

**Montage und Programmierung
eines Roboters für
den Hessen SolarCup
Disziplin: SolaRobot
Teil 2.3: Magnetsensor digital/Motor**

Roboter aufbocken!!!

Wenn wir das Programm auf den Roboter hochgeladen haben, dann **fährt er sofort los!!!**

Damit er nicht vom Tisch hüpfen bocken wir ihn auf und stellen ein Hindernis in den Weg.

Dann schauen wir uns die Sensorwerte an: **Mag_x_Value** besprechen wir später.

Dreht den Roboter auf der Stelle.
Welche Werte werden bei **BNO055_Value** angezeigt?

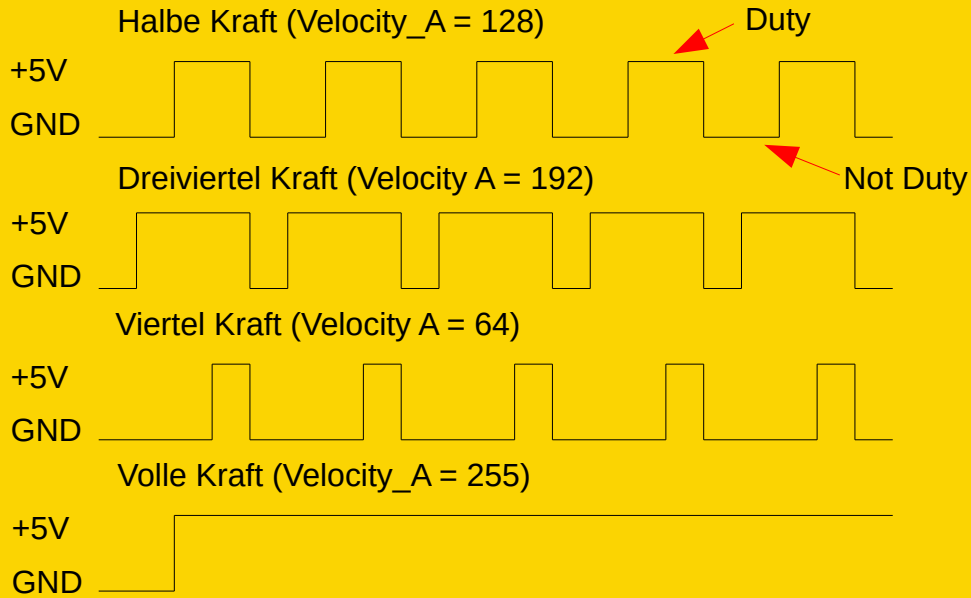
Geht zur Magnetstreifenbahn und schaut euch die **Mag_xy_Value** Werte an. Wann sind sie 0 wann sind sie 1.
Was ist mit den beiden linken Sensoren?

Die **US_Werte** haben wir ja schon behandelt.

Mag_x_Value	BNO055_Value	Mag_L1_Value	Mag_LO_Value	Mag_R0_Value	Mag_R1_Value	Mag_R2_Value	Mag_R3_Value	US_Wert_L	US_Wert_R
-	-	---	---	-	-	-	-	-	-
50	0	826	823	1	1	1	1	5	9
37	0	827	824	1	1	1	1	6	9
45	0	827	823	1	1	1	1	6	9
50	0	827	825	1	1	1	1	5	9
30	0	828	825	1	1	1	1	5	9
37	0	825	824	1	1				

```
04_Magnetsensor_digital_Motor | Arduino IDE 2.3.8
Datei Bearbeiten Sketch Werkzeuge Hilfe
Arduino Nano
04_Magnetsensor_digital_Motor.ino Drive_Functions.h Init.h
101 }
102 /**Testschleife Ende*****/
103
104 /***/
105 //Roboter aufbocken - - - Roboter aufbocken - - - Roboter aufbocken - - -
106 /***/
107 Toggle_On_Off = 1; //LED blinken an
108 if (US_Wert_L < 10) {
109     Stop();
110 }
111 else {
112     Forward(128, 128);
113 }
114 Clock(); //Werte auf Monitor anzeigen, LED blinken, Werte ermitteln
115
116 }
117
118 void Data_Visualizer() {
119     Data[0] = Mag_x_Value; //Analoge Magnetwerte über I2C
120     Data[1] = BNO055_Value; //Werte vom Gyroskop
121     Data[2] = Mag_L1_Value; //Analoge Magnetwerte, 0 bis 255
122     Data[3] = Mag_LO_Value; //0 -255
123     Data[4] = Mag_R0_Value; //Digitale Magnetwerte, 0 oder 1
124     Data[5] = Mag_R1_Value; //Digital
125     Data[6] = Mag_R2_Value; //Digital
126     Data[7] = Mag_R3_Value; //Digital
127     Data[8] = US_Wert_L; //Abstand vom linken US Sensor
128     Data[9] = US_Wert_R; //Abstand vom rechten US Sensor
129 }
130
Ausgabe
Der Sketch verwendet 10604 Bytes (34%) des Programmspeicherplatzes. Das Maximum sind 30720
Globale Variablen verwenden 794 Bytes (38%) des dynamischen Speichers, 1254 Bytes für lokal
Zeile 139, Spalte 39 Arduino Nano an COM5 2
```

PWM: Pulse Width Modulation



Warum gibt es Velocity_A = 256 NICHT?

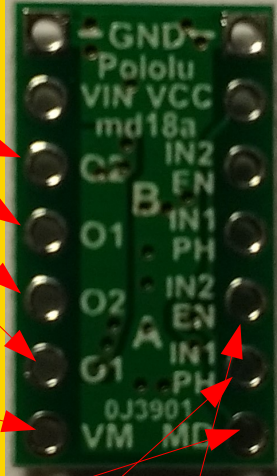
The screenshot shows the Arduino IDE 2.3.8 interface. The title bar reads '04_Magnetsensor_digital_Motor - Drive_Functions.h | Arduino IDE 2.3.8'. The menu bar includes 'Datei', 'Bearbeiten', 'Sketch', 'Werkzeuge', and 'Hilfe'. The toolbar contains icons for saving, running, and uploading. The file explorer shows three files: '04_Magnetsensor_digital_Motor.ino', 'Drive_Functions.h', and 'Init.h'. The main editor displays the following C++ code:

```
13
14 void Forward (unsigned char, unsigned char); //Direction with velocity A and B, for d
15 void Backward (unsigned char, unsigned char); //Direction with velocity A and B, for
16
17 void Stop (void); //Stop all Motors
18
19 //Motors forward
20 void Forward (unsigned char velocity_A, unsigned char velocity_B) {
21     analogWrite(PWMA_Pin, velocity_A); //Pulwidth modulation, MotorA
22     analogWrite(PWMB_Pin, velocity_B); //Pulwidth modulation, MotorB
23     digitalWrite(Motor_A_Pin, 0); //MotorA
24     digitalWrite(Motor_B_Pin, 0); //MotorB
25 }
26
27 //Motors backward
28 void Backward (unsigned char velocity_A, unsigned char velocity_B) {
29     analogWrite(PWMA_Pin, velocity_A); //Pulwidth modulation, MotorA
30     analogWrite(PWMB_Pin, velocity_B); //Pulwidth modulation, MotorB
31     digitalWrite(Motor_A_Pin, 1); //MotorA
32     digitalWrite(Motor_B_Pin, 1); //MotorB
33 }
34
35
36 // //Stop the motors
37 void Stop (void) {
38     analogWrite(PWMA_Pin, 0); //Pulwidth modulation, MotorA
39     analogWrite(PWMB_Pin, 0); //Pulwidth modulation, MotorB
40 }
41
```

The output window at the bottom shows the following message: 'Der Sketch verwendet 10604 Bytes (34%) des Programmspeicherplatzes. Das Maximum sind 30720. Globale Variablen verwenden 794 Bytes (38%) des dynamischen Speichers, 1254 Bytes für loka'. The status bar at the bottom indicates 'Zeile 12, Spalte 18 Arduino Nano an COM5 [keine Verbindung]'.

Auf Polung achten!!!

Motortreiber



Motorausgänge

IN2(B) In1(B)

Steuer-
spannungs-
eingänge

Motorspannung

Motor_A_Pin

PWMA_Pin

Mode	PH(A)	EN (A)	Motor
High	X	Low	Aktiv bremsen
High	Low	PWM	Vorwärts
High	High	PWM	Rückwärts

X = Egal
High = +5V
Low = GND

Wann fährt euer Roboter
schnurgeradeaus?

04_Magnetsensor_digital_Motor - Drive_Functions.h | Arduino IDE 2.3.8

Datei Bearbeiten Sketch Werkzeuge Hilfe

Arduino Nano

```

13
14 void Forward (unsigned char, unsigned char); //Direction with velocity A and B, for d
15 void Backward (unsigned char, unsigned char); //Direction with velocity A and B, for
16
17 void Stop (void); //Stop all Motors
18
19 //Motors forward
20 void Forward (unsigned char velocity_A, unsigned char velocity_B) {
21     analogWrite(PWMA_Pin, velocity_A); //Pulwidth modulation, MotorA
22     analogWrite(PWMB_Pin, velocity_B); //Pulwidth modulation, MotorB
23     digitalWrite(Motor_A_Pin, 0); //MotorA
24     digitalWrite(Motor_B_Pin, 0); //MotorB
25 }
26
27 //Motors backward
28 void Backward (unsigned char velocity_A, unsigned char velocity_B) {
29     analogWrite(PWMA_Pin, velocity_A); //Pulwidth modulation, MotorA
30     analogWrite(PWMB_Pin, velocity_B); //Pulwidth modulation, MotorB
31     digitalWrite(Motor_A_Pin, 1); //MotorA
32     digitalWrite(Motor_B_Pin, 1); //MotorB
33 }
34
35
36 // //Stop the motors
37 void Stop (void) {
38     analogWrite(PWMA_Pin, 0); //Pulwidth modulation, MotorA
39     analogWrite(PWMB_Pin, 0); //Pulwidth modulation, MotorB
40 }
41

```

Ausgabe

Der Sketch verwendet 10604 Bytes (34%) des Programmspeicherplatzes. Das Maximum sind 30720
Globale Variablen verwenden 794 Bytes (38%) des dynamischen Speichers, 1254 Bytes für loka

Zeile 12, Spalte 18 Arduino Nano an COM5 [keine Verbindung]

Montage und Programmierung
eines Roboters für
den Hessen SolarCup
Disziplin: SolaRobot
Teil 2.4: Magnetspur folgen