Projektarbeit: **PanzerRoboter** auf dem STM32F407-ORB-Board

Luan Bytyqi
Hochschule Bonn-Rhein-Sieg

Juli 2025

Inhaltsverzeichnis

1 Einleitung

Im Rahmen dieser Projektarbeit wurde ein fernsteuerbarer **Roboter** auf Basis des STM32F407-ORB-Board entwickelt. Ziel war es, mithilfe von C++ und der EmbSys-Library einen steuerbaren Roboter mit modularer Klassenarchitektur zu implementieren, der per Tastatureingabe über das Terminal kontrolliert werden kann.

2 Funktionalität

Der Roboter kann sich in vier Richtungen bewegen und durch Benutzereingaben live gesteuert werden. Die Bewegungsbefehle werden über die Tastatur an das System gesendet. Zusätzlich lässt sich die PWM-Leistung verändern.

- `W`: Vorwärts fahren
- `S`: Rückwärts fahren
- `A`: Links drehen
- `D`: Rechts drehen
- Pfeiltasten ↑/↓: PWM erhöhen/verringern
- `Space`: Stop

3 Systemarchitektur

Das Projekt basiert auf einem objektorientierten Design. Die Steuerlogik ist entkoppelt von der konkreten Hardware. Es wurden Interfaces für Motoren und Roboter implementiert, um Austauschbarkeit und Testbarkeit zu gewährleisten.

3.1 Klassenübersicht

- `PanzerRoboter`: Implementiert das Verhalten eines Roboters mit zwei Motoren
- `Motor`: Konkrete Implementierung eines Motors mit PWM- und Richtungssteuerung
- `IRoboter_`, `IMotor`, `IPin`, `IAnalogOut`: Abstrakte Interfaces
- `AnalogOutAdapter`, `DigitalAdapter`: Hardwareadapter für das Board
- `MathUtils`: Enthält Hilfsfunktionen wie `clamp()`

3.2 UML-Diagramm

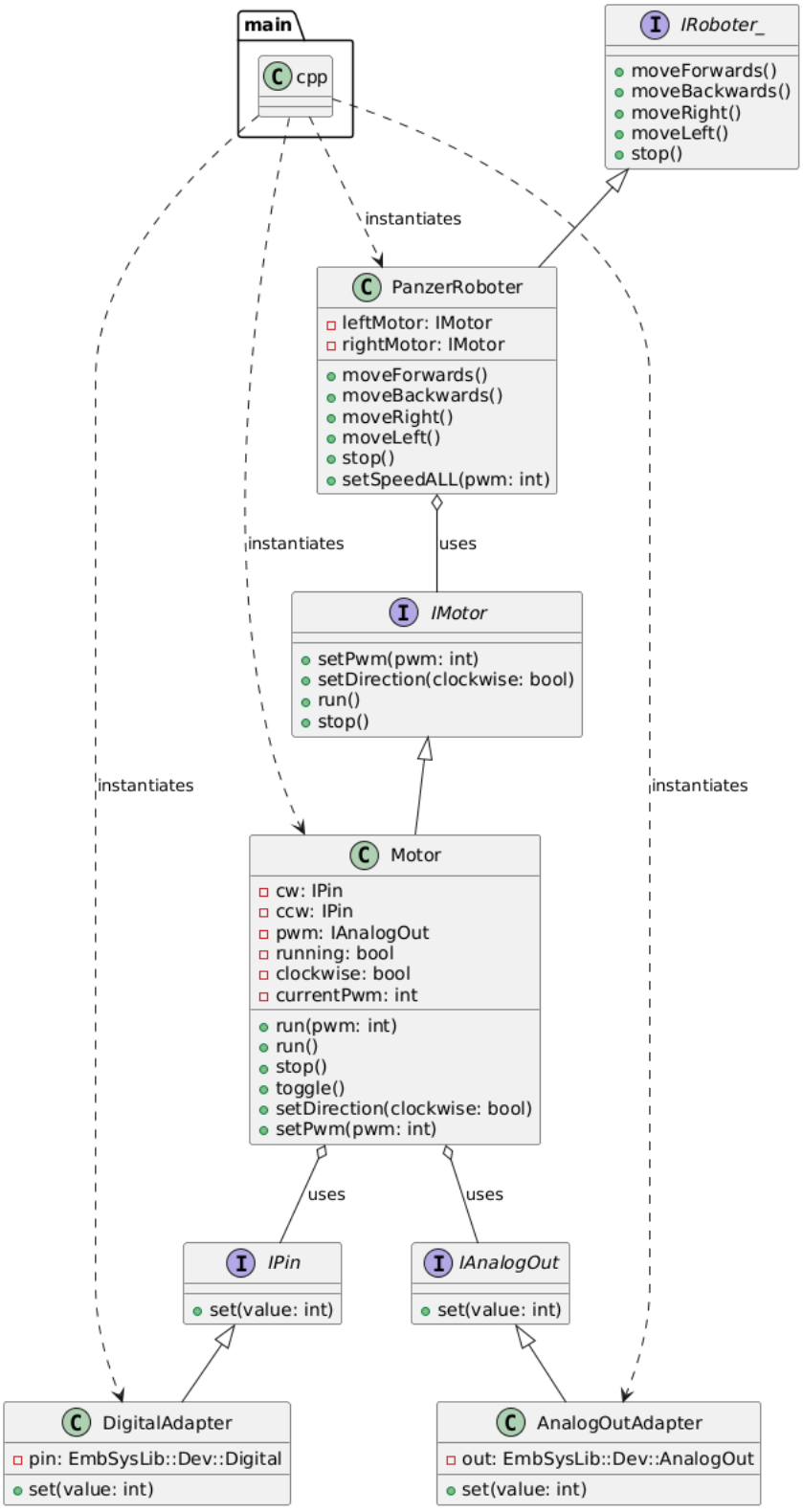

Abbildung 1: UML-Diagramm der Systemarchitektur

4 Codeausschnitt

Ein zentrales Beispiel für die Architektur dieses Projekts ist die Implementierung der Motorsteuerung. Die Klasse `Motor` kapselt die Logik für Drehrichtung, PWM-Steuerung und Adapterverwendung und basiert auf dem Interface `IMotor`.

Listing 1: Implementierung der Motorsteuerung

```
class Motor : public IMotor {
private:
    IPin &cw;
    IPin &ccw;
    IAnalogOut &pwm;
    bool running;
    bool clockwise;
    int currentPwm;

    void updateDirectionPins() {
        cw.set(clockwise ? 1 : 0);
        ccw.set(clockwise ? 0 : 1);
    }

    void updatePwmOutput() {
        pwm.set(running ? currentPwm : 0);
    }

public:
    explicit Motor(IPin &_cw, IPin &_ccw, IAnalogOut &_pwm)
        : cw(_cw), ccw(_ccw), pwm(_pwm),
          running(false), clockwise(false), currentPwm(0) {}

    void run(int pwmValue) {
        currentPwm = clamp(pwmValue, 0, 800);
        running = true;
        updateDirectionPins();
        updatePwmOutput();
    }
    ...
};
```

Die Motorsteuerung kann so unabhängig vom konkreten Board getestet oder wiederverwendet werden.

Listing 2: Motorübergabe

```
PanzerRoboter robot(motorLinks, motorRechts, 400);
```

Die konkrete Hardwareanbindung erfolgt durch Übergabe von Motorinstanzen, welche über Adapter mit den PWM- und Digital-Pins des STM32 verbunden sind.

5 Besonderheiten und Herausforderungen

- Entkopplung der Hardware-Implementierung durch Adapter und Interfaces
- PWM-Anpassung in Echtzeit über Benutzereingabe
- Testbarkeit durch abstrakte Motor- und Roboter-Schnittstellen

6 Fazit

Die Arbeit an diesem Projekt hat mir nicht nur Spaß gemacht, sondern mir auch ein tieferes Verständnis für die objektorientierte Entwicklung auf Embedded-Systemen vermittelt. Besonders spannend war die Umsetzung einer modularen Architektur mit klar getrennten Zuständigkeiten und austauschbaren Komponenten.

Das Projekt bildet eine solide Basis für zukünftige Erweiterungen, etwa um Sensorik, autonomes Verhalten oder eine visuelle Rückmeldung über ein Display.

7 Anhang

- Projektdateien: `main.cpp`, `Motor.h`, `Roboter.h`, etc.
- Quellverzeichnis: Embitz Projekt mit Header- und Implementierungsdateien