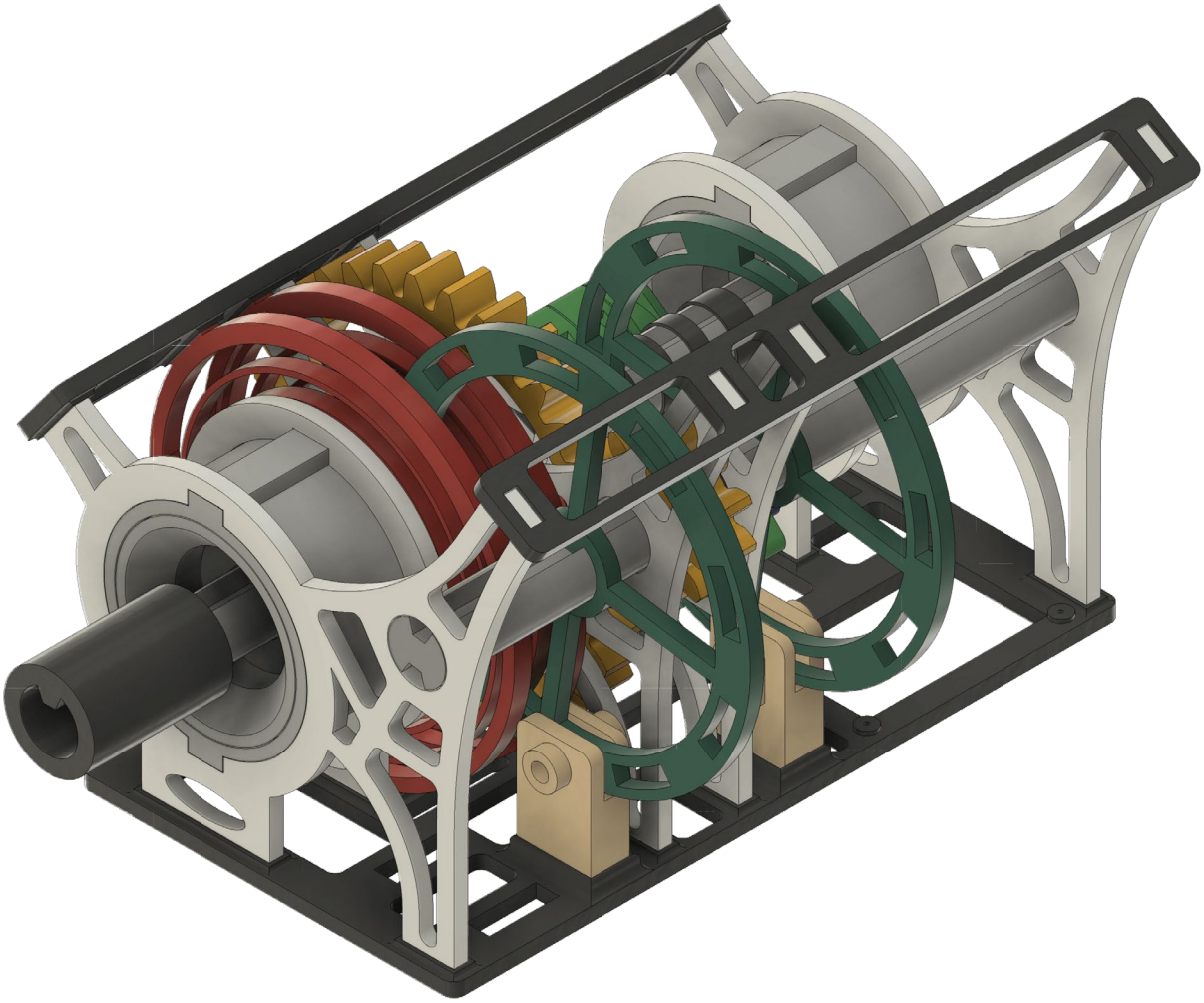
Racing Simulator

Kein fertiges Produkt, sondern ein Experiment:
Ein Racing-Simulator, der Technik erlebbar macht
– von CAD bis Code!



Ein spannendes Projekt zum Mitdenken von

Niklas Schwabauer

Inhaltsverzeichnis

1. Mein eigener Rennsimulator

Technik begeistert – insbesondere dann, wenn sie Präzision, Ästhetik und Ingenieurskunst miteinander vereint. Fahrzeuge renommierter deutscher Hersteller wie Porsche oder BMW sind weit mehr als bloße Fortbewegungsmittel: Sie sind Beispiele technischer Exzellenz, wirtschaftlicher Bedeutung und funktionaler Gestaltung. Als technikaffiner Autoliebhaber fasziniert mich dieser Anspruch – auch im Kontext des Motorsports, wo Disziplinen wie DTM, WEC oder die Formel 1 für Innovation, Leistung und Effizienz stehen.

Über diesen Weg entstand mein Interesse am Simracing. Professionelle Anbieter wie Simucube, Heusinkveld oder GSI haben den Bereich längst aus der Nische geführt – ihre Produkte stehen für Präzision und Qualität, sind aber für den Rennsport typisch auch preislich in diesem Segment angeordnet. Die Idee war daher klar: Ein eigenes System entwickeln. Nicht als Nachbau kommerzieller Lösungen, sondern als technisches Experiment, in dem das Lernen und Verstehen im Vordergrund steht.

So entstand mein selbst entwickelter Rennsimulator – ein interdisziplinäres Projekt, das Mechanik, Elektronik und Software vereint. Grundlage war der Wunsch, praxisorientiert zu arbeiten: Statt Theorie dominierte hier die Anwendung – jedes Bauteil wurde konstruiert, gedruckt, getestet und optimiert. Mit Fusion 360 entstand das mechanische Design, der 3D-Druck wurde über Bambu Studio umgesetzt. Ich entwickelte eigene Federsysteme, arbeitete mit Fotowiderständen und realisierte die Signalverarbeitung mithilfe eines ESP8266-Mikrocontrollers und eines ADS1115-AD-Wandlers. Ziel war es, Sensordaten möglichst präzise zu erfassen, zu analysieren und digital in Eingaben umzusetzen.

Mein technisches Vorwissen entstand schrittweise: von LEGO über fischertechnik bis hin zu Arduino-Projekten und ersten CAD-Erfahrungen mit FreeCAD. Doch vieles war neu – insbesondere die Kommunikation via Bluetooth, die Echtzeit-Auswertung analoger Signale und die Integration über Tools wie vJoy zur Weitergabe von Steuerbefehlen an Spiele wie Forza Horizon.

Begleitet wurde das Projekt durch den Einsatz von ChatGPT – nicht als Ersatz für eigenes Denken, sondern als effektives Werkzeug zur Problemlösung: bei Fehlersuche, Codeinterpretation und der konzeptionellen Planung technischer Teilbereiche. Diese Form der digitalen Unterstützung ermöglichte eine effiziente, lösungsorientierte Herangehensweise – vergleichbar mit einer interaktiven technischen Rechercheplattform.

Das Ergebnis ist kein marktfertiges Produkt, sondern ein durchdachtes, technisch tiefgehendes System. Es soll nicht zum Nachbau anregen, sondern zum Verstehen. Ziel ist es, komplexe Abläufe und Fachbegriffe so aufzubereiten, dass sie nachvollziehbar und greifbar werden – auch für jene, die sich bislang nicht intensiv mit Mikrocontrollern, Sensorik oder CAD befasst haben.

2. Überblick über den Aufbau

Der selbstentwickelte Simulator **folgt** einem einfachen, aber technisch durchdachten Aufbau: Eine direkt gelagerte Lenkeinheit, ein selbstgedruckter Drehfederelement-Satz, ein optisch arbeitender Encoder sowie eine Bluetooth-Anbindung an den PC bilden das funktionale Grundgerüst. Der Schwerpunkt des Projekts lag auf einem möglichst modularen, vollständig selbst herstellbaren Aufbau – insbesondere im Bereich der Mechanik und Sensorik.

2.1 Grundkonzept des Lenkmechanismus

Zentrales Element ist die rotierende Lenkwelle, die über zwei gegenüberliegend gespannte, spiralförmige PLA-Torsionsfedern gegen ihre Drehrichtung arbeitet. Die Kraftübertragung erfolgt direkt, ohne zwischengeschaltete Getriebestufen zwischen Hand und Feder – ein Ansatz, der für lineare Rückstellmomente und eine kompakte Bauform sorgt. Die Verbindung zur Sensorik erfolgt über eine per Zahnrad übersetzte Nebenwelle, auf der die Encoderscheiben montiert sind.

2.2 Sensorik und Signalauswertung

Zur Erfassung der Rotationsbewegung kommt ein selbstentwickeltes optisches Encodersystem zum Einsatz. Zwei leicht versetzt angeordnete, aus PLA gefertigte Codierscheiben mit abwechselnd lichtdurchlässigen und lichtundurchlässigen Segmenten erzeugen bei Drehung ein nahezu sinusförmiges Signal. Die Lichtstärke wird analog erfasst und zur Bestimmung der Drehrichtung sowie der aktuellen Stellung genutzt. Die Genauigkeit des Systems hängt dabei stark von der Rotationsgeschwindigkeit und der Auslesefrequenz ab – ein Aspekt, der im weiteren Verlauf noch vertieft wird.

2.3 Kommunikation mit dem PC

Die Verarbeitung der analogen Signale erfolgt auf einem Mikrocontroller (ESP8266), der sie per Bluetooth an den PC überträgt. Dort werden die Werte mit Hilfe von vJoy in virtuelle Gamepad-Signale umgesetzt. Die Verbindung über USB ist parallel für die Programmierung und Konfiguration nutzbar. Die Integration in Spiele ist grundsätzlich möglich, wenngleich bestimmte Grenzen bestehen – etwa in Bezug auf Reaktionsgeschwindigkeit und Präzision.

2.4 Technische Herausforderungen und Grenzen

Schon während des Aufbaus wurde deutlich, dass das System seine Grenzen nicht im mechanischen Aufbau, sondern in der Datenverarbeitung erreicht: Die Kombination aus selbstgebautem Encoder, analoger Signalverarbeitung und limitierter Abtastrate des Mikrocontrollers führt zu Genauigkeitsverlusten – insbesondere bei schnellen Bewegungen. Diese Einschränkungen schränken den praktischen Einsatz als vollwertiger Simulator ein, stellten aber gleichzeitig den gewünschten technischen Reiz dar.

Trotz (oder gerade wegen) dieser Herausforderungen wurde das Projekt als erfolgreich abgeschlossen gewertet. Der Fokus lag nicht auf industrieller Reproduzierbarkeit, sondern auf einem vollständigen, funktionalen Eigenentwurf mit Lerncharakter – inklusive mechanischer Konstruktion, Sensorikentwicklung, Signalverarbeitung und PC-Anbindung.

3. Die Mechanik im Detail

Wie lassen sich Lenkung, Kraftübertragung und Positionsmessung mechanisch lösen – ohne auf teure Fertigteile zurückzugreifen? In diesem Kapitel geht es um genau diese Frage. Alle mechanischen Komponenten des Projekts wurden entweder selbst entwickelt oder mit dem 3D-Drucker hergestellt. Im Zentrum steht dabei die Lenkwelle als tragendes Bauteil, auf das alle weiteren Funktionen aufbauen: Kraftübertragung, Drehmomentmessung, Lagerung und Signalübertragung.

3.1 Die Lenkwelle: Mehr als nur ein Träger

Zentrales Element ist die rotierende Lenkwelle, die über zwei gegenüberliegend gespannte, spiralförmige PLA-Torsionsfedern gegen ihre Drehrichtung arbeitet. Die Kraftübertragung erfolgt direkt, ohne zwischengeschaltete Getriebestufen zwischen Hand und Feder – ein Ansatz, der für lineare Rückstellmomente und eine kompakte Bauform sorgt. Die Verbindung zur Sensorik erfolgt über eine per Zahnrad übersetzte Nebenwelle, auf der die Encoderscheiben montiert sind.

3.2 Zahnräder: Kraftübertragung mit Übersetzung

Die eigentliche Drehbewegung wird über zwei 3D-gedruckte Zahnräder übertragen: Ein kleineres Zahnrad (11 Zähne) ist direkt auf der Lenkwelle montiert, ein größeres (33 Zähne) greift in dieses ein. Damit ergibt sich eine Übersetzung von etwa 3 : 1 – langsame Lenkbewegungen erzeugen so eine verstärkte Reaktion im Sensorbereich, was die Auflösung der Auswertung verbessert.

Beide Zahnräder greifen sauber ineinander. Zwar gibt es durch die Fertigungstoleranzen im 3D-Druck ein geringes Spiel, doch dieses ist in der Praxis vernachlässigbar – insbesondere, da andere Teile des Systems wie die Torsionsfederlagerung oder die Schleifkontakte ohnehin deutlich reibungsintensiver arbeiten. Die Zahnräder sind zwar hörbar, aber nicht störend – lauter ist die Reibung zwischen Encoderscheiben und Sensoraufnahmen. Diese Reibung ist dabei nicht nur ein Nebeneffekt, sondern sogar gewollt: Um die Sensoren zuverlässig gegen Fremdlicht zu schützen, muss der Lichtspalt möglichst eng geführt werden – was unweigerlich zu leichtem Schleifkontakt führt.

3.3 Stromübertragung über Schleifkontakte

Um Tasten auf dem Lenkrad abfragen zu können, müssen Signale von der rotierenden Lenkwelle zum feststehenden Mikrocontroller übertragen werden. Die Lösung: Ein Schleifkontakt – umgesetzt mit klassischen Mitteln, aber untypischer Ausführung.

Zunächst wurden Kupferdrähte abisoliert und spiralförmig um die Lenkwelle gewickelt. Dadurch entstehen geschlossene Kontaktbereiche, die ringförmig auf der Welle aufliegen. Um die Leitfähigkeit zu erhöhen, wurden diese Ringe zusätzlich verzinnt und anschließend mit dünneren Kabeln verlötet, die zur Elektronik führen. Die Kontaktierung selbst erfolgt über federnd gelagerte Abnehmer, die von unten im 45°-Winkel gegen die rotierende Welle drücken. Die Federn sind so angeordnet, dass sie tangential zur Welle wirken und gleichzeitig stabil geführt werden – ganz ohne äußere Führungsschiene. Durch die enge Platzierung der Abnehmer ergibt sich ein fast spielfreier Aufbau.

Das ganze Modul ist an einer zusätzlichen (deutlich dünneren) PLA-Stange montiert, die parallel zur Lenkwelle verläuft und alle Abnehmer stabil hält. Insgesamt sind zwei Gruppen verbaut: Eine mit zwei, eine mit drei Abnehmern – getrennt durch eine 4 mm starke Distanzstruktur. Die Kontaktqualität war im Betrieb stabil genug, da für die Schalterabfrage keine dauerhafte Signalübertragung nötig ist. Kurzzeitige Kontaktverluste werden durch die Trägheit des Tastendrucks überbrückt.

3.4 Das Kugellager: Leichtlauf mit Airsoft-Kugeln

Die Lagerung der Lenkwelle erfolgt über ein eigens entwickeltes 3D-gedrucktes Kugellager mit Airsoft-Kugeln als Wälzkörper. In zwei Kammern wurden je 12 Kugeln eingesetzt, also

insgesamt 24 pro Lager. Die Kugeln laufen trocken – also ohne Schmierung – was zwar zu einer gewissen Geräuschentwicklung führt, aber in der Praxis kaum ins Gewicht fällt. Entscheidend ist: Der Lauf ist extrem leichtgängig. Die Welle dreht sich mühelos und läuft nach dem Andrehen deutlich nach – ganz wie bei einem echten Kugellager. Axiale und radiale Kräfte werden durch die doppelte Lagerung zuverlässig abgefangen. Brüche oder Materialversagen wurden während des Tests nicht beobachtet.

3.5 Die Torsionsfedern: Gedruckte Elastizität

Die Rückstellkraft des Lenkrads wird durch spiralförmige PLA-Torsionsfedern erzeugt. Diese wurden parametrisch in Fusion 360 modelliert und umschließen die Lenkwelle formschlüssig. In ihrer Mitte besitzen die Federn eine runde Aufnahme mit zentralem Loch und Aussparungen, die exakt zur Wellengeometrie passen – so wird das Drehmoment direkt übertragen. Der äußere Bereich ist ebenfalls mit einer gezielten Geometrie versehen, um die Feder am Gehäuse fixieren zu können.

Eine genaue Kraftmessung wurde nicht durchgeführt. Subjektiv ist die Rückstellkraft deutlich spürbar, aber nicht übermäßig – vergleichbar mit einem leichten Fahrzeuglenkrad ohne Servounterstützung. Da die Federn im eingebauten Zustand gerade belastet sind, lassen sich Unterschiede in der Steifigkeit schwer nachprüfen. Wahrscheinlich bestehen leichte Varianzen durch Drucktemperatur, Wandstärke oder Restspannung im Material. Typische Kraftverläufe solcher Torsionsfedern folgen einer nahezu linearen Kennlinie im elastischen Bereich, idealisiert durch die Beziehung:

$$M = k \cdot \theta$$

wobei M das Drehmoment, θ der Verdrehwinkel und k die Torsionsfederkonstante ist. Die Form der Feder bestimmt maßgeblich die Steifigkeit.

3.6 Montage und Wartung

Trotz des komplexen Aufbaus zeigten sich im Betrieb nur wenige Probleme. Lediglich das Gehäuse lockerte sich gelegentlich – ein Problem, das sich durch Einkleben dauerhaft beheben ließe, aber derzeit aus Wartungsgründen bewusst offen bleibt. Die restliche Mechanik ist überraschend robust: Weder mussten Bauteile ersetzt werden, noch traten relevante mechanische Schwächen auf. Ein Indiz dafür, dass die Planung sorgfältig war – und dass 3D-Druck in Kombination mit handwerklicher Nacharbeit durchaus eine ernstzunehmende Option für funktionale Mechanik darstellt.

3.7 Einordnung: Was war spannend – was aufwendig?

Besonders spannend war das Experimentieren mit den Encoderscheiben: Zahlreiche Varianten wurden gedruckt und getestet, um eine möglichst präzise und dennoch robuste Auswertung zu ermöglichen. Hier zeigte sich unmittelbar der Zusammenhang zwischen Geometrie, Drehgeschwindigkeit und Auslesegenauigkeit – und auch die Grenze des technisch Machbaren. Wie bei der Quantenmechanik gibt es gewissermaßen eine Art Messunschärfe: Die exakte Position, Geschwindigkeit und Richtung können nicht beliebig genau gleichzeitig bestimmt werden – zumindest nicht bei dieser Hardware.

Aufwendig war vor allem die Entwicklung der Schleifkontakte. Erste Versuche mit Alufolie scheiterten erwartungsgemäß an Haltbarkeit und Leitfähigkeit. Die Lösung mit verzinnten Kupferdrähten sieht nicht nur sauber aus, sondern funktioniert auch – war aber in der Umsetzung langwierig. Jeder Ring, jeder Kontakt musste exakt positioniert, verlötet und stabilisiert werden. Am Ende aber hat sich der Aufwand gelohnt: Die Mechanik bildet das Rückgrat des Projekts – und macht auch auf den zweiten Blick klar, wie viel Technik in einem simplen Lenkrad stecken kann.

4. Der optische Encoder: Licht, Schatten und Richtung

Die präzise Bestimmung der Lenkradposition ist das zentrale Element jedes Simulators. In meinem Fall fiel die Wahl nicht auf einen handelsüblichen Encoder, sondern – ganz im Sinne des Projekts – auf einen selbstgebauten, rein optisch funktionierenden Drehgeber. Ein bewusster Schritt, der nicht nur technische Herausforderungen mit sich brachte, sondern auch eine Menge an Lerngelegenheiten bot.

4.1 Die Encoderscheiben – Lichttore aus PLA

Kernstück des Systems sind zwei identische, selbst designte Encoderscheiben. Diese bestehen aus schwarzem PLA und setzen sich jeweils aus einem Innen- und Außenring zusammen. Der Innenring dient der Befestigung an der drehenden Welle, während der Außenring die entscheidenden Aussparungen trägt.

Jede Scheibe besitzt insgesamt 18 gleich große Segmente, von denen 9 lichtdurchlässig (mit Aussparung) und 9 lichtundurchlässig sind. Die lichtdurchlässigen Elemente haben eine Breite von 5 mm und sind mittig im Außenring eingelassen – sie wirken wie leicht gebogene Rechtecke, was durch die Kreisform bedingt ist. Der Außendurchmesser der Scheibe beträgt 105 mm, der Innendurchmesser des Außenrings liegt bei 85 mm. Die Dicke der gesamten Scheibe beträgt 4 mm.

Die genaue Positionierung und Symmetrie der Segmente war das Ergebnis mehrerer Iterationen. Anfangs experimentierte ich mit verschieden großen und geformten Ausschnitten (u.a. kreisförmig oder schmaler), doch am Ende setzte sich ein gleichmäßig segmentiertes Design durch. Die damit erreichte Auflösung entsprach rechnerisch 20 Grad pro Segment (360° ÷ 18), was zwar unter meiner ursprünglich angestrebten Zielgröße von 5 Grad blieb, aber aufgrund der vielen praktischen Einflüsse (Drehgeschwindigkeit, Latenzen, Toleranzen) realistischer umsetzbar war.

4.2 Das Test-Rig – Prototyp für Lichtspiele

Noch vor dem eigentlichen Simulator entstand ein kleines Test-Rig: eine einfache Kurbelkonstruktion mit montierter Scheibe und einem ersten Sensorhalter. Hier konnte ich die Grundidee der optischen Abtastung überprüfen – per Hand angetrieben, manuell ausgerichtet und Schritt für Schritt validiert. Dieses Mini-Projekt diente als wichtiger Machbarkeitsnachweis, bevor die Idee später in das eigentliche Lenksystem überführt wurde.

4.3 Sensorik – Licht und Widerstand

Jede Encoderscheibe wird mit einem Sensorpaar aus LED und Fotowiderstand abgetastet. Die LED sitzt auf der einen Seite der Scheibe, der Fotowiderstand exakt gegenüber. Bei Drehung der Scheibe fällt im Rhythmus der Aussparungen entweder Licht auf den Sensor oder wird blockiert – daraus ergibt sich ein analoger Lichtwert, der später digital interpretiert wird.

Diese Sensorpaare sind als einzelne Module aufgebaut: LED und Sensor sind jeweils fest montiert, und das Modul wird exakt so ins Gehäuse eingesetzt, dass die Encoderscheibe mittig durchläuft. Der Abstand zwischen LED, Scheibe und Sensor wurde bewusst minimal gewählt, um Streulicht oder LED-Helligkeitsschwankungen zu minimieren.

4.4 Zwei Scheiben für die Drehrichtung

Ein besonders cleverer Aspekt des Designs: Zur Erkennung der Drehrichtung werden zwei Encoderscheiben verwendet – beide auf derselben Welle montiert, aber zueinander um 90° phasenversetzt (π/2). Damit ergibt sich bei Drehung ein zeitlicher Versatz zwischen den beiden Sensorsignalen, aus dem sich die Richtung eindeutig ableiten lässt – ganz im Prinzip eines Quadraturgebers.

Obwohl man alternativ auch zwei Sensoren an einer Scheibe hätte platzieren können, war die Entscheidung für zwei separate Scheiben im Nachhinein sogar vorteilhaft: So konnten

identische Bauteile doppelt genutzt werden – ideal für ein Prototypensystem, in dem Wiederverwendbarkeit und Klarheit im Vordergrund standen.

4.5 Auswertung auf dem ESP8266

Die Signale beider Fotowiderstände werden analog über den ADS1115 ausgelesen, einem hochauflösenden A/D-Wandler, der via I²C mit dem ESP8266 kommuniziert. In der Software wurden die kontinuierlichen Helligkeitswerte anschließend durch festgelegte Schwellwerte (Stichwort: Thresholding) in digitale Zustände („Licht“ / „kein Licht“) überführt.

Die Herausforderung: Bei schnellen Drehbewegungen wurde der Schwellwert teilweise nicht rechtzeitig erreicht, was zu Fehlinterpretationen oder „vertauschten“ Signalzuständen führte. Besonders problematisch war dies, wenn die Werte beider Sensoren sich annäherten oder nahezu gleichzeitig die Schwelle über- bzw. unterschritten – die Richtungserkennung versagte dann.

Ein möglicher Grund: optische Dämpfungseffekte. Ähnlich wie bei einem PWM-Signal könnte bei schnellen Bewegungen das Licht nicht konstant genug durchdringen oder vollständig blockiert werden, was zu verrauschten, mittleren Signalwerten führte.

4.6 Erkennung & Genauigkeit

Die Drehrichtung ließ sich bei korrekt justierten Schwellenwerten und moderater Drehgeschwindigkeit zuverlässig erkennen. Eine feinere Justierung der Grenzwerte hätte theoretisch die Auflösung erhöht, gleichzeitig aber die Empfindlichkeit gegenüber Geschwindigkeit verstärkt. Hier zeigte sich das grundlegende Dilemma:

„Maximale Genauigkeit bedeutet minimale Geschwindigkeit – maximale Geschwindigkeit bedeutet minimale Genauigkeit.“

Trotzdem war das System in der Praxis nutzbar – solange sich die Bewegung im realistischen Rahmen eines Rennspiels bewegte, funktionierte die Erkennung überzeugend.

4.7 Softwareseitige Umsetzung und vJoy

Die Auswertung erfolgte über ein Python-Skript, das die eingehenden Werte interpretierte. Die erste Encoderscheibe lieferte den Abtastimpuls: Bei jedem Wechsel von Hell zu Dunkel (bzw. umgekehrt) wurde die interne Zählposition um eins erhöht oder verringert – abhängig von der Zustandsänderung der zweiten, versetzten Scheibe. Letztere diente rein der Richtungserkennung, trug aber indirekt ebenfalls zur Positionierung bei.

Die berechnete Position wurde anschließend per vJoy-Schnittstelle an den PC übergeben und dort als virtuelles Gamepad-Signal interpretiert. In vielen Spielen war dies problemlos nutzbar – lediglich in Spielen wie Forza Horizon wäre zusätzlich eine umfassende Treiberumgebung nötig gewesen, die ich bewusst nicht installieren wollte aufgrund von potenziellen Anticheat-Problemen in anderen Spielen.

4.8 Fazit – Funktion vs. Erfahrung

Der selbstgebaute optische Encoder funktionierte überraschend zuverlässig – zumindest im Rahmen der gewählten technischen Mittel. Kleinste Bewegungen wurden nicht immer erkannt, die Genauigkeit variierte mit der Drehgeschwindigkeit, und natürlich war das System nicht mit einem industriellen Drehgeber vergleichbar.
Aber: Es war mein Design, mein Experiment, mein Fortschritt.
Ein kommerzieller Encoder hätte zweifellos besser funktioniert. Doch Ziel dieses Projekts war nicht Effizienz – sondern Verständnis. Und genau das wurde mit jedem Ausschnitt, jedem Sensorwert und jedem Aha-Moment tiefer.

5. Signalübertragung per Bluetooth

Wer glaubt, dass die größte Herausforderung eines DIY-Simulators im mechanischen Bau liegt, wird schnell eines Besseren belehrt – denn auch das Thema Datenübertragung verdient Aufmerksamkeit. Wie kommen die Lenksignale vom selbstgebauten Eingabegerät eigentlich ins Spiel? Die Antwort: erstaunlich unspektakulär, aber absolut funktional – über Bluetooth. Ein simples HC-05-Modul übernimmt die Rolle des stillen Vermittlers zwischen Mikrocontroller und PC – und leistet dabei mehr, als man auf den ersten Blick vermuten würde.

5.1 Warum eigentlich Bluetooth?

Die Entscheidung, Bluetooth als Übertragungsweg zu nutzen, war keine reine Frage der Technik, sondern vielmehr das Ergebnis einer pragmatischen Abwägung. USB schied frühzeitig aus – nicht etwa, weil der verwendete Mikrocontroller, ein ESP8266 (Wemos D1 mini), keinen USB-Anschluss besäße, sondern weil dieser nur zur Stromversorgung genutzt werden kann. Als Eingabegerät kann sich der ESP nicht ohne weiteres gegenüber dem PC ausgeben. Auch Wi-Fi wurde bewusst ausgeschlossen: Zwar hätte es eine deutlich höhere Übertragungsgeschwindigkeit ermöglicht, doch im Kontext von grafikintensiven Spielen war eine stabile Internetverbindung wichtiger als interne Datenraten – und genau diese hätte eine WLAN-Übertragung stören können.

Entscheidend war letztlich aber auch der Blick in die Bastelkiste: Ein HC-05-Bluetooth-Modul aus einem alten Arduino-Set lag schon länger ungenutzt herum – und bot sich damit als ideale Lösung an. Einfach zu verschalten, zuverlässig und für serielle Daten wie geschaffen. Zwar ist Bluetooth in puncto Geschwindigkeit nicht konkurrenzfähig zu modernen Controllern wie dem Xbox Wireless, doch für diesen Anwendungsfall reichte es vollkommen aus. Die Verbindung ist stabil, das Pairing funktioniert nach kurzer Wartezeit, und sobald der Kanal steht, wird jede Lenkbewegung zuverlässig übertragen. Kurz gesagt: Bluetooth war nicht nur die einfache, sondern auch die sinnvolle Lösung.

5.2 Die Hardware: Einfach, aber effektiv

Der HC-05 wurde direkt an den ESP8266 angeschlossen – ohne zusätzliche Spannungsquelle, direkt über RX/TX. Trotz vieler Internetbeiträge, die auf potenzielle Probleme bei der Stromversorgung hinweisen, lief alles stabil. Der verwendete D1 Mini scheint ausreichend Leistung bereitzustellen – oder anders gesagt: Glück gehabt, aber mit System.

Die Verkabelung war unkompliziert – einzig die klassische RX-TX-Verwirrung kostete kurzzeitig Nerven, war aber schnell behoben. Auch die Flash-Problematik (während RX und TX belegt sind) wurde elegant gelöst: Zwei kleine Taster auf der Platine unterbrechen manuell die Verbindung zum HC-05, wenn ein neues Programm aufgespielt werden muss. Einfach, clever, zuverlässig.

5.3 Die Software auf dem ESP: AD-Wandlung, Schmitt-Trigger, Encoder-Logik

Auf dem ESP8266 läuft ein Arduino-Sketch, der mit dem I²C-basierten ADS1115-Wandler vier analoge Eingänge ausliest. Für das Lenkrad relevant sind hier insbesondere zwei Kanäle: Die A- und B-Signale eines mechanischen Quadratur-Encoders. Diese werden digitalisiert und dann mithilfe eines softwareseitigen Schmitt-Triggers geglättet.

Der Trigger vermeidet zufällige Schaltflanken durch Wackler oder Störungen. Erst wenn eine Spannung einen definierten Schwellenwert über- oder unterschreitet, wird ein Zustandswechsel angenommen – konkret hier mit folgenden Grenzen:

```
#define thresholdHighA 15000
#define thresholdLowA 5000
```

Die Encoder-Auswertung selbst erfolgt durch Vergleich der aktuellen und vorherigen Zustände der A- und B-Spuren. Je nachdem, in welcher Reihenfolge sie ihre Pegel ändern, wird entweder ein Schritt im oder gegen den Uhrzeigersinn gezählt.
Das Ergebnis: eine variable "position", die die Lenkbewegung in diskrete Schritte auflöst – und schließlich als serieller Wert (mit Serial.println(position * 25);) über Bluetooth verschickt wird.

5.4 Die Empfängerseite: vJoy als Gamecontroller-Emulator

Auf dem PC lauscht ein kleines Python-Skript der seriellen Verbindung. Sobald der HC-05 vom Betriebssystem erkannt und einem COM-Port zugewiesen wurde (z.B. COM6), beginnt das Skript mit dem Auslesen der Werte:

```
line = ser.readline().decode('utf-8').strip()
lenkradWert = int(line)
mapped_value = int((lenkradWert / 1023) * 32767)
```

Dabei wird der gelesene Encoderwert auf den Wertebereich einer vJoy-Achse (0–32767) skaliert. Anschließend schreibt das Skript den Wert auf die virtuelle X-Achse eines vJoy-Gamecontrollers. Spiele wie Forza Horizon interpretieren das Ganze dann wie ein echtes Lenkradsignal – und das ganz ohne teure Hardware oder komplexe Treiber.
Das Python-Skript selbst ist extrem schlank, robust und läuft im Grunde „einfach so“. Kein Schnickschnack – aber eben auch keine Probleme.

5.5 Robustheit, Geschwindigkeit, Alltagstauglichkeit

Die Verbindung ist überraschend stabil – auch wenn sie im Vergleich zu professionellen Controllern wie einem Xbox Wireless Pad beim Verbindungsaufbau etwas träge wirkt. Doch: Einmal verbunden, läuft alles sauber.

Da Latenzen für dieses Projekt keine kritische Rolle spielten, wurden keine weiteren Optimierungen in Richtung Echtzeitübertragung umgesetzt – eine bewusste Entscheidung zugunsten von Einfachheit und Nachvollziehbarkeit.

Wichtig: Das Setup ist keine Plug-and-Play-Lösung, sondern bewusst so offen gehalten, dass man daraus lernen kann. Das Ziel war nie, ein perfektes kommerzielles Produkt zu schaffen – sondern ein funktionierendes, verstehbares System, das zeigt, wie man vom elektrischen Signal zur Spielsteuerung gelangt.

5.6 Reflexion: Besser geht’s (fast) nicht

Rückblickend war die Entscheidung für Bluetooth genau richtig. Es war schnell verfügbar, einfach integrierbar und erfüllte seinen Zweck zu 100 %. Die Kombination aus ESP, ADS1115, Encoder, HC-05 und Python ist schlank, logisch und stabil.

Natürlich gäbe es noch Optimierungsmöglichkeiten: feinere Triggergrenzen, softwareseitige Glättung der Werte, adaptive Mapping-Funktionen – aber nötig ist das alles nicht. Das System funktioniert. Punkt.

5.7 Exkurs: Schmitt-Trigger, ADS1115, vJoy – ein kurzer Technikblick

Schmitt-Trigger:
Ein Schmitt-Trigger ist ein Logikbaustein, der ein Signal erst dann als „geändert“ betrachtet, wenn es eine obere oder untere Schwelle überschreitet. Diese Technik wird genutzt, um verrauschte oder unsaubere Signale zu stabilisieren – insbesondere bei mechanischen Kontakten oder analogen Wandlern.
ADS1115:
Der ADS1115 ist ein 16-bit-Analog-Digital-Wandler mit vier Eingängen, I²C-Schnittstelle und hoher Genauigkeit. Er eignet sich ideal für Sensoranwendungen, bei denen Spannungsschwankungen präzise erkannt werden sollen – z.B. bei Potentiometern, Hall-Sensoren oder eben Encoder-Signalen.
vJoy:
vJoy ist ein virtueller Gamecontroller-Treiber für Windows. Über eine API kann man Achsen, Buttons und andere Eingabeelemente simulieren. Das macht vJoy zum idealen Bindeglied zwischen Bastelprojekten und kommerziellen Spielen – ohne dass man eigene Treiber entwickeln muss.

6. Reflexion & Weiterdenken

Manches funktioniert direkt. Manches nicht. Entscheidend ist, was man daraus macht.

Was bleibt hängen, wenn das Projekt abgeschlossen ist? Diese Frage stellt sich spätestens dann, wenn der letzte Druck abgeschlossen, der letzte Code geschrieben und die letzte Schraube angezogen wurde. In diesem Kapitel geht es nicht darum, den Simulator an käufliche Fertigprodukte zu messen. Es geht um Eigenleistung, um Ideen, und darum, was sich daraus entwickelt – technisch wie gedanklich.

6.1 Lieblingsbaustellen und elegante Lösungen

Manche Bauteile sind nicht nur funktional, sondern machen einfach Freude – sei es beim Konstruieren, beim Zusammenbauen oder im Ergebnis. Zwei Elemente haben sich dabei besonders hervorgetan: Zum einen die Spiral-Torsionsfedern, die nicht nur ungewöhnlich gewählt, sondern auch überraschend effizient waren. Sie verleihen der Lenkung eine gleichmäßige Rückstellkraft, ohne dabei viel Platz zu benötigen – und vor allem: Sie funktionieren besser, als zunächst erwartet.

Zum anderen ist es die Encodertechnik, die, auch wenn nicht in Perfektion umgesetzt, das Herzstück der Signalerfassung bildet. In Kombination mit den eingesetzten Kugellagern ergibt sich eine sehr elegante mechanische Lösung. Die Lager laufen nahezu reibungslos, verteilen die Kräfte gleichmäßig und tragen wesentlich dazu bei, dass sich das Lenksystem weich und direkt anfühlt. Natürlich gäbe es an anderer Stelle Optimierungspotenzial – etwa bei der Gestaltung der Stromabnehmer oder der Effizienz der Encoderaufnahme. Aber als Gesamtkonstruktion überzeugt der Aufbau durch ein durchdachtes Zusammenspiel aller Teile.

6.2 Pläne, die nicht realisiert wurden – und warum das okay ist

Nicht jeder Gedanke, der während des Bauprozesses entsteht, muss auch zu einem vollendeten Ergebnis führen. Ursprünglich war angedacht, den Simulator modularer aufzubauen – mit Adaptern für Tische oder ergänzenden Halterungen. Auch hier zeigte sich aber, dass sich der Nutzen nicht in Relation zum Aufwand setzte. Der Fokus verlagerte sich stattdessen auf die zentrale technische Machbarkeit.

Wirklich hartnäckig zeigten sich die Stromabnehmer – und obwohl die finale Lösung funktional war, bleibt sie ein kleiner Kritikpunkt. Trotzdem gilt: Das System funktioniert. Und das war von Anfang an die zentrale Überzeugung – auch wenn nicht alles auf Anhieb gelang. Das Wissen, dass sich das Vorhaben grundsätzlich umsetzen lässt, war über den gesamten Zeitraum präsent – ein wertvoller Motor für das Durchhalten

6.3 Was ich gelernt habe

Das Projekt war vor allem eines: eine technische Lernreise. Ich konnte mein Wissen in verschiedenen Bereichen deutlich erweitern – angefangen beim CAD-Design und 3D-Druck, wo ich lernte, mechanische Komponenten funktional zu konstruieren, Toleranzen gezielt einzuplanen und Vorspannung kontrolliert einzubringen. Die Verbindung von Software (Fusion 360) und Hardware (Bambu Studio, Drucker) eröffnete mir ein neues Verständnis für präzise Bauteilgestaltung.

Auch mechanisch hat sich mein Blick geschärft: Kraftverläufe, Lagerungen, Federwirkung – all das wurde vom bloßen Konzept zur spürbaren Realität. Gerade das Zusammenspiel aus Federn und Kugellagern erforderte ein gewisses Feingefühl, das ich mir erst im Laufe der Arbeit aneignen musste.

Im Bereich der Programmierung betrat ich fast vollständig neues Terrain. Während ich zuvor schon mit Arduino gearbeitet hatte, war die Welt von Python, serieller Kommunikation per Bluetooth und der Einbindung in virtuelle Eingabegeräte (vJoy) neu für mich – aber genau deshalb auch besonders spannend. Das Zusammenspiel aus Mikrocontroller,

Signalübertragung und PC-Steuerung funktionierte am Ende nicht nur zuverlässig, sondern vermittelte mir ein ganz neues Verständnis davon, wie Soft- und Hardware ineinandergreifen.

Hilfreich war dabei vor allem die Mischung aus systematischem Vorgehen, Ausprobieren und stetigem Verstehen. Selbst das Arbeiten mit der Kommandozeile, das mir anfangs eher fremd erschien, wurde mit der Zeit zur Selbstverständlichkeit.

6.4 Ideen, die bleiben – auch wenn das Projekt abgeschlossen ist

Der Simulator wird aktuell nicht aktiv verwendet. Zwar ist er funktional, allerdings wären gezielte Verbesserungen an der Encodertechnik nötig, um ihn dauerhaft in Spielen wie Forza Horizon einzusetzen. Doch auch wenn das Projekt nicht produktiv weiterverwendet wird – die Gedanken, die es angestoßen hat, bleiben.

Einer dieser Gedanken ist die Idee, den Controller zu einem echten Fernsteuergerät für ein selbstgebautes RC-Auto weiterzuentwickeln. Ein Auto, das nicht durch Spielzeugtechnik, sondern durch durchdachtes Fahrwerksdesign, Lenktechnik und Kommunikationslogik überzeugt. Ohne Motor vielleicht – aber mit dem vollen Fokus auf die Aspekte, die mich auch am echten Auto faszinieren: Fahrwerk, Kraftverteilung, Lenkgefühl.

7. Künstliche Intelligenz: Mitdenken auf Abruf

Komplexe Projekte wie dieser Simulator lassen sich selten völlig im Alleingang stemmen – zumindest nicht dann, wenn man gleichzeitig neue Bereiche erkunden, Fehler verstehen und gleichzeitig an der Umsetzung feilen will. Umso wertvoller war es, während der Entwicklung auf ChatGPT von OpenAI zurückgreifen zu können. Nicht als Lösungslieferant oder gar automatischer Codegenerator, sondern als technischer Sparringspartner.

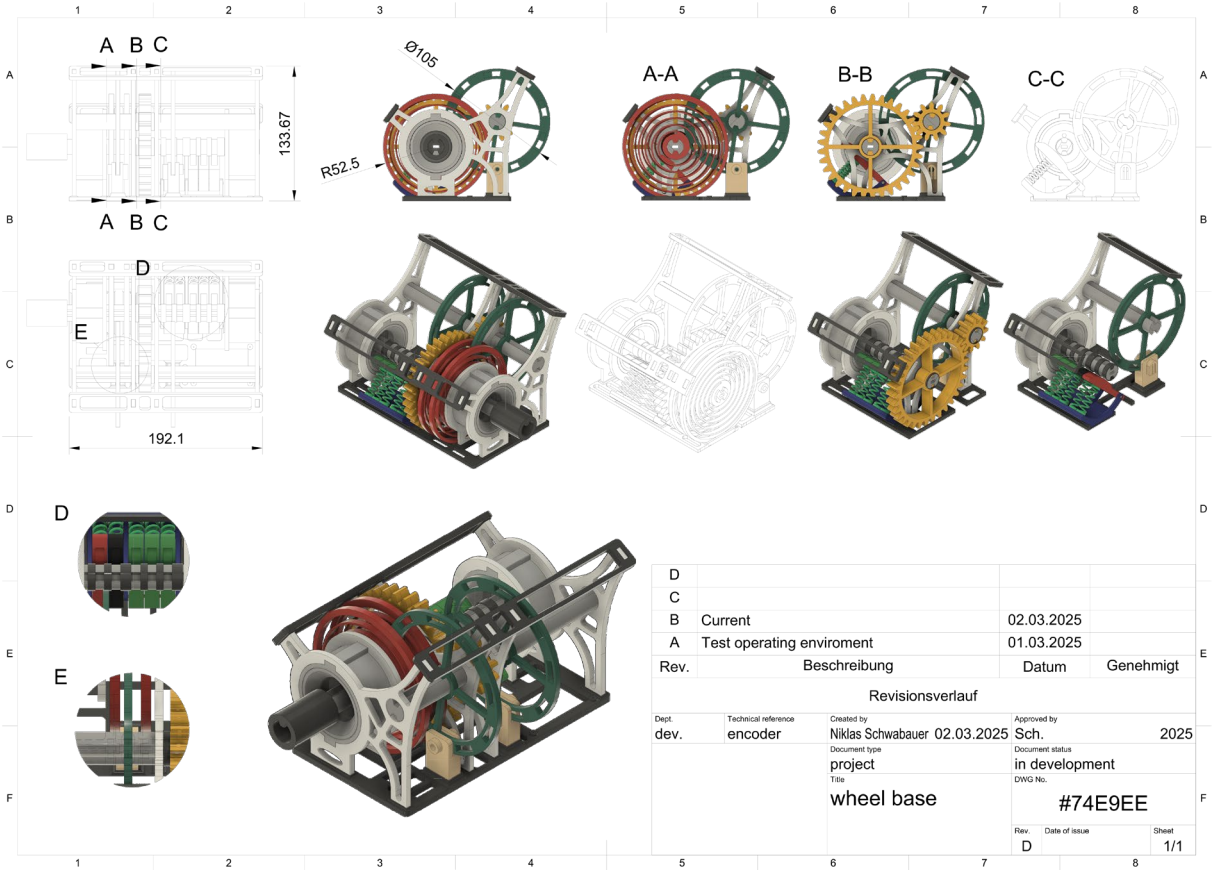
Die Rolle, die ChatGPT dabei spielte, ähnelte am ehesten einem kompetenten Mitdenker: ein Gesprächspartner, der immer dann zur Verfügung stand, wenn ein kleiner Impuls, ein Denkanstoß oder eine zweite Sichtweise nötig war – sei es beim Aufsetzen des Python-Skripts, bei der Auswertung der Sensordaten oder beim Debugging der Bluetooth-Verbindung. Besonders hilfreich war das bei Themen, in die ich mich erst neu einarbeiten musste: die Anbindung über vJoy, der Umgang mit cmd oder das korrekte Timing bei der Signalverarbeitung.

Natürlich verlief nicht jede Hilfestellung reibungslos. Manchmal waren mehrere Anläufe nötig, bis der Vorschlag zu meinem Aufbau passte – und oft waren es gerade die Rückfragen und Nachbesserungen, durch die ich mein eigenes Verständnis vertiefen konnte. Das macht die Zusammenarbeit umso wertvoller: nicht als Ersatz für eigenes Denken, sondern als Katalysator für genau dieses Denken.

ChatGPT hat in diesem Projekt keine Entwürfe gezeichnet, keine Bauteile ausgedruckt und auch keinen einzigen Draht gelötet. Die Idee, die Umsetzung und die konstruktiven Entscheidungen lagen vollständig bei mir. Aber die Möglichkeit, technische Fragestellungen direkt mit einer KI durchzuspielen, hat den Entwurfsprozess effizienter, präziser – und manchmal auch einfach entspannter gemacht. Gerade bei umfangreichen Eigenprojekten ist das ein Vorteil, der kaum zu überschätzen ist.

Mitdenken auf Abruf – das beschreibt ziemlich genau, wie sich diese Art der Zusammenarbeit anfühlt. Und wahrscheinlich wird sie bei kommenden Projekten nicht weniger wichtig sein.

8. Anhang



Rev.	Beschreibung	Datum	Genehmigt
D			
C			
B	Current	02.03.2025	
A	Test operating enviroment	01.03.2025	

Revisionsverlauf

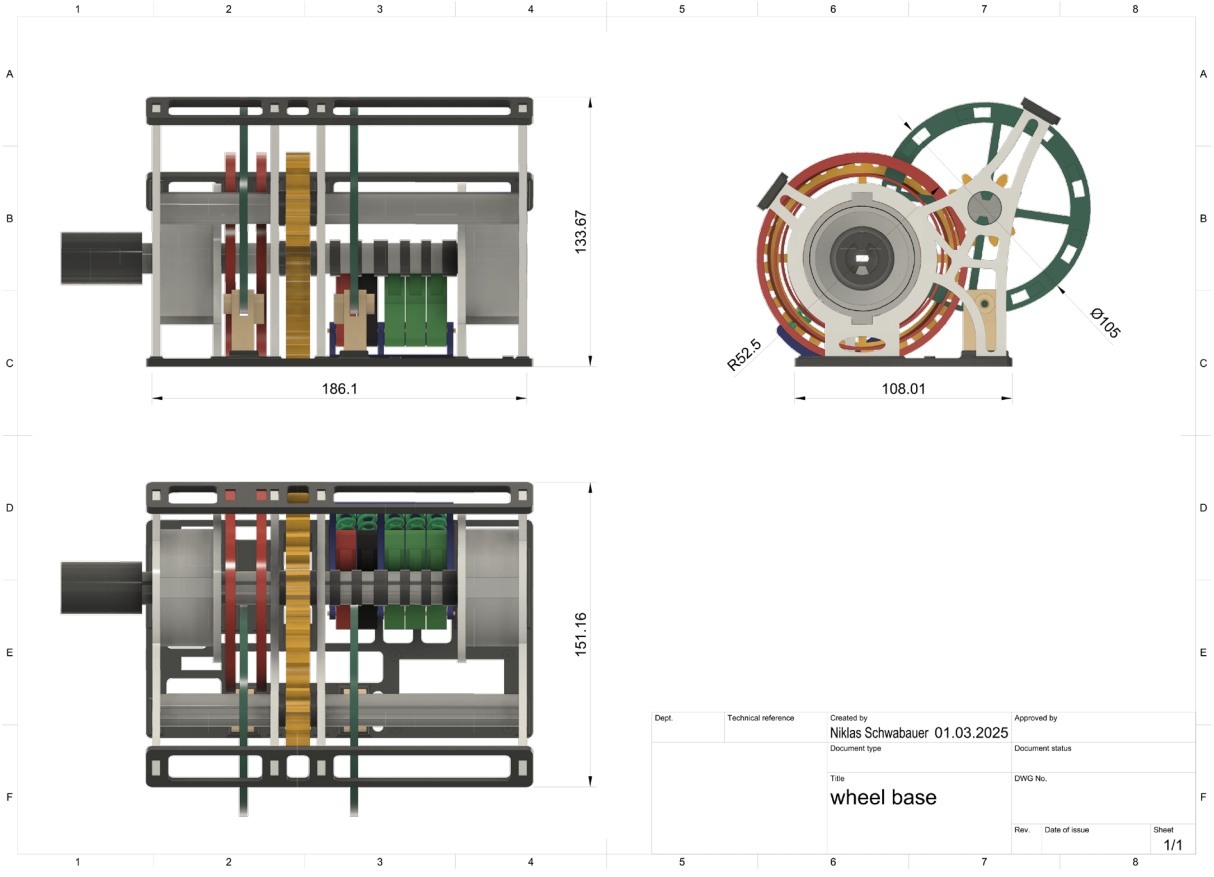
Dept.	Technical reference	Created by	Approved by
dev.	encoder	Niklas Schwabauer 02.03.2025	Sch. 2025

Document type	Document status
project	in development

Title: wheel base

DWG No.: #74E9EE

Rev.	Date of issue	Sheet
D		1/1



Dept.	Technical reference	Created by	Approved by
		Niklas Schwabauer 01.03.2025	

Document type	Document status

Title: wheel base

DWG No.:

Rev.	Date of issue	Sheet
		1/1