Radio Direction Finder

RDF

Funkpeiler 6

DK5BD

Inhalt

Funkpeiler 6

Einleitung

Ich hatte schon lange überlegt, einen stationären Peiler zu bauen. Jetzt sollte das Projekt akut werden, da wir auf unserer Rundenfrequenz 145,250 MHz von einem starken Signal gestört werden.

Der erste Probeaufbau (Peiler 5) erfolgte mit sehr kleinen schwachen Motoren. Die Technik funktionierte gut, war aber nicht in der Lage, eine größere Antenne zu drehen. Also mußte eine stärkere Version her, wo ein Scheibenwischermotor die Antenne drehen kann.

Nachdem die Technik funktionierte, mußte ich ein wassergeschütztes Gehäuse besorgen und dieses mit entsprechenden Schnallen auf eine Bodenplatte befestigen.

Konzept

Das Konzept hatte sich ja bewährt: 3 Plattformen aus Laminat wurden mit 3 Flanschlagern ausgestattet, die einen 12mm Gewindestab tragen und führen können.

Die Platten wurden mit 8mm Gewindestäben zusammengehalten. Später kam noch eine Platte hinzu, damit die Technik in das Gehäuse passend eingebaut werden konnte. Als Gehäuse habe ich einen Küchenabfallbehälter von Ikea genommen.

Spezialmaterial

Die erforderlichen Zahnräder habe ich im Internet bei der Firma **igus® GmbH, Spicher Str. 1a, 51147 Köln** bestellt.

Das Hauptzahnrad hat 140 Zähne, die beiden kleinen 30er Räder sind für den Motorantrieb und das 10 Gang-Poti bestimmt.

Pos.	Anzahl	Art.-Nr.	Bezeichnung			Einzel	Gesamt
1	2	P360GM-ST-100-030-00-060-R	igutek P360 Zahnrad			7,26 EUR	14,52 EUR
2	1	P360GM-ST-100-140-00-120-R	igutek P360 Zahnrad			22,87 EUR	22,87 EUR

Den Scheibenwischermotor habe ich bei der Firma

Cleverlog-Autoteile GmbH, Westhafenstraße 1, 13353 Berlin

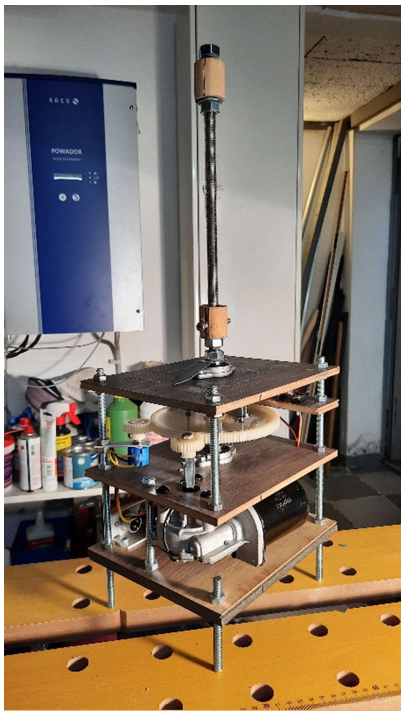
bestellt.

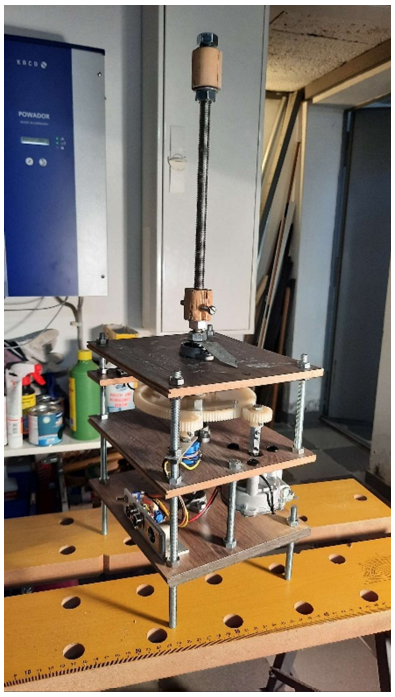
1 HP108 792 85011099 Scheibenwischermotor

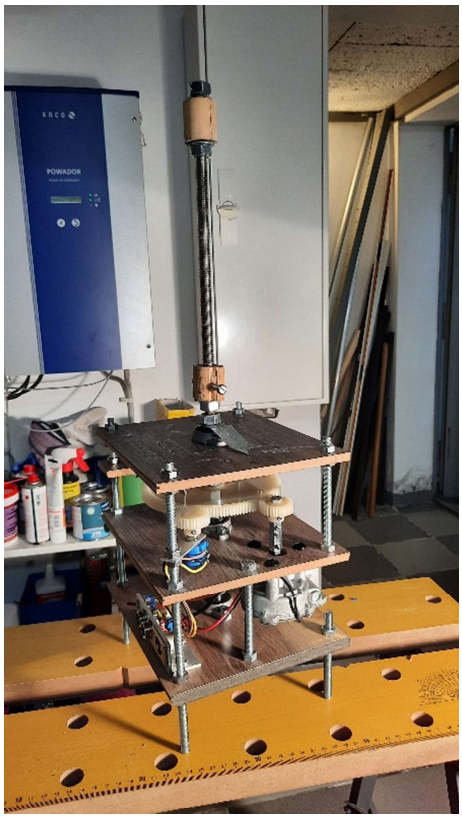
Standardmaterial

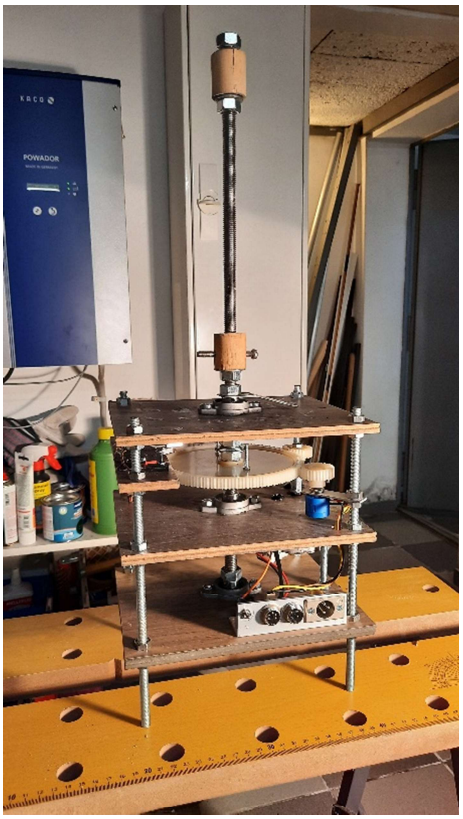
Die Gewindestäbe, Schrauben, Muttern und Scheiben habe im hiesigen Bauhaus Baumarkt gekauft. Die Laminatplatten habe ich aus kostenlosem Überschußlaminat unserer lokalen Holzhandlung zugeschnitten. Laminat ist formstabil und läßt sich leicht bearbeiten-

4 Ansichten des Peilers

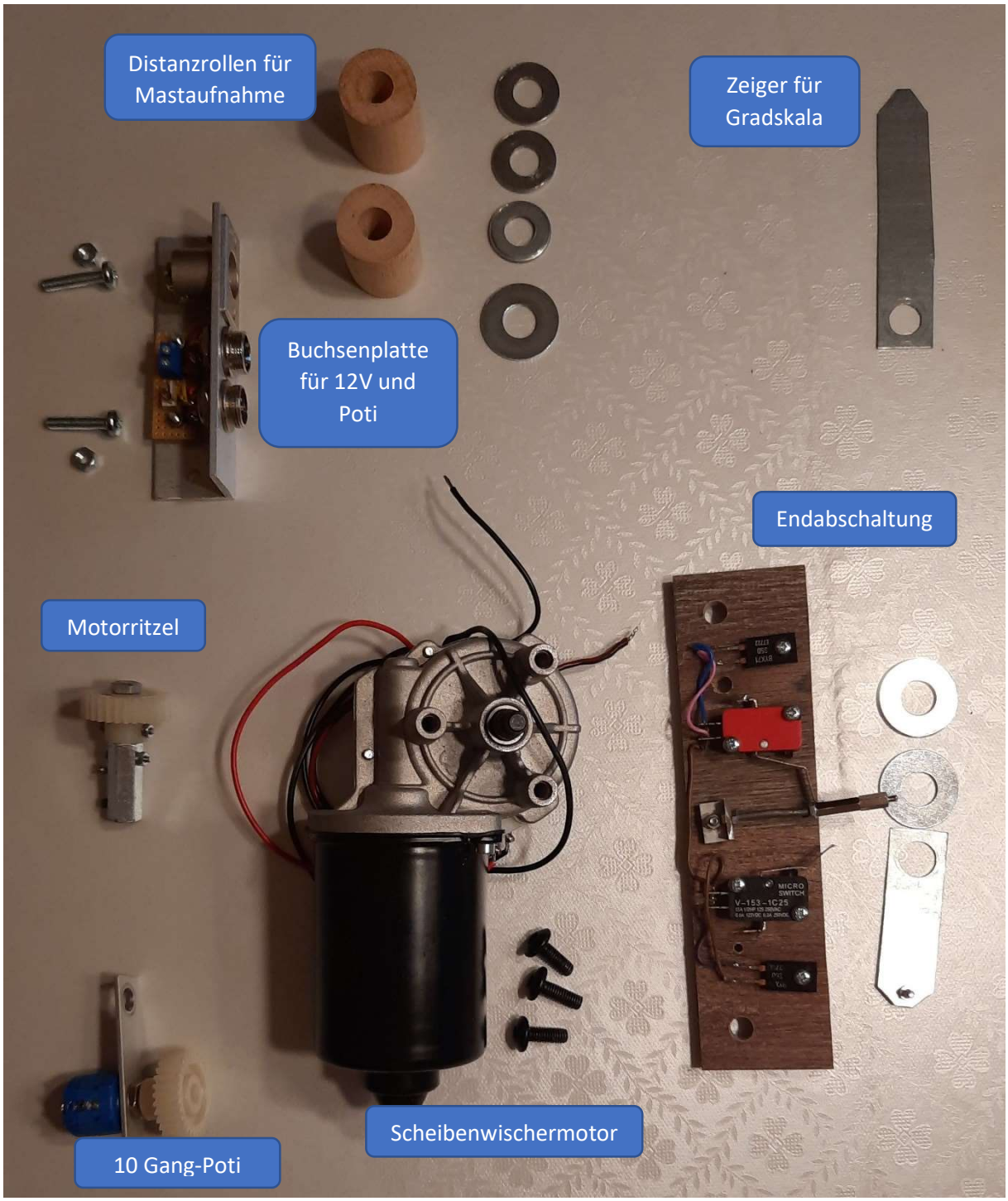





KACO
POWADOR


KACO
POWADOR

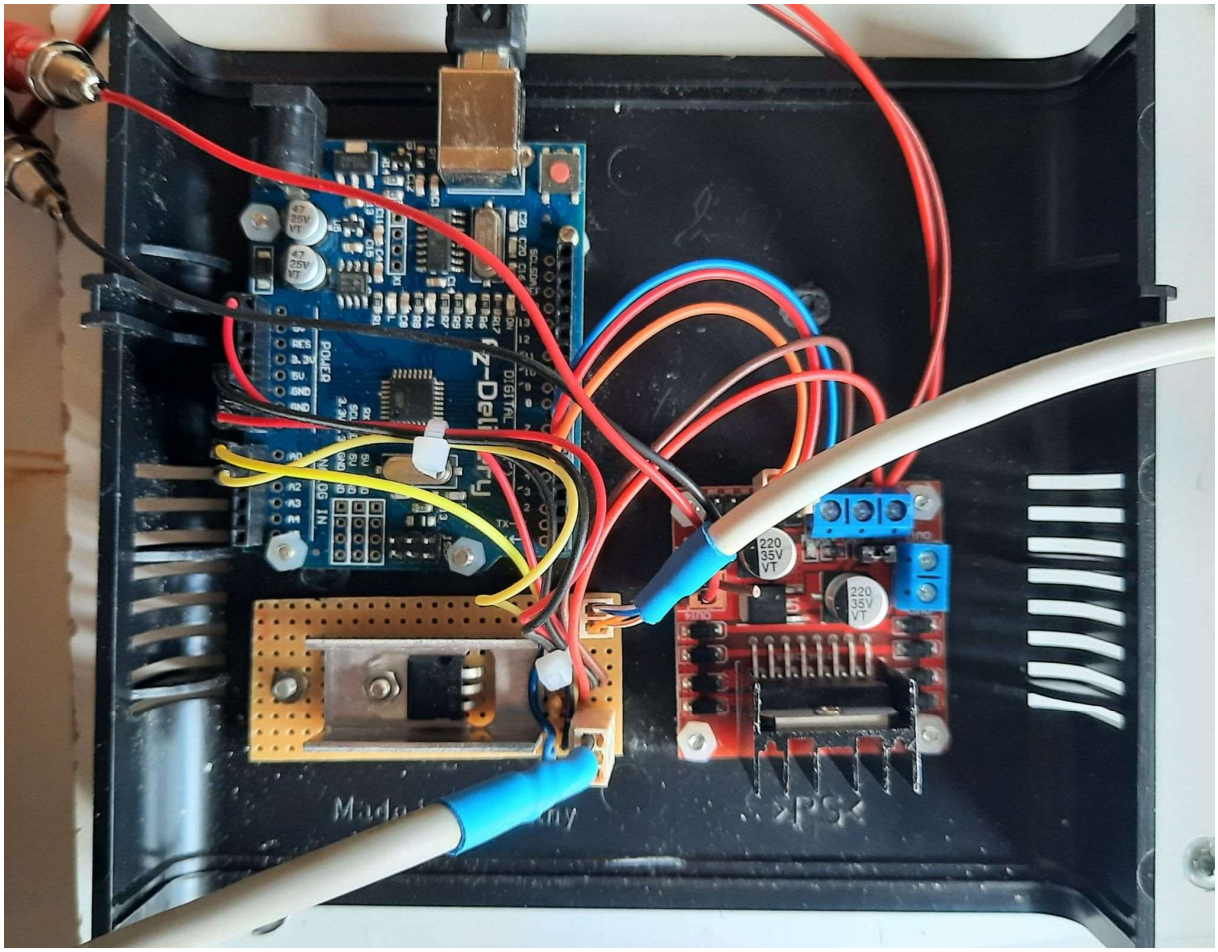
Weiteres Material

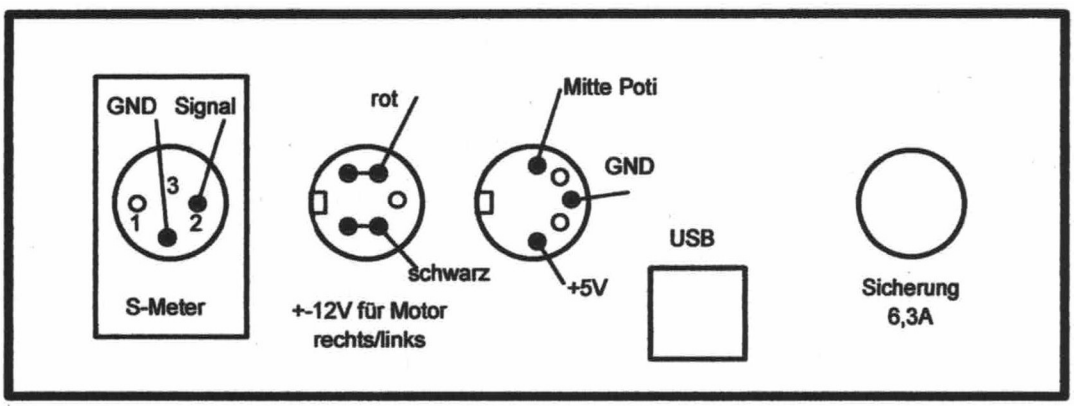


Die Distanzrollen aus Holz haben sich nicht bewährt. Bei Feuchtigkeit haben sie sich ausgedehnt und somit konnte das Mastrohr nicht mehr entfernt werden. Jetzt werde ich eine Alu Version nehmen, die mir Helmut anfertigt,

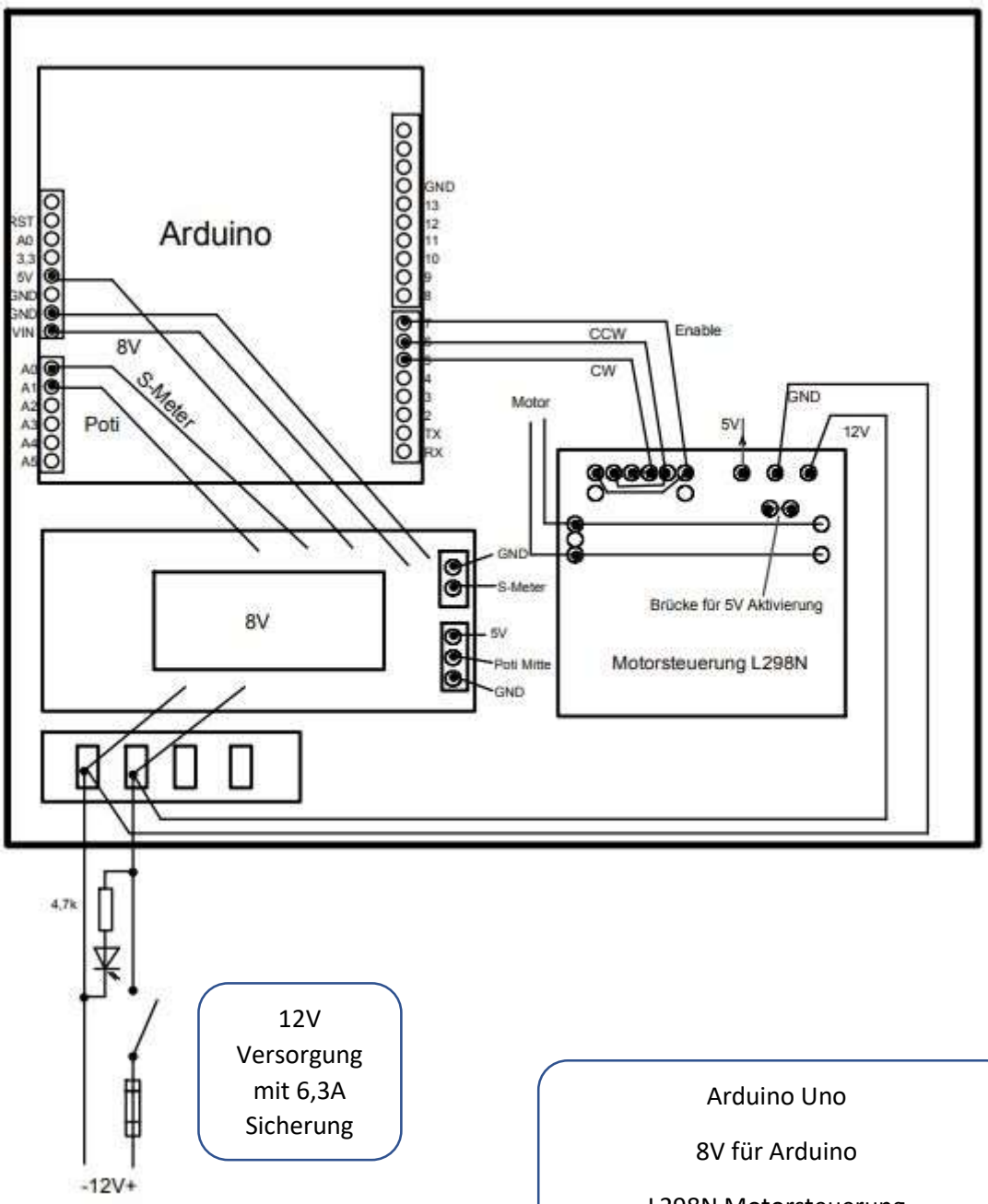
Steuerung

Bei den bisher gebauten Peilern war die Steuerelektronik im jeweiligen Peilergehäuse untergebracht. Diesmal wollte ich die Steuerung separat im Shack lassen, so dass zum Peiler nur zwei Kabel erforderlich sind, eins für die +-12V für den Motor und eins für das 10 Gang-Poti für die Richtungsanzeige. Die 12V werden von der Steuerung in der Polarität gewechselt; für Rechts- und Linkskauf.





Verdrahtung



12V Versorgung mit 6,3A Sicherung

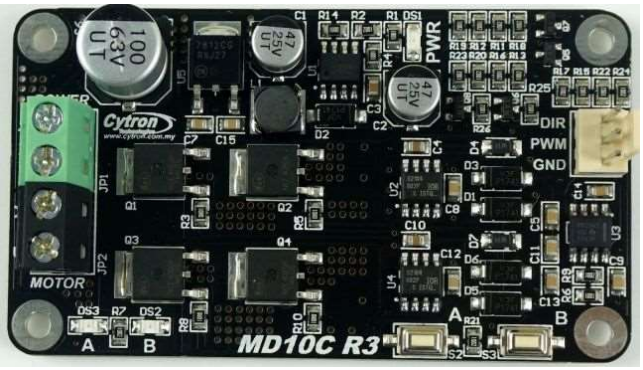
Arduino Uno

8V für Arduino

L298N Motorsteuerung

Probleme der Motorsteuerung

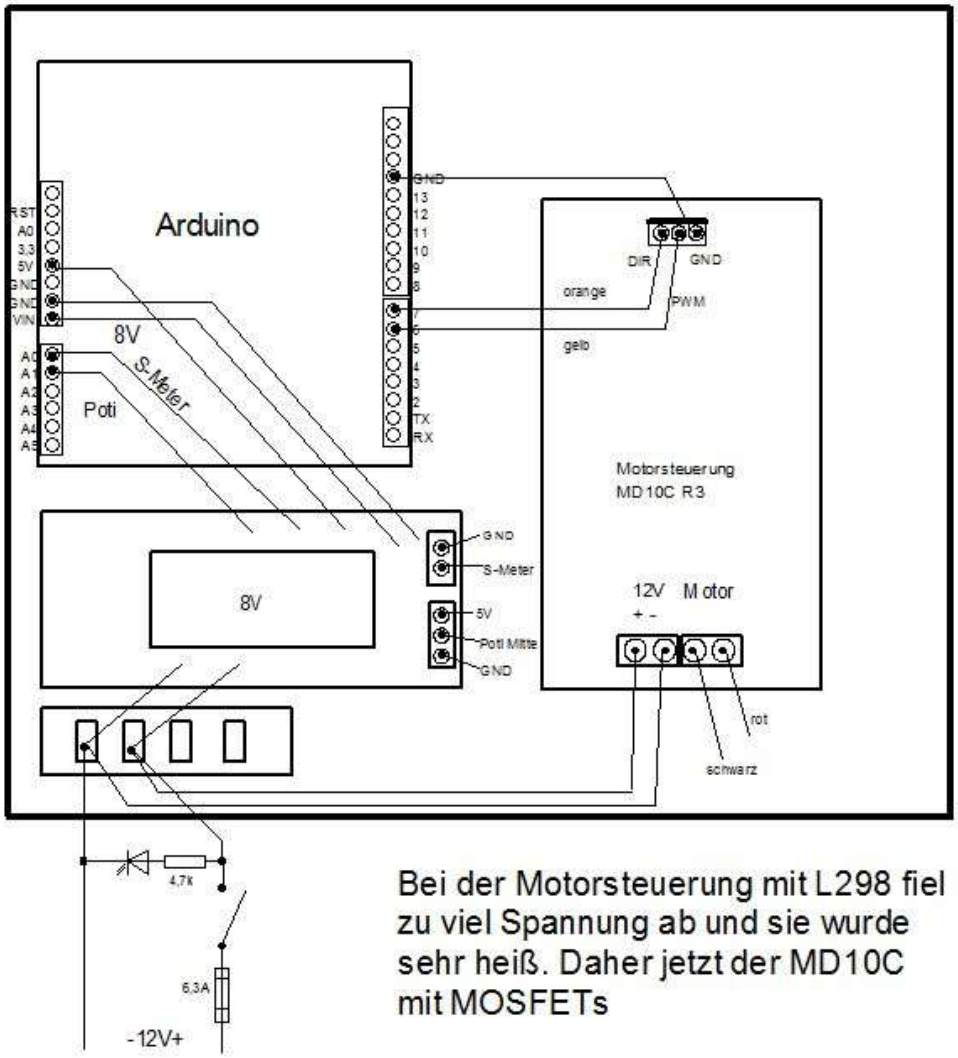
Die Steuerplatine mit dem L298 IC wurde bei maximaler Motorgeschwindigkeit sehr heiß. Es fiel eine Spannung von fast 4V ab, die auf der einen Seite am Motor fehlten und auf der anderen Seite Ursache für die Hitzeentwicklung war.



Hier ist nun die Lösung:

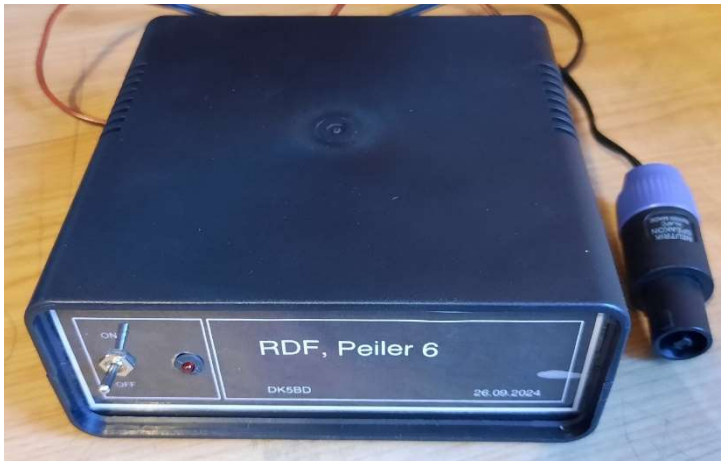
Mittlerweise gibt es aber günstige Motorsteuerungen, die Mit einer MOSFET H-Schaltung versehen sind, wie hier z. B. der MD10C.

Neue Verdrahtung

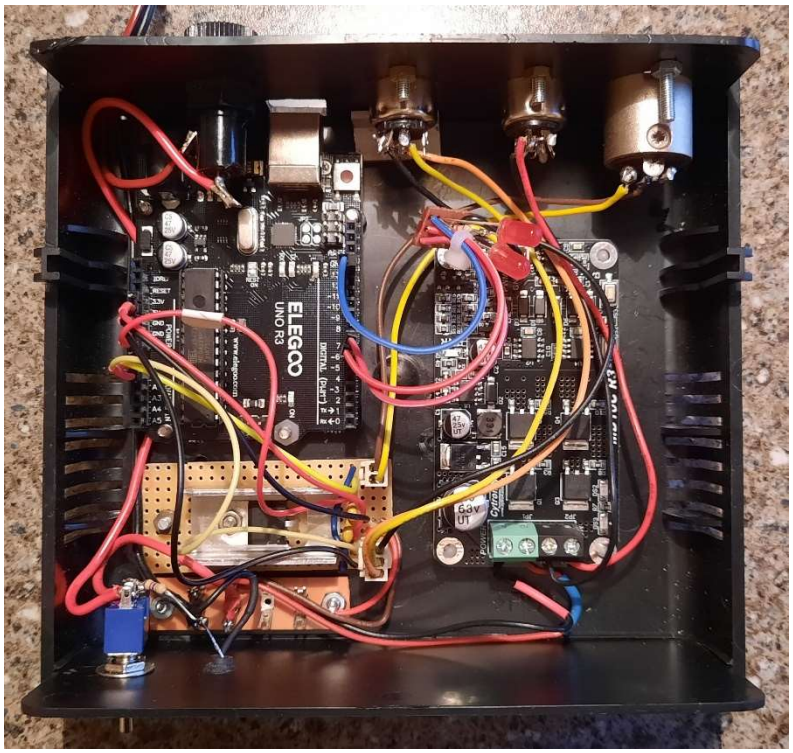


Bei der Motorsteuerung mit L298 fiel zu viel Spannung ab und sie wurde sehr heiß. Daher jetzt der MD10C mit MOSFETs

Steuergerät



Rückseite mit Buchsen für S-Meter, +-12V, 10 Gang-Poti, USB. Und 6,3A Sicherung



Neues Layout jetzt mit MD10C Steuerung und zwei Kontroll-LEDs an den Arduino Ports 7 und 6.

Software

Der Arduino wird über den USB Anschluß mit der allgemeinen Peiler Software Hunter.ino geladen, in der vorher im IDE der Peiler 5 (gilt auch für Peiler 6) eingestellt werden muß. Im Hunter.exe muß jetzt der Peiler 6 (hier nicht 5) ausgewählt werden. Die Stromversorgung für den Arduino soll im Betrieb nicht über die 5V aus dem USB Port erfolgen, weil sich bei Betrieb an unterschiedlichen Rechnern durch geringfügige Spannungsdifferenzen Richtungsdifferenzen ergeben können. Daher habe ich eine separate 8V Stromversorgung für den Arduino vorgesehen und die 5V für das Poti kommen jetzt vom Arduino. Die 5V Leitung im USB Kabel habe ich unterbrochen.

Jetzt läuft Peiler 6 an Jans komplexer Hunter.exe Software.

Stückliste

1	Bodenplatte	
1	Mittelplatte	
1	Topplatte	
3	Flanschlager	12mm
1	Gewindestab	12mm
6	Gewindestäbe	8mm
10	Muttern	M12
34	Muttern	M8
3	Scheiben, groß	12mm
3	Scheiben, klein	12mm
1	Endabschaltung mit 2 Mikroschaltern und 2 Dioden	
1	Motor mit 3 Schrauben M5	
1	Antriebswelle mit Stirnrad	
1	10 Gang Poti mit Stirnrad und Halter	
1	Hebel für Endabschaltung	
1	Winkel mit Buchsen für 12V und Potikabel	
2	Schrauben M5 mit Muttern	
1	Stirnrad, groß mit 2 M3 Schrauben zur Aktivierung der Enabschaltung	
2	Holzrollen zur Mastaufnahme	

Das oben genannte Material wurde in ein Plastikgehäuse montiert, das auf einer passenden Bodenplatte geschnallt wurde. Die Bohrung für die Mastdurchführung wurde mit einem Flansch versehen, damit der Peiler wassergeschützt ist.

Software für Arduino „Hunter.ino“

```
#include <Wire.h>

byte charFastLeft            =  60; // <
byte charSlowLeft            = 123; // {
byte charStepLeft            =  91; // [
byte charStop                = 124; // |
byte charStepRight           =  93; // ]
byte charSlowRight           = 125; // }
byte charFastRight           =  62; // >

byte charI2C                 =  73; // I
byte charIncreaseVolume      =  43; // +
byte charDecreaseVolume      =  45; // -
byte charAmplifier           =  65; // A

byte charSetup               =  83; // S

int hasReceivedSetup         = 0;
int directionFinderModel     = 3; // 0 for DJ4TA, 1 for Eimer, 2 for DF3AL,
3 for P5

int pinMotorEnable           = directionFinderModel == 2 ? 6 : 7;
int pinMotorClockwise        = 5;
int pinMotorCounterClockwise = directionFinderModel == 2 ? 7 : 6;

int pinSignalStrength        = A0;
int pinDirectionPoti         = A1;

int pinDirectionEnable       = 12;
int pinDirectionClock        = 10; // (has to be between 8 and 13 because
it uses direct access to PINB and PORTB - see https://is.gd/Kf85yW)
int pinDirectionData         = 11; // (has to be between 8 and 13)

int pinVolume                = 10;
int pinAmplifier             = 11;

int useAmplifier             = directionFinderModel == 2 ? 1 : 0;
int usePotiForDirection      = directionFinderModel == 0 ? 0 : 1;
int useEnablePinForPWM       = directionFinderModel == 2 ? 1 : 0;

int millisecondsForMotorStep = 40;
int slowSpeedPWMdutyCycle    = directionFinderModel == 3 ? 75 : 100;
int fastSpeedPWMdutyCycle    = directionFinderModel == 0 ? 255 :
(directionFinderModel == 3 ? 150 : 200);
int bitsInGrayCode           = 22;

int readThreeDigitIntFromSerial() {
  // Wait for four bytes to be available
  while (Serial.available() < 4)
    delay(1);

  Serial.read(); // ignore leading ;

  // Read the three bytes
  char buffer[4];
  buffer[0] = Serial.read();
  buffer[1] = Serial.read();
  buffer[2] = Serial.read();
  buffer[3] = '\0'; // Null-terminate the string
```

Hier auch für Peiler 6 den Peiler 5 auswählen, also die Nummer 3

```
  // Convert the ASCII digits to an integer
  int value = atoi(buffer);

  Serial.print("GOT:");
  Serial.println(value);

  return value;
}

void receiveSettings() {
  pinMotorEnable            = readThreeDigitIntFromSerial();
  pinMotorClockwise         = readThreeDigitIntFromSerial();
  pinMotorCounterClockwise  = readThreeDigitIntFromSerial();

  pinSignalStrength         = readThreeDigitIntFromSerial();
  pinDirectionPoti          = readThreeDigitIntFromSerial();

  pinDirectionEnable        = readThreeDigitIntFromSerial();
  pinDirectionClock         = readThreeDigitIntFromSerial();
  pinDirectionData          = readThreeDigitIntFromSerial();

  pinVolume                 = readThreeDigitIntFromSerial();
  pinAmplifier              = readThreeDigitIntFromSerial();

  useAmplifier              = readThreeDigitIntFromSerial();
  usePotiForDirection       = readThreeDigitIntFromSerial();
  useEnablePinForPWM        = readThreeDigitIntFromSerial();

  millisecondsForMotorStep  = readThreeDigitIntFromSerial();
  slowSpeedPWMdutyCycle     = readThreeDigitIntFromSerial();
  fastSpeedPWMdutyCycle     = readThreeDigitIntFromSerial();
  bitsInGrayCode            = readThreeDigitIntFromSerial();

  pinMode(pinMotorEnable, OUTPUT);
  pinMode(pinMotorClockwise, OUTPUT);
  pinMode(pinMotorCounterClockwise, OUTPUT);

  pinMode(pinDirectionEnable, OUTPUT);
  pinMode(pinDirectionClock, OUTPUT);

  if(useAmplifier == 1)
    pinMode(pinAmplifier, OUTPUT);

  digitalWrite(pinDirectionEnable, LOW);

  hasReceivedSetup = 1;
}

const int directionClockToLow  = (1 << (pinDirectionClock-8)) ^ B11111111;
const int directionClockToHigh =  1 << (pinDirectionClock-8);
const int isDirectionDataHigh  =  1 << (pinDirectionData -8);

unsigned long stopMotorAtMillis;
int signalStrength;
int directionPoti;
bool currentDirection[50];

void setup() {
  Wire.begin();
  Serial.begin(57600);
}
```

```
void loop() {
  if(hasReceivedSetup) {
    motorAndReceiverControl();
    readSignalStrength();
    if(usePotiForDirection == 0)
      readDirection();
    else
      readDirectionPoti();
    sendDatagram();
  } else {
    Serial.println("SETUP");
    if(Serial.available() > 0) {
      Serial.println("SERIAL");
      int inByte = Serial.read();
      if(inByte == charSetup) {
        Serial.println("RECEIVE");
        receiveSettings();
      } else {
        Serial.print("UNKNOWN:");
        Serial.println(inByte);
      }
    }
  }
}

void motorAndReceiverControl() {
  bool stopMotorCommandReceived = 0;
  if(Serial.available() > 0) {
    int inByte = Serial.read();
    if(inByte == charI2C) {
      while(Serial.available() < 2)
        delay(1);
      int address   = Serial.read();
      int byteCount = Serial.read();
      while(Serial.available() < byteCount)
        delay(1);

      Wire.beginTransmission(address);
      for(int iByte=0; iByte<byteCount; ++iByte)
        Wire.write(byte(Serial.read()));
      Wire.endTransmission();
    }
    if(inByte == charAmplifier) {
      while(Serial.available() < 1)
        delay(1);
      analogWrite(pinAmplifier, Serial.read());
    }
    if(inByte == charIncreaseVolume) {
      pinMode(pinVolume, OUTPUT);
      digitalWrite(pinVolume, HIGH);
      pinMode(pinVolume, INPUT);
    }
    if(inByte == charDecreaseVolume) {
      pinMode(pinVolume, OUTPUT);
      digitalWrite(pinVolume, LOW);
      pinMode(pinVolume, INPUT);
    }
    if(inByte == charFastLeft || inByte == charStepLeft || inByte ==
charSlowLeft) {
      digitalWrite(pinMotorClockwise, LOW);
      digitalWrite(useEnablePinForPWM == 1 ? pinMotorCounterClockwise :
pinMotorEnable, HIGH);
```

```
      analogWrite(useEnablePinForPWM == 1 ? pinMotorEnable :
pinMotorCounterClockwise, inByte == charFastLeft ? fastSpeedPWMdutyCycle :
slowSpeedPWMdutyCycle);
    }
    if(inByte == charFastRight || inByte == charStepRight || inByte ==
charSlowRight) {
      digitalWrite(useEnablePinForPWM == 1 ? pinMotorClockwise :
pinMotorEnable, HIGH);
      digitalWrite(pinMotorCounterClockwise, LOW);
      analogWrite(useEnablePinForPWM == 1 ? pinMotorEnable :
pinMotorClockwise, inByte == charFastRight ? fastSpeedPWMdutyCycle :
slowSpeedPWMdutyCycle);
    }
    if(inByte == charStop)
      stopMotorCommandReceived = 1;
    if(inByte == charStepLeft || inByte == charStepRight) {
      stopMotorAtMillis = millis() + millisecondsForMotorStep;
    }
  }
  if(stopMotorCommandReceived || stopMotorAtMillis > 0 && stopMotorAtMillis
< millis()) {
    digitalWrite(pinMotorClockwise, LOW);
    digitalWrite(pinMotorCounterClockwise, LOW);
    digitalWrite(pinMotorEnable, LOW);
    stopMotorAtMillis = 0;
  }
}

void readSignalStrength() {
  signalStrength = analogRead(pinSignalStrength);
}

void readDirectionPoti() {
  directionPoti = analogRead(pinDirectionPoti);
}

void readDirection() {
  /* inspired by
http://forum.arduino.cc/index.php?topic=47045.msg339531#msg339531 (2016-10-
05) */
  while(!digitalRead(pinDirectionData))
    delayMicroseconds(1);
  for(int i=0; i<bitsInGrayCode; ++i) {
    PORTB &= directionClockToLow;
    currentDirection[i]= PINB & isDirectionDataHigh ? 1 : 0;
    PORTB |= directionClockToHigh;
  }
}

void sendDatagram() {
  if(usePotiForDirection == 0) {
    for(int i=0; i<bitsInGrayCode; ++i)
      Serial.print(currentDirection[i]);
  } else {
    Serial.print(directionPoti);
  }
  Serial.print(";");
  Serial.println(signalStrength);
}
```