## Beilage: 2

## Roboter_Software_links_langsamer

```
/*****************************************************************************/
/***                                                                       ***/
/*** Version: 1.1                                                          ***/
/*** Datum: 08.008.2014                                                    ***/
/***                                                                       ***/
/*** Dies ist der Programmcode für das Starter Kit von:                    ***/
/*** www.BotMakers.de                                                      ***/
/***                                                                       ***/
/*** Kleinere Korrekturen und Anpassungen von Markus Knapp.                ***/
/*** http://robotiklabor.de                                                ***/
/***                                                                       ***/
/*** Weiterentwicklung erwünscht! Aber bitte erzählt uns davon... ;-)      ***/
/***                                                                       ***/
/*** Florian > Profil: http://botmakers.de/mitglieder/florian/             ***/
/*****************************************************************************/

// Nützliche Funktionen für die Servo Ansteuerung kommen von hier:
#include <Servo.h>

// Welcher Pin hat welche Funktion???
//     Funktion  Pin  Beschreibung
#define IR_SENSOR 2    // Der Sharp IR Sesor ist an Pin 2
#define DIR_A     12   // Richtung A
#define DIR_B     13   // Richtung B
#define PWM_A     3    // Geschwindigkeit A
```

```
#define PWM_B     11   // Geschwindigkeit B

#define BRAKE_A   9    // Bremse A

#define BRAKE_B   8    // Bremse B

#define SERVO     5    // Servo wird an Pin 5 angesteuert



// Nützliche Einstellungen:

#define FULL_SPEED_A      180     // Vollgas ist 255. 180 für Motor A, da dieser langsamer läuft!! (Fahrtrichtung rechts)

#define FULL_SPEED_B      100     // Vollgas ist 255. 100 für Motor B.                    (Fahrtrichtung links)

#define TURN_SPEED        255     // schnell Drehen

#define LEFT              LOW     // Links drehen bei LOW

#define RIGHT             HIGH    // ... und Rechts bei HIGH

#define CLOSEST_DISTANCE  280     // Bei diesem Wert am IR Sensor soll der Roboter anhalten --> Ist der Wert KLEINER hält der Roboter FRÜHER an

#define SERVO_LEFT        45     // Bei welchem Winkel stößt der Sensor auf dem Servo links an?   < < Ggf. anpassen!  10

#define SERVO_RIGHT       135     // Bei welchem Winkel stößt der Sensor auf dem Servo rechts an?  < <  Ggf. anpassen! 180



Servo SensorServo;          // Mit diesem Element wird der Servo gesteuert

byte ServoPosition = 90;    // Mittelposition des Servos (geradeaus)

boolean TurnServo = RIGHT;  // Drehrichtung des Servos



// An welcher Servo Position ist die Distanz zur Gefahr am höchsten?

int SeekingPositionWithClosestDanger()

{
```

```
// Erst mal anhalten
digitalWrite(BRAKE_A, HIGH);
digitalWrite(BRAKE_B, HIGH);
int ServoPosition;
int MinimumDistance = 0;
int MinimumServoPosition = 0;

// Von rechts (SERVO_RIGHT) nach links (SERVO_LEFT) alle Servo Positionen anfahren.
// Mit ServoPosition-- wird der Wert immer um 1 verringert.
for(ServoPosition = SERVO_RIGHT; ServoPosition >= SERVO_LEFT; ServoPosition--)
{
 // Servo Wert einstellen
 SensorServo.write(ServoPosition);
 // Warten bis Servoposition erreicht
 delay(10);

 // ist der aktuelle Wert näher, als der minimale Wert?
 if(analogRead( IR_SENSOR ) > MinimumDistance )
 {
  // Ja: aktueller Wert ist neues Minimum
  MinimumDistance = analogRead( IR_SENSOR );

  // Außerdem merken wir uns die Servo Position
  MinimumServoPosition = ServoPosition;
 }
}

// Die gefundene Position wieder einstellen und Wert zurückgeben
```

```
  SensorServo.write(MinimumServoPosition);

  return MinimumServoPosition;
}

// Vorwärts fahren und dabei den Sensor hin und her schwenken
void DriveForward()
{
  SensorServo.write( ServoPosition );
  //Beide Motoren auf Geradeaus stellen, ...
  digitalWrite( DIR_A, HIGH );
  digitalWrite( DIR_B, HIGH );

  // ... Vollgas und ...
  analogWrite( PWM_A, FULL_SPEED_A );
  analogWrite( PWM_B, FULL_SPEED_B );

  // ..Bremsen lösen!
  digitalWrite( BRAKE_A, LOW );
  digitalWrite( BRAKE_B, LOW );

  // Dreht sich der Servo nach links?
  // Weiter nach links drehen, also 1 vom Wert abziehen
  if( TurnServo == LEFT )
    ServoPosition--;

  // Dreht sich der Servo nach rechts?
```

```
  // Weiter nach rechts drehen, d.h. Wert um 1 erhöhen
  if( TurnServo == RIGHT )
    ServoPosition++;

  // Hat der Servo das linke Ende erreicht?
  // Dann nach rechts drehen
  if( ServoPosition < SERVO_LEFT )
    TurnServo = RIGHT;

  // Hat der Servo das rechte Ende erreicht?
  // Dann nach links drehen
  if( ServoPosition > SERVO_RIGHT )
    TurnServo = LEFT;
}

// Drehen! Aber in welche Richtung?
// LEFT für links (gegen den Uhrzeiger)
// RIGHT für rechts (im Uhrzeigersinn)
void Turn( boolean Direction )
{
  // Bremsen
  digitalWrite( BRAKE_A, HIGH );
  digitalWrite( BRAKE_B, HIGH );
  delay( 500 );

  // Motor A in entgegen der "RICHTUNG" drehen
  digitalWrite( DIR_A, !Direction );
```

```
  // Motor B in die "RICHTUNG" drehen
  digitalWrite( DIR_B, Direction );

  // Geschwindigkeit für das Drehen einstellen
  analogWrite( PWM_A, TURN_SPEED );
  analogWrite( PWM_B, TURN_SPEED );

  // Bremsen lösen
  digitalWrite( BRAKE_A, LOW );
  digitalWrite( BRAKE_B, LOW );

  // Solange drehen, bis die vom Sensor gemessene Entfernung, größer als CLOSEST_DISTANCE ist.
Also "freie Sicht" herrscht.
  while( CLOSEST_DISTANCE < analogRead( IR_SENSOR ) )
  {
    delay( 50 );
  }
  // Halt!
  digitalWrite( BRAKE_A, HIGH );
  digitalWrite( BRAKE_B, HIGH );
  delay(1000);
}


// Bevor es los geht... Diese Funktion wird am Anfang genau einmal ausgeführt.
void setup( )
{
  //Motor A (rechts) initialisieren
```

```
 pinMode( DIR_A, OUTPUT );    // Pin für Richtung Motor A als Ausgang definieren
 pinMode( BRAKE_A, OUTPUT );  // Pin für Bremse Motor A als Ausgang definieren
 //Motor B (links) initialisieren
 pinMode( DIR_B, OUTPUT );    // Pin für Richtung Motor B als Ausgang definieren
 pinMode( BRAKE_B, OUTPUT );  // Pin für Bremse Motor B als Ausgang definieren

 // Beide Bremsen anziehen, HIGH = Bremsen!
 digitalWrite( BRAKE_A, HIGH );
 digitalWrite( BRAKE_B, HIGH );

 // Servo initialiseren und auf 90° stellen
 SensorServo.attach( SERVO );
 SensorServo.write( 90 );
 delay( 500 );

 // Warten bis etwas vor dem Sensor ist
 while( CLOSEST_DISTANCE  > analogRead( IR_SENSOR ) )
 {
   // solange warten...
   delay( 100 );
 }

 // Los geht's!!!
}
```

// Der eigentliche Programmablauf. Nachdem setup() fertig ist wird die Funktion loop() endlos nacheinander ausgeführt.

```
void loop( )

{

  int DangerPosition;

  int Distance;


  Distance = analogRead( IR_SENSOR );    // was misst der Sensor?

  if(Distance > CLOSEST_DISTANCE )       // nah genug? Ist ein Hindernis nah, kommt ein großer Wert zurück. Sonst ein kleiner.

  {

    DangerPosition = SeekingPositionWithClosestDanger();  // Nochmal alles scannen um zu wissen, wo genau die Gefahr am nächsten ist...


    // Gefahr Links?

    if(DangerPosition > 90)   // <= 90  geändert am 24.08.20

    {

      Turn( RIGHT );  // Rechts drehen

    }

    // Oder doch Rechts?

    if( DangerPosition <= 90 )  // > 90  geändert am 24.08.20

    {

      Turn( LEFT );   // dann Links drehen

    }

  }


  DriveForward();  // Immer gerade aus!!!

  delay( 10 );

}
```