2.3. DC-Motoren

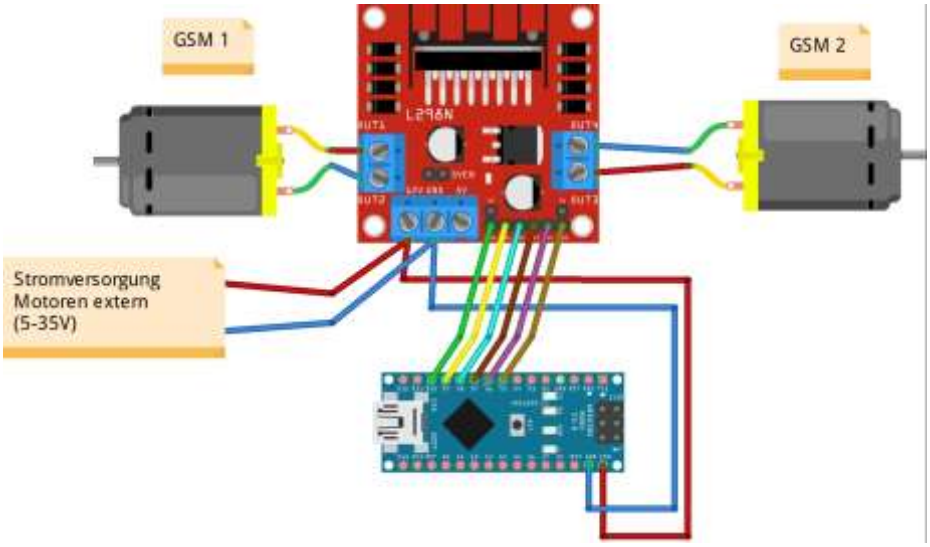
An den folgenden Beispielen soll der Einsatz und die Steuerung von DC-Motoren (später ans Gleis für unsere Loks) gezeigt werden. Ich habe dazu einfache Motoren verwendet. Die sollten unbedingt über eine externe Stromversorgung betrieben werden!!

2 1,5- 9V

1 6V

2.3.1.Variante 1: 2 Motoren mit verschiedenen Geschwindigkeiten vor- und rückwärts.

Dazu benötigen wir einen Motortreiber. Ich nehme hier den L298N (H-Brücke zu Steuerung von 2 Motoren. Hier der Aufbau (mal ohne Breadboard)



Der Sketch dazu. Der Ablauf erfolgt automatisch gesteuert vom Nano :

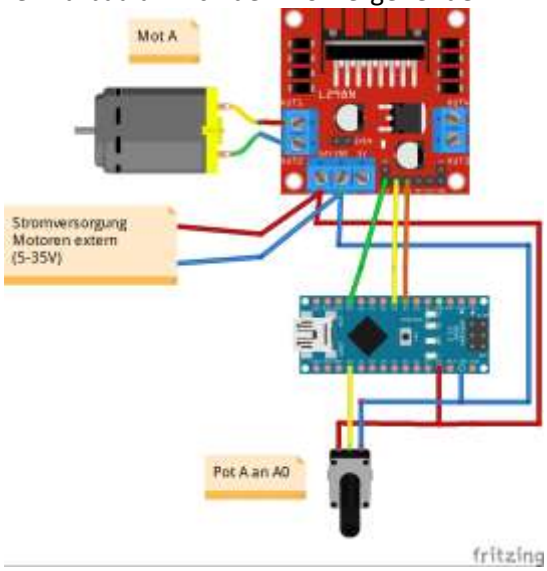
```
// Gleichstrommotor 1

int GSM1 = 10;           // Pin 5 und 10 können (wenn an PWM-Pin) Geschwindigkeit
regeln
int in1 = 9;             // Pin 6-9 zum Steuern der Motoren
int in2 = 8;

// Gleichstrommotor 2

int GSM2 = 5;
int in3 = 7;
int in4 = 6;

void setup()
{
  pinMode(GSM1, OUTPUT);    // Festlegen der Pins als Ausgang
  pinMode(GSM2, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}
void loop()
{
```

```
  digitalWrite(in1, HIGH);  //Motor1 beginnt zu rotieren
  digitalWrite(in2, LOW);
  analogWrite(GSM1, 200);   //Motor1 soll mit der Geschwindigkeit "200" (max. 255)
rotieren
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW); // Motor 2 beginnt zu rotieren ebenfalls mit der
Geschwindigkeit "200" rotieren
  analogWrite(GSM2, 200);   // Motor 2 soll (max. 255)
  delay(2000);


  digitalWrite(in1, LOW);   //Durch die Veränderung von HIGH auf LOW (bzw. LOW auf
HIGH) wird die Richtung der Rotation verändert.
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  delay(2000);

  digitalWrite(in1, LOW);   // Anschließend sollen die Motoren 2 Sekunden ruhen.
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);

  delay(2000); // kurze Pause und dann von vorn
}
```

Das Schöne, mit dem L298N müssen nicht mal neue Bibliotheken eingebaut werden. Über analogWrite((GSMNr, Zahl zwischen 0 und 255) lässt sich die Motorgeschwindigkeit (in Abhängigkeit der externen Spannungsversorgung) einstellen. Ich empfehle: PROBIEREN.

2.3.2.Variante 1: Motor mit Poti zur einfachen Geschwindigkeitsregelung

Das wird jetzt die Vorstufe zum einfachen Fahrregler. Hier geht es um das Kennenlernen der map() - Funktion, die wir später noch brauchen. Es soll also mit einem Poti die Geschwindigkeit stufenlos (als PWM) vorwärts gesteuert werden.
Der Aufbau ähnlich dem vorhergehenden:



Übrigens, für 2 Motoren, ein 2. Poti an z.B. D8,7 und 6 sowie an A1. Das Ganze geht auch mit Schieberegler, wer das lieber mag.

Der Sketch dazu:

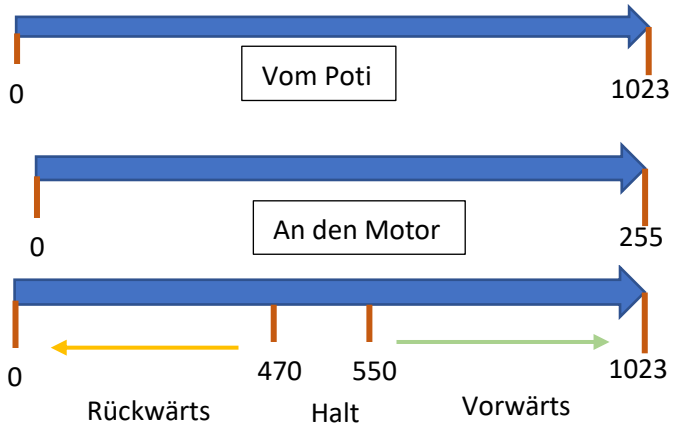
```
// Gleichstrommotor 1

int MotA = 9;
int in1 = 5;
```

```
int in2 = 4;
int Speedcontr = A0;
int Motorspeed = 0;

void setup()
{
 Serial.begin(9600);
 pinMode(MotA, OUTPUT);
 pinMode(in1, OUTPUT);
 pinMode(in2, OUTPUT);

}
void loop()
{

 digitalWrite(in1, HIGH);  // Motor 1 beginnt zu rotieren
 digitalWrite(in2, LOW);
 Motorspeed  = analogRead(Speedcontr);
 Motorspeed = map(Motorspeed, 0, 1023, 0, 255);
 /* die map Funktion liest die Werte von Motorspeed ein,
  die kommen in einem Bereich von 0-1023 (das liefert das Poti)
  und wandelt die passend in einen Bereich von 0-255 ( das kann der Motor)
 */
 if (Motorspeed < 8)Motorspeed = 0;// setzt  die Empfindlichkeit für „Stopp“ höher, muss man
ausprobieren in Abhängigkeit vom Poti
 analogWrite(MotA, Motorspeed);   //Motor1 dreht  in Abhängigkeit vom Poti
Serial.println(Motorspeed); //dient der Anzeige der Werte bei eingeschaltetem seriellem Monitor
}
```

Schön, PWM gesteuerte Motorregelung, macht sich gut für unsere Loks. Aber eben nur vorwärts ☹
Um jetzt mit einem Poti Geschwindigkeit und Fahrtrichtung zu ändern, gibt es mehrere Varianten: Umpolen mit Taster oder Relais, über Magnetschalter (Reed oder Hallsensor) oder gleich über das Poti. Ich gehe hier auf das Poti ein, finde ich am elegantesten und kommt einem Fahrregler am nächsten.

2.3.3. DC-Motor über Poti mit Geschwindigkeits- und Richtungssteuerung

Eigentlich gibt’s da nicht all zu viel zu tun. Der Aufbau ist analog zu 2.3.2. Im Sketch müssen wir was ändern, die map-Funktion ist eine Lösung. Wir wollen, dass in Mittelstellung des Potis „Halt“, nach rechts „Vorwärts, nach links „Rückwärts“ realisiert wird. Also wie beim echten Trafo. Das Poti liefert analoge Werte von 0-1023, die wir dann umwandeln in den Bereich 0-255. Das heißt, wir müssen den Lesebereich des Potis einteilen und mit If-Abfragen in die verschiedenen Ausgabebereiche umwandeln. Dazu eine kleine erklärende Skizze.



Als Satz formuliert:
Wenn das vom Poti < 470, setze In1 auf LOW, in2 auf HIGH (rückwärts), wenn Poti > als 550, in1 auf HIGH, in2 auf LOW (vorwärts), ansonsten (zwischen 470 und 550) Motorspeed auf 0 (Halt). Entsprechend wird das im Map umgesetzt auf die Ausgabebereiche.

Hier der Sketch:

```
/*
  DC-Motorsteuerung mit L298N und Poti zur
  Steuerung der Geschwindigkeit und Fahrtrichtung
  über Mittelstellung eines Potis (Schiebereglers)
  von Schorsch für ZFI
*/
// ======================================================
int enA =9;
int in1= 4;
int in2=5;
int motorspeed = 0;

void setup() {
  Serial.begin(9600);
 pinMode (enA, OUTPUT);
 pinMode (in1, OUTPUT);
 pinMode (in2, OUTPUT);
}

void loop() {
  int PotiMitte=analogRead(A0);     //Mittelabgriff Poti lesen
  if (PotiMitte<470){
    digitalWrite(in1,LOW);         // Wenn <470, Motoreingänge so stellen
    digitalWrite(in2,HIGH);
    motorspeed=map(PotiMitte,470,0,0,255); //und die Werte "rückwärts lesen"
    }
    else if(PotiMitte>550){
      digitalWrite(in1,HIGH);      // Wenn > 555, Motoreingänge umpolen
      digitalWrite(in2,LOW);
      motorspeed=map(PotiMitte,550,1023,0,255);// und ab 550 aufwärts lesen
    }
    else{                        // ansonsten (zwische 470 und 550) Motor aus
      motorspeed=0;              // Toleranzbereich des Potis, kann man verändern
    }
    analogWrite(enA, motorspeed);   // fahr los
    Serial.println(motorspeed);     // dient der Ausgabe auf seriellen Monitor, um sich die
Ausgabewerte anzeigen zu lassen
}
```

Soochen, das wäre dann schon mal ein einfacher PWM-Regler, Aufwand gering, Kosten ca. 15,-. Mit dem L298N lassen sich zwei Motoren (Loks) steuern, einfach alles doppeln. Das lässt sich dann auch irgendwann (späterns) über Bluetooth und eine schicke App steuern.

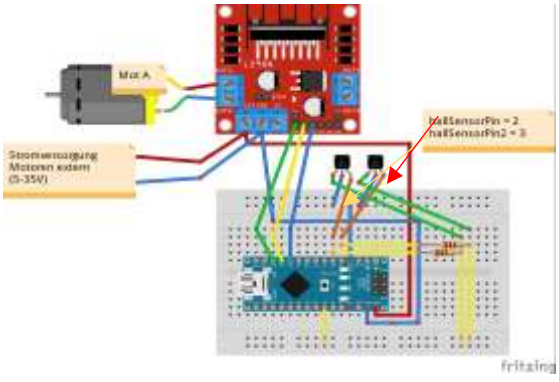
2.3.4.DC-Motor automatisch langsam anfahren/bremsen

Nächste Idee für spätere automatisierte Abläufe. Der Motor soll langsam anfahren bis Höchstgeschwindigkeit bzw. bremsen bis zum Halt.
Gleicher Aufbau, wie im vorherigen. Der Sketch dazu:

```
/*
  DC-Motorsteuerung mit L298N zum Regeln
  von Anfahr- und Bremsverhalten ohne Auslöser
  von Schorsch für ZFI
  27.03.21
*/
```

```
// Gleichstrommotor 1

int GSM1 = 9;
int in1 = 5;
int in2 = 4;

void setup()
{
  Serial.begin(9600);
  pinMode(GSM1, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);

}
void loop()
{
  for (int i = 0; i <=240; i += 20) { //in 20-iger Schritten bis Maximalgeschwindigkeit
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);   // Motor 1 beginnt zu rotieren

    analogWrite(GSM1, i); // Motor 1 beschleunigt
    delay(1000);          // das delay steuert die Geschwindigkeit des Hochzählens
  Serial.println(i);
  }
  delay(2000); //
  int i=240; // für das Bremsen sollte das i den gleichen Wert haben, wie das max. i beim Anfahren
// (hier 240), sonst „springt“ oder „bockt“ der  Motor
  for (int i = 240; i >= 0; i -= 20) { // die Schrittweite (i-=20) sollte keine Reste übrig lassen!!
    digitalWrite(in1, LOW);  // Motor 1 beginnt zu rotieren
    digitalWrite(in2, HIGH);
    analogWrite(GSM1, i);
    delay(1000);
  Serial.println(i);
  }
  delay(4000);
}
```

Wenn das jetzt funktioniert, könnte man doch mal daran gehen und Auslöser einbauen, die das Anfahren/Bremsen einleiten. Ich liebe Hallsensoren. Reed, Taster für manuelle Steuerung, Relais tuns natürlich auch, sind mir persönlich zu aufwendig. Jetzt kommen die Grundlagen aus 1.7.2 ins Spiel. Wir (also ich) hatten mit einem Hallsensor eine LED eingeschaltet. Das Prinzip ist anwendbar auf unseren Motor. Erst mal der Aufbau.



Ich hoffe, man kann es erkennen.
Fragen geht auch ☺

Und nun der Sketch, es wird schon etwas komplexer:

```
/*
  DC-Motorsteuerung mit L298N zum Regeln
  von Anfahr- und Bremsverhalten mit 2 Hallsensoren
  von Schorsch für ZFI
  27.03.21
*/

// Gleichstrommotor 1
const int hallSensorPin = 2;
const int hallSensorPin2 =3;
int hallStatus=0;     // Ausgagnswert zum Beschleunigen
int hallStatus2=240;  // Ausgagnswert zum Bremsen
int GSM1 = 9;
int in1 = 5;
int in2 = 4;
void setup()
{
  Serial.begin(9600);
  pinMode(GSM1, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
pinMode(hallSensorPin, INPUT);
pinMode(hallSensorPin2, INPUT);
}
//============================
void loop(){
    hallStatus=digitalRead(hallSensorPin);
    hallStatus2=digitalRead(hallSensorPin2);
  if (hallStatus ==LOW) {
// Beschleunigen
  for (int i = 0; i <=240; i += 40) { //in 40-iger Schritten bis Maximalgeschwindigkeit
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);   // Motor 1 beginnt zu rotieren

    analogWrite(GSM1, i); // Motor 1 beschleunigt
    delay(1000);         // Verzögerung beim Hochzählen
    Serial.println(i);
  }
  }
//=======================================
 if (hallStatus2 ==LOW) {
// Bremsen
  for (int i = 240; i >=0; i -= 40) { //in 20-iger Schritten bis Maximalgeschwindigkeit
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);   // Motor 1 beginnt zu rotieren

    analogWrite(GSM1, i); // Motor 1 beschleunigt
    delay(1000);
    Serial.println(i);
  }
  }
  }
```

Eigentlich ganz einfach (sagt der Willi immer zu mir). Im Loop werden die beiden HallPins eingelesen. Dann erfolgt in jeweils 2 IF's die Abfrage auf Zustand. Je nachdem, an welchem Hallsensor der Magnet vorbeikam, wird entweder beschleunigt oder gebremst. Jetzt muss man ausprobieren, wie schnell beschleunigt/gebremst werden soll (das sind die Zählschleifen i+=40 bzw. i-=40), wie lange es dauern soll (die delay's) und was die Maximalgeschwindigkeit sein soll (hier jeweils die 240).
Natürlich kommt statt des Motors das Ganze ans Gleis ☺. Damit lässt sich das an jede Lok mit ihren Eigenschaften anpassen.
So, erst mal Futter für den nächsten Zeitraum. Probiert es aus.