Entwicklung und Steuerung eines inversen Doppelpendels mittels 5G-Datenübertragungstechnologie

Owis Alsabbagh, Elektro- und Informationstechnik B.Sc.
Otto-von-Guericke-Universität, Magdeburg, Fakultät für Elektro- und Informationstechnik
IFAK–Institut für Automation und Kommunikation e.V., Magdeburg

***Zusammenfassung*—Der Aufbau und die Regelung eines inversen Doppelpendels unter Verwendung der 5G-Technologie als Übertragungsmedium ist Gegenstand dieser Forschungsarbeit. Das inverse Doppelpendel ist ein hochkomplexes und instabiles System, das zur Stabilisierung eine präzise Regelung benötigt. Herkömmliche Regelungsverfahren stoßen dabei an ihre Grenzen, insbesondere in Bezug auf Echtzeitkommunikation und Energieeffizienz. In dieser Studie wird eine innovative Lösung vorgestellt, die auf der Nutzung der 5G-Technologie für die drahtlose Datenübertragung und Echtzeitkommunikation basiert. Die Vorteile von 5G, wie hohe Datenraten, geringe Latenzzeiten und hohe Zuverlässigkeit, werden genutzt, um eine effiziente Regelung des inversen Doppelpendels zu ermöglichen. Durch die Integration von 5G in das Regelungssystem wird eine verbesserte Reaktionsfähigkeit und präzise Steuerung des Pendels erreicht. Die Ergebnisse dieser Forschung tragen zur Weiterentwicklung der Regelungstechnik für komplexe und instabile Systeme bei und zeigen das Potenzial der 5G-Technologie in der Regelungstechnik.**

***Schlagwörter*—5G, Doppelpendel, nichtlinear, Regelung, Stabilisierung**

I. Nomenklatur

m_i	Masse
l_i	Abstand zwischen Drehpunkt und Schwerpunkt des i-ten Pendelarms, entspricht der Länge des Pendelarms
x	Position des Bewegungskörpers
θ_i	Winkel zwischen dem i-ten Pendelarm und der Y-Achse
I_i	Trägheitsmoment am Massenschwerpunkt des i-ten Pendelarms
g	Ortsfaktor
u, F	Steuerkraft
T	kinetische Energie
U	potentielle Energie
L	Lagrange-Funktion

II. Einleitung

DIE Entwicklung und Steuerung eines inversen Doppelpendels stellt eine klassische Herausforderung in der Regelungstechnik und Dynamik dar, die durch inhärente Instabilität und Nichtlinearität gekennzeichnet ist. Ein inverses Doppelpendel besteht aus zwei starr miteinander verbundenen Stäben, die an einem gemeinsamen Drehpunkt befestigt sind und deren Schwerpunkt über diesem Punkt liegt. Die präzise Steuerung eines solchen Systems erfordert hochentwickelte Algorithmen und schnelle Reaktionszeiten, um die Stabilität und die gewünschte Bewegungsdynamik zu gewährleisten.

In jüngster Zeit ist die 5G-Datenübertragungstechnologie als potenziell bahnbrechend für die Steuerungs- und Regelungstechnik in den Blickpunkt gerückt. Mit ihrer enormen Bandbreite, extrem niedrigen Latenz und der Fähigkeit, Millionen von Geräten gleichzeitig zu verbinden, bietet 5G eine beispiellose Plattform für die Echtzeitsteuerung komplexer Systeme. Diese Technologie ermöglicht nicht nur die schnelle und zuverlässige Übertragung von Steuerungsdaten, sondern auch die Implementierung verteilter Steuerungssysteme, bei denen die Rechenlast auf mehrere Knoten verteilt werden kann.

In diesem Zusammenhang eröffnet der Einsatz von 5G neue Perspektiven für die Echtzeitsteuerung inverser Doppelpendel. Herkömmliche Methoden, die auf kabelgebundener oder weniger zuverlässiger drahtloser Kommunikation beruhen, stoßen oft an ihre Grenzen, wenn es darum geht, die hohen Datenraten und geringen Latenzzeiten zu bewältigen, die für eine effiziente Regelung erforderlich sind. Durch den Einsatz von 5G kann jedoch eine nahtlose und ultraschnelle Kommunikation zwischen Sensoren, Aktoren und Reglern gewährleistet werden, wodurch die Reaktionsfähigkeit des Systems erheblich verbessert wird.

Ziel dieser Arbeit ist es, die theoretischen Grundlagen und praktischen Anwendungen der 5G-Technologie bei der Regelung eines inversen Doppelpendels zu untersuchen. Zunächst werden die dynamischen Modelle des inversen Doppelpendels und die grundlegenden Regelungsstrategien erläutert. Anschließend wird der Fokus auf die spezifischen Herausforderungen und Lösungen gelegt, die sich aus der Integration der 5G Datenübertragung in dieses

System ergeben. Dazu gehören die Untersuchung der Kommunikationsarchitektur, die Analyse der Latenzzeiten und deren Einfluss auf die Systemstabilität sowie die Implementierung und Validierung der Regelalgorithmen in einer realen 5G-Umgebung.

Durch die Kombination fortschrittlicher Regelungstechnik mit moderner Kommunikationstechnologie soll ein wesentlicher Beitrag zur Optimierung der Steuerung komplexer dynamischer Systeme geleistet werden. Die Ergebnisse könnten nicht nur für die akademische Forschung von Interesse sein, sondern auch praktische Anwendungen in Bereichen wie Robotik, Automatisierung und vernetzte Fahrzeuge finden. Durch die Nutzung des Potenzials der 5G-Technologie können wir die Grenzen des Machbaren in der Regelungstechnik neu definieren und innovative Lösungen für einige der anspruchsvollsten Probleme der modernen Ingenieurwissenschaften entwickeln.

III. Vorbetrachtungen

In diesem Abschnitt werden mathematische, technische und softwarebezogene Vorüberlegungen angestellt.

A. Überlegungen zur mathematischen Modellierung

Als Vorbereitung für die Erstellung eines mathematischen Modells des inversen Doppelpendels wird zunächst das Modell eines einfachen inversen Pendels auf einem Wagen betrachtet.

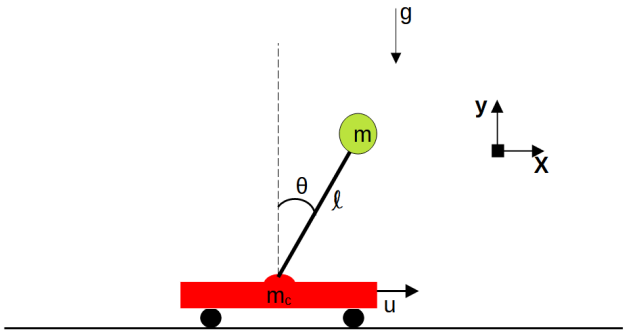

Abbildung 1. Schematisches Modell eines inversen Pendels

Zerst werden die Positionsvariablen in x und y ausgedrückt:

$$x_c = x$$

$$y_c = 0$$

Und somit gilt für die Positionsvariablen der Masse des Pendelarms:

$$x_p = x_c + l\sin\theta = x + l\sin\theta$$

$$y_p = y_c + l\cos\theta = l\cos\theta$$

Außerdem werden die Geschwindigkeiten für die Berechnungen benötigt und daher werden die zeitlichen Ableitungen wie folgt berechnet:

$$\dot{x_p} = \dot{x} + l\dot{\theta}\cos\theta$$

$$\dot{y_p} = -l\dot{\theta}\sin\theta$$

Nun werden sowohl die kinetischen als auch die potentiellen Energien berechnet.
Für die kinetische Energie des Bewegungkörpers gilt:

$$T_c = \frac{1}{2}m_c\dot{x}^2 \tag{1}$$

Für den Pendelarm wird die kinetische Energie T_p wie folgt berechnet:

$$T_p = \frac{1}{2}m\left(\dot{x_p}^2 + \dot{y_p}^2\right) + \frac{1}{2}I\dot{\theta}^2$$

$$= \frac{1}{2}m\left[\left(\dot{x}^2 + 2\dot{x}l\dot{\theta}\cos\theta + l^2\dot{\theta}^2\cos^2\theta\right) + l^2\dot{\theta}^2\sin^2\theta\right] + \frac{1}{2}I\dot{\theta}^2$$

Aufgrund des trigonometrischen Beziehung $\sin^2 x + \cos^2 x = 1$ gilt:

$$T_p = \frac{1}{2}m\dot{x}^2 + m\dot{x}l\dot{\theta}\cos\theta + \frac{1}{2}ml^2\dot{\theta}^2 + \frac{1}{2}I\dot{\theta}^2$$

Und somit gilt für T_p:

$$T_p = \frac{1}{2}m\dot{x}^2 + m\dot{x}l\dot{\theta}\cos\theta + \frac{1}{2}\left(ml^2 + I\right)\dot{\theta}^2 \tag{2}$$

Aus den Gleichungen 1 und 2 und mit $M = m_C + m$ gilt für die gesamte kinetische energie des Systems T:

$$T = \frac{1}{2}M\dot{x}^2 + m\dot{x}l\dot{\theta}\cos\theta + \frac{1}{2}\left(ml^2 + I\right)\dot{\theta}^2 \tag{3}$$

Analog wird die potentielle Energie des Systems ermittelt.
Da der Bewegungskörper stets bei $y = 0$ liegt, gilt:

$$U_c = 0 \tag{4}$$

Allerdings gilt für den Pendelarm:

$$U_p = mgy_p$$

$$U_p = mgl\cos\theta \tag{5}$$

Aus den Gleichungen 4 udn 5 gilt für die gesamte potentielle Energie des Systems:

$$U = mgl\cos\theta \tag{6}$$

Nun soll aus den Gleichungen 3 und 6 die Lagrange-Funktion erstellt werden. Allgemein gilt:

$$L = T - U$$

Und somit gilt für L:

$$L = \frac{1}{2}M\dot{x}^2 + m\dot{x}l\dot{\theta}\cos\theta + \frac{1}{2}\left(ml^2 + I\right)\dot{\theta}^2 - mgl\cos\theta \tag{7}$$

Mithilfe der Gleichung 7 werden die Bewegungsgleichungen erstellt. Dafür wird die sogenannte Lagrange-Gleichung verwendet, für die allgemein gilt:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0 \tag{8}$$

Dabei sind q_i generalisierte Koordinaten. Außerdem gilt für diesen Fall nicht den rechten Gleichungsteil 0, sondern:

$$\underline{U} = \begin{bmatrix} u \\ 0 \end{bmatrix}$$

Damit gilt für die erste Variable x:

$$\frac{\partial L}{\partial \dot{x}} = M\dot{x} + ml\dot{\theta}\cos\theta \tag{9}$$

und davon wird die erste Ableitung wie folgt berechnet:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = M\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{10}$$

Da kein Term in L von x abhängt gilt:

$$\frac{\partial L}{\partial x} = 0 \tag{11}$$

Die erste Bewegungsgleichung des Systems ergibt sich aus den Gleichungen 9,10 und 11:

$$M\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = u \tag{12}$$

Analog wird die Bewegungsgleichung für die Koordinate θ aufgestellt.

$$\frac{\partial L}{\partial \dot{\theta}} = ml\dot{x}\cos\theta + \left(ml^2 + I\right)\dot{\theta} \tag{13}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = ml\cos\theta\ddot{x} - ml\dot{\theta}\sin\theta\dot{x} + \left(ml^2 + I\right)\ddot{\theta} \tag{14}$$

$$\frac{\partial L}{\partial \theta} = -ml\dot{x}\dot{\theta}\sin\theta + mgl\sin\theta \tag{15}$$

Aus den Gleichungen 13,14 und 15 gilt für die 2. Bewegungsgleichung:

$$ml\cos\theta\ddot{x} + \left(ml^2 + I\right)\ddot{\theta} - mgl\sin\theta = 0 \tag{16}$$

An dieser Stelle wird die Kleinwinkelapproximation angewendet, um das Gleichungssystem zu vereinfachen.
Unter der Kleinwinkelapproximation wird die mathematische Näherung verstanden, bei der angenommen wird, der Winkel x sei so hinreichend klein, dass man seinen Sinus oder Tangens durch den Winkel selbst (in Radiant) und den Kosinus durch 1 ersetzen kann. [1]
Es ist in den Abbildungen 2 und 3 ersichtlich, dass für einen hinreichend kleinen Winkel x:

$$\cos x = 1 \tag{17}$$

$$\sin x = x \tag{18}$$

Außerdem gilt:

$$\dot{\theta}^2 \approx 0 \tag{19}$$

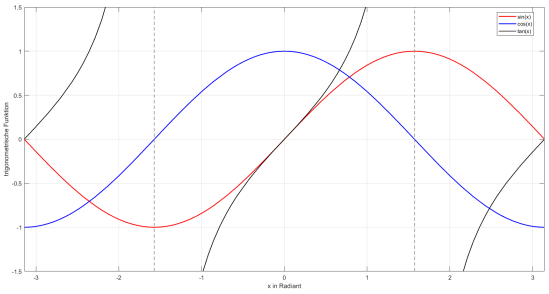

Abbildung 2. Die Winkelfunktionen Sinus, Kosinus und Tangens

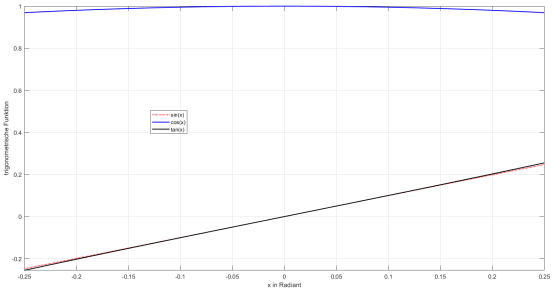

Abbildung 3. Die Winkelfunktionen Sinus, Kosinus und Tangens mit einem hinreichend kleinen Defintionsbereich

Aus den Gleichungen 12,16 und den Zusammenhängen 17, 18 und 19 gilt für die vereinfachten Bewegungsgleichungen:

$$M\ddot{x} + ml\ddot{\theta} = u \tag{20}$$

$$ml\ddot{x} + \left(ml^2 + I\right)\ddot{\theta} - mgl\theta = 0 \tag{21}$$

In Matrixform gilt für das System:

$$\begin{bmatrix} M & ml \\ ml & \left(ml^2 + I\right) \end{bmatrix}\begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -mgl \end{bmatrix}\begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} u \\ 0 \end{bmatrix} \tag{22}$$

Zur Überführung des Gleichungssystems in die Zustandsraumdarstellung, muss folgende Form der Gleichung eingehalten werden:

$$\dot{\underline{x}} = \underline{A}\cdot\underline{x} + \underline{B}\cdot\underline{U} \tag{23}$$

Mit:

$$\underline{A} = \underline{M}^{-1}\cdot\underline{N}$$

$$\underline{B} = \underline{M}^{-1}\cdot\underline{U}$$

Dabei gilt:

$$\underline{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & M & ml \\ 0 & 0 & ml & \left(ml^2 + I\right) \end{bmatrix} \tag{24}$$

$$\underline{N} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & -mgl & 0 & 0 \end{bmatrix} \tag{25}$$

$$\underline{U} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{26}$$

Und somit gilt für das lineasierte System:

$$\dot{\underline{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{m^2 l^2 g}{MI+Mml^2-m^2l^2} & 0 & 0 \\ 0 & \frac{-Mmgl}{MI+Mml^2-m^2l^2} & 0 & 0 \end{bmatrix} \cdot \underline{x} + \begin{bmatrix} 0 \\ 0 \\ \frac{u \cdot \left(ml^2+I\right)}{MI+Mml^2-m^2l^2} \\ \frac{-mlu}{MI+Mml^2-m^2l^2} \end{bmatrix} \cdot \underline{U} \tag{27}$$

B. Überlegungen zur Auswahl der Komponenten

Bei der Auswahl der Komponenten müssen mehrere Aspekte berücksichtigt werden. Einerseits muss der Mikrocontroller mit dem Motor, seinem Treiber und den konventionellen Sensoren kompatibel sein, andererseits muss er die 5G-Zellulartechnologie mit oder ohne zusätzliche Komponenten nutzen können. Auch der Kostenaspekt darf nicht vergessen werden.
Zwei der am häufigsten für solche Anwendungen verwendeten Mikrocontroller sind RaspberryPi und Arduino. Nach dem derzeitigen Stand der Technik ist es nicht möglich, die 5G-Mobilfunktechnologie mit dem Arduino zu verwenden. Für den RaspberryPi gibt es jedoch Aufsätze, die üblicherweise als HATs (engl. hardware attached on top) bezeichnet werden und in diesem Fall die Nutzung von 5G mit einem 5G-Modul ermöglichen können.
Allerdings hat die Verwendung des RaspberryPi einige Nachteile, z.B. dass die digitalen Signalpins des RaspberryPi, auch GPIOs (engl. general-purpose input/output) genannt, nur eine maximale Signalspannung von $3{,}3\,\mathrm{V}$ erlauben.
Diese Spannung ist problematisch, da sie im Vergleich zu den Signalspannungen der meisten herkömmlichen Sensoren zu niedrig ist. Dies schränkt die Auswahl an Sensoren ein und stellt ein neues Kriterium dar, das streng beachtet werden muss.
Nun müssen die Kriterien für die Auswahl des Motors und der Sensoren festgelegt werden. Logischerweise muss der Motor den Anforderungen der Geschwindigkeit erfüllen, indem genug Kraft einbringt. Auch hier darf der Kostenaspekt nicht vergessen werden.
Anzahl, Art und Eigenschaften der Sensoren müssen festgelegt werden. Beispielsweise werden für die Messung der Winkelposition der beiden Pendelarme zwei Inkrementalencoder benötigt, die über eine ausreichende Anzahl von Impulsen pro Umdrehung (PPR) verfügen, um den gesamten Wertebereich der Drehbewegung der Pendelarme abzudecken. Idealerweise ist auch die Möglichkeit zur Messung der Winkelgeschwindigkeiten vorgesehen.

C. Überlegungen zur Software

Die Software für dieses Projekt ist in zwei Hauptteile unterteilt: Der erste Teil läuft auf einem Mikrocontroller, der die Sensordaten erfasst und über UDP (via Ethernet, WIFI oder 5G) weiterleitet. Der zweite Teil besteht aus einem Controller, der diese Daten als Eingabeparameter verwendet, um Steuerwerte zu berechnen und an den ersten Teil zurückzusenden, um den Motor zu steuern. Eine wichtige Überlegung bei der Softwareentwicklung ist die Wahl der Programmiersprache. Python wird als Hauptvorschlag diskutiert, da es eine weit verbreitete Sprache für die Programmierung von Mikrocontrollern ist.
Bei der Auswahl der Programmiersprache für dieses Projekt sind drei Hauptkriterien zu berücksichtigen:

Einfachheit der Implementierung: Python ist bekannt für seine einfache und lesbare Syntax, was die Entwicklung und Wartung des Codes erleichtert. Dies ist besonders wichtig, da die Software sowohl Sensordaten erfassen als auch in Echtzeit darauf reagieren muss.

Echtzeitfähigkeit: Da die Steuerung eines inversen Doppelpendels eine schnelle und präzise Reaktion erfordert, ist die Echtzeitfähigkeit der Software von entscheidender Bedeutung. Python bietet zwar nicht die Echtzeitfähigkeit von Low-Level-Sprachen wie C, aber durch die Verwendung geeigneter Bibliotheken und Optimierungen kann eine ausreichende Performance erreicht werden.

Kompatibilität mit dem Controller: Das Projekt verwendet einen Raspberry Pi als Mikrocontroller. Python ist vollständig kompatibel mit dem Raspberry Pi und unterstützt zahlreiche Bibliotheken, die für dieses Projekt nützlich sind.

Für die Sensorintegration können Bibliotheken wie gpiozero und RPi.GPIO verwendet werden. Diese Bibliotheken ermöglichen eine einfache und effiziente Steuerung der GPIO-Pins des Raspberry Pi, was für die Erfassung der Sensordaten notwendig ist.

Für die Netzwerkkommunikation bietet Python mit socket eine leistungsfähige Bibliothek, die die Implementierung der UDP-Kommunikation erleichtert. Die time-Bibliothek kann genutzt werden, um Zeitstempel zu erstellen und Verzögerungen zu steuern, was für die Echtzeitverarbeitung wichtig ist.

Insgesamt bietet Python durch seine Einfachheit, seine umfassende Bibliotheksunterstützung und seine Kompatibilität mit dem Raspberry Pi eine solide Basis für die Entwicklung der Software für das Projekt.

D. Überlegung zum Entwurf des Conrtollers

Im Rahmen des Projekts zur Entwicklung und Steuerung eines inversen Doppelpendels stehen verschiedene Methoden für den Reglerentwurf zur Verfügung. Die Hauptoptionen sind Model Predictive Control (MPC), maschinelles Lernen, konventionelle PID-Regler und lineare quadratische Regler (LQR). Die Entscheidung für eine dieser Methoden basiert auf den Kriterien Einfachheit der Implementierung mit Python, Echtzeitfähigkeit und Kompatibilität mit komplexen nichtlinearen Systemen.

Model Predictive Control (MPC) ist eine fortschrittliche Methode, bei der zukünftige Systemzustände auf der Grundlage eines mathematischen Modells vorhergesagt werden. MPC kann besonders gut mit komplexen und nichtlinearen Systemen umgehen und bietet die Möglichkeit, verschiedene Einschränkungen direkt in das Regelungsgesetz zu integrieren. Die Implementierung in Python kann jedoch anspruchsvoll sein und erfordert umfangreiche Rechenressourcen, was die Echtzeitfähigkeit beeinträchtigen kann. Daher ist MPC möglicherweise weniger geeignet, wenn Einfachheit und Echtzeitfähigkeit im Vordergrund stehen.

Das Machine Learning, insbesondere das Reinforcement Learning, hat in den letzten Jahren große Fortschritte gemacht und kann für den Entwurf von aus Erfahrung lernenden Reglern eingesetzt werden. Diese Methode ist besonders leistungsfähig für nichtlineare Systeme und kann adaptive Regelstrategien entwickeln. Allerdings ist die Implementierung komplex und die Trainingsphase kann zeitaufwändig sein. Außerdem erfordert die Echtzeitfähigkeit eine sorgfältige Optimierung der Algorithmen, um sicherzustellen, dass sie schnell genug reagieren.

Der PID-Regler (Proportional-Integral-Derivative) ist ein bewährtes und weit verbreitetes Verfahren zur Regelung von Systemen. Er zeichnet sich durch seine Einfachheit und die einfache Implementierung in Python aus. Ein PID-Regler kann jedoch bei stark nichtlinearen Systemen an seine Grenzen stoßen, da er in erster Linie für lineare Systeme entwickelt wurde. Trotz dieser Einschränkungen bietet der PID-Regler eine gute Echtzeitfähigkeit und ist aufgrund seiner Einfachheit eine attraktive Option für erste Implementierungen und Tests.

Der lineare quadratische Regler ist ein Verfahren zur Optimierung der Regelung eines linearen Systems durch Minimierung einer Kostenfunktion. LQR ist mathematisch fundiert und kann zu sehr effizienten Regelstrategien führen. Die Implementierung in Python ist möglich, erfordert aber ein gutes Verständnis der Systemdynamik und der Optimierung. Während LQR für lineare Systeme gut geeignet ist, kann die Anpassung an nichtlineare Systeme eine Herausforderung darstellen. In Bezug auf Echtzeitfähigkeit und Implementierungskomplexität liegt LQR zwischen PID-Reglern und MPC.

Bei der Auswahl des Reglers für das inverse Doppelpendel sollte auf einfache Implementierung, Echtzeitfähigkeit und Kompatibilität mit komplexen nichtlinearen Systemen geachtet werden. Herkömmliche PID-Regler bieten eine einfache und schnelle Implementierung und eignen sich gut für erste Tests und weniger komplexe Szenarien. Für komplexere Anforderungen können MPC oder maschinelles Lernen in Betracht gezogen werden, wobei die Komplexität der Implementierung und die Rechenanforderungen berücksichtigt werden müssen. LQR stellt einen Kompromiss dar, der eine mathematisch fundierte Regelung ermöglicht, aber auch Herausforderungen bei der Anwendung auf nichtlineare Systeme mit sich bringt.

IV. Systemimplementierung

Dieser Abschnitt beschreibt die Implementierung des Systems und gliedert sich in mathematische, technische und softwaretechnische Aspekte.

A. Das mathematische Modell des inversen Doppelpendels

Das Verfahren zur Erstellung des mathematischen Modells des inversen Doppelpendels ist analog zum Modell des inversen Pendels im Unterabschnitt III-A.

Zuerst müssen die Positionsvariablen aller Massen und ihrer

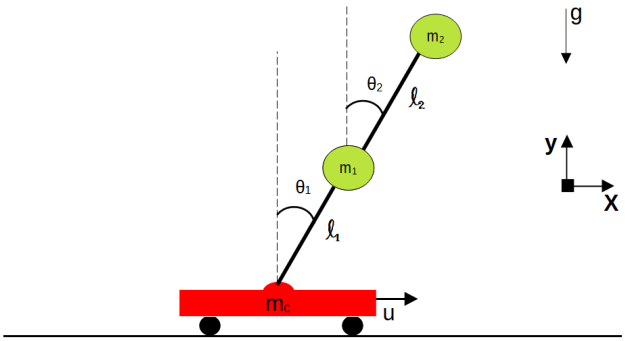

Abbildung 4. Schmatisches Modell des inversen Doppelpendels

Ableitungen bestimmt werden.Dabei gilt:

$$x_c = x$$

$$y_c = 0$$

$$x_1 = x_c + l_1 \sin\theta_1 = x + l_1 \sin\theta_1$$

$$y_1 = y_c + l_1 \cos\theta_1 = l_1 \cos\theta_1$$

$$x_2 = x_1 + l_2 \sin\theta_2 = x + l_1 \sin\theta_1 + l_2 \sin\theta_2$$

$$y_2 = y_1 + l_2 \cos\theta 2 = l_1 \cos\theta_1 + l_2 \cos\theta 2$$

$$\dot{x_c} = \dot{x}$$

$$\dot{x_1} = \dot{x} + l_1\dot{\theta_1}\cos\theta_1$$

$$\dot{y_1} = -l_1\dot{\theta_1}\cos\theta_1$$

$$\dot{x_2} = \dot{x} + +l_1\dot{\theta_1}\cos\theta_1 + +l_2\dot{\theta_2}\cos\theta_2$$

$$\dot{y_2} = -l_1\dot{\theta_1}\cos\theta_1 - l_2\dot{\theta_2}\cos\theta_2$$

Analog zu Abschnitt III-A werden nun die kinetischen Energien T_c, T_1 und T_2 wie folgt berechnet:

$$T_c = \frac{1}{2} m_c \dot{x}^2 \tag{28}$$

$$T_1 = \frac{1}{2}\left(\dot{x_1}^2 + \dot{y_1}^2\right) + \frac{1}{2} I_1 \dot{\theta_1}^2$$

$$T_1 = \frac{1}{2}m_1\dot{x}^2 + \frac{1}{2}\left(m_1 l_1^2 + I_1\right)\dot{\theta_1}^2 + m_1 l_1 \dot{x}\dot{\theta_1}\cos\theta_1 \quad (29)$$

$$T_2 = \frac{1}{2}\left(\dot{x_2}^2 + \dot{y_2}^2\right) + \frac{1}{2}I_2\dot{\theta_2}^2$$

$$\begin{aligned} T_2 = {} & \frac{1}{2}m_2\dot{x}^2 + m_2\dot{x}\dot{\theta_1}l_1\cos\theta_1 + m_2\dot{x}\dot{\theta_2}l_2\cos\theta_2 \\ & + \frac{1}{2}\left(m_2 l_2^2 + I_2\right)\dot{\theta_2}^2 + \frac{1}{2}m_2 l_1^2\dot{\theta_1}^2 \\ & + m_2 l_1 l_2 \dot{\theta_1}\dot{\theta_2}\cos\left(\theta_1 - \theta_2\right) \end{aligned} \quad (30)$$

$$T = T_c + T_1 + T_2$$

$$\begin{aligned} T = {} & \frac{1}{2}M\dot{x}^2 + \frac{1}{2}\left[\left(m_1 + m_2\right)l_1^2 + I_1\right]\dot{\theta_1}^2 \\ & + \left(m_1 + m_2\right)l_1\dot{x}\dot{\theta_1}\cos\theta_1 + \frac{1}{2}\left(m_2 l_2^2 + I_2\right)\dot{\theta_2}^2 \\ & + m_2 l_2 \dot{x}\dot{\theta_2}\cos\theta_2 + m_2 l_1 l_2 \dot{\theta_1}\dot{\theta_2}\cos\left(\theta_1 - \theta_2\right) \end{aligned} \quad (31)$$

Mit $M = m_c + m_1 + m_2$.
Analog gilt für die potentielle Energie:

$$U_c = 0 \quad (32)$$

$$U_1 = mgl_1\cos\theta_1 \quad (33)$$

$$U_2 = m_2 g\left(l_1\cos\theta_1 + l_2\cos\theta_2\right) \quad (34)$$

$$U = \left(m_1 l_1 + m_2 l_2\right)g\cos\theta_1 + m_2 l_2 g\cos\theta_2 \quad (35)$$

Somit gilt für die Lagrange-Funktion L:

$$\begin{aligned} L = {} & \frac{1}{2}M\dot{x}^2 + \frac{1}{2}\left[\left(m_1 + m_2\right)l_1^2 + I_1\right]\dot{\theta_1}^2 \\ & + \left(m_1 + m_2\right)l_1\dot{x}\dot{\theta_1}\cos\theta_1 + \frac{1}{2}\left(m_2 l_2^2 + I_2\right)\dot{\theta_2}^2 \\ & + m_2 l_2 \dot{x}\dot{\theta_2}\cos\theta_2 + m_2 l_1 l_2 \dot{\theta_1}\dot{\theta_2}\cos\left(\theta_1 - \theta_2\right) \\ & - \left(m_1 l_1 + m_2 l_2\right)g\cos\theta_1 - m_2 l_2 g\cos\theta_2 \end{aligned} \quad (36)$$

Analog zum Ansatz im Unterabschnitt III-A wird hier die Lagrange-Euler-Gleichung zur Erstellung der Bewegungsgleichungen verwendet:

$$\frac{\partial L}{\partial \dot{x}} = M\dot{x} + \left(m_1 + m_2\right)l_1\cos\theta_1\dot{\theta_1} + m_2 l_2\cos\theta_2\dot{\theta_2} \quad (37)$$

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = {} & M\ddot{x} + \left(m_1 + m_2\right)l_1\cos\theta_1\ddot{\theta_1} \\ & + m_2 l_2\cos\theta_2\ddot{\theta_2} - \left(m_1 + m_2\right)l_1\sin\theta_1\dot{\theta_1}^2 \\ & - m_2 l_2\sin\theta_2\dot{\theta_2}^2 \end{aligned} \quad (38)$$

Da $\frac{\partial L}{\partial x} = 0$, gilt für die erste Bewegungsgleichung:

$$\begin{aligned} & M\ddot{x} + \left(m_1 + m_2\right)l_1\cos\theta_1\ddot{\theta_1} + m_2 l_2\cos\theta_2\ddot{\theta_2} \\ & - \left(m_1 + m_2\right)l_1\sin\theta_1\dot{\theta_1}^2 - m_2 l_2\sin\theta_2\dot{\theta_2}^2 = u \end{aligned} \quad (39)$$

$$\begin{aligned} \frac{\partial L}{\partial \dot{\theta_1}} = {} & \left(m_1 + m_2\right)l_1\cos\theta_1\dot{x} + \left[\left(m_1 + m_2\right)l_1^2 + I_1\right]\dot{\theta_1} \\ & + m_2 l_1 l_2\cos\left(\theta_1 - \theta_2\right)\dot{\theta_2} \end{aligned} \quad (40)$$

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_1}}\right) = {} & \left(m_1 + m_2\right)l_1\cos\theta_1\ddot{x} + \left[\left(m_1 + m_2\right)l_1^2 + I_1\right]\ddot{\theta_1} \\ & + m_2 l_1 l_2\cos\left(\theta_1 - \theta_2\right)\ddot{\theta_2} - \left(m_1 + m_2\right) \\ & l_1\sin\theta_1\dot{x}\dot{\theta_1} - m_2 l_1 l_2\left(\dot{\theta_1} - \dot{\theta_2}\right)\sin\left(\theta_1 - \theta_2\right)\dot{\theta_2} \end{aligned} \quad (41)$$

$$\begin{aligned} \frac{\partial L}{\partial \theta_1} = {} & -\left(m_1 + m_2\right)l_1\dot{x}\dot{\theta_1}\sin\theta_1 - m_2 l_1 l_2\dot{\theta_1}\dot{\theta_2}\sin\left(\theta_1 - \theta_2\right) \\ & + \left(m_1 l_1 + m_2 l_2\right)g\sin\theta_1 \end{aligned} \quad (42)$$

Damit gilt für die zweite Bewegungsgleichung des Systems:

$$\begin{aligned} & \left(m_1 + m_2\right)l_1\cos\theta_1\ddot{x} + \left[\left(m_1 + m_2\right)l_1^2 + I_1\right]\ddot{\theta_1} \\ & + m_2 l_1 l_2\cos\left(\theta_1 - \theta_2\right)\ddot{\theta_2} + m_2 l_1 l_2\sin\left(\theta_1 - \theta_2\right)\dot{\theta_2}^2 \\ & - \left(m_1 l_1 + m_2 l_2\right)g\sin\theta_1 = 0 \end{aligned} \quad (43)$$

$$\begin{aligned} \frac{\partial L}{\partial \dot{\theta_2}} = {} & \left(m_2 l_2^2 + I_2\right)\dot{\theta_2} + m_2 l_2\dot{x}\cos\theta_2 \\ & + m_2 l_1 l_2\dot{\theta_1}\cos\left(\theta_1 - \theta_2\right) \end{aligned} \quad (44)$$

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta_2}}\right) = {} & m_2 l_2\cos\theta_2\ddot{x} + m_2 l_1 l_2\cos\left(\theta_1 - \theta_2\right)\ddot{\theta_1} \\ & + \left(m_2 l_2^2 + I_2\right)\ddot{\theta_2} - m_2 l_2\sin\theta_2\dot{x}\dot{\theta_2} \\ & - m_2 l_1 l_2\dot{\theta_1}\left(\dot{\theta_1} - \dot{\theta_2}\right)\sin\left(\theta_1 - \theta_2\right) \end{aligned} \quad (45)$$

$$\begin{aligned} \frac{\partial L}{\partial \theta_2} = {} & -m_2 l_2\dot{x}\dot{\theta_2}\sin\theta_2 + m_2 l_1 l_2\dot{\theta_1}\dot{\theta_2}\sin\left(\theta_1 - \theta_2\right) \\ & + m_2 l_2 g\sin\theta_2 \end{aligned} \quad (46)$$

Damit gilt für die dritte Bewegungsgleichung:

$$\begin{aligned} & m_2 l_2\cos\theta_2\ddot{x} + m_2 l_1 l_2\cos\left(\theta_1 - \theta_2\right)\ddot{\theta_1} + \left(m_2 l_2^2 + I_2\right)\ddot{\theta_2} \\ & - m_2 l_1 l_2\sin\left(\theta_1 - \theta_2\right)\dot{\theta_1}^2 - m_2 l_2 g\sin\theta_2 = 0 \end{aligned} \quad (47)$$

Bei Anwendung der Kleinwinkelapproximation aus Unterabschnitt III-A auf die Gleichungen 39, 43 und 47 gilt für das System:

$$M\ddot{x} + \left(m_1 + m_2\right)l_1\ddot{\theta_1} + m_2 l_2\ddot{\theta_2} = u \quad (48)$$

$$\begin{aligned} & \left(m_1 + m_2\right)l_1\ddot{x} + \left[\left(m_1 + m_2\right)l_1^2 + I_1\right]\ddot{\theta_1} \\ & + m_2 l_1 l_2\ddot{\theta_2} - \left(m_1 l_1 + m_2 l_2\right)g\theta_1 = 0 \end{aligned} \quad (49)$$

$$m_2 l_2\ddot{x} + m_2 l_1 l_2\ddot{\theta_1} + \left(m_2 l_2^2 + I_2\right)\ddot{\theta_2} - m_2 l_2 g\theta_2 = 0 \quad (50)$$

In Matrixform:

$$\begin{aligned} & \begin{bmatrix} M & \left(m_1 + m_2\right)l_1 & m_2 l_2 \\ \left(m_1 + m_2\right)l_1 & \left[\left(m_1 + m_2\right)l_1^2 + I_1\right] & m_2 l_1 l_2 \\ m_2 l_2 & m_2 l_1 l_2 & \left(m_2 l_2^2 + I_2\right) \end{bmatrix} \cdot \begin{bmatrix} \ddot{x} \\ \ddot{\theta_1} \\ \ddot{\theta_2} \end{bmatrix} \\ & + \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\left(m_1 l_1 + m_2 l_2\right)g & 0 \\ 0 & 0 & -m_2 l_2 g \end{bmatrix} \cdot \begin{bmatrix} x \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} u \\ 0 \\ 0 \end{bmatrix} \end{aligned} \quad (51)$$

Mit:

$$\underline{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & M & (m_1+m_2)\,l_1 & m_2 l_2 \\ 0 & 0 & 0 & (m_1+m_2)\,l_1 & \left[(m_1+m_2)\,l_1^2 + I_1\right] & m_2 l_1 l_2 \\ 0 & 0 & 0 & m_2 l_2 & m_2 l_1 l_2 & \left(m_2 l_2^2 + I_2\right) \end{bmatrix} \tag{52}$$

$$\underline{N} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -(m_1 l_1 + m_2 l_2)\,g & 0 & 0 & 0 & 0 \\ 0 & 0 & -m_2 l_2 g & 0 & 0 & 0 \end{bmatrix} \tag{53}$$

$$\underline{U} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{54}$$

Um das linearisierte System in das Zustandsraummodell zu überführen, müssen folgende Matrizen berechnet werden:

$$\underline{A} = \underline{M}^{-1}\underline{N}, \underline{B} = \underline{M}^{-1}\underline{F} \tag{55}$$

Und somit erreicht man die Gleichung:

$$\dot{\underline{x}} = \underline{A} \cdot \underline{x} + \underline{B} \cdot \underline{u} \tag{56}$$

B. Auswahl, Aufbau und Zusammensetzung der Komponenten

Bei der Auswahl der Komponenten sind mehrere Kriterien zu berücksichtigen. Einige davon wurden bereits im Unterabschnitt III-B diskutiert.

Wie bereits im Abschnitt III-B angedeutet wurde, ist die beste Auswahl des Mikrocontollers der Raspberry Pi 4.
Auswahlkriterien sind die Fähigkeit zur Sensorintegration, die

Abbildung 5. RaspberryPi 4 model B [2]

Möglichkeit, Python-Programme direkt auf dem Controller auszuführen und die Möglichkeit, 5G-Technologie zu integrieren, indem ein HAT und ein entsprechendes 5G-Modul mit dem Controller kombiniert werden.

Ein weiteres Problem ist, dass die meisten herkömmlichen Sensoren und Aktoren eine Signalspannung von mindestens 5 V benötigen, während die GPIO-Pins des RaspberryPis eine Signalspannung von maximal 3,3 V liefern können.
Eine Lösung für den Spannungsdefizit wird durch den Einsatz von Pegelwandlern gelöst. Dieser ist ist in Abbildung 6 zu sehen.
Die HAT und 5G-Modul sind in Abbildung 7 dargestellt.

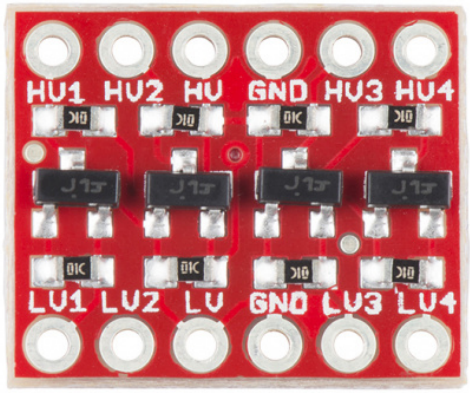

Abbildung 6. Pegelwandler [3]

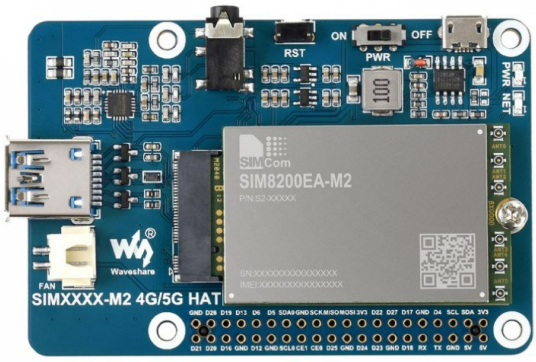

Abbildung 7. 5G-HAT (engl. Hardware attached on Top) und das 5G-Modul SIM8200EA-M2 [4]

Bei der Auswahl des Motors wurde in Absprache mit dem Werkstattleiter entschieden, den Schrittmotor MOT-AN-S-060-020-056-M-C-AAAC des Herstellers IGUS zu verwenden. Dieser ist in Abbildung 8 zu sehen.

In Verbindung mit einem Motortreiber bildet der Motor die Steuereinheit, die für die Ausführung der Steuerwerte verantwortlich ist.
Der Treiber ist in Abbildung 9 dargestellt.
Nach ausgiebigen Tests wurde festgestellt, dass die Steuereinheit genügend Kraft aufbringen kann, um den beweglichen Körper des Pendels angemessen zu bewegen.

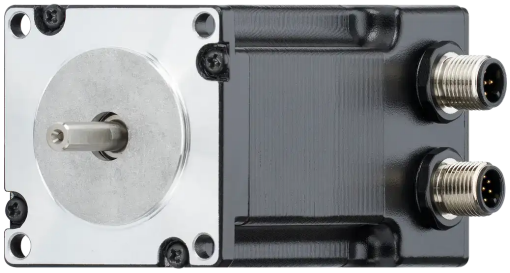

Abbildung 8. Schrittmotor mit integriertem Inkrementalencoder [5]

Abbildung 9. Schrittmotortreiber [6]

Die Abbildung 10 zeigt den Lesekopf des Drehgebers zur Messung der Winkelposition bzw. der Winkelgeschwindigkeit des Pendelarms. Dieser zählt mit Hilfe eines Codierrades die auf dem Codierrad eingravierten Impulse und gibt die Werte des Winkels und der Winkelgeschwindigkeit aus.

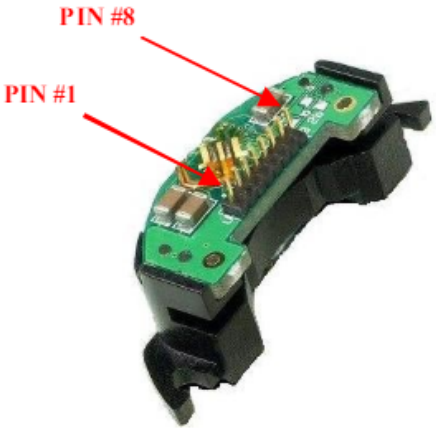

Abbildung 10. Winkelpositions-/Geschwindigkeitssensor [7]

Der letzte benötigte Sensor ist ein fotoelektronischer Sensor, der als Endschalter an jedem Ende der Linearführung angebracht wird, um zu verhindern, dass der bewegte Körper mit den Enden der Linearführung kollidiert.
An dieser Stelle trat ein weiteres Problem auf. Die Signalspannung des Sensors kann 10 V erreichen. Dies ist zu hoch für die GPIOs des Raspberry Pis.
Eine Lösung für die Überspannung ist die Verwendung eines Spannungsteilers.

Abbildung 11. Endschalter, fotoelektronischer Sensor [8]

Die Abbildung 12 ist eine schematische Darstellung, wie die Komponenten mit einander verkabelt sind. Jedoch fehlen die Endschalter, da sie bis zum Zeitpunkt der Erstellung des Designs noch nicht vorhanden waren.

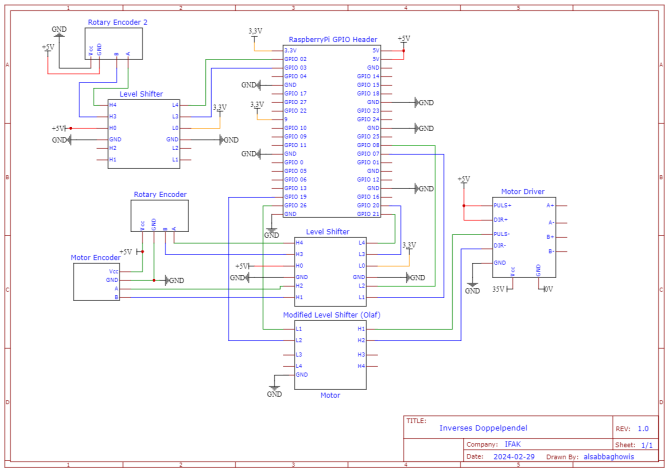

Abbildung 12. Schematische Darstellung der Verschaltung der Komponenten

C. Implementierung des Softwares

In diesem Abschnitt werden Codebeispiele diskutiert und vorgestellt. Die Software ist in zwei Teile gegliedert. Der erste Teil läuft auf dem Raspberry Pi und ist in mehrere „Blöcke“

unterteilt.
Der erste Block ist für den Aufbau der UDP bzw. TCP-Verbindung zuständig, indem durch Initiierung der Instanz ein UDP- bzw. TCP-Server bzw. Client initiiert wird und ist wie Folgend dargestellt.

```
#This a the code for the communication interface
    from the Rpi side
import socket

class COM:
 def __init__(self,IP,PORT,TYPE):
  self.IP_addr = IP
  self.PORT_num = PORT
  self.TYPEofCOM = TYPE  # if x = 1 ,2 than
      Com_type is udp ,tcp/ip respectively

 def CreateServer(self):
  if self.TYPEofCOM == 1:
   ServerSocket =
       socket.socket(family=socket.AF_INET, type =
       socket.SOCK_DGRAM)
   ServerSocket.bind((self.IP_addr,self.PORT_num))
  elif self.TYPEofCOM == 2:
   ServerSocket =
       socket.socket(family=socket.AF_INET, type =
       socket.SOCK_STREAM)
   ServerSocket.bind((self.IP_addr,self.PORT_num))
   ServerSocket.listen()
  return ServerSocket

 def CreateCLient(self):
  if self.TYPEofCOM == 1:
   ClientSocket =
       socket.socket(family=socket.AF_INET,type
       =socket.SOCK_DGRAM)
  elif self.TYPEofCOM ==2:
   CleintSocket =
       socket.socket(family=socket.AF_INET,type
       =socket.SOCK_STREAM)
   ClientSocket.connect((self.IP_addr,self.PORT_num))
  return ClientSocket

 def SendData(self,data):
  if self.TYPEofCOM == 1:
   self.CreateCLient().sendto(str.encode(data),
    (self.IP_addr,self.PORT_num))
  elif self.TYPEofCOM == 2:
   self.CreateClient().sendall(str.encode(data))
 def RecData(self):
  if self.TYPEofCOM == 1:
   msgfromServer =
       self.CreateServer().recvfrom(1024)
  elif self.TYPEofCOM == 2:
   conn , addr = self.CreateServer().accept()
   msgfromServer = conn.recv(1024)
  data = float(msgfromServer[0])
  return data
```

Listing 1. Class COM zum Aufbau des Verbindungsprotokolls

Der zweite und dritte Block sind für das Einlesen der Sensorwerte zuständig und verwenden die Bibliothek gpiozero mit dem Modul RotaryEncoder.

```
from gpiozero import RotaryEncoder #to creat Rotary
    encoder instance
import numpy as np
import time

class ReadRotaryEncoder(RotaryEncoder):

 def readPosition(self,CPR):
  sensorFloat = (2*np.pi/ CPR*self.steps)
  return sensorFloat

 def readVelocity(self,CPR,l_t ,l_s):
  time.sleep(0.002)
  current_time = time.time()
  current_steps = self.steps
  # Calculate the time difference and step
      difference
  time_diff = current_time - l_t
  step_diff = current_steps - l_s

  # Calculate angular velocity (in radians per
      second)
  angular_velocity = (step_diff * (2 * np.pi /
      CPR)) / time_diff

  # Update last_time and last_steps
  l_t = current_time
  l_s = current_steps
  return angular_velocity
```

Listing 2. Class ReadRotaryEncoder zum Lesen des Winkelsensorwerte

```
from gpiozero import RotaryEncoder #to creat Rotary
    encoder instance
import time
import numpy as np
from gpiozero import RotaryEncoder

class ReadMotorEncoder(RotaryEncoder):
 def readPosition(self,CPR):
  # Calculate the angle and position
  angle = (2 * np.pi / CPR) * self.steps
  position = (angle * 25) / 1000  # m
  return position

 def readVelocity(self,CPR,l_t,l_s):
  time.sleep(0.002)
  current_time = time.time()
  current_steps = self.steps

  # Calculate the time difference and step
      difference
  time_diff = current_time - l_t
  step_diff = current_steps - l_s

  # Calculate speed (in cm per second)
  velocity = (step_diff * (2 * np.pi / CPR) * (25 /
      1000)) / time_diff

  # Update last_time and last_steps
  l_t = current_time
  l_s = current_steps
  return velocity
```

Listing 3. Class ReadMotorEncoder zum Lesen des Motorsensorwerte

Der folgende Codeausschnitt zeigt die Zusammenarbeit der drei Blöcke und die Weiterleitung der Daten über Ethernet.

```
#This is the code for the ethernet interface for
    communnicating sensor and control data through
    the ethernet interface of the RPi
from com import COM
from gpiozero import RotaryEncoder #to creat Rotary
    encoder instance
from rotaryencoder import ReadRotaryEncoder#child
    class wherer all sensor reading related methods
    are stored
from motorencoder import ReadMotorEncoder
import numpy as np
import time
```

```
#Constant Values
ETH_SERVER_IP = "10.0.8.40"
ETH_CLIENT = "10.0.8.55"
ETH_SERVER_PORT_A = 890
ETH_SERVER_PORT_A_DOT = 990
ETH_SERVER_PORT_B = 892
ETH_SERVER_PORT_B_DOT = 992
ETH_SERVER_PORT_X = 893
ETH_SERVER_PORT_X_DOT = 993
ETH_SERVER_PORT_CTRL = 5000
UDP = 1
TCP = 2

# Creating encoder object using GPIO pins 7 and 8
    in BCM mode
encoder_m     = ReadMotorEncoder(7, 8, max_steps=0)
cpr_m         = 500

# Creating encoder object using GPIO pins 02 and 03
encoder_2      =
    ReadRotaryEncoder(02,03,max_steps=625,wrap=True)
encoder_2.steps = 625

# Creating encoder object using GPIO pins 20 and 21
encoder        =
    ReadRotaryEncoder(21,20,max_steps=625,wrap=True)
encoder.steps = 625
cpr           = 1250

# Initialize variables for angular velocity
    measurement
last_time     = time.time()
last_steps    = encoder.steps
last_steps_m  = encoder_m.steps

#Communiaction Clients initiation
udpA          =
    COM(ETH_SERVER_IP,ETH_SERVER_PORT_A,UDP)
udpAdot       =
    COM(ETH_SERVER_IP,ETH_SERVER_PORT_A_DOT,UDP)
udpB          =
    COM(ETH_SERVER_IP,ETH_SERVER_PORT_B,UDP)
udpBdot         =
    COM(ETH_SERVER_IP,ETH_SERVER_PORT_B_DOT,UDP)
udpX          =
    COM(ETH_SERVER_IP,ETH_SERVER_PORT_X,UDP)
udpXdot       =
    COM(ETH_SERVER_IP,ETH_SERVER_PORT_X_DOT,UDP)

udpCTRL       =
    COM(ETH_CLIENT,ETH_SERVER_PORT_CTRL,UDP)

while True:
 try:
  if encoder_m.wait_for_rotate() or
      encoder.wait_for_rotate():
   udpA.SendData(str(encoder.readPosition(cpr)))
    udpB.SendData(str(encoder_2.readPosition(cpr)))
   udpX.SendData(str(encoder_m.readPosition(cpr_m)))
   udpAdot.SendData(str(encoder.readVelocity(cpr,
    last_time,last_steps)))
    udpBdot.SendData(str(encoder_2.readVelocity(cpr,
    last_time,last_steps)))
   udpXdot.SendData(str(encoder_m.readVelocity(cpr_m,
    last_time,last_steps_m)))
   U = udpCTRL.RecData()
   print(U)
 except KeyboardInterrupt:
  break
```

Listing 4. Progarmmbeispiel via Ethernet

Der zweite Teil läuft auf dem Computer und ist eine Python-Implementierung des LQR-Controllers.

```
import numpy as np
import control as ct
import matplotlib.pyplot as plt
from scipy.linalg import solve_continuous_are
import time
from com import COM

# define UDP server
ETH_IP = "10.0.8.40"
ETH_IP_PI = "10.0.8.55"
FIVEG_IP = "10.0.3.55"
FIVEG_IP_PI = "10.0.5.13"
UDP_PORT_RECV_ALPHA = 890
UDP_PORT_RECV_ALPHA_DOT = 990
UDP_PORT_RECV_BETA = 892
UDP_PORT_RECV_BETA_DOT = 992
UDP_PORT_RECV_X = 893
UDP_PORT_RECV_X_DOT = 993
UDP_PORT_SEND = 5000
UDP = 1

# Parameter defintion
pi = np.pi
mc = 0.232
m1 = 0.127
m2 = 0.127
L1 = 0.3
L2 = 0.3
LC1 = 0.3
LC2 = 0.3
I1 = m1 * LC1**2
I2 = m2 * LC2**2
g = 9.81


# Intermediates
h1 = mc + m1 + m2
h2 = m1 * LC1 + m2 * L1
h3 = m2 * LC2
h4 = m2 * LC1**2 + m2 * L1**2 + I1
h5 = m2 * LC2 * L1
h6 = m2 * LC2**2 + I2
h7 = m1 * LC1 * g + m2 * L1 * g
h8 = m2 * LC2 * g

udpA = COM(ETH_IP, UDP_PORT_RECV_ALPHA,UDP)
udpAdot = COM(ETH_IP, UDP_PORT_RECV_ALPHA_DOT,UDP)
udpB = COM(ETH_IP, UDP_PORT_RECV_BETA,UDP)
udpBdot = COM(ETH_IP, UDP_PORT_RECV_BETA_DOT,UDP)
udpX = COM(ETH_IP, UDP_PORT_RECV_X,UDP)
udpXdot = COM(ETH_IP, UDP_PORT_RECV_X_DOT,UDP)

udpSend = COM(ETH_IP_PI,UDP_PORT_SEND,UDP)


# Dynamics
M = np.array(
    [
        [1, 0, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0],
        [0, 0, 0, h1, h2, h3],
        [0, 0, 0, h2, h4, h5],
        [0, 0, 0, h3, h5, h6],
    ]
)
N = np.array(
    [
        [0, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 1, 0],
        [0, 0, 0, 0, 0, 1],
        [0, 0, 0, 0, 0, 0],
        [0, -h7, 0, 0, 0, 0],
        [0, 0, -h8, 0, 0, 0],
    ]
```

```
)
F = np.array([[0], [0], [0], [1], [0], [0]])

A = np.linalg.solve(M, N)
B = np.linalg.solve(M, F)
C = np.array([1, 0, 0, 0, 0, 0])
D = 0

Q = np.array(
    [
        [3000, 0, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0],
        [0, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 1, 0],
        [0, 0, 0, 0, 0, 1],
    ]
)
R = np.array([[10]])

P = solve_continuous_are(A, B, Q, R)
K = np.linalg.inv(R) @ B.T @ P


def lqr_contol(x):
    return np.clip(-K @ x, -20, 20)

while True:
    try:
        Alpha = float(udpA.Rec_Message())
        AlphaDot = float(udpAdot.Rec_Message())
        Beta = float(udpB.Rec_Message())
        BetaDot = float(udpBdot.Rec_Message())
        X = float(udpX.Rec_Message())
        XDot = float(udpXdot.Rec_Message())
        x0 = np.array([[X], [Alpha], [Beta],
             [XDot], [AlphaDot], [BetaDot]])
        time.sleep(0.001)
        u = lqr_contol(x0)
        print(u[0][0])
        udpSend.Send_Message(float(u[0][0]))
    except KeyboardInterrupt:
        break
```

Listing 5. LQR.py

Die Übertragung mittels 5G-Technologie erfolgt auf eine andere Art als bei Ethernet. Die Verbindung zwischen dem Raspberry Pi und dem Modul erfolgt über eine serielle Schnittstelle, über die die Daten mittels AT-Befehlen weitergeleitet werden. Der folgende Codeausschnitt zeigt beispielhaft, wie die Daten mit Hilfe der Bibliothek Pyserial gesendet und empfangen werden. Die Kommunikationsschnittstelle wurde bis zum Ende des Praktikums jedoch noch nicht perfektioniert.

```
#This is the code for the 5G interface for
    communnicating sensor and control data through
    the 5G Module interface of the RPi
from com import COM
from gpiozero import RotaryEncoder #to creat Rotary
    encoder instance
from rotaryencoder import ReadRotaryEncoder
import numpy as np
import time
import serial
from curses import ascii

# Creating encoder object using GPIO pins 20 and 21
encoder =
    RotaryEncoder(21,20,max_steps=625,wrap=True)
encoder.steps = 625
cpr= 1250

# Initiating USB Serial port instance
s = serial.Serial("/dev/ttyUSB2",115200) # creates
    a serial instance through which the
    communicates with 5G mode through at-commands
    and different data types
rec_buff= '' # temp variable


def initiateUDP():
 send_at('AT','OK',1)
 send_at('ATE1','OK',1)
 if send_at('AT+NETOPEN?','+NETOPEN: 0',1):
  send_at('AT+NETOPEN','OK',1)
  time.sleep(0.01)
 send_at('AT+CIPOPEN=1,"UDP",,,5000','OK',1)

def send_at(command,back,timeout):
 rec_buff= ''
 s.write((command+ '\r\n').encode())
 time.sleep(timeout)
 if s.inWaiting():
  time.sleep(0.01)
  rec_buff=s.read(s.inWaiting())
 if back not in rec_buff.decode():
  print(command + 'ERROR')
  print(command + ' back:\t' + rec_buff.decode())
  return 0
 else:
  print(rec_buff.decode())
  return back
def readValue(CPR):
  if encoder.wait_for_rotate():
   sensorFloat = (2*np.pi/ cpr*encoder.steps)
   if sensorFloat >= 0:
    sensorFloat = sensorFloat
   if sensorFloat < 0:
    sensorFloat = sensorFloat
   sensor = str(sensorFloat)
   return sensor
def ask_ME(CPR):
   sensor = readValue(CPR)
   AT = 'AT+CIPSEND=1,,"10.0.3.55",664'
   send_at(AT,'',1)
   time.sleep(0.5)
   s.write(sensor.encode())
   ctrl = chr(26)
   s.write(ctrl.encode())
   time.sleep(0.05)

def CloseConnection():
 send_at('AT+CIPCLOSE=0','OK',1)
 send_at('AT+NETCLOSE','OK',1)


initiateUDP()
while True:
 try:
  ask_ME(1250)
  time.sleep(0.5)
  s.write(chr(10).encode())
  data = send_at('AT+CIPRXGET=0','',1)
  print(data)
 except KeyboardInterrupt:
  break
CloseConnection()
```

Listing 6. fiveg.py

V. Stand des Projekts

Das Projekt zur Entwicklung und Steuerung eines inversen Doppelpendels ist bis zum Ende des Praktikums noch nicht abgeschlossen. Es gibt mehrere Bereiche, in denen noch Fortschritte gemacht werden müssen, um das Projekt erfolgreich abzuschließen.

Die Implementierung der 5G-Kommunikationsschnittstelle, die in diesem Projekt eine entscheidende Rolle spielt, ist noch nicht vollständig ausgereift. Trotz einiger Fortschritte sind noch technische Herausforderungen zu bewältigen, um eine stabile und zuverlässige Datenübertragung zu gewährleisten. Eine reibungslose 5G-Verbindung ist für die Echtzeitsteuerung des inversen Doppelpendels von entscheidender Bedeutung und muss daher sorgfältig optimiert werden.

Für den physischen Aufbau des inversen Doppelpendels fehlen noch einige wichtige Komponenten. Diese fehlenden Teile haben den Aufbau und die Inbetriebnahme des Systems verzögert. Es ist notwendig, diese Komponenten so schnell wie möglich zu beschaffen und zu integrieren, um die Hardware-Seite des Projekts abzuschließen und mit den umfangreichen Tests beginnen zu können.

Ein weiterer wichtiger Punkt, der bisher noch nicht ausreichend behandelt wurde, ist das Testen der Reglerparameter. Ohne diese Tests kann die Leistungsfähigkeit und Stabilität des Reglers nicht validiert werden. Die Abstimmung der Reglerparameter ist entscheidend für eine präzise und stabile Regelung des inversen Doppelpendels. Bisherige Verzögerungen in anderen Projektbereichen haben sich auch auf diesen wichtigen Schritt ausgewirkt.

Zusammenfassend ist festzustellen, dass das Projekt zur Entwicklung und Steuerung eines inversen Doppelpendels trotz erheblicher Fortschritte noch nicht abgeschlossen ist. Die Optimierung der 5G-Kommunikationsschnittstelle, die Beschaffung fehlender Komponenten und das Testen der Reglerparameter sind entscheidende Schritte, die noch durchgeführt werden müssen. Diese Herausforderungen müssen bewältigt werden, um das Projekt erfolgreich abzuschließen und die gesteckten Ziele zu erreichen.

VI. Simulationsergebnisse und Ausblick

In diesem Abschnitt werden die Simulationsergebnisse des inversen Doppelpendels mit Hilfe von MATLAB dargestellt. Als Regelstrategie wurde der lineare quadratische Regler (LQR) verwendet. Durch die Simulationen konnten wichtige Erkenntnisse über das Systemverhalten und die Effizienz des LQR gewonnen werden.

Die Simulationen zeigen, dass der LQR eine effektive Methode zur Regelung des inversen Doppelpendels darstellt. Die Stabilität des Systems konnte unter verschiedenen Bedingungen erfolgreich gewährleistet werden. Besonders positiv hervorzuheben ist die Fähigkeit der LQR, trotz der nichtlinearen Dynamik des Pendels eine präzise Regelung zu ermöglichen. Diese Ergebnisse bestätigen die theoretischen Erwartungen und zeigen das Potential des LQR für den praktischen Einsatz.

Die Aussichten für den Abschluss des Projekts sind positiv, wenn alle bestehenden Probleme gelöst sind. Sobald die

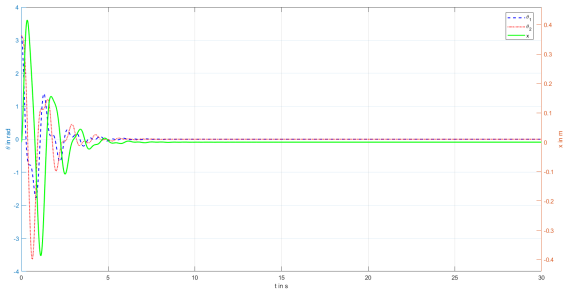

Abbildung 13. Simulationsdarstellung von x, θ_1 und θ_2 als Funktion der Zeit

5G-Kommunikationsschnittstelle vollständig optimiert ist, wird die Echtzeitübertragung von Sensordaten und Steuerbefehlen gewährleistet sein, was eine reibungslose und präzise Steuerung ermöglicht.

Mit der Beschaffung der fehlenden Komponenten kann der physische Aufbau des Systems abgeschlossen und in Betrieb genommen werden. Dies ermöglicht umfangreiche Tests und Feinabstimmungen, die notwendig sind, um die theoretischen Ergebnisse der Simulationen in die Praxis umzusetzen.

Das Testen und Optimieren der LQR-Parameter wird entscheidend zur Stabilität und Leistungsfähigkeit des Systems beitragen. Mit den Simulationsergebnissen haben wir bereits eine solide Basis, auf der weiter aufgebaut werden kann. Mit den gewonnenen Erkenntnissen kann der LQR für den praktischen Einsatz optimiert und angepasst werden.

Insgesamt zeigt der Ausblick, dass das Projekt trotz der bestehenden Herausforderungen auf einem guten Weg ist. Die bisherigen Simulationsergebnisse stellen eine solide Basis dar, auf der weiter aufgebaut werden kann. Wenn alle identifizierten Probleme erfolgreich gelöst werden, kann das Projekt erfolgreich abgeschlossen und ein funktionierendes System zur Steuerung des inversen Doppelpendels realisiert werden.

Literaturverzeichnis

[1] Autoren, Wikipedia: *Kleinwinkelnäherung*. https://de.wikipedia.org/wiki/Kleinwinkeln%C3%A4herung. Version: August 2022

[2] RaspberryPi: *RaspberryPi 4*. https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[3] logic, SparkFun: *SparkFun Logic Level Converter, Bi-directional*. https://www.sparkfun.com/products/12009

[4] Waveshare: *SIM8200EA-M2 5G HAT for Raspberry Pi*. https://www.waveshare.com/product/raspberry-pi/hats/iot/sim8200ea-m2-5g-hat.htm

[5] IGUS (Hrsg.): *MMOT-AN-S-..._DE_20240403*. IGUS, https://www.igus.de/product/MOT-AN-S-060-020-056-M-C-AAAC?artnr=MOT-AN-S-060-020-056-M-C-AAAC

[6] Leadshine: *Stepper motor driver Leadshine M880A*. https://mecheltron.com/en/product/m880a

[7] AVAGO (Hrsg.): *AEDB-9340 Series*. AVAGO, https://docs.broadcom.com/doc/AV02-0075EN

[8] Cortinex: *LTK-0507-301-502 Datasheet (PDF) - Contrinex AG Industrial Electronics*, https://pdf1.alldatasheet.com/datasheet-pdf/view/1004901/CONTRINEX/LTK-0507-301-502.html