habe ermöglicht es zudem auch Grafiken auf die Matrix zu zeichnen, was ich zwar am Ende nicht genutzt habe, aber sich trotzdem als ein sehr nettes feature herausstellt.

Probleme und Lösung:

- Ich habe zur LED Matrix so gut wie nichts brauchbares für C Code gefunden, nach gut 6h. Deswegen bin ich dann auf die Luma.LED_Matrix library ausgewichen, obwohl diese nur für python ausgelegt ist. Da aber alles andere mit C funktioniert habe ich überlegen müssen, wie ich die zu zeichnenden Punkte nun mit dieser library auf die Matrix bringe, ohne alles bestehende in Python neu zu gestalten. Am Ende habe ich dann einfach pipes verwendet um von einem .c programm, eine 2D Bitmap zu einem .py programm zu schicken, welches dann mithilfe der Luma library diese 2D Bitmap (aka. Die Koordinaten aller einzuschaltenden LEDs) ausliest und an die tatsächliche LED-Matrix weiterleitet. Die offizielle Dokumentation zur Luma library finden Sie hier:

https://luma-led-matrix.readthedocs.io/en/latest/install.html#

Der .py Code in meinem testcase zur LED Matrix ist NICHT VON MIR. Ich habe ehrlicherweise python noch gar nicht verwendet bis zu diesem Zeitpunkt und habe mich von ChatGPT unterstützen lassen, um die .py seite zum laufen zu bringen. Das pipe System habe ich ebenfalls gemeinsam mit ChatGPT erarbeitet.

- AD Wandler und Matrix werden beide via SPI angesprochen. Wie in meinem Video vorweggenommen, haben beide gemeinsam nicht funktioniert, auch wo die Matrix noch gemacht hat, was sie soll. Ansich sollten die 2 sich schon vertragen, da genau für diese Fälle die Chipselect Pins (CS0 & CS1) eine selektive Kommunikation ermöglichen -> CS0 wird auf GND gezogen, LED Matrix hört zu, CS1 wird auf GND gezogen, AD Wandler hört zu.

  Hierzu habe ich aber des öfteren Online gelesene, dass der Raspberry Pi gerne CS0 und CS1 auf GND zieht, obwohl nur einer davon auf GND gezogen werden soll. Das war vermutlich auch am Ende des Tages das Problem, warum beide gemeinsam nicht funktioren auf demselben SPI Bus. Wie gesagt, kann ich das leider nicht nachmessen, da ich kein Multimeter Zuhause habe 😟

- Die Matrix hat vom einen auf den anderen Tag nicht mehr funktioniert, und tut sie nachwievor nicht. Warum ich glaube, dass das Netzteil schuld ist, ist jener, dass die 8x32 Matrix nicht vom Raspberry Pi allein versorgt werden kann. Sie ist nur mit einer eigenen Versorgung funktionstüchtig und die 2 Transistoren auf dem Netzteil welches sie mir mitgegeben haben, sind (wo die Matrix noch funktioniert hat) so heiß geworden, dass ich mir davon fast eine Brandblase eingeholt habe … Da hätte ich definitiv die Matrix nicht so fordern sollen, wie ich es in meinen Tests getan habe.

  Es ärgert mich wirklich sehr, dass ich meinen Matrix Test nicht herzeigen kann. Ich kann Ihnen allerdings garantieren, dass mein test in “/test_cases/wokringLedTest_nowXandY_checkersboard” zu 100% funktioniert. Sie können diesen gerne selbst nachtesten wenn sie möchten, der Schaltungsaufbau verwendet diese Pins:

GPIO pin-outs

MAX7219 Devices (SPI)

The breakout board has two headers to allow daisy-chaining:

Board Pin	Name	Remarks	RPi Pin	RPi Function
1	VCC	+5V Power	2	5V0
2	GND	Ground	6	GND
3	DIN	Data In	19	GPIO 10 (MOSI)
4	CS	Chip Select	24	GPIO 8 (SPI CE0)
5	CLK	Clock	23	GPIO 11 (SPI CLK)

Hierfür einfach c_matrix_demo.c kompilieren und ausführen, dann das .py file ausführen.

> In der Konsole ist dann sichtbar, welche Daten .py empfangen hat und beschreibt die Matrix dann dementsprechend.

Falls Sie genauere Fragen zu meinem Projekt haben, melden Sie sich bitte bei mir. Ansonsten ein schönes Wochenende ^^.