



Hochschule für Technik,
Wirtschaft und Kultur Leipzig

FAKULTÄT INGENIEURWISSENSCHAFTEN

E492 - EMBEDDED SYSTEMS I

Doom und Trackmania auf einem Balance Board

<i>Autor</i>	Mat.Nr.
Justin Pöhl	79793
Max Tschirschwitz	80098

https://github.com/coffeebeanster/doom_balance_board/tree/main

Inhaltsverzeichnis

Abbildungsverzeichnis	i
1 Aufgabenstellung	1
2 Implementierung von Doom	1
3 Einrichtung des Raspberry Pi Pico	2
4 Der Source Code	4
4.1 Initialisierung	4
4.2 Das Kernelement: der „while True:“ -Loop	4
4.3 Auswertung der vom G-Sensor übermittelten Werte	4
4.4 Berechnung der Mausgeschwindigkeit	5
5 Einbau in das Balance Board	6
6 Anpassung des Source Codes und Funktionstests	7
6.1 Anpassung der Totpunkte	7
6.2 Problembehebung beim Keyboard-Buffer	7
7 Übersicht der verwendeten Materialien	9
8 Ausblick	10

Abbildungsverzeichnis

1 Prototyp unserer Steuerung	2
2 Berechnung der Mausgeschwindigkeit. Rot: rechts, Blau: links, Grün: Totpunkt . .	5
3 Einbau Sensor und Pi im Balance Board	6

1 Aufgabenstellung

Mithilfe eines G-Sensors sollte eine Bewegungssteuerung implementiert werden, mit der die Spiele Doom und Trackmania Nations Forever auf einem Balanceboard gespielt werden können. Zum Interfacing standen ein Raspberry Pi Pico und ein STM32 zur Verfügung.

2 Implementierung von Doom

Die Shareware Version lässt sich legal aus dem Online Archiv herunterladen:

https://archive.org/details/doom_20231012

Zur Implementierung nutzen wir das Open Source Projekt DosBox:

<https://www.dosbox.com/download.php?main=1>

DosBox ist ein Emulator, mit dem man DOS-Anwendungen und Spiele auf neueren Rechnern und Betriebssystemen verwenden kann.

Sobald DosBox gestartet wurde, landet der Benutzer im klassischen DOS-Prompt. Dort muss für den Emulator ein virtuelles Laufwerk erstellt werden. In folgendem Beispiel wird ein virtuelles Laufwerk mit dem Laufwerksbuchstaben c erstellt, auf dem sich die Daten des doom-Ordners befinden der sich im Root-Verzeichnis des Host-Laufwerks c findet:

```
mount c c:\doom
```

Anschließend kann mit dem Befehl **c:** auf das virtuelle c-Laufwerk gewechselt werden. Mit dem Befehl **doom** wird die dort befindliche Start-Datei für Doom geladen.

Um das ganze beim Start von DosBox zu automatisieren, muss die DosBox-Configfile editiert werden, sie befindet sich im Pfad:

```
C:\Users\[USERNAME]\AppData\Local\DOSSBox
```

Am Ende der Datei unter [autoexec] werden die oben genannten Befehle in chronologischer Reihenfolge eingetragen:

```
mount c c:\doom
c:
doom
exit
```

Durch **exit** bekommt das DOS-Prompt beim Beenden von Doom den Befehl, DosBox zu beenden.

3 Einrichtung des Raspberry Pi Pico

Die Einrichtung erfolgt mit Hilfe von CircuitPython (stable 8.9.2), welches auf dem Raspberry Pi Pico installiert wird. CircuitPython ist ein Fork von MicroPython von Adafruit. Ebenso ist unser G-Sensor (BNO085) von Adafruit. Dies macht es uns möglich mit den entsprechenden Bibliotheken den Pi mit unserem Sensor kompatibel zu machen. Der Pico wird an den Rechner angeschlossen, dabei wird die Bootsel-Taste gedrückt um den Pi in den BootLoader-Mode zu versetzen. Anschließend wird die MicroPython.uf2-Datei auf den Pico kopiert. Dieser startet sich anschließend neu und taucht als Massenspeichermedium „CIRCUITPY“ auf. Zur Programmierung wird kein Compiler benötigt. Der Pi bringt mit der CircuitPython-Firmware einen eigenen Interpreter mit, sodass der unveränderte Source-Code als „code.py“ direkt auf den Pico kopiert werden kann. Um zusätzliche Bibliotheken verwenden zu können, müssen diese lediglich in den **lib**-Ordner des Picos kopiert werden. Alle Bibliotheken findet man auf <https://circuitpython.org/libraries>. Die benötigten Bibliotheken für die Verbindung und Nutzung des Sensors sind „adafruit_bno08x“ und „adafruit_bus_device“.

Der Prototyp unserer Steuerung auf einem Breadboard war zu groß für eine Installation im Balanceboard.

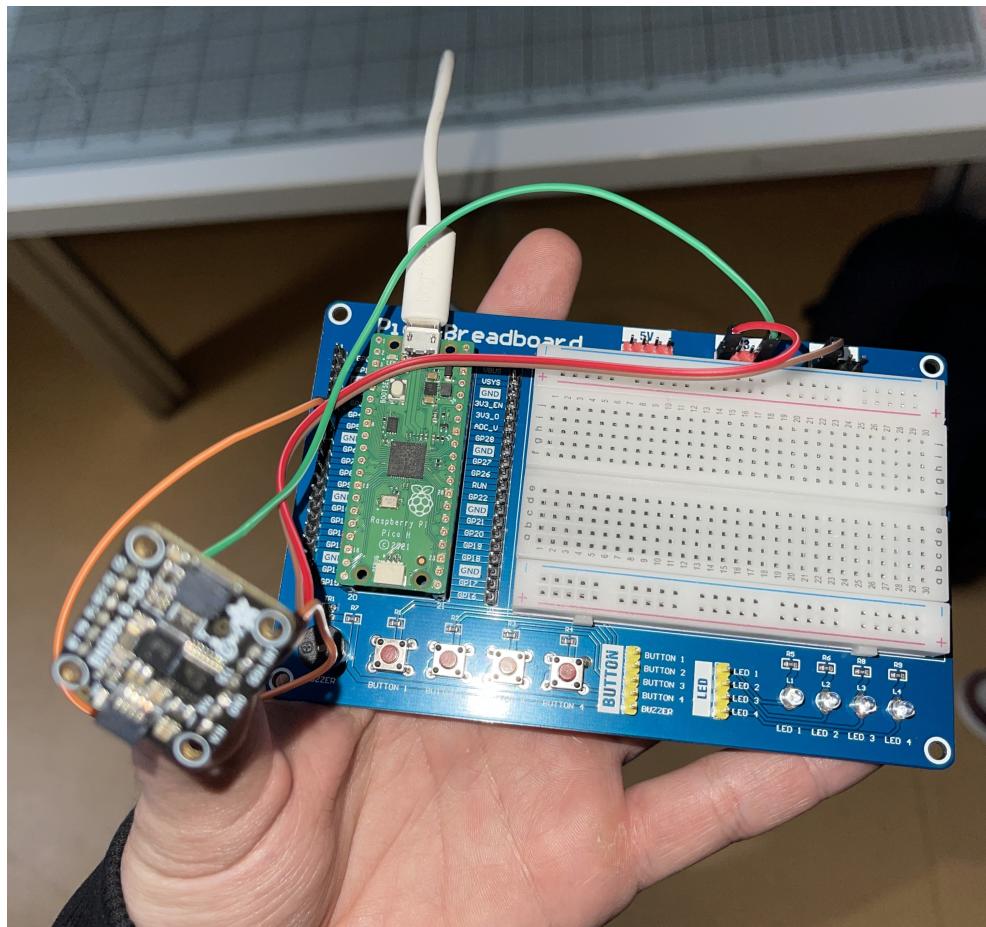


Abbildung 1: Prototyp unserer Steuerung

Um den Aufbau zu verkleinern entschieden wir uns dazu, den Sensor direkt mit dem Raspberry Pi zu verbinden und zum Betrieb dessen Onboard 3,3 V Spannungsquelle zu nutzen. Die Verbindung erfolgte über die folgenden Pins:

Raspberry Pi Pico	G-Sensor	Farbe
--------------------------	-----------------	--------------

<i>GND</i>	<i>GND</i>	<i>blau</i>
<i>3,3V</i>	<i>Vin</i>	<i>rot</i>
<i>3,3V</i>	<i>P0</i>	<i>braun</i>
<i>GP5</i>	<i>SDA</i>	<i>orange</i>

Über ein USB-Kabel haben wir eine serielle Verbindung zwischen Pi und Rechner hergestellt, welches auch gleichzeitig unsere Stromversorgung ist. Anschließend wird der Mu-Editor heruntergeladen und installiert. Beim ersten Start muss der Modus „CircuitPython“ ausgewählt und mit dem Button „Seriell“ das REPL geöffnet werden um mit der weiteren Konfiguration fortzufahren.

4 Der Source Code

Funktionsbeschreibung der/des Source Code*in (Gender unbekannt):

Der Source-Code ist in unserem Github Repository ersichtlich: https://github.com/coffeebeanster/doom_balance_board/tree/main

Diese Beschreibung soll eine Ergänzung zu den im Source Code enthaltenen Kommentaren sein.

4.1 Initialisierung

Kernelement des Programms ist der „while True:“ -Loop. Davor werden alle benötigten Variablen erstellt, Bibliotheken importiert und die UART-Verbindung zwischen G-Sensor und Port UART1:RX des Picos konfiguriert. Zudem wird der Benutzer über die Kommandozeile / das REPL gefragt, welches Spiel gespielt und wann das Balance Board justiert werden soll.

4.2 Das Kernelement: der „while True:“ -Loop

Im while-Loop werden laufend die vom G-Sensor übermittelten Werte abgefragt. Damit das Balance Board nicht zu empfindlich auf Wackeln reagiert, haben wir in die if-Bedingungen zur Triggerung der Tastendrücke und Mausbewegungen „Totpunkte“ einprogrammiert, sodass das Programm erst ab einem bestimmten Wert handelt. Sollte eine oder mehrere der Bedingungen durch eine erkannte Neigung des Balanceboards erfüllt werden, so wird der entsprechende Keycode oder eine Mausbewegung gesendet. Auch wird in der Programmschleife laufend die Mausgeschwindigkeit berechnet, falls im nächsten Zyklus die Maus bewegt werden soll.

4.3 Auswertung der vom G-Sensor übermittelten Werte

Da der Sensor die abgefragten Werte **YAW**, **PITCH** und **ROLL** in Relation zum Magnetfeld der Erde ausgibt (und wir nicht erwarten können, dass später jeder Benutzer das Balance Board nach Norden ausrichtet um Spielen zu können), haben wir die Variablen **yaw_start**, **pitch_start** und **roll_start** erstellt. Das Programm speichert die ersten vom Sensor übermittelten Werte als dauerhafte Referenz für den while-Loop ab. Dort werden die aktuellen Werte zyklisch erfasst und von den Startwerten subtrahiert. Die Differenz wird in den Variablen **yaw_aktuell**, **roll_aktuell** bzw. **pitch_aktuell** gespeichert. Diese wiederum werden dann entsprechend vom Programm ausgewertet.

4.4 Berechnung der Mausgeschwindigkeit

Ein weiteres Ziel war es, dass man sich bei Doom mit einer variablen Geschwindigkeit umsehen kann. Je stärker der Drehwinkel („yaw“) des Boards, desto schneller soll man sich im Spiel auch drehen können. Dies ermöglicht schnelle Drehungen wenn man sich umsehen und langsame präzise Drehungen, wenn man einen Gegner anvisieren möchte. Zur Berechnung der dazu benötigten Mausgeschwindigkeit haben wir zwei lineare Funktionen (eine für die Links- und eine für die Rechtsdrehung) erstellt:

```
mausgeschwindigkeit_links(yaw_aktuell) = -(5/4) * yaw_aktuell + (5/2)  
mausgeschwindigkeit_rechts(yaw_aktuell) = -(5/4) * yaw_aktuell - (5/2)
```

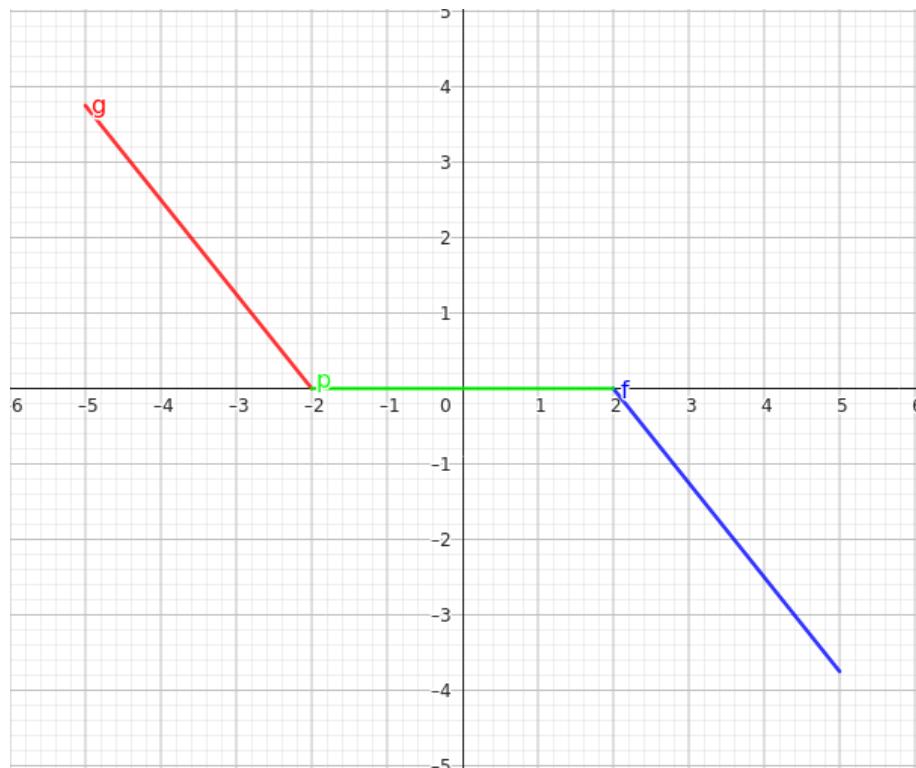


Abbildung 2: Berechnung der Mausgeschwindigkeit. Rot: rechts, Blau: links, Grün: Totpunkt

Die berechneten Mausgeschwindigkeiten werden in den Variablen `mausgeschwindigkeit_links` bzw. `mausgeschwindigkeit_rechts` gespeichert und können für den Folgezyklus verwendet werden.

5 Einbau in das Balance Board

Wir haben uns dafür entschieden, den G-Sensor und den Pi direkt in den Fuß des Balance Boards einzubauen. Das hat einen platzsparenden Zusammenbau der Komponenten erfordert. Dies hatte zur Folge, dass wir das beim Prototyping verwendete Breadboard komplett entfernt und, wie bereits in [3] beschrieben, den Raspberry Pi direkt mit dem G-Sensor verbunden haben. Das Plastik im Fuß des Balance Boards haben wir mit einem Heißluftgebläse etwas geschmolzen und eingedrückt. Dies ermöglichte es uns zum einen den G-Sensor gerade (parallel zur x-y-Ebene) zu verbauen und zusätzlich Platz zur Unterbringung des Raspberry Pi Pico zu generieren.



Abbildung 3: Einbau Sensor und Pi im Balance Board

Mit diesem Aufbau haben wir alle nötigen Komponenten direkt im Balance Board verbaut ohne die Stabilität und Robustheit zu verringern. Darüber hinaus führt nun nur noch ein Kabel vom Balance Board zum Rechner.

6 Anpassung des Source Codes und Funktionstests

6.1 Anpassung der Totpunkte

Nachdem der Python-Code und das Balance-Board fertig waren, unterzogen wir es einigen Funktionstests. Zuerst fiel uns auf, dass die gewählten Totpunkte vergrößert werden müssen. In den Füßen, stehend auf dem Balance Board hat man weniger Feingefühl als erwartet, sodass das Board auf unsere Bewegungen zu empfindlich reagiert hat. Ein kleiner Wackler mit den Füßen führte schon zu ungewollten Eingaben. Uns fiel auf, dass wir für Trackmania und Doom verschiedene Totpunkte einprogrammieren müssen. Durch verschiedene Kombinationen und dem Trial-and-Error-Verfahren kamen wir auf Werte, mit denen sowohl Doom als auch Trackmania spielbar sind.

6.2 Problembehebung beim Keyboard-Buffer

Beim Test von Doom fiel uns auf dass der Charakter im Spiel für eine gewisse Zeit einfach weiterlief, obwohl das Balanceboard keine Vorwärtsneigung mehr hatte. Das gleiche Verhalten zeigte sich auch bei allen anderen Eingaben: Diese hörten nicht auf, sondern hatten einen Nachlauf. So kam es auch bei Trackmania vor, dass der Wagen weiterhin beschleunigte, obwohl man das Balanceboard bereits zum Bremsen nach hinten neigte. Dieses Verhalten schlossen wir darauf zurück, dass das Balanceboard solange es eine Neigung registriert, laufend die entsprechenden Tastendrücke simuliert. Es ist das gleiche Verhalten wie wenn man mit einer extrem hohen Geschwindigkeit mehrmals eine Taste betätigt: Der Keyboard-Buffer enthält all diese gedrückten Tasten, diese müssen vom Computer erst verarbeitet werden. Es bildet sich eine „Warteschlange“, die abgearbeitet werden muss.

Um diesem Problem entgegenzuwirken, haben wir die Variablen

```
w_flushed = True  
s_flushed = True  
w_pressed = False  
s_pressed = False  
a_flushed = True  
d_flushed = True  
a_pressed = False  
d_pressed = False  
up_flushed = True  
up_pressed = False  
down_flushed = True  
down_pressed = False
```

```
right_flushed = True  
right_pressed = False  
left_flushed = True  
left_pressed = False
```

eingeführt.

pressed

Sobald das Balanceboard eine Bewegung erkannt und den entsprechenden Tastendruck gesendet hat, wird diese Variable auf **True** gesetzt. Sollte das Balanceboard im nächsten Zyklus der while-Schleife dieselbe Bewegung wieder registrieren, wird der Tastendruck somit nicht erneut gesendet. Wird diese Bewegung nicht mehr erkannt, so sendet das Balanceboard dass die Taste losgelassen wurde (**flush**) und setzt die Variable wieder auf False. Die Variablen dienen also als Merker für das Programm, ob der hier erkannte Tastendruck nicht vielleicht im letzten Zyklus bereits gesendet wurde.

flushed

Damit auch das Loslassen der Taste(n) nur einmal gesendet wird, haben wir zusätzlich die „Flush-Variablen“ eingeführt. Wenn das Balanceboard die entsprechende Neigung nicht mehr registriert, wird ein *button_release* an den Computer gesendet und die dazugehörige flush-Variable auf **True** gesetzt. So wird der Computer nicht mit *key-flushes* überströmt, wenn sich das Balanceboard zum Beispiel in seiner Ruheposition befindet. Auch hier haben wir also wieder einen Merker, ob der *flush* der Taste nicht schon im letzten Zyklus gesendet wurde.

Der Code ist also darauf ausgelegt, nur **Änderungen** der Zustände zu übermitteln.

7 Übersicht der verwendeten Materialien

Gegeben waren folgende Materialien:

- ATmega328P Entwicklungsboard
- Programmer ATmega8
- Raspberry Pi Pico W
- Raspberry Pi Pico H
- Adafruit BMO085 G-Sensor
- Pico Breadboard
- Jumper Wire
- USB-Kabel
- USB TTL UART Adapterkabel
- STM32

Die Entwicklung der Steuerung stand uns frei und so auch die Nutzung der Materialien. Wir haben uns für einen simplen und platzsparenden Aufbau entschieden was dazu geführt hat, dass wir nur den Raspberry Pi Pico H, den Adafruit BMO085 G-Sensor, vier Jumper Wire und ein USB-Kabel zur Stromversorgung und Datenübertragung gebraucht haben. Der Raspberry Pi Pico ist kompatibel mit der Adafruit CircuitPython Bibliothek, deswegen fiel uns die Auswahl des Mikrocontrollers recht leicht. Der G-Sensor ist perfekt für eine solche Anwendung, da er einfach zu nutzen ist und mit seiner umfangreichen Stabilitätserkennung und den signifikanten Bewegungsdetektoren passend für diese Anwendung geeignet ist. Seine Sensordaten wie Beschleunigungs- oder Rotationsgeschwindigkeitsvektoren stehen mit wenigen Zeilen Code zur Auslesung und Nutzung bereit.

8 Ausblick

Die Entwicklung der Steuerung war simpel, ging schnell von der Hand und ist leicht replizierbar. Die Steuerung funktioniert tadellos und ohne Verzögerungen oder Lags. Es ist eine lustige Idee, aber für den tatsächlichen Gebrauch eher unbrauchbar, da es die Funktionen im Spiel eher minimiert als diese verbessert. Im Vergleich zu einer Steuerung mit einem Joystick ist das Balance Board klobig und ungenau, kurzum: unpraktisch. Bei der Benutzung des Balance Boards ist es immer empfehlenswert sich irgendwo festzuhalten, da sonst die Stabilität fehlt. Das erschwert die Nutzung eines weiteren Gerätes, beispielsweise um in Doom zu schießen oder die Waffen zu wechseln.

Ein weiteres Problem stellt die Lenkung in Trackmania dar. Da durch das Balance Board nur Tastendrücke gesendet werden, hat man im Spiel stets einen vollen Lenkeinschlag. Durch kurzes Drücken der Pfeiltasten lässt sich im Spiel präzise genug fahren, dieses Konzept lässt sich aber nicht auf ein Balanceboard übertragen, sodass man zum Fahren von „Zickzack-Linien“ gezwungen ist. Zur Verbesserung der Spielerfahrung könnte man zukünftig eine Simulation eines analogen Joysticks im Balanceboard implementieren, sodass der Lenkeinschlag in Trackmania direkt auf die Neigung oder Drehung des Boards übertragen werden kann.