Zusammenfassung

Erkundung des Hindernisvermeidungsmoduls

Pin-belegung Infrarot-Hindernisvermeidungsmodul

- **GND (Ground):** Dient als gemeinsamer Bezugspunkt für die Stromversorgung und das Signal des Moduls, essentiell für die Stabilität und Funktionalität des Schaltkreises.
- **+ (VCC):** Die Stromversorgungsleitung, die typischerweise zwischen 3,3V und 5V DC liegt. Die Flexibilität in der Stromversorgung macht das Modul kompatibel mit einer Vielzahl von Mikrocontrollern und Entwicklungsboards.
- **Out (Output):** Dieser Pin liefert das Signal, das den Zustand der Hinderniserkennung anzeigt. Ein hoher Pegel signalisiert keine Hindernisse, während ein niedriger Pegel die Erkennung eines Objekts anzeigt. Diese binäre Signalübertragung ermöglicht eine direkte Integration in Logikschaltungen und Mikrocontroller-Inputs.
- **EN (Enable):** Ermöglicht die Aktivierung oder Deaktivierung des Moduls. Eine direkte Verbindung mit GND bedeutet eine ständige Aktivierung des Moduls. Die Möglichkeit, diesen Pin zu steuern, eröffnet erweiterte Anwendungsgebiete, in denen die Sensortätigkeit programmatisch umgeschaltet werden soll.

Arbeitsprinzip des Infrarot-Hindernisvermeidungsmoduls

Das Herzstück des Moduls bilden ein Infrarot-Sender und -Empfänger, die zusammenarbeiten, um die Präsenz und Distanz von Objekten zu detektieren. Der Sender emittiert kontinuierlich Infrarotstrahlung, die vom Empfänger detektiert wird, wenn sie von einem Hindernis reflektiert wird. Diese Art der Reflexion ermöglicht eine präzise Hinderniserkennung.

Störunterdrückung und Einfluss der Objektfarbe

Die Effektivität der Hinderniserkennung kann durch die Oberflächenbeschaffenheit und Farbe des Objekts beeinflusst werden. Dunkle Farben absorbieren Infrarotlicht stärker, wodurch die Reflexion minimiert und die Erkennungsdistanz reduziert wird. Helle Oberflächen, insbesondere Weiß, reflektieren Infrarotstrahlung effizienter, was die maximale Erkennungsdistanz des Moduls erhöht.

Auslesen der 2 Module

1. **Initialisierung:**
   - Die Pins für die IR-Module werden als Eingänge konfiguriert, um die Signale der Infrarot-Sensoren lesen zu können.
   - Die serielle Kommunikation ermöglicht die Übertragung der Sensorwerte an den Computer zur Anzeige im seriellen Monitor.
2. **Hauptprogrammschleife (loop):**
   - Die digitalen Werte der beiden IR-Module werden gelesen. Diese Werte geben an, ob ein Hindernis erkannt wurde (`0`) oder nicht (`1`).

- Die gelesenen Werte werden über den seriellen Port ausgegeben. Für jedes Modul wird angegeben, ob ein Hindernis vorliegt oder nicht.
- Das Programm wartet 200 Millisekunden (`delay(200)`), um die Lesbarkeit der ausgegebenen Werte zu verbessern und eine kontinuierliche Flut von Daten zu vermeiden.

Beschreibung des Programmflusses

Programmfluss - Visualisieren in einem Flussdiagramm

- **Setup-Phase:**
  - **PIN-Modus einstellen:** Zuerst werden die Pins, an denen die IR-Sensoren angeschlossen sind, als Eingänge (`INPUT`) definiert. Dies ist notwendig, damit der Arduino die von den Sensoren kommenden Signale lesen kann.
  - **Serielle Kommunikation starten:** Anschließend wird die serielle Kommunikation initialisiert, was für die Datenübertragung zum Computer erforderlich ist. Die Baudrate von 115200 wird gewählt, da sie eine gängige Rate für viele Arduino-Projekte darstellt und eine ausreichende Übertragungsgeschwindigkeit bietet.
- **Loop-Phase:**
  - **Sensorwerte lesen:** In jedem Durchlauf der Hauptschleife (`loop`) werden die digitalen Signale der IR-Sensoren gelesen. Diese Signale zeigen an, ob die Sensoren ein Hindernis vor sich wahrnehmen.
  - **Werte ausgeben:** Die gelesenen Werte werden dann zusammen mit einer beschreibenden Nachricht über den seriellen Port ausgegeben. Diese Ausgabe ermöglicht es dem Benutzer, die Sensorwerte in Echtzeit zu überwachen.
  - **Verzögerung einfügen:** Zwischen den Ausgaben wird eine kurze Verzögerung eingefügt, um die Ausgabe im seriellen Monitor lesbarer zu machen und dem Benutzer Zeit zu geben, die Informationen zu verarbeiten.

```
// Definiere die Pins für die IR-Module
#define IR_RIGHT 7
#define IR_LEFT 8

void setup() {
  // Setze die Pins der IR-Module als Eingänge
  pinMode(IR_RIGHT, INPUT);
  pinMode(IR_LEFT, INPUT);

  // Beginne die serielle Kommunikation
  Serial.begin(115200);
}

void loop() {
  // Lese die Werte von den IR-Modulen
  int rightValue = digitalRead(IR_RIGHT);
  int leftValue = digitalRead(IR_LEFT);

  // Ausgabe auf den seriellen Monitor
  Serial.print("Rechter IR: ");
  Serial.println(rightValue);
  Serial.print("Linker IR: ");
  Serial.println(leftValue);

  // Warte kurz, um die Lesbarkeit zu verbessern
  delay(200);
}
```

Anpassen der Erkennungsdistanz

- **Bedeutung der Anpassung:**
  - Die voreingestellten Erkennungsdistanzen der Infrarotmodule sind möglicherweise nicht ideal für alle Umgebungen.
  - Zu kurze Distanzen könnten zu Kollisionen mit Hindernissen führen.
  - Zu weite Distanzen könnten unnötige Ausweichmanöver verursachen, selbst wenn Hindernisse noch weit entfernt sind.
- **Anpassungsprozess:**
  1. **Rechtes Hindernisvermeidungsmodul justieren:**
     - Überprüfe das Modul auf eventuelle Fehlausrichtungen durch Transportstöße und richte es gegebenenfalls gerade.
     - Platziere ein Hindernis (z.B. die Verpackungsbox des Rover-Kits) etwa 20 cm vor dem Modul und 30°.
     - Drehe das Potentiometer am Modul, bis die Anzeigeleuchte aktiv wird.
     - Teste die Einstellung, indem du das Hindernis bewegst, um die Genauigkeit der Erkennung zu überprüfen. Stelle bei Bedarf das andere Potentiometer ein.
  2. **Linkes Hindernisvermeidungsmodul justieren:**
     - Wiederhole den gleichen Prozess wie beim rechten Modul, um eine konsistente Erkennungsdistanz zu gewährleisten.

Entwurf eines automatischen Hindernisvermeidungssystems

Programmfluss steuert einen Rover mithilfe von Infrarot-Hindernisvermeidungsmodulen und SoftPWM zur Geschwindigkeitsregelung. Der Rover kann vorwärts, rückwärts, nach rechts und nach links navigieren, abhängig von den Signalen der IR-Sensoren. Hier ist eine Zusammenfassung des Programmflusses:

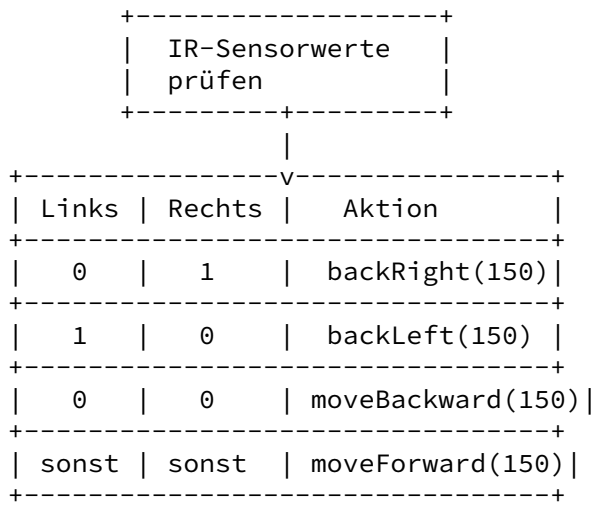
1. **Initialisierung:**

- Die `SoftPWM` Bibliothek wird initialisiert, um eine präzise Geschwindigkeitskontrolle der Motoren zu ermöglichen.
- Die Pins für die IR-Sensoren werden als Eingänge konfiguriert, um die Hinderniserkennung zu ermöglichen.

2. **Hauptprogrammschleife (loop):**
   - Die Werte von beiden IR-Sensoren (rechts und links) werden gelesen.
   - Basierend auf den gelesenen Werten der IR-Sensoren entscheidet das Programm, in welche Richtung der Rover bewegt werden soll:
     - Wenn der Weg rechts blockiert ist (`rightValue == 0` und `leftValue == 1`), dreht der Rover nach hinten rechts.
     - Wenn der Weg links blockiert ist (`rightValue == 1` und `leftValue == 0`), dreht der Rover nach hinten links.
     - Wenn beide Wege blockiert sind (`rightValue == 0` und `leftValue == 0`), bewegt sich der Rover rückwärts.
     - Wenn beide Wege frei sind, bewegt sich der Rover vorwärts.
3. **Bewegungsfunktionen:**
   - `moveBackward(int speed)`: Setzt die Motoren so, dass der Rover rückwärts fährt.
   - `moveForward(int speed)`: Setzt die Motoren so, dass der Rover vorwärts fährt.
   - `backRight(int speed)`: Dreht den Rover nach hinten rechts.
   - `backLeft(int speed)`: Dreht den Rover nach hinten links.

Jede Bewegungsfunktion nimmt eine Geschwindigkeit (`speed`) als Parameter, die mittels `SoftPWMSet()` auf die entsprechenden Motorpins angewendet wird, um die Motoren in die gewünschte Richtung zu drehen.

Visualisieren in einem Flussdiagramm bei einem Wert von 0 wird ein Hindernis angenommen, und der Rover reagiert entsprechend, um Kollisionen zu vermeiden.



```
/**
 * @file main.cpp
 * @brief Entwurf eines automatischen Hindernisvermeidungssystems mit 2x
    Infarotsensoren
 */
#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)

// Definition der Pins für die IR-Module
#define IR_RIGHT 7
#define IR_LEFT 8

// Zustandskonstanten für die IR-Sensoren
#define OBSTACLE 0
#define CLEAR 1

void moveBackward(int speed);
void moveForward(int speed);
void backRight(int speed);
void backLeft(int speed);

void setup() {
  // Initialisiere SoftPWM
  SoftPWMBegin();

  // Setze die Pins der IR-Module als Eingänge
  pinMode(IR_RIGHT, INPUT);
  pinMode(IR_LEFT, INPUT);

  // Konfiguriere Motorpins als Ausgänge
  pinMode(LEFT_MOTOR_FORWARD_PIN, OUTPUT);
  pinMode(LEFT_MOTOR_REVERSE_PIN, OUTPUT);
  pinMode(RIGHT_MOTOR_FORWARD_PIN, OUTPUT);
  pinMode(RIGHT_MOTOR_REVERSE_PIN, OUTPUT);
}

void loop() {
  // Lese Werte von den IR-Sensoren
  int rightValue = digitalRead(IR_RIGHT);
  int leftValue = digitalRead(IR_LEFT);

  // Steuere die Bewegungen des Rovers basierend auf den Werten der IR-Sensoren
  if (rightValue == OBSTACLE && leftValue == CLEAR) {  // Weg rechts blockiert
    backRight(70);
  } else if (rightValue == CLEAR && leftValue == OBSTACLE) {  // Weg links
     blockiert
    backLeft(70);
```

```
  } else if (rightValue == OBSTACLE && leftValue == OBSTACLE) {  // Beide Wege
     blockiert
    moveBackward(70);
  } else {  // Wege frei
    moveForward(70);
  }

  delay(100); // Einfache Entprellungsverzögerung
}

void moveBackward(int speed) {
  SoftPWMSet(LEFT_MOTOR_REVERSE_PIN, speed);
  SoftPWMSet(RIGHT_MOTOR_REVERSE_PIN, speed);
  SoftPWMSet(LEFT_MOTOR_FORWARD_PIN, 0);
  SoftPWMSet(RIGHT_MOTOR_FORWARD_PIN, 0);
}

void moveForward(int speed) {
  SoftPWMSet(LEFT_MOTOR_FORWARD_PIN, speed);
  SoftPWMSet(RIGHT_MOTOR_FORWARD_PIN, speed);
  SoftPWMSet(LEFT_MOTOR_REVERSE_PIN, 0);
  SoftPWMSet(RIGHT_MOTOR_REVERSE_PIN, 0);
}

void backRight(int speed) {
  SoftPWMSet(LEFT_MOTOR_REVERSE_PIN, speed);
  SoftPWMSet(RIGHT_MOTOR_FORWARD_PIN, 0);
  SoftPWMSet(LEFT_MOTOR_FORWARD_PIN, 0);
  SoftPWMSet(RIGHT_MOTOR_REVERSE_PIN, 0);
}

void backLeft(int speed) {
  SoftPWMSet(RIGHT_MOTOR_REVERSE_PIN, speed);
  SoftPWMSet(LEFT_MOTOR_FORWARD_PIN, 0);
  SoftPWMSet(RIGHT_MOTOR_FORWARD_PIN, 0);
  SoftPWMSet(LEFT_MOTOR_REVERSE_PIN, 0);
}
```

Reflexion Infarotsensor

- Beobachten Sie, ob sich der Rover so bewegt, wie Sie es erwartet haben.
- Oder setzen Sie ihn verschiedenen Lichtverhältnissen aus, um zu sehen, wie sich seine Bewegungen ändern.
- Während er geschickt Hindernissen zu seiner Linken und Rechten ausweicht, könnte er Schwierigkeiten haben, kleinere Hindernisse direkt vor ihm zu erkennen. Wie können wir diese Herausforderung meistern?