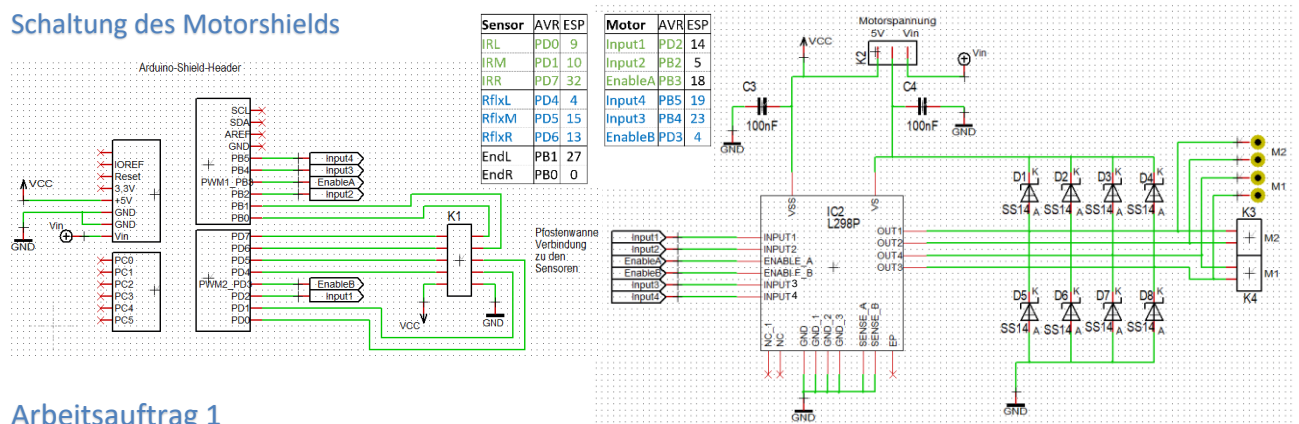
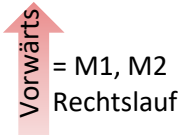
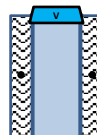
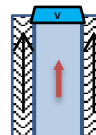
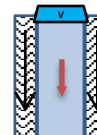

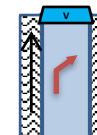
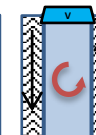

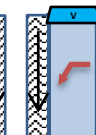



Schaltung des Motorshields



Arbeitsauftrag 1

- Ergänzen Sie die Ansteuertabelle für die Roboterbewegungen.

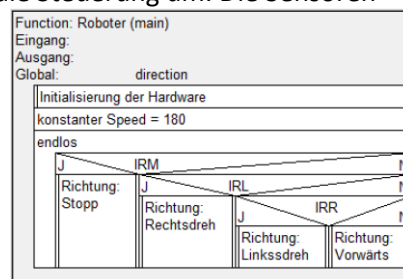
Bewegungsrichtung Roboter		Stopp	Vorwärts	Rückwärts	Links-Vor	Rechts-Vor	Links-Dreh	Rechts-Dreh	Links-Rück	Rechts-Rück
 Vorwärts = M1, M2 Rechtslauf										
Motor M2 Links (Brücke A)	Input1	0	0							
	Input2	0	1							
Motor M1 Rechts (Brücke B)	Input3	0	0							
	Input4	0	1							

- Tragen Sie die Tabellenwerte in das Array **directions[][]** ein und ergänzen Sie die Definition des **enum Richtung**.
- Ergänzen Sie die Funktion **robby_richtung()**.

```
void robby_richtung(uint8_t dir)
{
  bit_write(_PORTD_,2, directions[dir][0]);
  ...
}
```

Die Geschwindigkeit (PWM) soll auf einen konstanten Wert gesetzt werden (duty_cycle = 180);

- Erstellen Sie nun ein Steuerprogramm für einen einfachen Ausweichroboter. Verwenden Sie die nebenstehenden Definitionen für die IR-Sensorsignale. Setzen Sie das abgebildete Struktogramm für die Steuerung um. Die Sensoren sind 0-aktiv !



```
#define SensorD _PORTD_
#define IRL      0      // PD0
#define IRM      1      // PD1
#define IRR      7      // PD7

const uint8_t directions[9][4] = {
  {0,0,0,0}, // stopp
  {0,1,0,1}, // vor
  ...
};

enum Richtungen {STOPP,VORWAERTS, ... };

void setup (void) // Initialisierungen
{
  // IR-Sensorsignale
  bit_init(SensorD,IRL,IN);
  bit_init(SensorD,IRM,IN);
  bit_init(SensorD,IRR,IN);
  // Motorsignale
  bit_init(_PORTD_,2,OUT); // Input 1
  bit_init(_PORTB_,2,OUT); // Input 2
  bit_init(_PORTB_,4,OUT); // Input 3
  bit_init(_PORTB_,5,OUT); // Input 4
  // PWM für Motordrehzahl
  pwm_init(); // Enable A (pwm)
  pwm2_init(); // Enable B
}
```

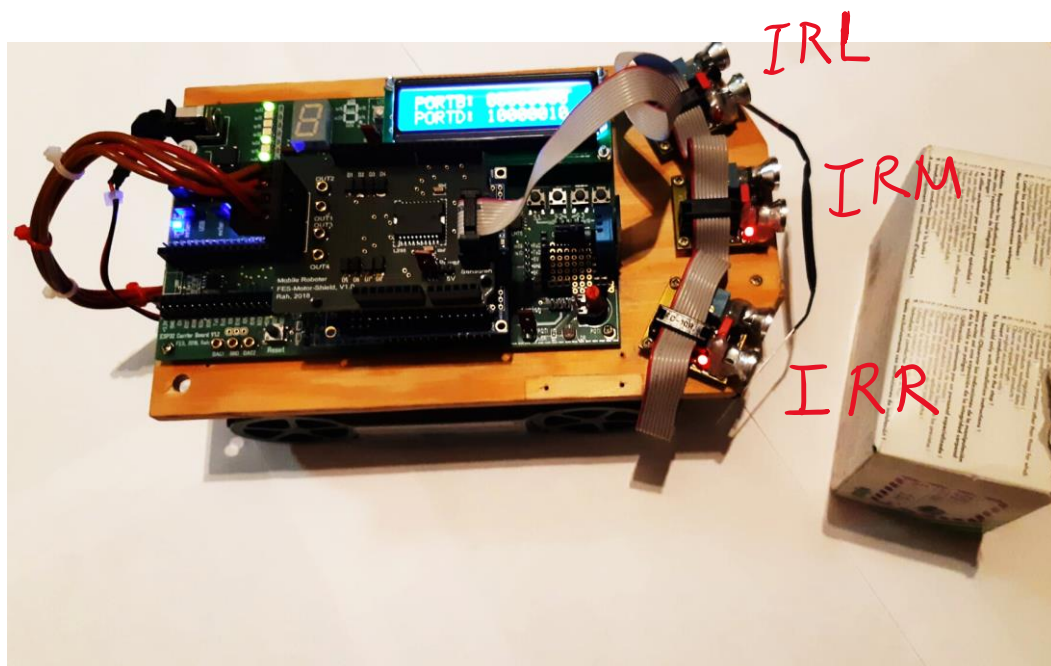
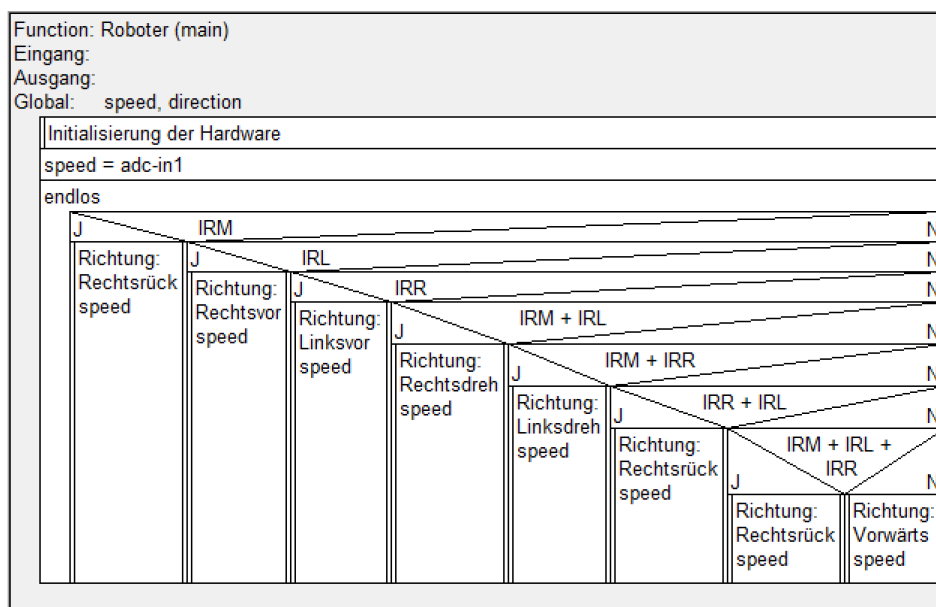
Im folgenden soll der Roboter „intelligenter“ werden.

Arbeitsauftrag 2

- Die Geschwindigkeit des Roboters soll über den AD-Wandler (Poti) eingestellt werden. Ändern Sie die Definition von `robby_richtung` entsprechende ab. Und fügen Sie die Einstellung der Geschwindigkeit in die Funktion `robby_richtung` ein.

```
void robby_richtung(uint8_t dir, uint8_t speed)
```

- Für ein flüssigeres Ausweichen sollen zusätzliche Sensorkombinationen von IRL, IRL und IRR ausgewertet werden. Setzen Sie den abgebildeten Algorithmus um. Überlegen Sie, wie sie die Abfrage von 2, bzw. von 3 Sensoren in einer if-Bedingung realisieren.



Arbeitsauftrag 3

- Alternativ können die Sensoren auch sehr elegant über eine Fallabfrage (switch ... case ...) ausgewertet werden. Dazu muss aber der Sensorwert als Bytewert eingelesen werden:

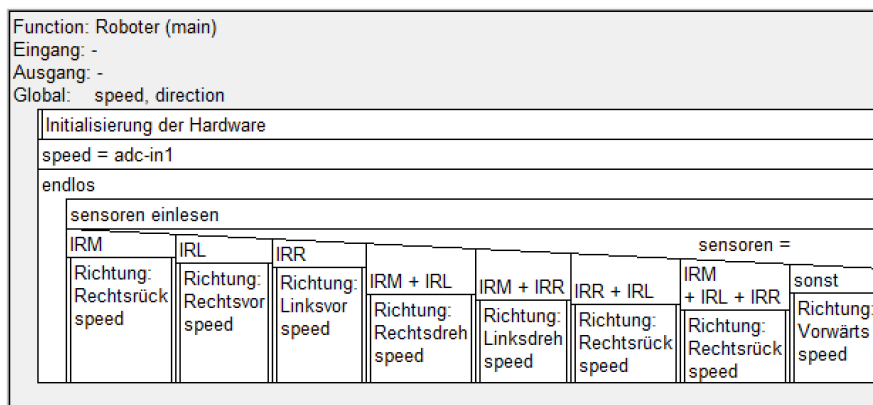
```
sensor = ~byte_read(_PORTD_);    // Sensoren sind 0-aktiv
```

Als switch-Vergleichswert wird die Sensorvariable so maskiert, dass nur noch die relevanten Bits (x) stehen bleiben. Die restlichen Bits werden zu 0 geforced. Im case-Wert darf nur eine Konstante stehen. Diese wird wie unten abgebildet vom Compiler erzeugt. Machen Sie sich die Konstruktion klar.

```
switch(sensor & ((1<<IRM) | (1<<IRL) | (1<<IRR))) // sensor => x0000xx
{
  case (1<<IRM):                                robby_richtung(RECHTSRUECK, speed); break;
  ...;
  ...;
  case ((1<<IRM)|(1<<IRL)):                        ...;                // Vergleichswert mit 2 Sensoren
  ...;
  ...;
  case ((1<<IRM)|(1<<IRR)|(1<<IRL)): ...;                // Vergleichswert mit 3 Sensoren
  default:                                       ...;
}
```

Stimmen switch-Vergleichswert und ein case-Wert überein, wird der case-Zweig ausgeführt.

- Ändern Sie nun das Programm so ab, dass der unten stehende Algorithmus umgesetzt wird.



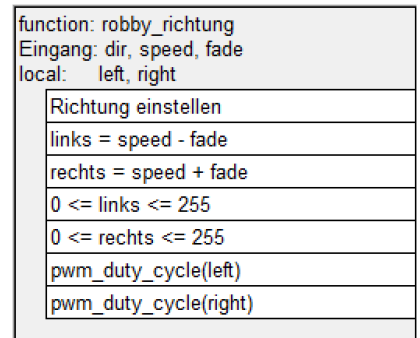
- Die Raupenantriebe des Roboters erlauben auch, dass sich die Motoren mit unterschiedlichem Speed bewegen. Dadurch können auch beliebige Kurvenradien gefahren werden. Ergänzen Sie die Funktion robby_richtung mit einem dritten Parameter:


```
void robby_richtung(uint8_t dir, uint8_t speed, int8_t fade)
```

Achten Sie darauf, dass die Variablen links und speed im nebenstehenden Algorithmus auch negativ werden können. Ebenso der Parameter „fade“.

Erzeugen Sie die Variablen „speed“ und „fade“ in der Funktion main() folgendermaßen:

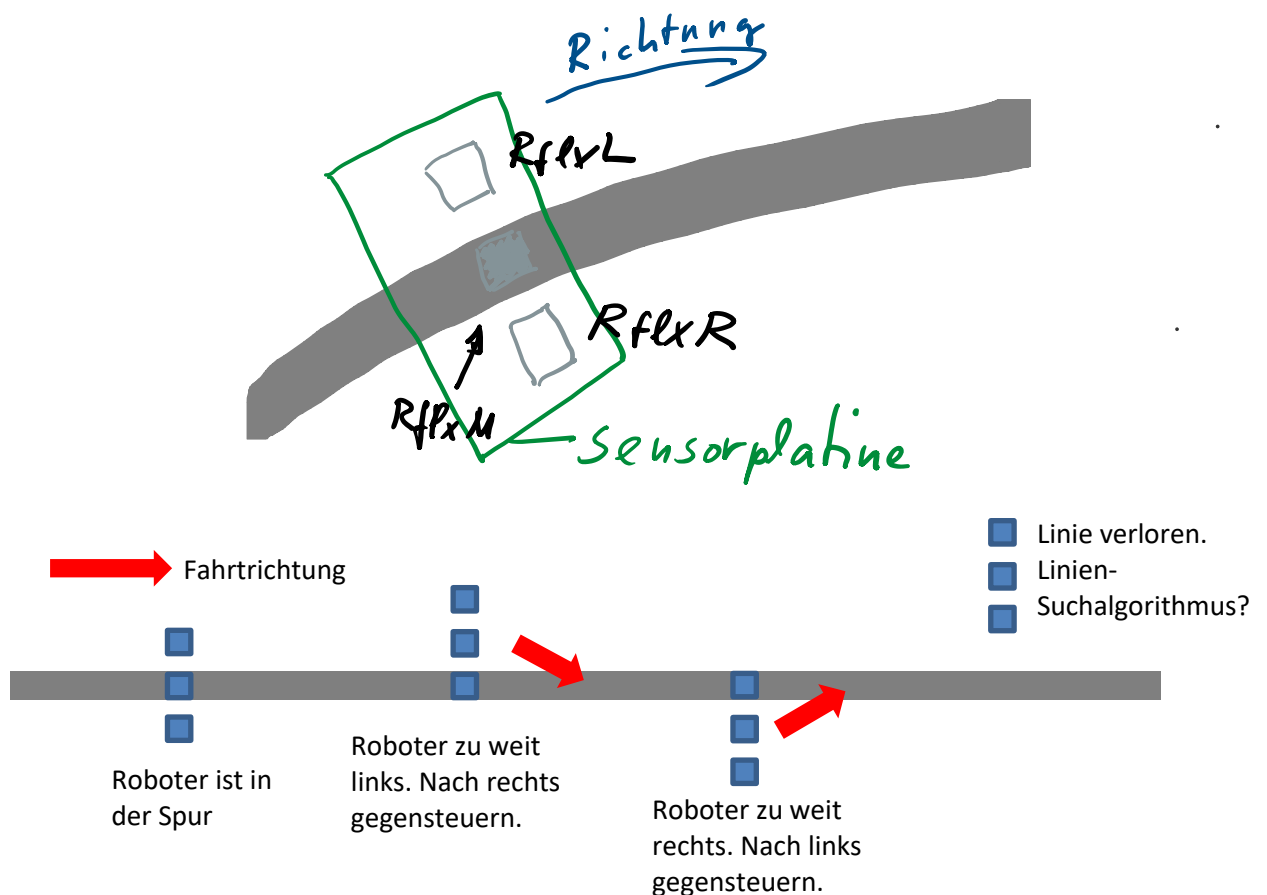
```
fade = 127 - adc_in1();    // fade = -128...0...+127
speed = 140;               // Festwert für speed
```




 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller	Name: Rahm Datum: 31.12.2019 6_5_Einfache_Robotersteuerung.docx
	Robotersteuerung	6.5.4

Weitere Aufgaben:

1. Das Ausweichverhalten kann noch verbessert werden, wenn der Roboter manchmal eine kurze Strecke sensorlos zurückfährt und anschließend eine kurze Richtungskorrektur vornimmt. Dann fährt er automatisch wieder mit einem anderen Winkel zum Hindernis noch vorne. Probieren Sie aus, wo dieses Verhalten wünschenswert ist.
2. Werten Sie auch die beiden Endtaster (EndR = PortB.0, EndL = PortB.1) aus. Bringen Sie den Roboter bei Kollision sofort zum stehen.
3. Schreiben Sie eine Robotersteuerung, die den Roboter immer auf einer schwarzen Linie entlang fahren lässt (Line Follower). Verwenden Sie dazu die drei Reflex-Lichttaster (RflxL, RflxM, RflxR) an der Unterseite des Roboters.



 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller	Name: Rahm Datum: 31.12.2019 6_5_Einfache_Robotersteuerung.docx
	Robotersteuerung	6.5.5

Lösung zu Arbeitsauftrag 1:

```
#include "controller.h"

#define SensorD _PORTD_
#define IRL      0      // PD0
#define IRM      1      // PD1
#define IRR      7      // PD7

const uint8_t directions[9][4] = {
    {0,0,0,0},    // stopp
    {0,1,0,1},    // vor
    {1,0,1,0},    // rueck
    {0,0,0,1},    // linksvor
    {0,1,0,0},    // rechtsvor
    {1,0,0,1},    // linksdreh
    {0,1,1,0},    // rechtsdreh
    {1,0,0,0},    // linksrueck
    {0,0,1,0},    // rechtsrueck
};

enum Richtungen {STOPP,VORWAERTS,RUECKWAERTS,LINKSVOR,RECHTSVOR,
    LINKSDREH,RECHTSDREH,LINKSRUECK,RECHTSRUECK};

//Funktionsprototypen
void robby_richtung(uint8_t dir);

void setup (void)    // Initialisierungen
{
    // IR-Sensorsignale
    bit_init(SensorD,IRL,IN);
    bit_init(SensorD,IRM,IN);
    bit_init(SensorD,IRR,IN);
    // Motorsignale
    bit_init(_PORTD_,2,OUT);    // Input 1
    bit_init(_PORTB_,2,OUT);    // Input 2
    bit_init(_PORTB_,4,OUT);    // Input 3
    bit_init(_PORTB_,5,OUT);    // Input 4
    // PWM für Motordrehzahl
    pwm_init();                // Enable A (pwm)
    pwm2_init();               // Enable B (pwm)
}


int main (void)
{
    uint8_t speed;

    setup();

    pwm_start();
    pwm2_start();
    pwm_duty_cycle(180);
    pwm2_duty_cycle(180);

    while(1)
    {
        if (bit_read(SensorD,IRM)==0) robby_richtung(STOPP, speed);
        else if (bit_read(SensorD,IRL)==0) robby_richtung(RECHTSDREH, speed);
        else if (bit_read(SensorD,IRR)==0) robby_richtung(LINKSDREH, speed);
        else robby_richtung(VORWAERTS, speed);
    }
}

// Bewegungsrichtung des Roboters festlegen
// dir:  0 ... 9      (Richtungen: STOPP,VORWAERTS,RUECKWAERTS,
//                      LINKSVOR,RECHTSVOR,LINKSDREH,
//                      RECHTSDREH,LINKSRUECK,RECHTSRUECK)
void robby_richtung(uint8_t dir)
{
    //Bewegungsrichtung
    bit_write(_PORTD_,2,directions[dir][0]);    // Input 1
    bit_write(_PORTB_,2,directions[dir][1]);    // Input 2
    bit_write(_PORTB_,4,directions[dir][2]);    // Input 3
    bit_write(_PORTB_,5,directions[dir][3]);    // Input 4
}
```

 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller	Name: Rahm Datum: 31.12.2019 6_5_Einfache_Robotersteuerung.docx
	Robotersteuerung	6.5.6

Lösung zu Arbeitsauftrag 2:

```
#include "controller.h"

#define SensorD _PORTD_
#define IRL 0 // PD0
#define IRM 1 // PD1
#define IRR 7 // PD7

const uint8_t directions[9][4] = {...}; (siehe A1)

enum Richtungen {STOPP, VORWAERTS, RUECKWAERTS, LINKSVOR,...}; (siehe A1)

//Funktionsprototypen
void robby_richtung(uint8_t dir, uint8_t speed);

void setup (void) // Initialisierungen
{
  ...(siehe A1)

  adc_init();
}

int main (void)
{
  uint8_t speed;

  setup();


  pwm_start();
  pwm2_start();

  while(1)
  {
    speed = adc_in1();

    if (!bit_read(SensorD,IRM))
      robby_richtung(RECHTSRUECK, speed);
    else if (!bit_read(SensorD,IRL))
      robby_richtung(RECHTSVOR, speed);
    else if (!bit_read(SensorD,IRR))
      robby_richtung(LINKSVOR, speed);
    else if (!bit_read(SensorD,IRM) && !bit_read(SensorD,IRL))
      robby_richtung(RECHTSDREH, speed);
    else if (!bit_read(SensorD,IRM) && !bit_read(SensorD,IRR))
      robby_richtung(LINKSDREH, speed);
    else if (!bit_read(SensorD,IRR) && !bit_read(SensorD,IRL))
      robby_richtung(RECHTSRUECK, speed);
    else if (!bit_read(SensorD,IRM) && !bit_read(SensorD,IRL) && !bit_read(SensorD,IRR))
      robby_richtung(RECHTSRUECK, speed);
    else
      robby_richtung(VORWAERTS, speed);
  }
}

// Bewegungsrichtung des Roboters festlegen
// dir: 0 ... 9 (Richtungen: STOPP, VORWAERTS, RUECKWAERTS,
// LINKSVOR, RECHTSVOR, LINKSDREH,
// RECHTSDREH, LINKSRUECK, RECHTSRUECK)
// speed: 0 ... 255 (Geschwindigkeit)
void robby_richtung(uint8_t dir, uint8_t speed)
{
  //Bewegungsrichtung
  bit_write(_PORTD_,2,directions[dir][0]); // Input 1
  bit_write(_PORTB_,2,directions[dir][1]); // Input 2
  bit_write(_PORTB_,4,directions[dir][2]); // Input 3
  bit_write(_PORTB_,5,directions[dir][3]); // Input 4

  pwm_duty_cycle(speed);
  pwm2_duty_cycle(speed);
}
```

 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller	Name: Rahm Datum: 31.12.2019 6_5_Einfache_Robotersteuerung.docx
	Robotersteuerung	6.5.7

Lösung zu Arbeitsauftrag 3:

```
#include "controller.h"

#define SensorD _PORTD_
#define IRL      0      // IR-Sensoren
#define IRM      1
#define IRR      7

const uint8_t directions[9][4] = {...}; (siehe A1)

enum Richtungen {STOPP,VORWAERTS,RUECKWAERTS,LINKSVOR,...}; (siehe A1)

//Funktionsprototypen
void robby_richtung(uint8_t dir, uint8_t speed, int8_t fade);

void setup (void)    // Initialisierungen
{
    ... (siehe A1,A2)
}

int main (void)
{
    volatile uint8_t speed, sensor;
    volatile int8_t fade;

    setup();

    pwm_start();
    pwm2_start();

    while(1)
    {
        fade = 127 - adc_in1();
        speed = 140;

        sensor = ~byte_read(_PORTD_); // Sensoren sind 0-aktiv

        switch(sensor & ((1<<IRL) | (1<<IRM) | (1<<IRR)))
        {
            case (1<<IRM):                robby_richtung(RECHTSRUECK, speed, fade); break;
            case (1<<IRL):                robby_richtung(RECHTSVOR, speed, fade); break;
            case (1<<IRR):                robby_richtung(LINKSVOR, speed, fade); break;
            case ((1<<IRM)|(1<<IRL)):      robby_richtung(RECHTSDREH, speed, fade); break;
            case ((1<<IRM)|(1<<IRR)):      robby_richtung(LINKSDREH, speed, fade); break;
            case ((1<<IRM)|(1<<IRL)):      robby_richtung(RECHTSRUECK, speed, fade); break;
            case ((1<<IRM)|(1<<IRR)|(1<<IRL)): robby_richtung(RECHTSRUECK, speed, fade); break;
            default:                    robby_richtung(VORWAERTS, speed, fade); break;
        }
    }
}

void robby_richtung(uint8_t dir, uint8_t speed, int8_t fade)
{
    int16_t left, right;

    //Bewegungsrichtung
    bit_write(_PORTD_,2,directions[dir][0]); // Input 1
    bit_write(_PORTB_,2,directions[dir][1]); // Input 2
    bit_write(_PORTB_,4,directions[dir][2]); // Input 3
    bit_write(_PORTB_,5,directions[dir][3]); // Input 4

    //Differenzial
    left = (int16_t)speed - fade;
    right = (int16_t)speed + fade;

    //Bereichsbegrenzung für PWM
    if (left > 255) left = 255;
    else if (left < 0) left = 0;
    if (right > 255) right = 255;
    else if (right < 0) right = 0;

    pwm_duty_cycle(left);
    pwm2_duty_cycle(right);
}
```