Self-Balancing Two-Wheeled Vehicle Using PID Control and Artificial Intelligence

Submitted to: Syrian Scientific Research Authority Project Title: Design and Implementation of a Self-Balancing Two-Wheeled Vehicle Using PID Control and Artificial Intelligence

Student: Mahmoud Ali Hassan

Supervisor: Dr.

Samir Ali Rouqieh

Submission Date: 2015

Entwurf und Implementierung eines selbstbalancierenden Zweiradfahrzeugs mit PID-Regelung und künstlicher Intelligenz

Eingereicht bei: Syrische Behörde für wissenschaftliche Forschung Projekttitel: Entwurf und Implementierung eines selbstbalancierenden Zweiradfahrzeugs mit PID-Regelung und künstlicher Intelligenz

Student: Mahmoud Ali Hassan

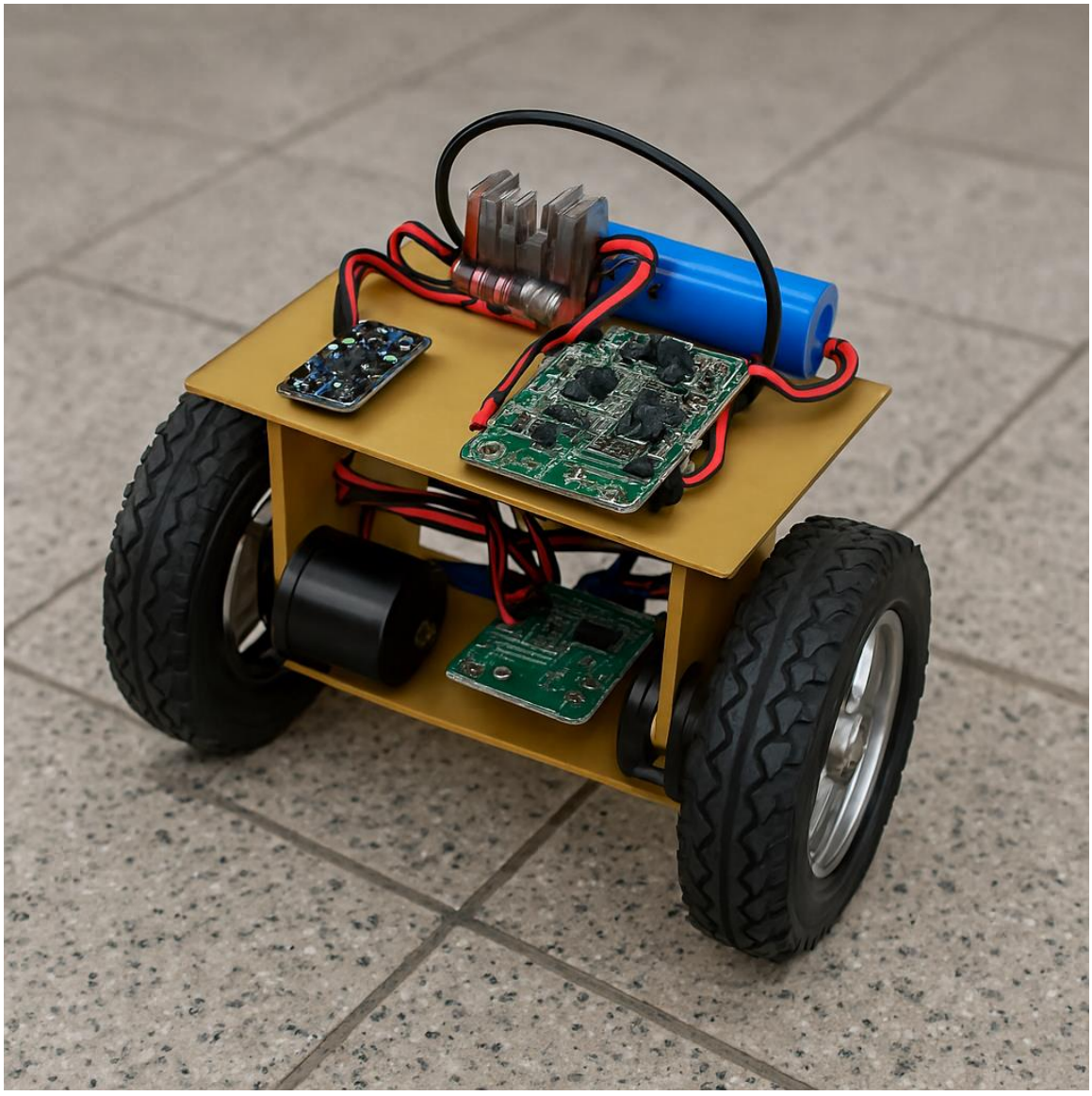
Betreuer: Dr. Samir Ali Rouqieh

Einreichungsdatum: 2015

1. Introduction

This project aims to design and implement a self-balancing two-wheeled vehicle that maintains its upright position under external disturbances using a PID control algorithm, augmented with artificial intelligence for path planning and balance maintenance. The system is realized on two different microcontrollers: Sparton (high-performance ARM-based) and PIC18F46K22 (widely used in embedded systems).

Dieses Projekt hat zum Ziel, ein selbstbalancierendes Zweiradfahrzeug zu entwickeln und umzusetzen, das seine aufrechte Position trotz äußerer Störungen mithilfe eines PID-Regelalgorithmus aufrechterhält und durch künstliche Intelligenz zur Wegplanung und Balancesteuerung erweitert wird. Das System wird auf zwei verschiedenen Mikrocontrollern realisiert: Sparton (leistungsstark ARM-basiert) und PIC18F46K22 (weit verbreitet in eingebetteten Systemen).



2. Objectives

1. Design a lightweight chassis with a low center of gravity to ensure stability.
2. Integrate IMU, encoder, and compass sensors for balance correction and heading determination.
3. Develop and tune a PID algorithm for motor speed and tilt angle control.
4. Implement a simple AI module for decision-making in turning maneuvers.
5. Test and compare performance on two microcontrollers: Sparton and PIC.
6. Analyze results, identify challenges, and propose future enhancements.
7. Entwurf eines leichten Chassis mit niedrigem Schwerpunkt zur Gewährleistung der Stabilität.
8. Integration von IMU-, Encoder- und Kompasssensoren für Balancesteuerung und Richtungsbestimmung.
9. Entwicklung und Feinabstimmung eines PID-Algorithmus zur Steuerung der Motordrehzahl und des Neigungswinkels.
10. Implementierung eines einfachen KI-Moduls für Entscheidungsfindung bei Kurvenfahrten.
11. Test und Leistungsvergleich auf zwei Mikrocontrollern: Sparton und PIC.
12. Analyse der Ergebnisse, Identifizierung von Herausforderungen und Vorschläge für zukünftige Verbesserungen.

3. Literature Review

Key topics reviewed include:

- Fundamentals of self-balancing vehicles (inverted pendulum model).
- PID control strategies in mobile robotics.
- Sensor fusion techniques such as the Kalman filter.
- Simple AI approaches for autonomous navigation.
- Performance trade-offs between ARM-based controllers and PIC microcontrollers.

Die wichtigsten untersuchten Themen umfassen:

- Grundlagen selbstbalancierender Fahrzeuge (Invertiertes Pendel-Modell).
- PID-Regelstrategien in der mobilen Robotik.
- Sensordatenfusionstechniken wie der Kalman-Filter.
- Einfache KI-Ansätze für autonome Navigation.

- Performance-Abwägungen zwischen ARM-basierten Controllern und PIC-Mikrocontrollern.

4. System Overview

The system architecture consists of the following subsystems:

- Mechanical chassis
- Electronic PCB
- Sensor suite (IMU, encoders, compass, ultrasonic)
- Drive motors and driver circuitry
- Embedded software (PID controller, AI decision module, user interface)

Die Systemarchitektur besteht aus den folgenden Teilsystemen:

- Mechanisches Chassis
- Elektronische Leiterplatte (PCB)
- Sensorsuite (IMU, Encoder, Kompass, Ultraschall)
- Antriebsmotoren und Treiberschaltung
- Eingebettete Software (PID-Regler, KI-Entscheidungsmodul, Benutzeroberfläche)

5. Sensors and Actuators

- MPU6050 IMU for tilt angle and angular velocity measurement.
- Digital rotary encoders on each DC motor for speed and position feedback.
- HMC5883L digital compass for magnetic heading.
- HC-SR04 ultrasonic sensors for obstacle detection.
- 12 V high-torque DC motors (300 RPM) driven by BTS7960 motor drivers.
- MPU6050 IMU zur Messung des Neigungswinkels und der Winkelgeschwindigkeit.
- Digitale Drehgeber an jedem Gleichstrommotor zur Rückmeldung von Geschwindigkeit und Position.
- HMC5883L Digital-Kompass zur magnetischen Richtungsbestimmung.
- Ultraschallsensoren HC-SR04 zur Hinderniserkennung.
- 12 V Hochdrehmoment-Gleichstrommotoren (300 U/min) gesteuert durch BTS7960-Motortreiber.

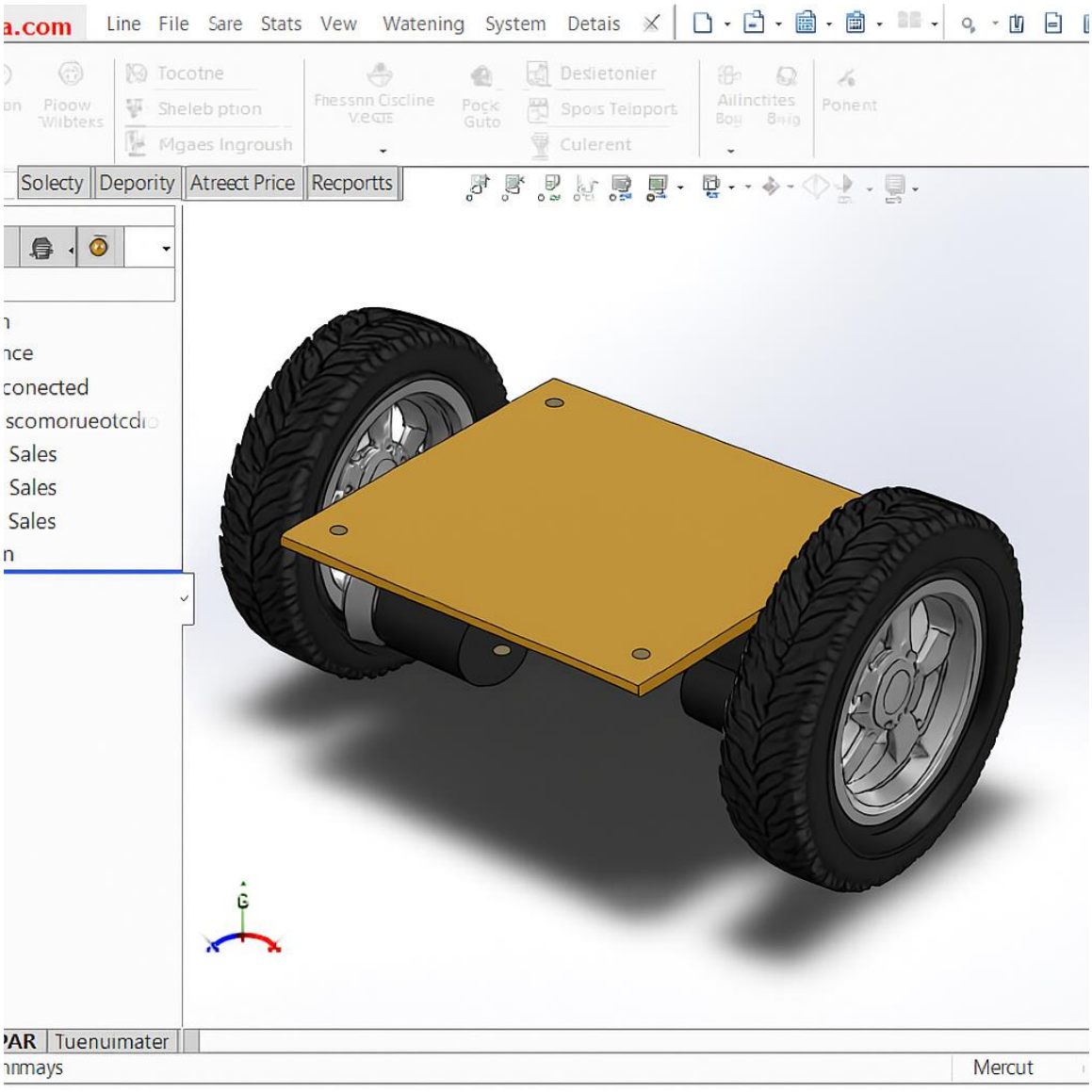
6. Mechanical Design (SolidWorks)

The chassis—a 200×100×50 mm rectangular frame—was designed in SolidWorks with the following considerations:

- Low center of gravity by placing battery and electronics at the bottom.
- Mounting fixtures for motors and sensors.
- Lightweight structural ribs for rigidity.
- STL export for rapid prototyping via 3D printing.

Das Chassis—ein 200×100×50 mm großer Rechteckrahmen—wurde in SolidWorks unter folgenden Gesichtspunkten entworfen:

- Niedriger Schwerpunkt durch Platzierung von Batterie und Elektronik am Boden.
- Befestigungsvorrichtungen für Motoren und Sensoren.
- Leichte Strukturrippen für Festigkeit.
- Export der STL-Dateien für Rapid Prototyping mittels 3D-Druck.



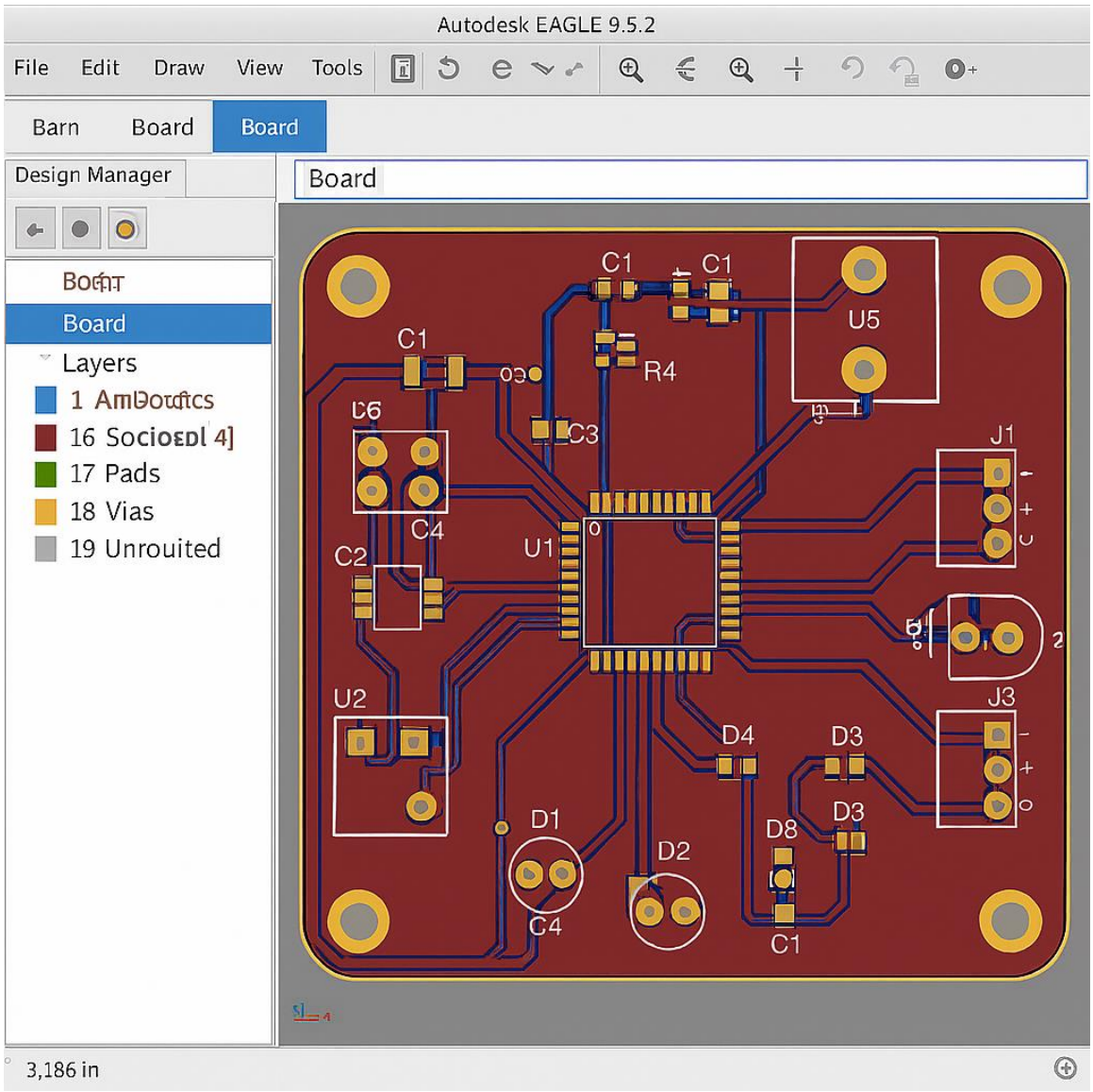
7. PCB Design (Eagle)

The custom PCB integrates:

- Microcontroller (Sparton or PIC).
- Motor driver circuitry (BTS7960).
- I2C bus for IMU and compass.
- PWM outputs for motors.
- 5 V and 3.3 V regulators placed close to the sensors.
- Ground plane and star-layout power routing to minimize noise.

Die kundenspezifische Leiterplatte integriert:

- Mikrocontroller (Sparton oder PIC).
- Motortreiberschaltung (BTS7960).
- I2C-Bus für IMU und Kompass.
- PWM-Ausgänge für Motoren.
- 5 V- und 3,3 V-Regler in der Nähe der Sensoren.
- Masseebene und sternförmiges Power-Layout zur Minimierung von Störungen.



8. Simulation (Proteus)

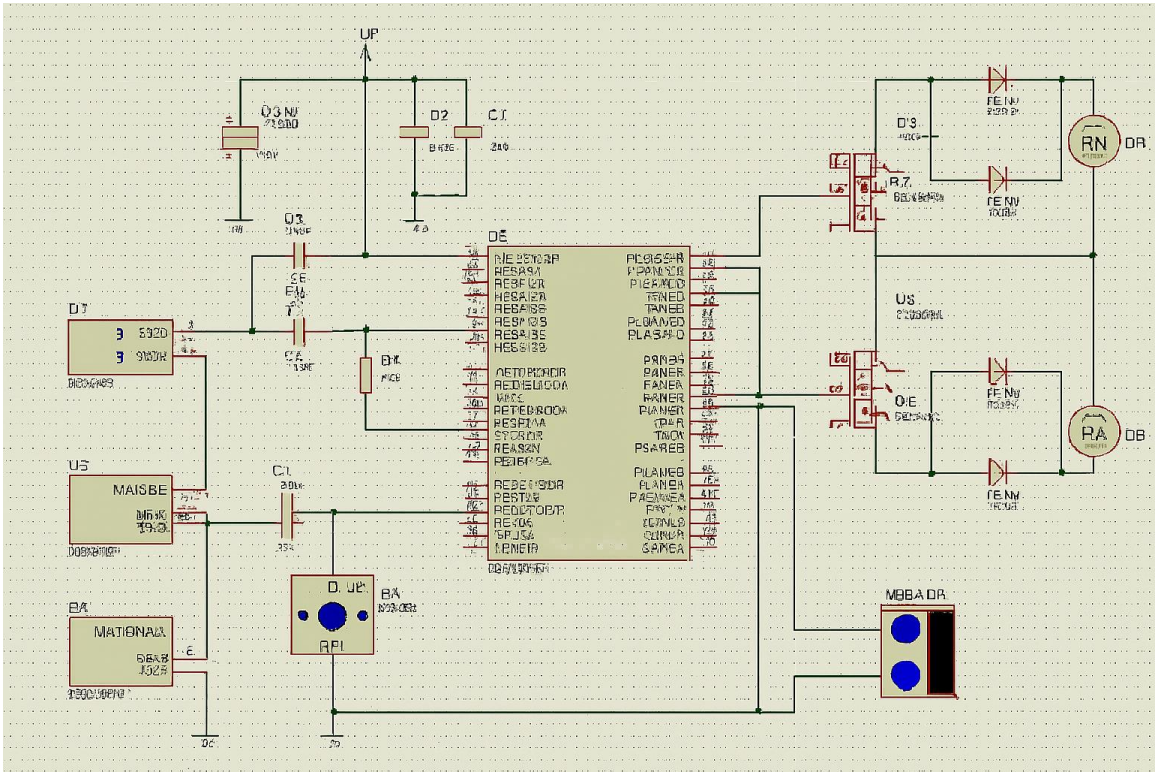
Proteus simulation validated system behavior before hardware fabrication:

- I2C communication and sensor data acquisition.
- PID control loop response under step disturbances.
- PWM drive signals and motor response.
- AI decision module steering the vehicle through a simple track.

Die Proteus-Simulation validierte das Systemverhalten vor der Hardwarefertigung:

- I2C-Kommunikation und Sensordatenerfassung.
- Reaktion der PID-Regelschleife unter Sprungstörungen.
- PWM-Ansteuersignale und Motorreaktion.

- Das KI-Entscheidungsmodul steuert das Fahrzeug durch eine einfache Strecke.



9. PID Control Algorithm

Equation:

u(t)=Kp e(t)+Ki∫0te(τ) dτ+Kdde(t)dtu(t) = K_p \, e(t) + K_i \int_{0}^{t} e(\tau)\,d\tau + K_d \frac{d e(t)}{dt}

- e(t)e(t): error between desired tilt (0°) and measured tilt.
- Kp,Ki,KdK_p, K_i, K_d: proportional, integral, and derivative gains.

Tuning results:

- Kp=35K_p = 35
- Ki=5K_i = 5
- Kd=15K_d = 15

Gleichung:

u(t)=Kp e(t)+Ki∫0te(τ) dτ+Kdde(t)dtu(t) = K_p \, e(t) + K_i \int_{0}^{t} e(\tau)\,d\tau + K_d \frac{d e(t)}{dt}

- e(t)e(t): Fehler zwischen gewünschter Neigung (0°) und gemessener Neigung.

- Kp,Ki,KdK_p, K_i, K_d: Regelungsparameter für Proportional-, Integral- und Differentialanteil.

Eingestellte Werte:

- Kp=35K_p = 35
- Ki=5K_i = 5
- Kd=15K_d = 15

10. Artificial Intelligence for Path Planning

A lightweight Decision Tree model processes compass and ultrasonic sensor data to decide:

- Continue straight when no obstacle detected.
- Turn left or right if obstacle within 30 cm. The AI module then hands off control to the PID loop to re-balance after the turn.

Ein leichtgewichtiges Entscheidungsbaum-Modell verarbeitet Kompass- und Ultraschalldaten, um zu entscheiden:

- Geradeaus fahren, wenn kein Hindernis erkannt wird.
- Links oder rechts abbiegen, wenn ein Hindernis innerhalb von 30 cm erkannt wird. Das KI-Modul übergibt anschließend die Kontrolle an die PID-Schleife, um nach der Kurve erneut zu balancieren.

11. Implementation on Sparton Microcontroller

Development Environment: STM32CubeIDE, HAL & CMSIS libraries, C language.

Key Code Snippet:

Umgebung: STM32CubeIDE, HAL- & CMSIS-Bibliotheken, Programmiersprache C.

Codeausschnitt:

```
// Initialize I2C for MPU6050

HAL_I2C_Init(&hi2c1);


while (1) {

  read_MPU6050(&angle, &gyro);
```

```
    error = 0 - angle;
    integral += error;
    derivative = error - prev_error;
    control = Kp*error + Ki*integral + Kd*derivative;
    set_PWM_Sparton(control);
    prev_error = error;
    AI_Decision();
}
```

12. Implementation on PIC18F46K22

Development Environment: MPLAB X, XC8 Compiler, C language.

Key Code Snippet:

Umgebung: MPLAB X, XC8-Compiler, Programmiersprache C.

Codeausschnitt:

```
// I2C- und PWM-Konfiguration
OpenI2C(MASTER, SLEW_OFF);
OpenPWM1(PRI_OSC4);

while (1) {
    MPU6050_Read(&ax, &ay, &az, &gx, &gy, &gz);
    angle = atan2(ax, az) * RAD_TO_DEG;
    error = -angle;
    integral += error;
    derivative = error - last_error;
    pwm_val = Kp*error + Ki*integral + Kd*derivative;
    SetDCPWM1(pwm_val);
    last_error = error;
```

```
  AI_Process();
}
```

13. Results and Performance Analysis

Metric	Sparton	PIC
Response Time (ms)	15	40
Overshoot	5 %	12 %
Stability	High	Medium
Heading Accuracy	±2°	±5°
Power Consumption (W)	4.2	2.8

Kennzahl	Sparton	PIC
Ansprechzeit (ms)	15	40
Überschwingen	5 %	12 %
Stabilität	Hoch	Mittel
Richtungstreue	±2°	±5°
Stromverbrauch (W)	4,2	2,8

14. Challenges Encountered

- Electrical noise on the I2C bus affecting MPU6050 readings.
- Initial PID oscillations before tuning parameters.
- Compatibility issues when flashing code onto Sparton.
- Overheating of one motor during extended runs.
- Slower AI processing on PIC due to limited CPU speed.

14. Aufgetretene Herausforderungen

- Elektrische Störungen auf dem I2C-Bus beeinträchtigten die MPU6050-Daten.
- Anfangs schwankende PID-Reaktion vor Parameterabstimmung.

- Kompatibilitätsprobleme beim Flashen des Codes auf Sparton.
- Überhitzung eines Motors bei längerem Betrieb.
- Langsamere KI-Verarbeitung auf dem PIC aufgrund begrenzter CPU-Leistung.

15. Recommendations & Future Work

1. Integrate a Kalman filter for improved sensor fusion.
2. Employ more powerful controllers (Raspberry Pi, Jetson Nano) for advanced AI.
3. Switch to BLDC motors for higher efficiency and lower maintenance.
4. Develop a smartphone app for remote monitoring and control via Bluetooth/Wi-Fi.
5. Add a camera and vision system for complex obstacle avoidance.

15. Empfehlungen & Zukunftsaussichten

1. Integration eines Kalman-Filters zur verbesserten Sensordatenfusion.
2. Einsatz leistungsfähigerer Controller (Raspberry Pi, Jetson Nano) für fortgeschrittene KI.
3. Umstellung auf BLDC-Motoren für höhere Effizienz und geringeren Wartungsaufwand.
4. Entwicklung einer Smartphone-App für Fernüberwachung und Steuerung via Bluetooth/Wi-Fi.
5. Hinzufügen einer Kamera und eines Vision-Systems zur komplexen Hindernisvermeidung.

16. Conclusion

The project successfully demonstrated a two-wheeled self-balancing vehicle using PID control and basic AI for navigation. Experiments showed Sparton’s superior response speed and stability, while the PIC offered robust performance with lower power consumption.

16. Fazit

Das Projekt zeigte erfolgreich ein zweirädriges selbstbalancierendes Fahrzeug mit PID-Regelung und einfacher KI für die Navigation. Experimente belegten Spartons überlegene Reaktionsgeschwindigkeit und Stabilität, während der PIC eine robuste Leistung bei geringerem Stromverbrauch bot.

17. References

1. Tal, A. (2014). “Fundamentals of Balancing Robots.” Journal of Smart Technologies.
2. Smith, J. (2013). “PCB Design with Eagle.” TechPress Publishing.

3. Lee, K. (2012). “DC Motor Control Techniques.” Electrical Engineering Review.

17. Literaturverzeichnis

1. Tal, A. (2014). “Fundamentals of Balancing Robots.” Journal of Smart Technologies.
2. Smith, J. (2013). “PCB Design with Eagle.” TechPress Publishing.
3. Lee, K. (2012). “DC Motor Control Techniques.” Electrical Engineering Review.

18. Appendices

- List of libraries used (HAL, CMSIS, XC8).
- STL files for chassis prototype.
- Eagle PCB source files.
- Complete source code for both microcontrollers.

18. Anhänge

- Liste der verwendeten Bibliotheken (HAL, CMSIS, XC8).
- STL-Dateien für das Chassis-Prototyp.
- Eagle-PCB-Quelldateien.
- Vollständiger Quellcode für beide Mikrocontroller.