

Kompakter pulswitengesteuerter proportionaler Fahrtregler für DC Motoren:

Programmierbarer, mikrocontrollergesteuerter Fahrtregler mit hochauflösenden PWM-Ein- und -Ausgängen, Drehrichtungsprogrammierung und -umschaltung, Gaskurve, Störungsunterdrückung und Einschaltsicherung.

Von: **Marc Schneider**

Technische Daten:

Versorgung und Last:

Spannung: 5,5-20V

Max. Strom: 45A

Eingangssignal:

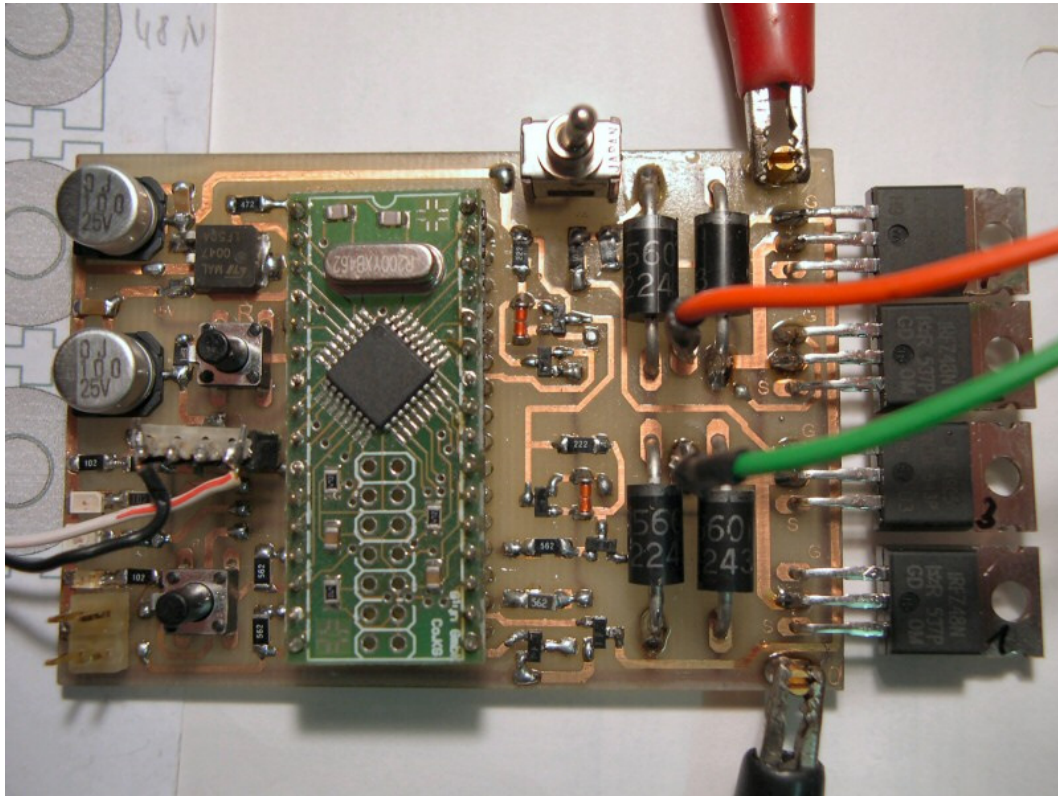
- pulsbreitenmoduliert, High-Pegel >3,3V
- dauerhaft geschützt gegen Überspannung
- Pulslängen-Messauflösung: ca. 1µs
(möglich durch Auswertung von Timer- und Prescaler-Register)
- maximale Pulslänge: ca. 65535µs

Ausgang:

- Pulsweitenmodulation: Frequenz 4kHz, Puls-Pausen-Verhältnis 0-100%,
Drehrichtungsumschaltung, 257 Stufen pro Drehrichtung
- Totband um Neutralstellung
- Status-LEDs

Features:

- geringer Innenwiderstand des Leistungsteils durch moderne FET Leistungsstufe
- kompakte Bauform mit geringer Anzahl externer Bauteile
- Steuerung durch R8C/13 Mikrocontroller
- Eingangspulslängen für Neutralstellung und Endanschläge kalibrierbar, Speicherung im internen FlashROM
- Kalibrierung ohne zusätzliche Hardware
- Keine mechanisch bewegten Teile am Fahrtregler
- Drehrichtungsumschaltung
- Verhältnis von Drehrichtung und Pulsweite invertierbar
- Bremsfunktion aktivierbar
- Pro Drehrichtung Drehzahl begrenzend konfigurierbar (im Controllerprogramm)
- sehr feinfühligste Steuerung durch hohe Ausgangspulsfrequenz und hohe Messauflösung des Eingangssignals
- Sicherheitsschaltung gegen Anlaufen bei Anschluss der Stromversorgung
- Signalfilterung gegen Störimpulse (intelligente Pulsbreitenbewertung mit History-Funktion, maximale Abweichung zu vorherigen Pulsmessungen in Prozent einstellbar im Controllerprogramm)
- Einfache Konfiguration der Gaskurven (im Controllerprogramm)
- Fail save bei Signalausfall konfigurierbar (im Controllerprogramm)



Beschreibung:

Bei diesem Projekt handelt es sich um einen pulswitengesteuerten proportionalen Fahrtregler für DC Motoren wie sie im allgemeinen Modellbau, in der experimentellen Robotik und auch im Automotive Bereich zum Einsatz kommen. Dabei wird die Stellgröße in Form eines pulswitengemodulierten Signals mit einem High-Pegel von ca. 5-6V vorgegeben. Zur Vereinfachung wird davon ausgegangen, dass eine Modellbauanwendung mit Funkfernsteuerung als Signalgeber vorliegt. Eine Anpassung auf andere Pulsbreiten ist dennoch sehr einfach möglich und erfordert nahezu keinen Programmieraufwand.

Bei Fernsteuerempfängern beträgt die positive Pulsbreite in der Regel zwischen 1ms und 2ms, die Pause ca. 20ms. Häufig ist dieses Signal aufgeteilt, so dass die Drehrichtung bei ca. 1,5ms umgeschaltet werden kann und so eine Pulsbreite von 1,5-2ms für die eine Drehrichtung und 1-1,5ms für die andere Drehrichtung genutzt wird (Bild 1). Der DC-Motor wird dann proportional zur Stellgröße im genannten Bereich mit einem pulsbreitenmodulierten (PWM) Signal zwischen 0% und 100% Pulsbreite versorgt. Damit wird erreicht, dass sich die Drehzahl entsprechend zwischen 0 und 100% verändert, wobei durch die hohe Pulsfrequenz von 3,9kHz der Rundlauf auch bei geringen Drehzahlen des Motors garantiert ist. Der Betriebszustand wird durch drei Leuchtdioden angezeigt. Sie geben Auskunft über die jeweilige Knüppelstellung Voll-Rückwärts, Neutral und Voll-Vorwärts.

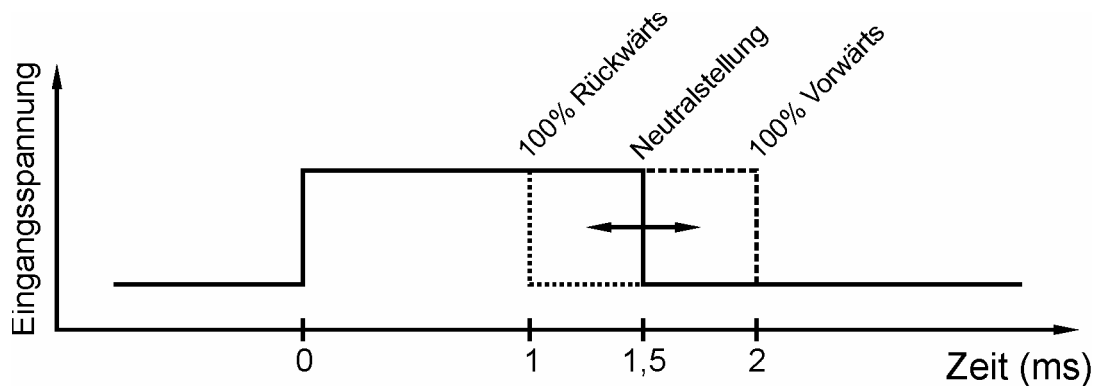


Bild 1: Eingangssignal des Fahrtreglers wie es idealerweise von einem Modellbau-Empfänger zur Verfügung gestellt wird (Ausschnitt).

Kommt der Fahrtregler in einer vorhandenen oder wechselnden Umgebung von anderen Systemkomponenten wie z.B. Fernsteueranlagen zum Einsatz, ist es erforderlich, die Reglergrenzen an das vorhandene System anzupassen, um den möglichen Stellbereich auch vollständig ausnutzen zu können. Dazu kann bei diesem Fahrtregler über einen Kalibriervorgang die Ruhstellung als auch die beiden 100% Grenzwerte der Zielumgebung direkt per Tastendruck am Regler und Vorgabe der entsprechenden Pulsweite z.B. vom Fernsteuersender aus, festgelegt werden. Dabei werden die LED's zur Anzeige der jeweils zu kalibrierenden Knüppelstellung genutzt. Anschließend zeigt ein nacheinander erfolgreiches Aufleuchten und Verlöschen der LEDs den Erfolg beim Abspeichern der Kalibrierung an. Bei Misserfolg leuchtet eine LED dauernd, abhängig vom aufgetretenen Fehler. Die genaue Abfolge zur Kalibrierung ist im Abschnitt Software näher erläutert.

Häufig kommt es im Aufbau vor, dass sich die Drehrichtung als falsch erweist, so dass dicke Hochstrom führende Leitungen zum Motor umgeklemmt oder gar umgelötet werden müssten, um die Drehrichtung zu korrigieren. Das ist hier nicht mehr erforderlich, da beim Kalibriervorgang auch die Drehrichtung mit gewählt werden kann. Natürlich kann dabei auch als 100% Grenzwert ein kleinerer Knüppelausschlag verwendet werden, so dass beispielsweise bereits nach 90% Knüppelausschlag Vollgas gegeben wird.

In stark störungsbehaftetem Umfeld z.B. bei großen Entfernungen zwischen Modell und Fernsteuersender kommt es häufig vor, dass das modulierte Signal der Stellgröße zeitweise nicht richtig empfangen werden kann und damit dem Fahrtregler zur Auswertung nicht korrekt zur Verfügung steht. Das führt bei Nichtbeachtung zu unkontrollierten Drehzahlwechseln am Motor, was neben erhöhtem Stromverbrauch und Bürstenverschleiß zu kritischem Flug- bzw. Fahrverhalten des Modells oder Roboters führen kann. Um derartige Störeinflüsse weitgehend zu unterdrücken, wird in diesem Fahrtregler eine intelligente Prüfung der Stellgröße durchgeführt. Erst wenn das Signal die Prüfung besteht, wird es zur Weiterverarbeitung zugelassen. Besteht es die Prüfung nicht, wird es aussortiert und der bisherige Wert der Stellgröße wird beibehalten, solange bis wieder aufeinander folgende, konsistente Signale empfangen werden. Fällt das Signal längere Zeit (ca. eine viertel Sekunde) vollständig aus, so tritt eine Fail Safe Schaltung in Kraft, die den Motor in einen definierten Zustand versetzt. Die gewählte Voreinstellung schaltet den Motor ab, es sind im Mikrocontrollerprogramm aber auch sehr einfach andere Verhaltensweisen z.B. eine vorkonfigurierte Motordrehzahl programmierbar.

Unter Umständen möchte der Benutzer eine asymmetrische Aufteilung zwischen den Drehzahlen für vorwärts und rückwärts wählen, schließlich parkt auch kein Autofahrer sein Fahrzeug mit voller Motorleistung rückwärts in eine Parklücke ein, dafür benötigt er aber eine erhöhte Auflösung um mit mehr Gefühl bei kleinen Drehzahlen gut positionieren zu können. Dafür bietet der hier vorgestellte Fahrtregler eine Funktion, für beide Drehrichtungen ge-

trennt, die maximale mittlere Ausgangsleistung auf einen Wert unterhalb von 100% zu limitieren, z.B. auf 50% und diesen Bereich auf den vollen Stellgrößenbereich zu spreizen.

Um die Neutralstellung bei 1,5ms Pulsbreite ist ein Todband definiert, in dem der Motor auch bei leichter Variation der empfangenen Pulsbreite abgeschaltet bleibt. Das Todband ist im Mikrocontrollerprogramm als Prozentwert mit 5% vom Vollausschlag voreingestellt und kann mit einem Parameter leicht an die jeweiligen Gegebenheiten angepasst werden.

Bei aufwändigen Modellen wollen Modellbauer häufig ein möglichst authentisches Flug-, bzw. Fahrverhalten erreichen. Um dies zu erreichen ist es für den Piloten hilfreich, so genannte Gaskurven konfigurieren zu können, die eine nicht lineare Abhängigkeit zwischen Knüppelstellung und Motorpulsweite ermöglichen. Z.B. kann so eine lineare Abhängigkeit zwischen Motorleistung und Knüppelstellung erreicht werden, in dem eine Quadratwurzelfunktion als Abhängigkeit definiert wird. Weitere Anwendungen erfordern ein sicheres Ansteuern einer Zwischendrehzahl, so dass eine Funktion mit lokal kleinerer Steigung wünschenswert wäre. Dieser Fahrtregler ermöglicht die freie Wahl von Gaskurven als Werteliste im Controllerprogramm, wobei zwischen den Stützwerten linear interpoliert wird. Der Nutzer kann durch einfache Anpassungen der Konfiguration beliebige Abhängigkeit zwischen Eingangs- und Ausgangsgröße wählen. Ein genaueres Verständnis des Programmablaufes ist zur Wahl der genannten Einstellmöglichkeiten nicht erforderlich. Zur Erstellung der Werteliste liegt ein Beispiel in Excel vor, das den Verlauf zeigt und die erforderlichen Werte errechnet.

Als weitere Option kann eine Motorbremse aktiviert werden. Eine Klappluftschraube z.B. klappt ohne Motorbremse nicht zusammen, da die Zentrifugalkraft der durch den Fahrtwind gedrehten Luftschraube stärker wirkt als der Luftwiderstand der diese einfallen soll um den Strömungswiderstand zu reduzieren. Die Motorbremse wird durch kurzschließen des Motors in der Neutralstellung realisiert.

Im Folgenden wird detaillierter auf die Hard- und Software des Fahrtreglers eingegangen.

Hardware:

Die Hardware besteht im Wesentlichen aus dem R8C/13 mit Spannungsversorgung, die auch gleich den Fernsteuerempfänger mit versorgt, und einer H-Brückenschaltung mit Power MOSFETs und zugehörigen Treiberstufen.

Verwendet wird das von Elektor gelieferte R8C/13 Modul der Firma Glyn, das in der notwendigen Grundbeschaltung für Mode-Eingang und Reset-Taster verwendet wird. Der Quarz wird hier nicht benötigt weil die internen Taktgeneratoren Verwendung finden. Zusätzlich ist ein Taster (S3) über einen Vorwiderstand an Port P16 (Pin 9) angeschlossen. Im Ruhezustand wird über den Widerstand R15 P16 auf Masse, gedrückt auf 5V gezogen. Damit lässt sich der Controller in den Programmiermodus für die Pulsbreitengrenzwerte bringen und die jeweiligen Knüppelstellungen übernehmen. Weiterhin sind die Status-LEDs LED1-LED3 direkt an die Ports P13-P15 angeschlossen.

Die Spannungsversorgung ist mit einem 5V Low-Drop-Spannungsregler realisiert, der auch die Empfängerstromversorgung mit übernehmen kann. Hier könnte auch ein Standard-Typ verwendet werden, ein Low-Drop-Typ bringt aber etwas mehr Spielraum beim Absinken der Akkuspannung.

Das Eingangssignal wird über den Vorwiderstand R16 an Pin 8, den Eingang von Timer X angelegt. Damit auch bei extern versorgtem Empfänger mit höherer Ausgangsspannung als 5V der Mikrocontroller nicht geschädigt wird, begrenzt eine Zener-Diode (D6) die Spannung auf 4,7V. Der minimale High-Pegel des Eingangssignals beträgt 3,3V, wobei eine gewisse Streuung bei den Mikrocontrollern zu beobachten ist.

Die H-Brücke für den Elektromotor besteht im Wesentlichen aus zwei p-Kanal- und zwei n-Kanal-Power-MOSFETs. Die p-Kanal-FETs Q3 und Q4 werden über einfache Transistor-

Treiberstufen angesteuert. Sie werden für die gewünschte Drehrichtung des Motors durchgeschaltet und erst wieder ausgeschaltet, wenn der Motor gestoppt wird oder die Drehrichtung umgeschaltet wird. Die Treibereingänge sind über Vorwiderstände mit den Ports P01 und P11 des Mikrocontrollers verbunden.

Die n-Kanal-FETs Q1 und Q2 werden dagegen über eine aufwändigere Treiberstufe versorgt, die auch im Elektor Halbleiterheft 2006 vorgestellt wurde. Die Treibereingänge sind mit den Timerausgängen Y und Z des Mikrocontrollers verbunden. Damit erfolgt die Pulsbreitenmodulation des Motorstroms. Diese Treiberschaltung ist notwendig, um trotz der recht großen Eingangskapazität der FETs steile Schaltflanken zu realisieren, so dass die Verlustleistung in den FETs gering bleibt.

Die Schalteigenschaften sind damit bei dieser Anwendung auch das Hauptauswahlkriterium für die FETs. Die Drain-Source-Widerstände im durchgeschalteten Zustand werden für den gewählten IRF4905 mit $20\text{m}\Omega$ und für den IRFZ48N mit $14\text{m}\Omega$ angegeben. Über einen noch niedrigeren On-Widerstand würde der n-Kanal-Power-MOSFET IRF3205Z verfügen ($6,5\text{m}\Omega$), allerdings ist dieser Typ schlechter erhältlich. Der IRFZ48N begrenzt auch den möglichen Dauerstrom auf 45-64A, je nach Temperatur. Das ergibt bei einer maximalen Versorgungsspannung von 20V immerhin mindestens 900W für den Motor. Wenn das nicht reicht können auch mehrere FETs parallel geschaltet werden. Bei hohen Motorleistungen sollte man den FETs allerdings einen Kühlkörper spendieren, der allerdings von den FETs elektrisch isoliert werden muss, da die Kühlfahnen jeweils mit Drain verbunden sind.

Die Schottky-Dioden D1-D4 dienen, zusätzlich zu den in den FETs integrierten Dioden, als Freilaufdioden, weil der Elektromotor eine induktive Last darstellt.

Natürlich dürfen nur jeweils diagonal gegenüberliegende FETs leiten, weil keinerlei Strombegrenzung in den Leistungspfad eingebaut ist. Das muss durch entsprechende Programmierung des Mikrocontrollers sichergestellt werden. Damit bei eigenen Experimenten sich nicht plötzlich die FETs in einem Rauchwölkchen verabschieden, gibt es die Möglichkeit, die Basen der oberen Treibertransistoren über Jumper auf Masse zu ziehen, so dass, unabhängig von den Portsignalen, die oberen FETs gesperrt bleiben. Nach ausführlicher Kontrolle mit einem Multimeter oder Oszilloskop kann man dann die Jumper abziehen und die Funktion mit einem Motor testen.

Grundsätzlich ist es aufgrund der hohen möglichen Motorströme zu vermeiden, diese über einen Schalter zu führen. Deshalb sieht diese Schaltung einen Schalter im Niederstromteil vor, der den Mikrocontroller und den angeschlossene Empfänger stromlos schaltet und die Treiberstufen für die unteren FETs spannungslos auf Nullpotential legt. Damit fällt deren Gatepotential ebenfalls auf Null und die FETs sperren. Der verbleibende Leckstrom ist mit jeweils ca. $25\mu\text{A}$ so gering, dass das nicht weiter stört, vor allem wenn man bedenkt, dass meist der verwendete Akku ohnehin zur Erhaltungsladung aus dem Modell entfernt wird.

Software:

Die eingesetzte Software für den R8C/13 wurde vollständig mit den von Renesas kostenlos zur Verfügung gestellten Entwicklungswerkzeugen in C geschrieben. Um die Konfiguration ohne Neuprogrammierung des Controllers einfach zu halten, wurde sie auf die Maximal- und Neutralstellungen der Knüppelpositionen beschränkt. Die weiteren Einstellungen, wie Totband um die Neutralstellung in Prozent, die maximale Pulsbreite für die jeweilige Drehrichtung, die Gaskurve, der maximale Pulsbreitenunterschied zwischen zwei Messungen um Störungen auszufiltern und auch die Voreinstellung für die maximal mögliche Pulslänge kann einfach in der globalen Variablendefinition des ausführlich kommentierten C-Programms eingestellt werden.

Das Hauptprogramm beginnt mit einer Initialisierungssequenz und gliedert sich dann in zwei Abschnitte, den normalen Fahrtreglerbetrieb und den so genannten Programmiermodus. In der Initialisierungssequenz werden die Taktfrequenz auf 8MHz eingestellt, die I/O-Richtung der Portpins definiert und das User-Flash und alle vier Timer des R8C/13 für den jeweiligen Betriebsmodus vorbereitet.

Timer X wird für die Pulsbreitenmessung des Eingangssignals konfiguriert. Bei einer steigenden Eingangsflanke wird der Zähler gestartet, bei einer fallenden Flanke gestoppt und ein Interrupt ausgelöst. Ebenfalls wird bei einem Zählerunterlauf, d.h. bei einem zu langen Puls, ein Interrupt ausgelöst. Ist dieser Fall eingetreten, wird bei einem nachfolgend ausgelösten Interrupt für das Zählende der Zähler wieder zurück auf seinen Ausgangswert gesetzt, sonst passiert nichts. Ansonsten wird geprüft, ob noch eine Bearbeitung der in einem früheren Interrupt gewonnenen Daten anhängig ist. Auch in diesem Fall wird nur der Zähler zurückgesetzt. Trat kein Unterlauf auf und ist auch die Bearbeitung vorheriger Daten abgeschlossen, wird der Zähler- und der Prescalerstand ausgelesen und in einer globalen Variablen gespeichert. Zusätzlich wird eine Flagge gesetzt, dass neue Daten zur Bearbeitung vorhanden sind und der Zähler wird zurückgesetzt.

Durch die Nutzung des Prescalerstandes kann sorgfältig und feinfühlig zwischen Auflösung und Messzeit des Zählers abgewägt werden. Eingestellt ist eine Messauflösung von 1µs. Daraus ergibt sich eine Auflösung von 1000 Stufen zwischen minimaler und maximaler Pulslänge eines Fernsteuerempfängers. Da die Timer-Register mit den maximal möglichen Werten von 255 geladen werden, ergibt sich eine maximal messbare Pulslänge von $65535\mu s = 65,535ms$, mit 1µs Auflösung, was den Fahrtregler sehr universell einsetzbar macht. Die Zeiten hängen direkt vom 8MHz-Ringoszillator des R8C ab, so dass sie nur so genau sind, wie der Oszillator. Allerdings sind die Ungenauigkeiten aufgrund der Kalibrierung (s.u.) unerheblich.

Timer Y und Timer Z sind als PWM-Generatoren mit einstellbarer Puls- und Pausenzeit programmiert.

Timer C dient der Betriebsüberwachung. Bei einem bestimmten Zählerstand wird ein Interrupt auslöst. In der Interruptroutine werden alle FETs gesperrt, so dass der Motor zum Stillstand kommt. Hier kann leicht ein anderes Verhalten wie z.B. z.B. 10% rückwärts, in der Hoffnung, dass das Modell (z.B. Auto oder Schiff) wieder in den Sendebereich der Fernsteuerung kommt, festgelegt werden. Damit im normalen Betrieb kein Interrupt ausgelöst wird, muss der Zähler immer wieder zurückgesetzt werden, was eine eigene Funktion an anderer Stelle erledigt.

Kalibrierung:

Nach den Initialisierungen wird geprüft, ob der Taster S3 an P16 gedrückt ist. Ist das der Fall, wird der Programmiermodus aktiviert. Danach leuchtet die LED an P13 als Indikator für die Knüppelstellung auf, die 100% Rückwärtsfahrt bedeuten soll. Dafür bringt man den Steuerknüppel der Fernsteuerung (oder entsprechendes) üblicherweise auf den unteren Anschlag, bei vertauschter Drehrichtung auf den oberen, und drückt kurz den Taster. Es werden jetzt 16 Messungen der Pulsbreite durchgeführt und gemittelt. Danach leuchtet die LED an P14 auf und der Controller wartet auf die Eingabe der Neutralstellung. Bringen Sie den Steuerknüppel in die entsprechende Position und betätigen wieder kurz den Taster. Wieder werden 16 Messwerte der Pulsbreite gemittelt. Dann leuchtet die LED an P15 auf. Bringen sie den Steuerknüppel in die Position für 100% Vorwärtsfahrt und drücken wieder kurz den Taster. Auch hier werden 16 Messwerte der Pulsbreite gemittelt. Jetzt sollten die gemittelten Messwerte für die entsprechenden Pulsbreiten ins FlashROM des R8C/13 geschrieben werden. Wenn alles geklappt hat, leuchten die drei LEDs kurz hintereinander auf und verlöschen wieder. Ist ein Fehler aufgetreten, leuchtet nur eine einzelne LED. Konnte das FlashROM nicht gelöscht werden, leuchtet die LED an P13, ist ein Fehler beim Schreiben des FlashROMs aufgetreten,

leuchtet die LED an P14 und wenn die gemessenen Pulsbreiten keine kontinuierliche steigende oder fallende Folge bilden, leuchtet die LED an P15. Im letzten Fall wurden die eventuell bereits im Flash vorhandenen Daten nicht verändert.

Diese Programmierung muss in jedem Fall bei der ersten Inbetriebnahme des Mikrocontrollers nach einer Übertragung des Hauptprogramms durchgeführt werden, weil das Flash Development Toolkit auch den Speicherbereich mit den Kalibrierungsdaten löscht.

Nach dem Programmieren resettet man den Controller. Weil jetzt die Taste an P16 nicht gedrückt ist, verzweigt das Hauptprogramm in den normalen Fahrtreglerbetrieb.

Dazu werden zunächst aus dem FlashROM die entsprechenden Pulsbreiten für Minimal-, Neutral- und Maximalstellung des Steuerknüppels ausgelesen. Je nach aufsteigender oder absteigender Reihenfolge wird eine entsprechende Flagge für normale oder invertierte Drehrichtung gesetzt, die für die spätere Weiterverarbeitung wichtig ist. Anschließend werden die entsprechenden Zeiten für das Todband um die Neutralstellung berechnet, gespeichert und in Abhängigkeit der Drehrichtungsflagge angepasst. Als nächstes wird Timer C gestartet und die Filterfunktion für das Ausfiltern von Störungen vorbereitet.

Damit bei ungünstiger Knüppelstellung der Motor jetzt nicht sofort losläuft, ist eine Sicherheitsfunktion eingebaut. Zunächst muss man den Steuerknüppel auf weniger als 5% der Maximalstellung bringen, anschließend auf mehr als 90% und dann wieder zurück auf weniger als 5%. Erst jetzt reagiert der Motor auf folgende Befehle. In der Praxis bedeutet das also, dass man den Steuerknüppel zunächst von der Neutralstellung auf Maximalstellung bringt und dann wieder loslässt. Damit das in beiden Drehrichtungskonfigurationen konsistent funktioniert, gibt es auch hierfür eine Fallunterscheidung.

Ab jetzt befindet sich der Controller in einer Endlosschleife, in der zunächst abgefragt wird, ob neue Messdaten der Pulsmessung anliegen. Wenn das der Fall ist, wird daraus eine neue Pulsbreite (PWMPower) für die Motoransteuerung berechnet. PWMPower ist dabei immer im Bereich zwischen -16383 und +16383, was sich allerdings mit der Variablen MaxPWMPower auch einstellen lässt. Der gegebene Wertebereich sollte für alle zu erwartenden Aufgaben ausreichen und ist auf jeden Fall größer als der eingestellte Messbereich von Timer X. Ist PWMPower größer als Null, wird der Strompfad von links oben nach rechts unten in der H-Brücke aktiviert, d.h. zunächst wird P11 auf Null gesetzt, damit der rechte obere FET sperrt, dann Timer Z ausgeschaltet und Timer Y auf die entsprechende Pulsbreite programmiert und zum Schluss P10 auf Eins gesetzt, damit der linke obere FET leitet. Ist PWMPower kleiner als Null wird das entsprechende mit dem Strompfad von rechts oben nach links unten und jeweils vertauschten Rollen von P10 und P11 bzw. Timer Y und Timer Z durchgeführt. Ist PWMPower gleich Null, werden alle FETs gesperrt. Hier besteht auch die Möglichkeit, durch Setzen einer Flagge im Controllerprogramm eine Bremsfunktion zu aktivieren. Dabei werden die beiden oberen FETs durchgeschaltet, um so eine Kurzschlussbremse für den Motor zu realisieren.

Im Folgenden werden die wesentlichen Routinen zur Berechnung von PWMPower aus den gemessenen Pulsbreiten und der Programmierung der Timer zur Ausgangssignalgenerierung beschrieben.

Die Berechnung von PWMPower geschieht in der Funktion CalcNewPWMPower(). Hier erfolgt die Verarbeitung zunächst in Abhängigkeit der Drehrichtungsvorgabe, normal oder invertiert. Erst werden je nach gemessener Pulsbreite die Leuchtdioden in einer dreistufigen Balkenanzeige angesteuert. Danach wird der Wert der gemessenen Pulsbreite bei Bedarf auf den programmierten Wertebereich beschnitten und um die Neutralstellung zentriert, so dass der Wertebereich jetzt vom negativen bis ins positive reicht. Da sicherlich nie exakt die Pulsbreite der Neutralstellung anliegt, wird hier zusätzlich ein Todband berechnet und der Wertebereich der zulässigen Messwerte auf den Standardbereich -MaxPWMPower ... +MaxPWMPower gespreizt. Dafür ist eine getrennte Behandlung von positiven und negativen

Werten notwendig. Die verwendeten Formeln unterscheiden sich aufgrund des Todbandes auch in Abhängigkeit des Drehrichtungsmodus, normal oder invertiert, wodurch hier die notwendige Unterscheidung zustande kommt. Die Abhängigkeiten sind in den Diagrammen für normale (Bild 2) und inverse Drehrichtung (Bild 3) dargestellt. Bei der Berechnung muss man besonders darauf achten, einerseits keine enormen Rundungsfehler durch eine falsche Berechnungsreihenfolge zu machen (Integerarithmetik mit Division!) und andererseits den verfügbaren Wertebereich der Variablen nicht zu überschreiten (Multiplikation). Daher sind im Programm explizite Typumwandlungen vorgegeben.

Nach der Berechnung der neuen Ausgangspulsbreite wird geprüft, ob sie im Vergleich zu den zwei vorherigen plausibel ist, indem geprüft wird, ob der neue Wert in einem bestimmten Bereich an erlaubten Werten liegt. Dieser Bereich wird mit der globalen Variablen MaxPWMPowerDifference eingestellt und ist auf ca. 15% des Maximalausschlags voreingestellt. Das heißt, dass sich der neue Wert für PWMPower nur um 15% vom vorherigen und 30% vom vorvorherigen Wert unterscheiden darf, sonst wird der bestehende Wert von PWMPower nicht verändert. Wird der erlaubte Bereich eingehalten, wird der neue Wert für PWMPower global gesetzt. In jedem Fall wird der neue Wert in einer Art Schieberegister gespeichert, in dem immer die beiden letzten PWMPower-Werte abgespeichert sind. Eine extrem schnelle Bewegung des Steuerknüppels (z.B. loslassen bei automatischer Rückstellung) führt also in jedem Fall nach drei Impulsen aus dem Fernsteuerempfänger, also in der Regel nach 60ms, zur gewünschten Einstellung.

Zum Schluss wird noch die Flagge zurückgesetzt, die angezeigt hat, dass neue Messwerte von Timer X vorliegen und zu bearbeiten sind.

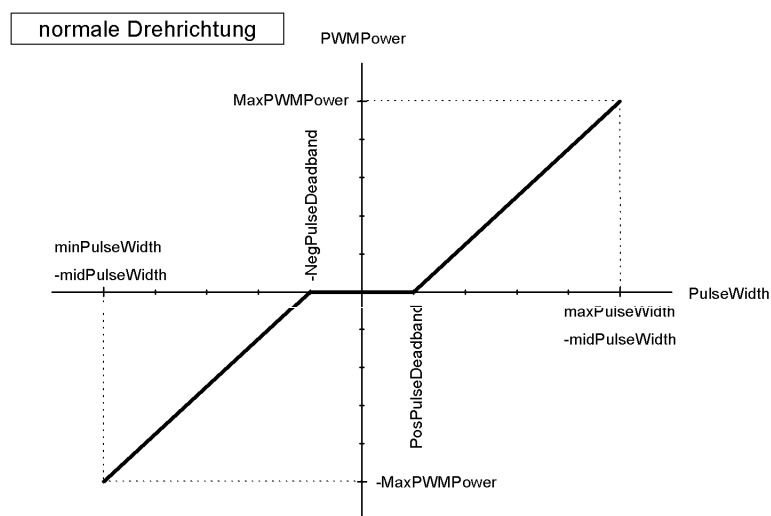


Bild 2: Umsetzung der gemessenen Pulsbreite auf einen Standardwertebereich mit Todband für normale Drehrichtung.

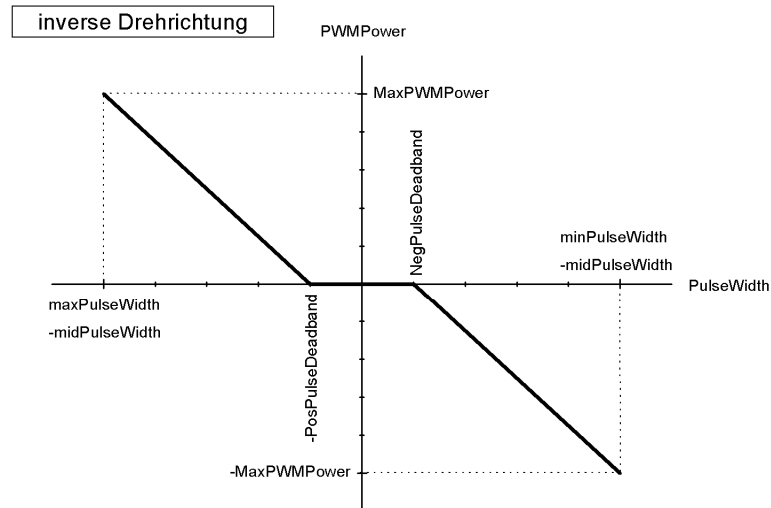


Bild 3: Umsetzung der gemessenen Pulsbreite auf einen Standardwertebereich mit Todband für inverse Drehrichtung.

Die Umsetzung von der Variablen PWMPower in ein pulsbreitenmoduliertes Ausgangssignal zur Ansteuerung des Motors erfolgt mit den Timern Y und Z in gleicher Weise in Timer_Y_Set bzw. Timer_Z_Set. Übergeben wird jeweils die gewünschte Pulsbreite aus dem Wertebereich -MaxPWMPower ...+MaxPWMPower. Dabei bedeutet eine negative Übergabe oder eine Null dass der Timer gestoppt wird. Er ist so programmiert, dass dann der Ausgang high wird. Die invertierende Treiberstufe sorgt dann dafür, dass der entsprechende FET sperrt. Ist die gewünschte Pulsbreite positiv, wird überprüft, ob die maximale Pulsbreite gewünscht ist. In diesem Fall wird der Ausgang als Portausgang konfiguriert und auf low geschaltet, so dass der entsprechende FET kontinuierlich leitet. Ist der Übergabewert an die Funktion positiv aber kleiner als der Maximalwert, wird berechnet, welches Puls-Pausenverhältnis notwendig ist. Dazu wird das Primary Register des jeweiligen Timers auf die Pulslänge und das Secondary Register auf die Pausenlänge programmiert, so dass immer eine Gesamtperiodenlänge von 257 entsteht (beide Register sind 8 Bit breit). Die Berechnung der Pulslänge für das Primary Register erfolgt in einer eigenen Funktion. Diese Funktion interpoliert aus vorgegebenen Tabellenwerten die gewünschte Pulsbreite, indem zwischen den dem gewünschten Wert benachbarten Stützwerten linear interpoliert wird. Durch Verändern der Tabelle können sehr einfach beliebige Funktionen realisiert werden. Voreingestellt ist eine wurzelförmige Abhängigkeit zwischen Eingangs- und Ausgangspulsbreite mit 33 Stützstellen. Damit ergibt sich eine nahezu lineare Abhängigkeit der Motorleistung von der Knüppelstellung. Mit der beiliegenden Excel-Tabelle lassen sich sehr einfach entsprechende Wertetabellen für andere funktionale Abhängigkeiten erstellen. Als Beispiel ist im Conrollerprogramm auch eine lineare Abhängigkeit als Kommentar angegeben.

Der Wert für das Secondary Register berechnet sich einfach, indem von 255 der Wert für das Primary Register subtrahiert wird. Zu bemerken ist hier, dass die tatsächlich ausgegebene Pulslänge um Eins größer ist, als der Wert in dem jeweiligen Register. Ausschließlich mit dem Timer lässt es sich also nicht realisieren, dass der FET kontinuierlich durchgeschaltet bleibt, weil bei einer Null im Secondary Register immer eine Pause von einem Zyklus eingefügt wird, daher ergibt sich eine Sonderbehandlung für Vollgas und eine ,krumme' Periodenlänge von $257 = (\text{Primary Register} + 1) + (\text{Secondary Register} + 1)$.

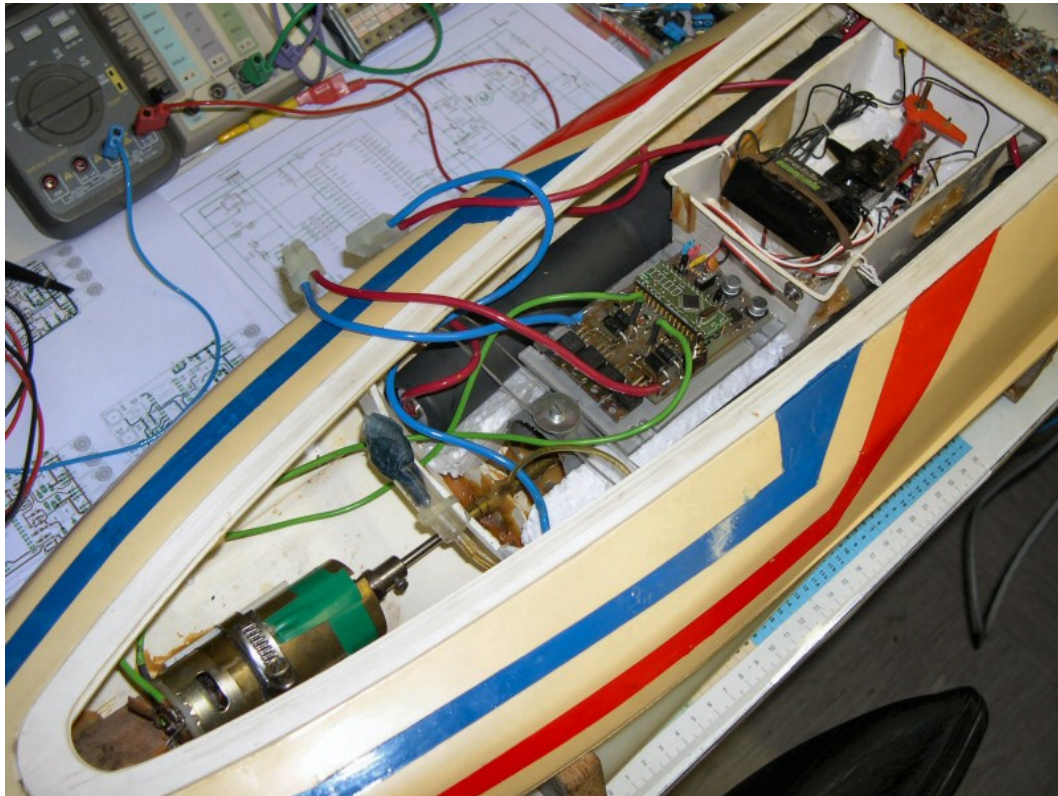


Bild 4: Fahrtregler in Modell-Rennboot eingebaut.

Servo-Stecker (z.B. Robbe)

