

Montage und Programmierung
eines Roboters für
ROBOCUP JUNIOR RESCUE
mit Arduino Nano
Teil 2.6: Gyroskop

Notwendige Änderungen an deinem Basisprogramm, wenn du das Gyroskop für die Drehung um 90° benutzen willst.

Neue Variablen:

```
unsigned char z; //Tests
unsigned char North = 0; //Direction of the robot
unsigned char East = 1; //
unsigned char South = 2; //
unsigned char West = 3; //
unsigned char direction = North; //Start position
signed int Angle = 0; //For turning 90°
```

Neue Funktionen:

```
//Funktionen
void Data_Visualizer (void); //Daten, die angezeigt werden sollen
void Warten_auf_Starttaster (void); //Nur Daten werden angezeigt
void Programm_unterbrechen(void);
void Turn_right_90_degree(void);
void Turn_left_90_degree(void);
```

Neue Startsequenz:

```
void loop() {
    _delay_ms(200); //Warten darauf, dass das System "hochgefahren" ist
    USART_Init(MYUBRR); //Initialisieren der Datenübertragung
    sei(); //Interrupt einschalten
    _delay_ms(50); //Warten auf erste Werte
    Warten_auf_Starttaster();
}
```

Anzeige einer int Variablen

```
Data[5] = Encoder;
Data[6] = Angle; //LO-Byte
Data[7] = Angle >> 8; //HI-Byte
```

Header „Init.h“

```
//TWBR = 0; //1.000.000Hz (1000kHz, 1MHz)
//TWBR = 12; //400.000Hz (400kHz)
TWCR = 0b11000101; //TWINT, clear; TWEA,
//TWSR no change to the initial values:
_delay_ms(300); //wichtig!!!
Ping_LCD(); //Überprüfen, ob das LCD ang
Ping_BNO055(); //Überprüfen, ob BNO055 (
BNO055_Init(); //BNO055 initialisieren
}
    Unten anhängen
```

Links um 90° drehen

```
//Links um 90° drehen
void Turn_left_90_degree(void) {
    Links_drehen(200); //Geschwindigkeit
    Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
    while (Angle > -90) { //Dreht womöglich ein bisschen
        Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
        _delay_us(1); //Sonst funzt das Programm nicht!
        Data_Visualizer(); //Data beeing transmitted to
    }
    Stop(); //Stop the motors
    _delay_ms(50); //Warten auf nächsten Wert
    Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
    Data_Visualizer(); //Anzeigen
    _delay_ms(3000); //Test
    if (direction == 0) direction = 4; //Einmal um die
    direction--; //
}
```

Rechts um 90° drehen

```
//Rechts um 90° drehen
void Turn_right_90_degree(void) {
    Rechts_drehen(200); //Geschwindigkeit
    Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
    while (Angle < 90) { //Dreht womöglich ein bisschen
        Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
        _delay_us(1); //Sonst funzt das Programm nicht!
        Data_Visualizer(); //Data beeing transmitted to
    }
    Stop(); //Stop the motors
    _delay_ms(50); //Warten auf nächsten Wert
    Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
    Data_Visualizer(); //Anzeigen
    _delay_ms(3000); //Test
    direction++; //
    if (direction == 4) direction = 0; //Einmal um die
}
```

```
unsigned char z; //Tests
unsigned char North = 0; //Direction of the robot
unsigned char East = 1; //
unsigned char South = 2; //
unsigned char West = 3; //
unsigned char direction = North; //Start position
signed int Angle = 0; //For turning 90°
```

Bei dieser Bezeichnung gehen wir davon aus, dass der Roboter am Start in Richtung „Norden“ zeigt. Ganz egal, wie er tatsächlich steht. Um nicht noch „aufwändig“ die Orientierung im Raum, bei der Berechnung des Winkels mitzuschleppen, **setzen wir den Roboter erst gut ausgerichtet in das Labyrinth und schalten ihn dann erst ein.**

Dann liefert das Gyroskop den Wert Null, +/-1°.

Wenn er nach rechts dreht wird die Richtung „größer“. North = 0 → East = 1 → South = 2...

```
//Rechts um 90° drehen
void Turn_right_90_degree(void) {
  Rechts_drehen(200); //Geschwindigkeit
  Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
  while (Angle < 90) { //Dreht womöglich ein bisschen
    Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
    _delay_us(1); //Sonst funzt das Programm nicht!
    Data_Visualizer(); //Data beeing transmitted to
  }
  Stop(); //Stop the motors
  _delay_ms(50); //Warten auf nächsten Wert
  Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
  Data_Visualizer(); //Anzeigen
  _delay_ms(3000); //Test
  direction++; //
  if (direction == 4) direction = 0; //Einmal um die
}
```

Wenn er sich nach links dreht, wird die Richtung „kleiner“.

```
//Links um 90° drehen
void Turn_left_90_degree(void) {
  Links_drehen(200); //Geschwindigkeit
  Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
  while (Angle > 90) { //Dreht womöglich ein bisschen
    Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
    _delay_us(1); //Sonst funzt das Programm nicht!
    Data_Visualizer(); //Data beeing transmitted to
  }
  Stop(); //Stop the motors
  _delay_ms(50); //Warten auf nächsten Wert
  Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
  Data_Visualizer(); //Anzeigen
  _delay_ms(3000); //Test
  if (direction == 0) direction = 4; //Einmal um die
  direction--; //
}
```

Die Initialisierung des LCD braucht seine Zeit. Vorher dürfen wir die Übertragung dorthin nicht beginnen. Daher warten wir 200ms bevor wir den Interrupt einschalten.

```
void loop() {
  _delay_ms(200); //Warten darauf, dass das System "hochgefahren" ist
  USART_Init(MYUBRR); //Initialisieren der Datenübertragung
  sei(); //Interrupt einschalten
  _delay_ms(50); //Warten auf erste Werte
  Warten_auf_Karttaster();
}
```

Dann warten wir noch auf die ersten Werte, die im Interrupt generiert werden.

Wir können auf unserem Display und auf dem Monitor nur 8-Bit große positive Zahlen anzeigen lassen.

Wollen wir z.B. eine Integer-Variable, 16Bit, anzeigen lassen, müssen wir sie in zwei Teilen ausgeben.

Geben wir die Variable einfach aus, dann werden die unteren 8Bit (LSB) ausgegeben. Für die oberen 8Bit (MSB) schieben wir die Variable 8-mal nach rechts, bevor wir sie anzeigen lassen.

```
Data[5] = Encode;
Data[6] = Angle; //LO-Byte
Data[7] = Angle >> 8; //HI-Byte
```

Jetzt ist Angle aber eine **signed int** Variable, kann also auch negative Werte annehmen. Daher bekommen wir für das High-Byte bei -1 den Wert 255 angezeigt!

Wenn wir nämlich von 0 eine 1 abziehen, dann ergibt das, an allen zur Verfügung stehenden Stellen, eine 1.

$11_b - 01_b = 10_b$, das ist dezimal nicht anders!
 $10_b - 01_b = 01_b$, dezimal ergibt das 09_d
Hierbei haben wir einen „Übertrag“.
Wenn wir jetzt rechnen $00_b - 01_b$ dann kommt bei zwei Stellen 11_b raus. Bei 8 Stellen eben 1111111_b . Das kennen wir umgekehrt schon vom Überlauf. Dort ergibt $255 + 1$ eine Null.

30	13	17	0	51	0	255	255	0	0
30	13	17	0	52	0	255	255	0	0
33	13	17	0	52	0	255	255	0	0
41	10	17	0	51	0	255	255	0	0
41	19	17	0	51	0	255	255	0	0

Wir haben also bei -1° die Anzeige 255 255 für Angle.
 Und, $256 - 90 = 166$,
 hat sich also der Roboter von 0° nach links um 90° gedreht,
 sollte im Display 166 255 erscheinen.

20	23	15	0	54	0	166	255	0	1
22	25	15	0	54	0	166	255	0	1
19	24	15	0	54	0	166	255	0	1
22	23	15	0	54	0	166	255	0	1
22	23	15	0	54	0	166	255	0	1

Bei der Rechtsdrehung haben wir dagegen keine Probleme, den aktuellen Winkel abzulesen.

19	13	15	0	50	0	90	0	255	2
18	26	15	0	50	0	90	0	255	2
17	16	15	0	50	0	90	0	255	2
17	16	15	0	50	0	90	0	255	2
18	20	15	0	50	0	90	0	255	2

Jetzt schauen wir uns noch die Berechnung des Winkels an,
 mit dessen Wert wir die Drehung steuern:

```
Links_drehen(200); //Geschwindigkeit
Angle = (Gyro_X - direction*90 + 180) % 360 - 180;
while (Angle > -90) { //Dreht womöglich ein bisschen
```

Gyro_X ist der aktuelle Winkel des Roboters im Verhältnis zu seiner Startposition. Irgendwas zwischen 0 und 359.

direction hat Werte von 0 bis 3, **North** bis **West**

direction * 90 hat also bei **North** den Wert 0.

D.h. die Klammer hat den Wert $Gyro_X + 180$.

% ist der Modulo-Operator. Modulo 360 bedeutet, dass ein Wert durch 360 geteilt wird. Der Rest der Division ist das Ergebnis von Modulo.

z.B. $365 \% 360$ ist 5, $725 \% 360$ ist auch 5.

Gyro_X – North*90	add 180	Modulo 360	sub 180
180	360	0	-180
270	450	90	-90
359	539	179	-1
0	180	180	0
1	181	181	1
90	270	270	90
179	359	359	179

Die nebenstehende Tabelle zeigt Ergebnisse der Berechnung für North (=0). Unser Problem ist ja, dass wir einen Sprung zwischen 0 und 359 haben, was uns die Abfrage (if, while) sehr erschwert.

Das können wir mit der Formel umgehen. Damit bekommen wir nämlich Werte zwischen -180 und 179.

Wir können also eine while-Schleife programmieren für eine 90° Drehung, indem wir als Bedingung kleiner 90° bei einer Rechtsdrehung und größer -90° bei einer Linksdrehung stellen.

Steht unser Roboter in einer anderen Richtung, kommen die gleichen Werte bei dieser Formel heraus. Könnt ihr gerne mal nachprüfen.

Montage und Programmierung
eines Roboters für
ROBOCUP JUNIOR RESCUE
mit Arduino Nano
Teil 2.7: ?