Zusammenfassung

Beherrschung des TT-Motors

- **Grundlagen von Motoren**:
  - Ein Motor wandelt elektrische in mechanische Energie um, basierend auf dem Prinzip der elektromagnetischen Induktion.
  - Der TT-Motor, ein Getriebemotor in einem Kunststoffgehäuse, erhöht durch Zahnräder das Drehmoment für effektive Bewegungen.
- **Steuerung von Motoren**:
  - Direktes Anschließen eines Motors an eine Batterie führt zur Drehung; die Umkehr der Anschlüsse ändert die Drehrichtung.
  - Arduino-Boards allein reichen nicht aus, um Motoren zu betreiben, da deren Signalpins nicht genügend Strom liefern.
  - Motor-Treiber dienen als Verstärker zwischen Arduino und Motor, um Bewegungen zu steuern.
- **Einsatz des GalaxyRVR-Shields**:
  - Das Shield dient als Schnittstelle für die Steuerung von bis zu sechs Motoren und verbindet Sensoren sowie Stromversorgung.
  - Die Steuerung erfolgt über spezifische Pins, die an Motor-Treiber-Chips angeschlossen sind, welche die Motoren aktivieren.
- **Programmierung zur Motorsteuerung**:
  - Grundlegende Befehle (wie `digitalWrite()` und `pinMode()`) steuern Richtung und Aktivität des Motors.
  - Die Anwendung der Pulsweitenmodulation (PWM) ermöglicht die Feinabstimmung der Motorgeschwindigkeit.
  - Arduino-Bibliotheken wie SoftPWM erweitern die Möglichkeiten zur Geschwindigkeitskontrolle durch Software.

Antriebslogik

TAB. 1

INA	INB	Motor
L	L	Standby
L	H	Im Uhrzeigersinn
H	L	Gegen den Uhrzeigersinn
H	H	Bremse

Anmerkungen:

Treiberchips mit den Pins 2, 3, 4 und 5 und verwenden die SoftPWM-Bibliothek von Arduino, um PWM auf diesen Pins zu ermöglichen.

```
/**
 * @file main.cpp
 * @brief Wie würdest du den Code ändern, um sechs Motoren gleichzeitig zu steuern?
 */
#include <Arduino.h>
const int in3 = 4;
const int in4 = 5;

void setup() {
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);
}

void loop() {
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
    delay(2000);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
    delay(2000);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, HIGH);
    delay(5000);
}
```

\newpage

Um sechs Motoren gleichzeitig zu steuern und dabei die Antriebslogik sowie die Nutzung der SoftPWM-Bibliothek für das Arduino-Board zu berücksichtigen, muss der vorgegebene Code erweitert werden.

>>`c++

```
// sechs Motoren gleichzeitig zu steuern
/**
 * @file main.cpp
 * @brief Rover vorwärts bewegen und Rover rückwärts bewegen
 */
#include <Arduino.h>
#include <SoftPWM.h>

// Definition der Pins für die linken Motoren A, B, C
const int motorA1 = 2; // INA Plus  für Motoren A, B, C sind parallel
const int motorA2 = 3; // INB Minus für Motoren A, B, C sind parallel

// Definition der Pins für die rechten Motoren D, E, F
const int motorB1 = 5; // INA Plus  für Motoren D, E, F sind parallel
const int motorB2 = 4; // INB Minus für Motoren D, E, F sind parallel

void setup() {
    // Initialisiert die SoftPWM-Bibliothek
    SoftPWMBegin();
}

void loop() {
    // Rover vorwärts bewegen
    // linke Motoren
```

```
    SoftPWMSet(motorA1, 120);  // etwa halbe Geschwindigkeit
    SoftPWMSet(motorA2, 0);    // Stop
    // rechte Motoren
    SoftPWMSet(motorB1, 120);  // etwa halbe Geschwindigkeit
    SoftPWMSet(motorB2, 0);    // Stop

    // Rover rückwärts bewegen
    // linke Motoren
    SoftPWMSet(motorA1, 0);    // Stop
    SoftPWMSet(motorA2, 120);  // etwa halbe Geschwindigkeit
    // rechte Motoren
    SoftPWMSet(motorB1, 0);    // Stop
    SoftPWMSet(motorB2, 120);  // etwa halbe Geschwindigkeit
}
```

Steuerung der Motorgeschwindigkeit (PWM)

```
/**
 * @file main.cpp
 * @brief Steuerung der Motorgeschwindigkeit (PWM)
 */
#include <Arduino.h>
#include <SoftPWM.h>

const int in1 = 2; // PWM-Pin für Motorrichtung 1
const int in2 = 3; // PWM-Pin für Motorrichtung 2

void setup() {
    // Beginnt die serielle Kommunikation
    Serial.begin(115200);
    // Initialisiert SoftPWM für alle verwendeten Pins
    SoftPWMBegin();
    // Setzt die PWM-Werte initial auf 0
    SoftPWMSet(in1, 0);
    SoftPWMSet(in2, 0);
}

void loop() {
    // Setzt in1 auf 0, um sicherzustellen, dass der Motor in eine Richtung dreht
    SoftPWMSet(in2, 0);
    // Schleife erhöht die Geschwindigkeit von 30 bis 255
    for (int i = 30; i <= 70; i++) {
        SoftPWMSet(in1, i); // Setzt die PWM für den Motor
        Serial.println("Steuerung der Motorgeschwindigkeit (PWM): " + String(i));
        delay(100); // Kurze Verzögerung zwischen den Geschwindigkeitsänderungen
        if (i == 70) {
            // Wenn i 255 erreicht, stoppt der Motor
            SoftPWMSet(in1, 0);
            Serial.println("Motor stoppt für 1 Sekunde.");
            delay(2000); // Wartet 1 Sekunde, bevor der Motor neu startet
            break; // Beendet die Schleife, damit sie von vorne beginnen kann
        }
    }
}
```

Hardware-PWM (Pulsweitenmodulation) und Software-PWM (z.B. mit der SoftPWM-Bibliothek für Arduino) bieten beide die Möglichkeit, die Ausgangsleistung an einem Pin zu steuern, unterscheiden sich jedoch in ihrer Implementierung und Leistungsfähigkeit.

Unterschiede zwischen Hardware-PWM und Software-PWM

- **Hardware-PWM**:
  - Wird direkt von den Mikrocontroller-Hardwareeinheiten bereitgestellt.
  - Bietet eine präzisere und stabilere PWM-Signalgenerierung im Vergleich zu Software-PWM, da sie nicht von der CPU-Auslastung oder Software-Verzögerungen beeinflusst wird.
  - Die Anzahl der verfügbaren Hardware-PWM-Pins ist begrenzt und hängt vom spezifischen Mikrocontroller ab.
  - Ermöglicht in der Regel höhere Frequenzen und eine bessere Auflösung des PWM-Signals.
- **Software-PWM**:
  - Wird durch Software-Algorithmen realisiert, die auf beliebigen digitalen Pins ausgeführt werden können.
  - Kann flexibler sein, da nahezu jeder Pin als PWM-Ausgang verwendet werden kann, ist aber weniger präzise und kann durch andere Vorgänge im Programm beeinträchtigt werden.
  - Die Frequenz und Auflösung des PWM-Signals kann durch die Prozessorgeschwindigkeit und die Effizienz der Software-Implementierung begrenzt sein.
  - Kann mehr CPU-Ressourcen verbrauchen, was die Leistung anderer Teile des Programms beeinträchtigen kann.

Beispiel für Hardware-PWM mit Arduino

Steuerung der Helligkeit einer LED. Die Arduino-Boards haben Pins, mit einem Tilde-Symbol (~ Pins 3, 5, 6, 9, 10 und 11 auf dem Arduino Uno).

```
int ledPin = 9; // Pin mit Hardware-PWM-Funktion

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  for (int i = 0; i <= 255; i++) {
    analogWrite(ledPin, i); // Setzt die Helligkeit der LED
    delay(10);
  }

  for (int i = 255; i >= 0; i--) {
    analogWrite(ledPin, i); // Verringert die Helligkeit der LED
    delay(10);
  }
}
```

Reflektieren und Verbessern

- Arbeitsprinzipien von Motoren
  - Motor das Prinzip der elektromagnetischen Induktionegen.
  - TT-Getriebemotor
- Wie steuert man ihre Richtung und Geschwindigkeit durch Programmierung.
- Wie würdest du die for-Schleife ändern, um die Motorgeschwindigkeit allmählich zu verringern?
- Wie würdest du den Motor so steuern, dass er beim Drehen gegen den Uhrzeigersinn beschleunigt oder verlangsamt?

Arbeitsprinzipien von Motoren

- **Elektromagnetische Induktion**: Die Drehbewegung eines Motors entsteht durch die Interaktion eines erzeugten Magnetfelds mit den Magneten im Motor. Dieses Prinzip ermöglicht es, elektrische Energie in mechanische Bewegung umzuwandeln.
- **TT-Getriebemotor**: Die Kombination aus Motor und Getriebe in einem TT-Getriebemotor ermöglicht es, das Drehmoment zu erhöhen. Dadurch kann der Motor größere Lasten bewegen, was ihn ideal für Anwendungen wie Rover auf unebenem Terrain macht.

Steuerung der Richtung und Geschwindigkeit durch Programmierung

- **Richtungssteuerung**: Die Richtung eines Motors wird durch Ändern der Polarität der an den Motor angelegten Spannung bestimmt. In einem Programm kann dies durch Wechseln der HIGH/LOW-Zustände der Pins erreicht werden, die den Motor steuern.
- **Geschwindigkeitssteuerung**: Die Geschwindigkeit eines Motors kann durch Pulsweitenmodulation (PWM) angepasst werden, wobei die durchschnittliche Spannung, die dem Motor zugeführt wird, durch Ändern des Tastverhältnisses des PWM-Signals variiert wird.

Motor beschleunigen/verlangsamen im und gegen den Uhrzeigersinn

```
/**
 * @file main.cpp
 * @brief Motor beschleunigen/verlangsamen im / gegen den Uhrzeigersinn
 */
#include <Arduino.h>
#include <SoftPWM.h>

const int in1 = 2; // PWM-Pin für Motorrichtung 1
const int in2 = 3; // PWM-Pin für Motorrichtung 2

void setup() {
    // Beginnt die serielle Kommunikation
    Serial.begin(115200);
    // Initialisiert SoftPWM für alle verwendeten Pins
    SoftPWMBegin();
    // Setzt die PWM-Werte initial auf 0
    SoftPWMSet(in1, 0);
    SoftPWMSet(in2, 0);
}

void loop() {
    // Schleife erhöht die Geschwindigkeit von 0 bis 255 anpassen!!!!!!!!

    // Beschleunigung im Uhrzeigersinn
    SoftPWMSet(in2, 0);//Minus
    for (int i = 30; i <= 70; i++) {
        SoftPWMSet(in1, i); // Setzt die PWM für den Motor
        Serial.println("Steuerung der Motorgeschwindigkeit (PWM): " + String(i));
        delay(100); // Kurze Verzögerung zwischen den Geschwindigkeitsänderungen
        if (i == 70) {
            // Wenn i 70 erreicht, stoppt der Motor
            SoftPWMSet(in1, 0);
            Serial.println("Motor stoppt für 1 Sekunde.");
            delay(20); // Wartet 1 Sekunde, bevor der Motor neu startet
            break; // Beendet die Schleife, damit sie von vorne beginnen kann
        }
    }

    // Kurze Pause
    delay(20);

    // verlangsamen im Uhrzeigersinn
    for (int i = 70; i >= 30; i--) {
        SoftPWMSet(in1, i); // Setzt die PWM für den Motor
        Serial.println("Steuerung der Motorgeschwindigkeit (PWM): " + String(i));
        delay(100); // Kurze Verzögerung zwischen den Geschwindigkeitsänderungen
        if (i == 30) {
            // Wenn i 70 erreicht, stoppt der Motor
            SoftPWMSet(in1, 0);
            Serial.println("Motor stoppt für 1 Sekunde.");
            delay(20); // Wartet 1 Sekunde, bevor der Motor neu startet
            break; // Beendet die Schleife, damit sie von vorne beginnen kann
        }
    }

    // Beschleunigung gegen Uhrzeigersinn
    SoftPWMSet(in1, 0);//Minus
```

```
    for (int i = 30; i <= 70; i++) {
        SoftPWMSet(in2, i); // Setzt die PWM für den Motor
        Serial.println("Steuerung der Motorgeschwindigkeit (PWM): " + String(i));
        delay(100); // Kurze Verzögerung zwischen den Geschwindigkeitsänderungen
        if (i == 70) {
            // Wenn i 70 erreicht, stoppt der Motor
            SoftPWMSet(in2, 0);
            Serial.println("Motor stoppt für 1 Sekunde.");
            delay(20); // Wartet 1 Sekunde, bevor der Motor neu startet
            break; // Beendet die Schleife, damit sie von vorne beginnen kann
        }
    }

    // Kurze Pause
    delay(20);

    // verlangsamen gegen Uhrzeigersinn
    for (int i = 70; i >= 30; i--) {
        SoftPWMSet(in2, i); // Setzt die PWM für den Motor
        Serial.println("Steuerung der Motorgeschwindigkeit (PWM): " + String(i));
        delay(100); // Kurze Verzögerung zwischen den Geschwindigkeitsänderungen
        if (i == 30) {
            // Wenn i 70 erreicht, stoppt der Motor
            SoftPWMSet(in2, 0);
            Serial.println("Motor stoppt für 1 Sekunde.");
            delay(20); // Wartet 1 Sekunde, bevor der Motor neu startet
            break; // Beendet die Schleife, damit sie von vorne beginnen kann
        }
    }

}
```

Entfesselung der Beweglichkeit des Mars Rovers

- **Integration von Motoren ins Rocker-Bogie-System**: Das Rocker-Bogie-System ist speziell für die Bewältigung der komplexen und unebenen Marslandschaften konzipiert. Die Einbindung von TT-Motoren in dieses System erweitert dessen Fähigkeit, sich an diverse Geländearten anzupassen, indem es eine verbesserte Mobilität und Stabilität bietet.
- **Programmierung des Mars Rovers**: Die Verwendung der Arduino-Plattform ermöglicht eine präzise Steuerung der Motoren, was die Grundlage für die Bewegungskontrolle des Rovers bildet. Die Programmierung umfasst die Steuerung der Motordrehrichtung und -geschwindigkeit, um Vorwärts-, Rückwärts- und Drehbewegungen zu realisieren.

Implementierungsschritte

1. **Bewegungssteuerung**: Durch Programmierung werden die Motoren so angesteuert, dass der Rover vorwärts und rückwärts fahren sowie nach links und rechts drehen kann. Dies wird durch Anpassung der Drehrichtung und Geschwindigkeit der Motoren erreicht.
2. **Programmbeispiele**: Die Bereitstellung von Codebeispielen illustriert, wie die SoftPWM-Bibliothek genutzt wird, um die Geschwindigkeit und Richtung der Motoren feinabzustimmen. Die Variation der PWM-Werte ermöglicht es, die Geschwindigkeit der Motoren dynamisch anzupassen, was eine differenzierte Steuerung der Bewegung des Rovers ermöglicht.
3. **Erweiterung der Bewegungssteuerung**: Die Entwicklung von Funktionen für spezifische Bewegungsabläufe erleichtert die Programmstrukturierung und erhöht die Wiederverwendbarkeit des Codes. Durch diese

Modularisierung wird der Code nicht nur übersichtlicher, sondern auch flexibler für zukünftige Anpassungen und Erweiterungen.

Motor beschleunigen/verlangsamen im / gegen den Uhrzeigersinn

```
/**
 * @file main.cpp
 * @brief Motor beschleunigen/verlangsamen im / gegen den Uhrzeigersinn
 */
#include <Arduino.h>
#include <SoftPWM.h>

// Definition der Pins für die Linken Motoren A, B, C
const int motorA1 = 2; // INA Plus  für Motoren A, B, C sind parallel
const int motorA2 = 3; // INB Minus für Motoren A, B, C sind parallel

// Definition der Pins für die Rechten Motoren D, E, F
const int motorB1 = 5; // INA Plus  für Motoren D, E, F sind parallel
const int motorB2 = 4; // INB Minus für Motoren D, E, F sind parallel

void setup() {
    Serial.begin(115200);
    SoftPWMBegin();
    SoftPWMSet(motorA1, 0);
    SoftPWMSet(motorA2, 0);
    SoftPWMSet(motorB1, 0);
    SoftPWMSet(motorB2, 0);
}

void controlMotorSpeed(int pin, int startSpeed, int endSpeed, int delayTime, bool
   increase) {
    if (increase) {
        for (int speed = startSpeed; speed <= endSpeed; speed++) {
            SoftPWMSet(pin, speed);
            Serial.println("Motor beschleunigen (PWM): " + String(speed));
            delay(delayTime);
        }
    } else {
        for (int speed = startSpeed; speed >= endSpeed; speed--) {
            SoftPWMSet(pin, speed);
            Serial.println("Motor verlangsamen (PWM): " + String(speed));
            delay(delayTime);
        }
    }
    SoftPWMSet(pin, 0); // Stoppt den Motor nach der Sequenz
    Serial.println("Motor stoppt für kurze Zeit.");
    delay(20); // Kurze Pause nach dem Stopp
}

void loop() {
    // Linke Seite des Rovers
    // Beschleunigung im Uhrzeigersinn
    controlMotorSpeed(motorA1, 30, 70, 100, true);
    // Verlangsamen im Uhrzeigersinn
    controlMotorSpeed(motorA1, 70, 30, 100, true);
    // Beschleunigung gegen Uhrzeigersinn
    controlMotorSpeed(motorA2, 30, 70, 100, true);
    // Verlangsamen gegen Uhrzeigersinn
    controlMotorSpeed(motorA2, 70, 30, 100, false);

    // Rechte Seite des Rovers
    // Beschleunigung im Uhrzeigersinn
```

```
    controlMotorSpeed(motorB1, 30, 70, 100, true);
    // Verlangsamen im Uhrzeigersinn
    controlMotorSpeed(motorB1, 70, 30, 100, true);
    // Beschleunigung gegen Uhrzeigersinn
    controlMotorSpeed(motorB2, 30, 70, 100, true);
    // Verlangsamen gegen Uhrzeigersinn
    controlMotorSpeed(motorB2, 70, 30, 100, false);
}
```

Steuert die Bewegungen eines Rovers

```
/**
 * @file main.cpp
 * @brief Steuert die Bewegungen eines Rovers, einschließlich Vorwärts-, Rückwä
    rtsbewegungen und Drehungen.
 */

#include <Arduino.h>
#include <SoftPWM.h>

// Definition der Pins für die linken Motoren A, B, C
#define LEFT_MOTOR_FORWARD_PIN 2 // Pin für Vorwärtsbewegung der linken Motoren (A,
    B, C)
#define LEFT_MOTOR_REVERSE_PIN 3 // Pin für Rückwärtsbewegung der linken Motoren (A
   , B, C)
// Definition der Pins für die rechten Motoren D, E, F
#define RIGHT_MOTOR_FORWARD_PIN 5 // Pin für Vorwärtsbewegung der rechten Motoren (
   D, E, F)
#define RIGHT_MOTOR_REVERSE_PIN 4 // Pin für Rückwärtsbewegung der rechten Motoren
   (D, E, F)

#define FORWARD_SPEED 255 // Maximalgeschwindigkeit
#define MIN_SPEED 30 // Minimalgeschwindigkeit, um MotorBrummen zu vermeiden
#define STOP 0 // Stopp-Signal

// Funktionsprototyp
void stopMotors();

/**
 * @brief Initialisiert die Motorsteuerung und die serielle Kommunikation.
 */
void setup() {
    Serial.begin(115200);
    SoftPWMBegin();
    // Initialisiert alle Motoren auf 0 (gestoppt)
    SoftPWMSet(LEFT_MOTOR_FORWARD_PIN, 0);
    SoftPWMSet(LEFT_MOTOR_REVERSE_PIN, 0);
    SoftPWMSet(RIGHT_MOTOR_FORWARD_PIN, 0);
    SoftPWMSet(RIGHT_MOTOR_REVERSE_PIN, 0);
}

/**
 * @brief Setzt die Geschwindigkeit der Motoren.
 *
 * @param speedLeftForward Geschwindigkeit für die linke Seite vorwärts.
 * @param speedLeftBackward Geschwindigkeit für die linke Seite rückwärts.
 * @param speedRightForward Geschwindigkeit für die rechte Seite vorwärts.
 * @param speedRightBackward Geschwindigkeit für die rechte Seite rückwärts.
 */
void setMotorSpeeds(int speedLeftForward, int speedLeftBackward, int
   speedRightForward, int speedRightBackward) {
    SoftPWMSet(LEFT_MOTOR_FORWARD_PIN, max(speedLeftForward, STOP));
    SoftPWMSet(LEFT_MOTOR_REVERSE_PIN, max(speedLeftBackward, STOP));
    SoftPWMSet(RIGHT_MOTOR_FORWARD_PIN, max(speedRightForward, STOP));
    SoftPWMSet(RIGHT_MOTOR_REVERSE_PIN, max(speedRightBackward, STOP));
}

/**
```

```
 * @brief Bewegt den Rover vorwärts.
 *
 * @param speed Optional. Die Geschwindigkeit für die Vorwärtsbewegung. Standard
    ist die Hälfte der maximalen Geschwindigkeit.
 */
void moveForward(int speed = FORWARD_SPEED / 2) {
    setMotorSpeeds(speed, STOP, speed, STOP);
    Serial.print("Rover bewegt sich vorwärts mit Geschwindigkeit: ");
    Serial.println(speed);
}
/**
 * @brief Bewegt den Rover rückwärts.
 *
 * @param speed Optional. Die Geschwindigkeit für die Rückwärtsbewegung. Standard
    ist ein Viertel der maximalen Geschwindigkeit.
 */
void moveBackward(int speed = FORWARD_SPEED / 4) {
    setMotorSpeeds(STOP, speed, STOP, speed);
    Serial.print("Rover bewegt sich rückwärts mit Geschwindigkeit: ");
    Serial.println(speed);
}
/**
 * @brief Dreht den Rover nach rechts.
 *
 * @param speed Optional. Die Geschwindigkeit für die Drehung. Standard ist ein
    Viertel der maximalen Geschwindigkeit.
 */
void turnRight(int speed = FORWARD_SPEED / 4) {
    setMotorSpeeds(speed, STOP, STOP, speed);
    Serial.print("Rover dreht nach rechts mit Geschwindigkeit: ");
    Serial.println(speed);
}
/**
 * @brief Dreht den Rover nach links.
 *
 * @param speed Optional. Die Geschwindigkeit für die Drehung. Standard ist die
    maximale Geschwindigkeit.
 */
void turnLeft(int speed = FORWARD_SPEED / 4) {
    setMotorSpeeds(STOP, speed, speed, STOP);
    Serial.print("Rover dreht nach links mit Geschwindigkeit: ");
    Serial.println(speed);
}
/**
 * @brief Stoppt alle Motoren des Rovers.
 */
void stopMotors() {
    setMotorSpeeds(STOP, STOP, STOP, STOP);
    Serial.println("Rover stoppt.");
}

/**
 * @brief Hauptprogrammschleife, steuert die Bewegungen des Rovers.
 */
void loop() {
    moveForward(); // Bewegt sich vorwärts mit halber Geschwindigkeit
    delay(800);
    stopMotors();
    delay(2000);
```

```
    moveBackward(); // Bewegt sich rückwärts mit einem Viertel der Geschwindigkeit
    delay(800);
    stopMotors();
    delay(2000);
    turnRight(); // Dreht nach rechts mit einem Viertel der Geschwindigkeit
    delay(3000);
    stopMotors();
    delay(2000);
    turnLeft(); // Dreht nach links mit voller Geschwindigkeit
    delay(3000);
    stopMotors();
    delay(2000);
}
```