Projektdokumentation:

RC-Control – Fernsteuerung von funkgesteuerten Modellhelikoptern und -Autos

David Bertram

Wintersemester 2010 / 2011

EDV-Nr: 20556 (MI-Projekt)
Betreuer: Professor Walter Kriha

Studiengang Medieninformatik (Bachelor)
Hochschule der Medien Stuttgart

HOCHSCHULE DER MEDIEN

Inhaltsverzeichnis

1. Einleitung und Motivation

Dieses Kapitel beschreibt wie die Idee zu diesem Projekt entstand, welche Ziele damit erreicht werden sollten, sowie den geplanten Arbeitsaufwand. Ausserdem wird ein vergleichbares kommerzielles Produkt vorgestellt. Es soll dazu dienen einen Überblick über diese Dokumentation und deren Inhalt zu geben.

1.1. Idee

Die Idee zu diesem Projekt entstand zum einen aus persönlichem Interesse am Modellbau, bzw der dabei verwendeten Elektronik und wie diese mit meinen Programmierkenntnissen zusammengebracht werden kann. Zum anderen war es bereits seit langem ein "offenes" Thema, sich einmal etwas mit der Programmierung von Microcontrollern und deren Einbindung in bestehende Systeme auseinanderzusetzen. Die Idee, zu versuchen ein ferngesteuertes Modell mit Hilfe eines PC's zu steuern lag nahezu auf der Hand.
Nachdem die Idee geboren war und ein paar Recherchen dazu einige sehr interessante Arbeitstunden versprachen fing ich kurzerhand damit an mich tiefer in die Materie einzuarbeiten. Einen kleinen Modellhubschrauber besaß ich bereits und auch wenn das bloße Fliegen damit bereits keine anspruchslose Beschäftigung war, wollte ich gerne etwas mehr, als nur das fertig gekaufte Produkt einschalten und fliegen zu lassen. Also verschaffte ich mir einen Überblick darüber, welchen Umfang ein solches Projekt annehmen könnte und was tatsächlich realisierbar wäre.
Relativ bald schätzte ich den Zeitaufwand, den man damit verbringen kann, als vergleichbar zu dem, den ein Projekt für mein Medieninformatik-Studium haben sollte[1], ein.
Nun hieß es also mehr oder weniger nur noch einen Betreuer zu finden. Nachdem ich einige Professoren per E-Mail angeschrieben hatte, meldete sich Herr Professor Kriha und zeigte Interesse an einem Projekt das sich mit Microcontrollern beschäftigt. Dankenswerter Weise ließ er mir sehr großen Freiraum bei der Projektgestaltung und so konnte ich tatsächlich sehr viel interessante Zeit mit diesem Projekt verbringen und alle eigenen Ideen mehr oder weniger vertiefen.

[1] 240h laut Studien- und Prüfungsordnung SS2008, Medieninformatik Bachelor

1.2. Aufgabenstellung und Projektziele

Einen etwas tiefergehenden Überblick über ferngesteuerte Modelle möchte ich erst in Kapitel 2 geben. Zunächst werden an dieser Stelle die Projektziele, wie sie zu Beginn des Projektes besprochen und festgehalten wurden, sowie die Aufgabenstellung etwas abstrakter beschrieben, da das zu entwickelnde System auch mehr oder weniger unabhängig vom verwendeten Modelltyp sein sollte. Das Hauptziel des Projekts war es überhaupt einmal eine Möglichkeit zu schaffen, mit deren Hilfe ein PC, bzw dessen Benutzer, in der Lage ist ein ferngesteuertes Modell zu steuern. Das zweite, wie sich herausgestellt hat, bereits etwas ehrgeizigere Ziel war es, die Software so flexibel zu gestalten, dass ein beliebiges Eingabegerät für eine beliebige Fernbedienung verwendet werden kann. Desweiteren wollte ich versuchen eine Funkkamera auf einem Modell anzubringen um damit sozusagen eine Ego-Perspektive auf den PC-Monitor zu übertragen. Technische Details folgen in späteren Kapiteln.

1.2.1. Steuerungsmöglichkeit

Da ich bereits einen Modellhelikopter besaß, lag es natürlich nahe zunächst einen Joystick als Eingabegerät einzusetzen. Auch diesen besaß ich bereits. Für diese erste Funktion kam also ein Saitek Cyborg Evo Force zum Einsatz. Dieser Joystick verfügt über insgesamt 4 analoge Eingabekanäle, was genau den 4 analogen Achsen einer Standard Fernsteuerung für solche kleinen Helikopter entspricht. Man kann diesen mit solch einem Joystick mit einer Hand - 3 Achsen: Nick[2], Roll[3] und Yaw[4] - kontrollieren. Mit der anderen Hand steuert man den sogenannten Throttle-Regler[5]. Diese Achsen sind bei einer herkömmlichen Modellbaufernsteuerung auf zwei nebeneinander angeordnete Joysticks verteilt, die mit den Daumen beider Hände gesteuert werden. Wie diese beiden Sticks konkret belegt sind ist unterschiedlich. Für die genauen Belegungen haben sich die Bezeichnungen Mode 1 bis Mode 4 eingebürgert. Ohne auf diese weiter eingehen zu wollen, nur kurz der Hinweis, dass die meisten Modellhelikopter mit Mode 2 gesteuert werden. Hierbei steuert man mit dem rechten Stick die horizontale Lage des Modells und mit dem linken Stick die Geschwindigkeitder beiden Rotoren, was entweder ein Steigen bzw. Sinken oder eine Drehung um die vertikale Achse bewirkt. Da diese Aufteilung der Achsen relativ unnatürlich und mehr oder weniger willkürlich festgelegt ist, vermutete ich, dass die Steuerung mittels Joystick sich sehr viel natürlicher anfühlen würde.

Als erstes Ziel setzte ich mir also, eine Verbindung zwischen diesen beiden Eingabegeräten zu schaffen. Als Helikopter kam dabei ein sehr kleines und leichtes Modell zum Einsatz.

[2] Nach vorne bzw hinten kippen
[3] Zur Seite kippen
[4] Drehen um die Vertikal-Achse
[5] Schubkontrolle

Das Modell Solo Pro vom chinesischen Hersteller Nine Eagles wiegt flugbereit inklusive Akku unter 30g.

1.2.2. Einbinden beliebiger Eingabegeräte

Die nächste Stufe des Projekts sollte dann diese vorerst relativ starre Kopplung zwischen zwei Eingabegeräten auflösen und flexibel sowie konfigurierbar gestalten. War es im ersten Projektabschnitt also noch relativ einfach gewesen die Eingabewerte des Joysticks auf die vorher festgelegten Achsen der Fernsteuerung zu übertragen, so stellten sich im zweiten Teil des Projekts sehr viel höhere Anforderungen an diese Zuordnungen. Da es nicht weiter sinnvoll und einleuchtend ist dies am Beispiel des Joysticks zu verdeutlichen, habe ich dafür ein komplett anderes Eingabegerät ausgewählt. Hierfür wurde ein Lenkrad besorgt, dass per USB an den PC angeschlossen wird und zusätzlich über ein Gas- und ein Bremspedal verfügt.
Da es auch nicht weiter sinnvoll erscheint mit diesem Eingabegerät einen Helikopter steuern zu wollen wurde auch ein anderes Modell ausgewählt, ein ferngesteuertes Modellauto. Von großem Vorteil war dabei, dass ich für die Fernsteuerung des Helikopters einen weiteren Empfänger besaß. Das bedeutete ich konnte zunächst mit der gleichen Verbindung zur Fernsteuerung arbeiten und mich voll und ganz auf die Eingabeseite am PC kümmern. Obwohl ein Modellauto nur über zwei Kanäle angesteuert wird, stellte sich diese Aufgabe als relativ anspruchsvoll heraus und es galt mehrere Schwierigkeiten in den Griff zu bekommen. Eine Anforderung an diesen Projektabschnitt war es, die beiden Pedale gemeinsam auf die Throttle-Achse zu legen. Die Theorie dazu war noch relativ einfach gewesen, so hatte ich ursprünglich gedacht man könnte eventuell das Bremspedal einfach dominieren lassen und damit einfach die Maximalauslenkung des Throttle-Wertes begrenzen. Dabei hatte ich allerdings nicht bedacht, dass Modellautos auch rückwärtsfahren können (und entsprechend auch können sollen) funktionierte dieser erste Plan nicht ganz so einfach. Schließlich fand ich eine Lösung, die auch zufriedenstellend funktioniert. Auch hierzu folgen in späteren Kapiteln weitere Details.

1.2.3. Einbinden einer Funkkamera

Nachdem die ersten beiden Projektziele erreicht wurden, blieb noch die Idee eine Funkkamera zu integrieren und den Videostream in der Software anzuzeigen. Prinzipiell ist auch dieses Ziel erreicht. Mit der im Moment eingesetzten Kamera ist die Qualität des Videobildes allerdings unzureichend. Die Kamera lässt sich zwar sehr gut im Cockpit des Modellautos platzieren, die Funkverbindung ist jedoch nicht wirklich gut, und bereits leichte Richtungsänderungen[6] verursachen grobe Bildstörungen. Die Ursache dafür liegt mit an Sicherheit grenzender Wahrscheinlichkeit in der Qualität der Kamera. Geeignete Kameras sind ab ca 100€ zu haben, die aktuell verfügbare Kamera liegt in der Preisklasse bis 25€. Die Software erlaubt das Einbinden und Konfigurieren einer beliebigen Kamera. Für meine Versuche habe ich einen USB-Videograbber verwendet der ein Composite Video Signal aufnehmen kann. Das bedeutet man könnte den verwendeten Videoempfänger direkt durch einen besseren ersetzen und mit dem bereits getesteten Video-Grabber verwenden.

Da die Videoqualität nicht zufriedenstellend war, habe ich damit bisher keine weiteren Funktionen implementiert. Denkbar wären allerdings einige weitere interessante Möglichkeiten, natürlich vorrausgesetzt man hat ein stabiles, sauberes Bild. Auf diese und weitere mögliche Funktionen gehe ich in Kapitel 6.4 ein.
Versuchsweise habe ich am Funkempfänger der Steuerung einen weiteren Servomotor[7] an das Signal für die Lenkung angeschlossen. Das bedeutet sobald die Räder ausgelenkt wurden, wurde die Kamera entsprechend mitgedreht. Damit lies sich das Modell allerdings keineswegs gut steuern. Man kann es sich eventuell vorstellen als ob man beim Autofahren beim Drehen des Lenkrads automatisch den Kopf mitdrehen muss. Da die Kamera eine relativ geringe Brennweite hat ist der Blick zur Seite ohnehin eingeschränkt. Beim Lenken nach links oder rechts aus dem Fenster schauen zu müssen, macht das Dirigieren des Fahrzeugs an eine gewünschte Position sehr schwer. Der bisher eingesetzte Empfänger hat leider nur zwei nutzbare Ausgänge die bereits für die Steuerung des Modells benötigt werden. Ich hoffe relativ bald eine andere Fernsteuerung inklusive Empfänger zu bekommen. Damit sollten 2-4 Kanäle frei bleiben um weitere Servomotoren ansteuern zu können. Mit zwei weiteren Kanälen könnte man die Kamera auf zwei separaten Achsen steuern, eine dritte Achse für die Kamera wäre nicht wirklich nötig oder sinnvoll, da die dritte Achse das Kamerabild nach links und rechts zur Seite drehen könnte, als ob man den Kopf nach links oder rechts neigt. Ebenfalls keine unbedingt hilfreiche Bewegung, wenn man versucht ein Fahrzeug zu steuern.

[6] Richtungsänderungen des Modells wirken sich stark aus auf die Empfangsqualität aus
[7] Motoren die durch Anlegen eines bestimmten Signals die Achse auf eine bestimmte Position stellen

1.3. Projekt- und Zeitplan

Neben den bereits geschilderten Aufgaben und Ideen waren noch einige weitere Dinge zu bearbeiten. Zum Beispiel musste die Verbindung zur Fernsteuerung geschaffen werden. Dazu gehörte zum einen die elektronische Seite. Kabel, Stecker, Buchsen, Platinen und IC-Chips mussten verlötet und verbaut werden. Ausserdem musste auch der Microcontroller programmiert werden. Auch hierzu folgen die technischen Details an späterer Stelle. Hier soll zunächst nur der ungefähre zeitliche Ablauf der einzelnen Arbeiten beschrieben werden. Angefangen habe ich mit den allerersten Recherchen und Planungen bereits vor Semesterbeginn, den Großteil der Aufgaben konnte ich daher bereits in den ersten Wochen des Wintersemesters erledigen.
Relativ viel Zeit hat die Recherche zu Beginn gekostet. Entscheidungen die in dieser Phase getroffen wurden, betrafen zum Beispiel:

- den Microcontroller
- die Programmiersprache
- mögliche Ausweichlösungen
- den realisierbaren Gesamtumfang und die Projektabschnitte
- den mindestens geforderten Funktionsumfang

Der zweite Schritt bestand darin, den ersten Ansatz einer Verbindung zwischen Microcontroller und Fernsteuerung über den Trainerport[8] der Fernsteuerung zu schaffen, zu testen und zu analysieren weshalb dieser Versuch scheiterte.
Da dieser erste Ansatz leider nicht von Erfolg gekrönt war, nahm ich die Alternativoption in Angriff. Das bedeutete tiefer in die Elektronik der Fernsteuerung einzugreifen und die mechanischen Steuersignale auf digitalem bzw elektronischem Weg zu generieren. Die Recherche welche elektronischen Bauteile genau benötigt wurden und vor allem welcher Händler diese in geringen Stückzahlen anbietet nahm erstaunlich viel Zeit in Anspruch.
Nachdem die Bauteile besorgt waren mussten diese natürlich verbaut werden. Diese Aufgabe bereitete an sich keine größeren Probleme. Die Datenblätter gaben nach etwas Rätselraten und Recherche ihre Geheimnisse preis. Nachdem die Schaltung entworfen und zusammengelötet war funktionierte sie einwandfrei und es konnte endlich ans Programmieren gehen.
Die Entwicklung eines allerersten Prototyps, noch ohne Joystick ging recht schnell von der Hand. Diesen schließlich einzubinden dauerte auch nicht übermäßig lange. Der nächste Schritt erforderte allerdings wiederum einige Denkarbeit.
Anschließend war nun der Entwurf einer halbwegs übersichtlichen und geeigneten Grafischen Oberfläche, sowie der weiter entwickelten Programmkomponenten an der Reihe. Während die Oberfläche und das Programm selbst relativ schnell entworfen waren, bereitete mir das bereits angesprochene Problem mit dem Rückwärtsfahren und die flexible Gestaltung der Zuordnung von Eingabe- auf Ausgabeachsen, etwas mehr Kopfzerbrechen. Ausserdem stellte sich heraus, dass der mechanische Speed-Controller[9] des zu diesem Zeitpunkt eingesetzten Modellautos nur 3 Geschwindigkeitsstufen ermöglichte. Also wurde auch dieses Bauteil durch einen elektronischen Speed-Controller ersetzt, der nun eine stufenlose Geschwindigkeitsregelung erlaubt.

[8] viele Fernsteuerungen erlauben es heutzutage, dass ein "Lehrer" und ein "Schüler" ihre Fernsteuerungen zusammenschalten, und somit der Lehrer Fehlsteuerungen des Schülers mit korrigierten Steuerbefehlen "überschreiben" kann. Mehr dazu in Kapitel 2.3.1

[9] Modellautos realisierten die Geschwindigkeitsregelung in den letzten Jahrzehnten durch Widerstände die den nicht benötigten Strom einfach in Hitze umwandelten, mittlerweile wird dies auf elektronischem Weg gelöst. Die veralteten Bauteile wurden MSC genannt, die moderne Variante heisst ESC

Nachdem schließlich auch diese Aufgabe erledigt war, stand noch das Einbinden der Videofunktion auf dem Plan. Auch dieser Teil konnte softwareseitig ohne größere Probleme umgesetzt werden und es folgten einige Versuche mit dem entwickelten System.
Da mittlerweile relativ viel Arbeitsaufwand und Ergebnisse erarbeitet waren, entschied ich mich dazu, an dieser Stelle einen (vorläufigen) Schlussstrich zu ziehen und die gesammelten Informationen, Daten und Erkenntnisse für diese Dokumentation aufzubereiten. Auch diese abschließenden Aufgaben nahmen ihre Zeit in Anspruch und waren von Anfang an Bestandteil der Zeitplanung. Alles in allem wurden bis zu diesem Zeitpunkt bereits 150 bis 200 Zeitstunden in das Projekt investiert. Da die Dokumentation und auch die Vorbereitung für den Präsentationstag[10] sowie der MediaNight[11] ebenfalls einiges an Zeit brauchen würden, denke ich der Zeitrahmen ist gut gefüllt und die Projektplanung kann als erfolgreich umgesetzt betrachtet werden.

1.4. Gliederung dieser Dokumentation

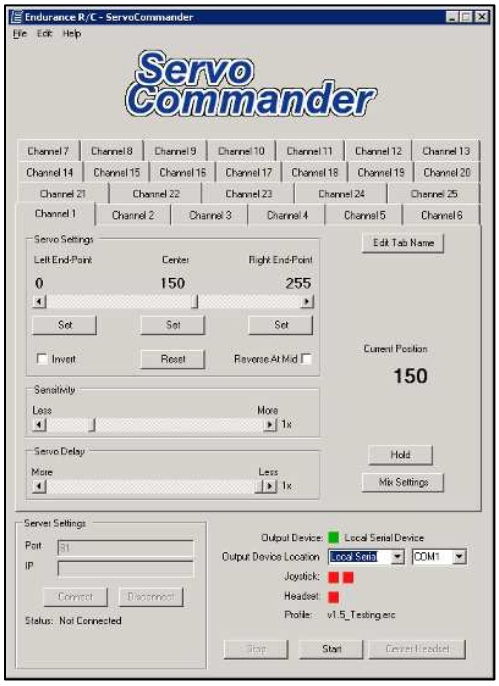
Diese Dokumentation besteht im Folgenden noch aus einem Punkt zum einleitenden ersten Kapitel der ein ähnliches System beschreibt das den Weg in den Modellbauhandel gefunden hat und von Endkunden mit Erfolg eingesetzt werden kann.
Es folgen 6 weitere Kapitel, die die technischen Details des Projekts sowie die Konkrete Umsetzung detaillierter beschreiben, eine persönliche Einschätzung und Ausblicke, sowie weitere Hintergrundinformationen bieten.
Kapitel 2 gibt einen Einblick in die verwendete Modellbautechnik und erläutert wie die Fernsteuerung der Modelle funktioniert.
In Kapitel 3 gehe ich auf den verwendeten Microcontroller und dessen Möglichkeiten sowie deren Anwendung und Programmierung ein.
Eine Gesamtübersicht und detaillierte Betrachtungen zum entwickelten System finden sich in Kapitel 4.
Kapitel 5 stellt die Entwicklung der Software dar, welche Werkzeuge eingesetzt und welche Libraries verwendet wurden. Ausserdem werden die verschiedenen Entwicklungsstufen des Prototyps dargestellt und beschrieben.
In Kapitel 6 folgt neben einer Zusammenfassung des Systems und dessen aktuellen Funktionsumfangs, sowie einer Reflektion des Projekts und einem persönliches Fazit, einige weitere Ideen, die bisher leider nicht verwirklicht werden konnten und dennoch nicht unerwähnt bleiben sollen.
Zum Schluss enthält Kapitel 7 schließlich einen Teil der Datenblätter, Informationsquellen, Schaltpläne und Entwürfe, die sonst keinen Platz in dieser Dokumentation gefunden haben, der Vollständigkeit halber aber dazu gehören.

[10] Studiengangsinterne Veranstaltung, bei der die Projekte eines Semesters vorgestellt werden
[11] Öffentliche Veranstaltung der Hochschule, die einem breiteren Publikum Einblick in die Arbeiten der Studierenden gibt

1.5. Ein vergleichbares kommerzielles System

Auch andere hatten bereits die Idee, ihre ferngesteuerten Modelle vom PC aus zu lenken. Eine US-Amerikanische Firma, die erfolgreich ein Produkt für diesen Einsatzzweck vertreibt, ist Endurance R/C[12]. Die Firma bietet seit 2005 verschiedene Produkte für den Modellbau- und Robotikbereich an.

Das Produkt zur Fernsteuerung von Modellen mit einem PC, nennt sich PCTx – PC to Transmitter Interface. Es besteht aus einem Microcontroller, der
auf der einen Seite mittels USB-Verbindung an einen PC und auf der anderen Seite an den Trainerport einer Fernsteuerung angeschlossen wird. Durch eine Vielzahl verschiedener Adapter wird versucht eine möglichst breite Palette von Fernsteuerungen zu unterstützen. Es gibt Versionen die eine Auflösung von 1024 Steuerungsstufen erlauben. Ausserdem sind eine API für dieses Produkt sowie Beispielprogramme verfügbar. Ergänzend zu diesem Produkt wird eine Software namens ServoCommander zum Kauf angeboten die es erlaubt beliebige Eingangskanäle auf verschiedene Ausgangskanäle zu mappen. Die Software und das Gerät wird ab einem Preis von 149,99 US$ angeboten. Es gibt weitere Varianten bei denen ein Joystick und ein Funksystem (399,99 US$) bis hin zu einem kompletten System inklusive einer Videobrille mit Head-Tracking Unterstützung (2499,99US$) mitgeliefert werden.



Den Funktionsumfang dieser Software habe ich erst nachdem ich mit meinem System bereits fast fertig war genauer angeschaut. Ich denke die von Endurance R/C angebotene Software hat ein sehr ausgereiftes Niveau erreicht und zeugt von einiger Erfahrung mit der Materie. Der Preis ist allerdings auch recht stolz, wird aber offenbar von der Kundschaft für solch ein System akzeptiert.

Eingesetzt wird bei diesem System der Anschluss an den Trainerport der Fernsteuerungen. Ob dieses System mit meiner Fernbedienung funktionieren würde ist nicht ganz sicher[13]. Ausprobieren konnte ich es leider nicht, ich bin nur bei der Recherche darauf gestoßen und halte die Lösung, abgesehn vom Preis, für durchaus interessant wenn die vorhandene Fernsteuerung das entsprechende PPM-Signal verarbeiten kann.

[12] Adresse und Website des Herstellers im Anhang
[13] Mehr darüber in Kapitel 2.3.1

2. Ferngesteuerte Modelle

Dieses Kapitel soll einen Überblick über die verschiedenen verwendeten Modellarten geben. Ausserdem wird die Technik, die zur Fernsteuerung eingesetzt wird, beschrieben und verschiedene Ansätze, wie man einen PC in diese Systeme einklinken kann, betrachtet. Die gesamte Modellbaubranche verbaut immer mehr zusätzliche Elektronik die dem Modell erlauben selbstständige Korrekturen und Stabilisierungen vorzunehmen. Vor allem die Modellhelikopter setzen Gyroskope ein um das Drehmoment der Hauptrotoren auszugleichen und für ein stabiles Flugverhalten zu sorgen. In der Anfangszeit mussten nahezu sämtliche Korrekturen vom Piloten manuell durchgeführt werden was sehr viel Erfahrung und Kenntnis des Flugverhaltens erfordert.
Neben den hier beschriebenen Elektromodellen gibt es bei sämtlichen Modelltypen, Helikoptern, Flugzeugen, Fahrzeugen, Schiffen, etc. auch Varianten die mit Verbrennungsmotoren angetrieben werden. Für dieses Projekt wurden ausschließlich elektrisch angetriebene Modelle eingesetzt.

2.1.1. Helikopter

Für den ersten Projektteil wurde ein Mini-Modellhelikopter eingesetzt. Modellhelikopter gibt es seit Anfang der 70er Jahre. Damals waren allerdings die technischen Bauteile noch weit entfernt von denen die heute verwendet werden. So waren die Akkus bzw. Batterien damals noch deutlich größer und schwerer als sie es heute sind. Ebenso waren die Motoren damals bei weitem nicht so leistungsfähig wie heutige Modelle. Vor allem diese beiden Punkte, aber ebenso die Funktechnik setzte dem Modellbau damals noch enge Schranken, und es gab nur wenige teure Modelle die in Serie produziert wurden und noch relativ groß waren.
In den letzten Jahren wurden beispielsweise die Bürstenmotoren[14] durch kontaktlose Brushless-Motoren[15] ersetzt. Die neuen Motoren bringen deutliche Leistungssteigerungen bei weniger Verbrauch und im Vergleich zu den herkömmlichen Modellen vernachlässigbarem Verschleiss. Dies und die Entwicklung von Akkus die mittlerweile bereits bei einem Gewicht von nur wenigen Gramm eine relativ große Energie speichern können, haben in den letzten Jahren sogenannte Indoor-Modellhelikopter sehr populär gemacht.
Bei den heute verfügbaren Modellhelikoptern gibt es eine große Palette von verschiedenen Techniken. So gibt es 2-Kanal Modelle, die nur die Höhe und die Richtung steuern können und meistens nur als Spielzeug betrachtet werden, da sie relativ ungenau zu steuern sind und zum Beispiel immer in einer leichten Vorwärtsbewegung sind. Ebenso gibt es 3-Kanal Modelle die zusätzlich zur Höhe und Richtung steuern können ob vorwärts oder rückwärts geflogen werden soll. Auch diese Modelle sind eher noch im Spielzeugbereich anzusiedeln.
Bei den 4-Kanal Modellen wird es bereits interessanter, diese können nun zusätzlich noch nach links und rechts zur Seite bewegt werden. Damit besitzen diese Modelle nahezu die vollen Bewegungsfreiheit wie ein richtiger Helikopter.
Bei den 3- und 4-Kanal Modellen gibt es jeweils Doppelrotorvarianten[16]. Diese haben durch physikalische Effekte bedingt ein deutlich stabileres Flugbild, sind allerdings auch ziemlich träge und reagieren daher immer etwas verzögert. Auf diese Unterschiede kann ich im

[14] Herkömmliche Elektromotoren mit Kohlestiften (Schleifbürsten)
[15] Drehstrom-Motoren
[16] Helikopter mit zwei koaxial angeordneten Hauptrotoren

Rahmen dieser Dokumentation nicht vertieft eingehen, im Anhang finden sich allerdings Linkverweise auf weitere Quellen zu diesen Themen.
Neben den beschriebenen Modellen gibt es noch Modelle mit 5 und mehr Kanälen. Ein fünfter Kanal erlaubt zum Beispiel eine Pitch-Kontrolle[17]. Das bedeutet durch Verstellen des Winkels der Blätter des Hauptrotors kann gesteuert werden wieviel Auf- oder Abtrieb der Helikopter erfährt. Bei Modellen ohne diese Funktion wird diese nur durch die Rotationsgeschwindigkeit des Hauptrotors erreicht. Weitere Kanäle werden meist für Zusatzfunktionen wie ein einfahrbares Fahrwerk oder ähnlichem verwendet.
Im Detail gibt es mittlerweile sehr viele verschiedene Varianten, das für dieses Projekt verwendete Modell verfügt über 4-Kanäle und kann demnach relativ frei bewegt werden.

2.1.2. Modellautos

Auch bei den Modellautos gibt es ähnlich viele Varianten und Möglichkeiten. Modellautos haben ebenso wie die Modellhelikopter von der Weiterentwicklung der Technik profitiert. So gibt es mittlerweile eine riesige Palette von Modellen. Es gibt welche, die so leicht sind, dass sie Wände hochfahren können in dem sie sich daran "festsaugen". Andere sind riesig und können richtige Lasten befördern. Im weitesten Sinne könnte man wohl auch Mars- und Mondrover als Modellauto bezeichnen. Ebenso gibt es Systeme die zur Bomben- und Minenentschärfung eingesetzt werden und über Greifarme und weitere Werkzeuge verfügen.
Im Hobbybereich gibt es Modelle die echten Fahrzeugen ins Detail nachgebaut werden und oft sogar von den Herstellern der Originalfahrzeuge mit Lizenzen ausgestattet werden. Manche Modellautos erreichen Geschwindigkeiten von über 70 km/h und erfordern entsprechende Erfahrung bei der Steuerung. Andere Modelle versuchen die echten Funktionen abbzubilden, zum Beispiel Baustellenfahrzeuge wie Bagger, LKWs jeglicher Bauart. Die gesamte Palette zu beschreiben ist ein relativ hoffnungsloses Unterfangen, ein großer Bereich sollte jedoch noch erwähnt werden. Dabei handelt es sich um Modelle die bei Wettbewerben, zum Beispiel Rennen eingesetzt werden. Oft existiert ein Regelwerk für diese Modelle das den Funktionsumfang bis hin zur verwendeten Technik vorschreibt.
Das hier verwendete Modell ist einem Monstertruck[18] nachempfunden, ein Tamiya – Big Foot. Es handelt sich um ein Modell das vor 10 bis 20 Jahren relativ weit verbreitet war und durchaus auch im Gelände gefahren werden kann. In der ursprünglichen Version wird die Geschwindigkeit dieses Modell durch einen Speed-Controller mit nur 3 Stufen gesteuert. Da die nutzbaren Geschwindigkeitsstufen für den Einsatz in diesem Projekt viel zu schnell und selbst bei niedriger Geschwindigkeit mit sehr hohem Energieverbrauch verbunden waren, wurde der Speed-Controller durch eine stufenlose modernere Variante ersetzt. Damit ist es nun möglich, die Maximalgeschwindigkeit zu begrenzen. Bei voller Geschwindigkeit kann dieses Modell durchaus Geschwindigkeiten von mindestens 20 bis 30 km/h erreichen. Für den Einsatz im Projekt sind zunächst allerdings Geschwindigkeiten von wenigen km/h mehr als ausreichend.

[17] Blattstellung des Hauptrotors
[18] Geländefahrzeug mit übergroßen Reifen

2.2. Prinzipien der Fernsteuerung

Die meisten Modelle setzen in der Regel auf eine Fernsteuerungsanlage die auf Funktechnik basiert und oft Reichweiten von mehreren 100 m und sogar bis zu wenigen Kilometern. In den letzten Jahren werden vermehrt auch Spielzeugmodelle verkauft, die mittels Infratrotsignalen kontrolliert werden. Für das Projekt wurde bisher nur Funktechnik verwendet, die Möglichkeit auch Infrarotsignale zu verwenden besteht allerdings und wird in Kapitel 2.3.3 kurz betrachtet betrachtet.

2.2.1. 27-, 35- und 40 MHz

Für den Modellbau gibt es bestimmte Frequenzbereiche[19], die für diesen Zweck reserviert wurden und ohne Registrierung oder Lizenzierung benutzt werden dürfen. Diese Frequenzbereiche sind jeweils nochmals untergliedert, und bestimmten Modelltypen zugewiesen. So gibt es beispielsweise für Flugmodelle eigene Frequenzbereiche. Die Frequenzbereiche sind nicht unbedingt international übereinstimmend, es gilt im Zweifelsfall also vor in Betriebnahme eines Modells zu klären ob der Betrieb am jeweiligen Standort erlaubt ist.
In den letzten Jahrzehnten wurden in Deutschland Frequenzen im Bereich von 27, 35 und 40 MHz für den Modellbau verwendet. Meist wurde dabei eine einfache Amplitudenmodulation oder Frequenzmodulation eingesetzt. Es haben sich mehrere Verfahren entwickelt, die sich allerdings im Zweifelsfall gegenseitig stören können. Daher wurden diese Frequenzen in feinere Bänder unterteilt, die mittels austauschbaren Quarzen auf Sender- und Empfängerseite ausgewählt werden. Damit wurde der Betrieb von mehreren Modellen zur gleichen Zeit, am gleichen Ort ermöglicht. Die Gefahr das die Funkverbindung gestört wird besteht bei diesen System allerdings fast jederzeit. Das hat zur Folge dass vor allem bei Flugmodellen jederzeit damit gerechnet werden muss die Kontrolle über das Modell zu verlieren, was häufig den Crash und teilweise oder komplette Zerstörung des Modells zur Folge hat. Ausserdem können weitere Schäden verursacht werden, daher ist es meist erforderlich sich vor in Betriebnahme von Flugmodellen (bzw Modellen überhaupt) um eine entsprechende Versicherung zu kümmern.
Bei meinen Versuchen am Campus Vaihingen hat sich gezeigt, dass dort sehr viel Störstrahlung vorhanden ist und das Modell bereits nach wenigen Metern kaum noch zu kontrollieren war.
Um dieses Problem in den Griff zu bekommen wurden in den letzten Jahren neue Techniken auf digitaler Basis entwickelt.

[19] Genaue Angaben über die nutzbaren Frequenzbereiche findet man über die Bundesnetzagentur sowie diversen Internetseiten über RC-Modellbau. Dabei muss darauf geachtet werden, dass die Angaben aktuell und regional gültig sind.

2.2.2. 2,4-GHz

Die Lösung des Problems der Funkstörungen bei der herkömmlichen Technik lässt sich durch Verwenden eines Digitalen Verfahrens in den Griff kriegen. Dabei muss allerdings vor der Inbetriebnahme eines Modells, dieses an die jeweilige Fernsteuerung "gebunden" werden. Das bedeutet, das Modell und die Fernsteuerung handeln eine gemeinsame ID aus, mit der jeder Steuerbefehl versehen wird. Damit wird relativ zuverlässig verhindert, dass ein Modell auf Steuersignale einer fremden Fernsteuerung reagiert. Diese Technik wird im 2,4 GHz Bereich eingesetzt, dem gleichen Frequenzbereich in dem auch herkömmliche WLans arbeiten. Es werden zum Teil herstellerabhängige proprietäre Verfahren angewandt. Einen richtigen Standard gibt es nicht wirklich, viele Hersteller von Funksystemen versuchen jedoch möglichst kompatibel zu anderen Systemen zu sein, vor allem teurere Fernsteuerungen erlauben hier mehr Flexibilität. Um möglichst hohe Störungresistenz zu erreichen werden unter anderem Frequency-Hopping[20] und Spectrum Spreading (Frequenzspreizung) verwendet. Ein relativ weit verbreitetes Verfahren nennt sich DSM und wurde von der Firma Spektrum entwickelt.
Bei diesem Projekt wurde ebenfalls eine 2,4 GHz Funkanlage verwendet, da diese Technik auch in Gebieten mit vielen Funksignalen den weitgehend störungsfreien Betrieb erlaubt.

2.2.3. Infrarot

Neben Funksystemen gibt es Modelle die durch Infrarotsignale gesteuert werden. Dies funktioniert ähnlich wie eine Fernbedienung für zum Beispiel TV-Geräte. Meist erlauben diese Modelle die Auswahl von 3 verschiedenen Sendekanälen und ermöglichen damit den gleichzeitigen Betrieb von mehreren Modellen gleichzeitig. Allerdings hat diese Technik gravierende Nachteile im Vergleich zur Funktechnik. Es muss eine ständige Sichtverbindung zum Modell existieren, wenn ein Modell also hinter einem Möbelstück verschwindet fällt die Steuerung aus. Ausserdem sind diese Modelle bei Tageslicht meist nicht verwendbar, da bereits Sonnenlicht einen hohen Infrarotanteil enthält und sehr große Störungen verursacht. Neben diesen Problemen, haben diese Modelle eine sehr begrenzte Reichweite von meist nur einigen Metern.
Eingesetzt wird diese Technik hauptsächlich bei sehr einfachen Spielzeugmodellen, der unteren Preisklassen.

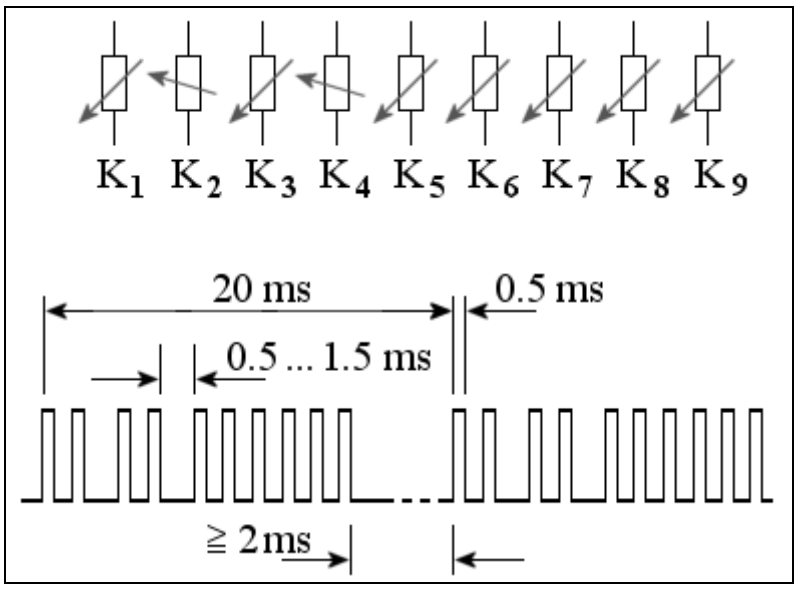
[20] Wechseln der Frequenz um lokale und kurzfristige Störungen zu minimieren.

2.3. Inputmöglichkeiten der Fernsteuerungen

Dieses Kapitel beschreibt verschiedene Möglichkeiten, wie der PC an die Fernsteuerungen angeschlossen werden kann. Zum einen gibt es die Möglichkeit sich als Trainer zu einer Schülerfernsteuerung zu verbinden, zum anderen kann man in die Elektronik eingreifen und die eigentlich mechanisch erzeugten Eingaben auf elektrischem Weg zu erzeugen. Eine weitere Möglichkeit besteht bei Infrarotmodellen in dem man die Signale direkt mit Hilfe von Infrarot-LEDs generiert.

2.3.1. Trainerport / PPM-Signal

Die prinzipiell einfachste Möglichkeit die auch von dem, in Kapitel 1.5 vorgestellten System, eingesetzt wird besteht darin, ein Trainersignal in die Fernsteuerung einzuspeisen. Sehr viele Hersteller verwenden dafür ein PPM-Signal, wobei PPM hier für Puls-Pausen-Modulation steht. Es existieren auch hier Unterschiede darin, wie verschiedene Hersteller diese Modulation realisieren. So wird das Signal manchmal invertiert oder die Pulslängen leicht verändert. Für manche Fälle gibt es Konverter, die zum Beispiel ein Signal invertieren können.



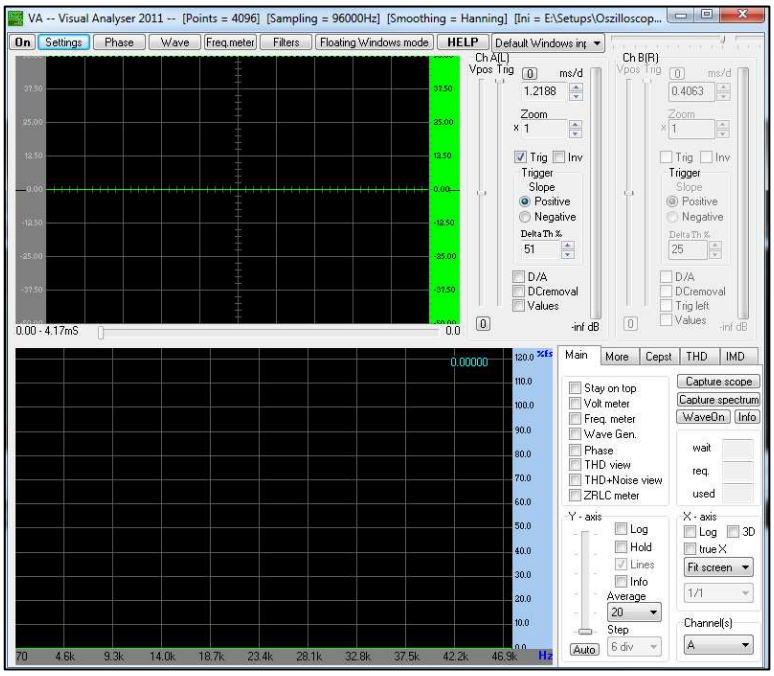
Diese Abbildung zeigt schematisch den Verlauf eines PPM-Signals, das 9 Kanäle beschreibt. Die "positiven" Impulse sind jeweils die Begrenzungen von einem Kanalwert zum nächsten. Diese Impulse sind bei einem Standard PPM-Signal 0,5 ms lang. Die Kanalwerte werden durch einen "negativen" bzw Null-Impuls variabler Länge moduliert. Die Symbole über dem Signalverlauf veranschaulichen die einzelnen Werte der Kanäle.
Die einzelnen Impulse werden in einen "Frame" konstanter Länge eingebettet. Das bedeutet nach dem der letzte Kanal übertragen wurde folgt ein Synchronimpuls ebenfalls variabler Länge. Die Framelänge ist so gewählt, dass der Synchronimpuls immer mindestens die Länge 2 ms hat und kann damit immer eindeutig von den anderen Impulsen unterschieden werden.

2.3.1.1. Experimente mit vorhandener Fernsteuerung

Ein solches PPM-Signal kann von einem Microcontroller ohne weiteres erzeugt werden. Dafür gibt es mindestens zwei Ansätze. Erstens kann man einen naiven Ansatz wählen, der vernachlässigt, dass alle durchgeführten Operationen auf dem Microcontroller selbst eine bestimmte Zeit benötigen. Dies lässt sich allerdings optimieren, indem man die Dauer der nötigen Operationen experimentell bestimmt und entsprechend einplant. Die meisten Microcontroller können einzelne Ausgänge digital ansteuern und damit das gewünschte Signal generieren. Man muss nur darauf achten, dass die Spannungspegel denen der Fernsteuerung entsprechen.
Ein anderer Ansatz nutzt Interrupts des Microcontrollers. Dabei wird ein Timer im Microcontroller so programmiert, dass er zu bestimmten Zeiten auslöst und die Signalgenerierung startet. Die konkrete Programmierung ist vor allem davon abhängig welche Fähigkeiten der verwendete Microcontroller besitzt.

2.3.1.2. Analysemöglichkeiten mit Soundkarten-Oszilloskop

In der Regel funktioniert die Erzeugung eines solchen PPM-Signal nicht auf Anhieb 100% korrekt mit einer beliebigen Fernsteuerung. Daher benötigt man meistens eine Analysemöglichkeit um das erzeugte Signal darstellen zu können. Ausserdem kann man meistens die Fernsteuerung selbst für eine Analyse des benötigten Signals im Trainermodus verwenden und sich das benötigte Signal darstellen.
Hierfür kann man entweder teuere Oszilloskope verwenden, die allerdings meistens mehrere 100 € kosten, oder man nutzt zum Beispiel eine Soundkarte am PC. Das funktioniert, da eine Soundkarte meistens eine Mikrofoneingang hat, der ein angelegtes analoges Signal digitalisieren und dem PC zur Verfügung stellen kann.
Für dieses Projekt habe ich mehrere solcher Software-Oszilloskope getestet. Es gibt einige frei verfügbare Tools. Als sehr brauchbar hat sich das Programm "Visual Analyzer" von Sillanumsoft erwiesen. Man sollte jedoch vor Einsatz dieser Methode sicherstellen, dass entweder die Soundkarte über entsprechende Schutzmechanismen vor Überspannungen und Kurzschlüssen verfügt, oder vorher die Pegel entsprechend anpassen. In meinem Fall hat jedoch ein einfaches Kabel mit entsprechenden Kontaktstiften auf der einen und einem Stereo-Klinkenstecker auf der anderen Seite ausgereicht.
Solche Programme bieten vielfältigste Analysemöglichkeiten, von Frequenzanlysen bis hin zu komplexeren Spektralanalysen. Für dieses Projekt reichte es allerdings aus ein angelegtes Signal auf einer Zeitachse zu visualisieren. Nebenstehende Abbildung zeigt die Oberfläche des Visual Analyzers.



2.3.1.3. Ausschluss dieser Möglichkeit

Leider konnte trotz sehr vielen Versuchen und verschiedenen Ansätzen keine funktionierende Lösung gefunden werden. Vergleiche zwischen dem benötigten und dem erzeugten Signal ließen am Ende keinerlei Unterschiede mehr erkennen.
Jedoch hat sich im Verlauf der Signalanalyse herausgestellt, dass diese Fernsteuerung keinesfalls ein Standardkonformes PPM-Signal verwendet. Zum Beispiel ist die Framelänge bei diesem Modell nicht konstant, sondern stattdessen die Länge des Synchronimpulses.
Nachdem das generierte Signal daran angepasst war, reagierte der Helikopter immerhin auf das angelegte Signal, allerdings nur durch wildes Zucken. Eine kontrollierte Steuerung war auf keinen Fall möglich.
Weitere Analysen, bei denen ich mit Hilfe des Microcontrollers die Impulse und Framelängen des benötigten Signals untersuchte, zeigten, dass einzelne Frames immer wieder länger als andere waren. Nachdem auch diese periodischen Schwankungen in die Signalerzeugung eingebaut waren und immer noch keine stabile Ansteuerung möglich war, und auch in Internetforen keiner mehr Rat wusste, entschied ich mich für die von vornherein eingeplante Alternativlösung.

2.3.1.4. eventuell zweiter Versuch mit anderer Fernsteuerung

Da das Misslingen der Ansteuerung über den Trainerport von mir auf ein nicht Standardkonformes PPM-Signal zurückgeführt wird, bleibt weiterhin die Möglichkeit offen, bei Einsatz einer anderen Fernsteuerung diese Möglichkeit noch einmal aufzugreifen. Die grundsätzliche Möglichkeit solch ein Ansteuerung besonders mit dem verwendeten Microcontroller besteht, es gibt mehrere Berichte und Beschreibungen von erfolgreichen Projekten. Diese verwendeten allerdings andere Fernsteuerungen.

2.3.2. Ersetzen der mechanischen durch digitale Potentiometer

Nachdem der erste Versuch also leider nicht erfolgreich war, kam die Alternative zum Einsatz. Die Sticks von Fernsteuerungen sind in der Regel an einfache mechanische Potentiometer angeschlossen. Das sind elektronische Bauteile mit 3 Kontakten, bei denen zwischen den beiden äusseren Kontakten ein Widerstand eingebaut ist. Der mittlere Kontakt ist üblicherweise an einen Schleifer an diesen Widerstand angeschlossen und erlaubt es einen variablen Widerstand abzugreifen. Damit werden verschiedene Schaltungen ermöglicht. Im Falle der Fernsteuerungen sind die Potentiometer als Spannungsteiler[21] eingesetzt.

Zu diesen mechanischen Potentiometern gibt es mittlerweile eine Alternative die durch einen IC-Chip[22] realisiert wird und Digitalpotentiometer genannt wird. Diese Chips können mittels eines bestimmten Protokolls angesprochen werden. Genaueres zur Beschaltung und Ansteuerung des verwendeten Digitalpotentiometers folgt in Kapitel 4.5.1. Es gibt Chips die bis zu 8 mechanische Potentiometer ersetzen können. Für dieses Projekt wurde eines mit 4 Potentiometern eingesetzt.
Diese Lösung funktioniert sehr gut und ist grundsätzlich für fast jede beliebige Fernsteuerung anwendbar, allerdings mit gewissem Bastelaufwand verbunden.

[21] Einfache Schaltung, Bestandteil elektrotechnischer Grundlagen
[22] Integrierter Schaltkreis, Mikrochip

2.3.3. Infrarot gesteuerte Modelle

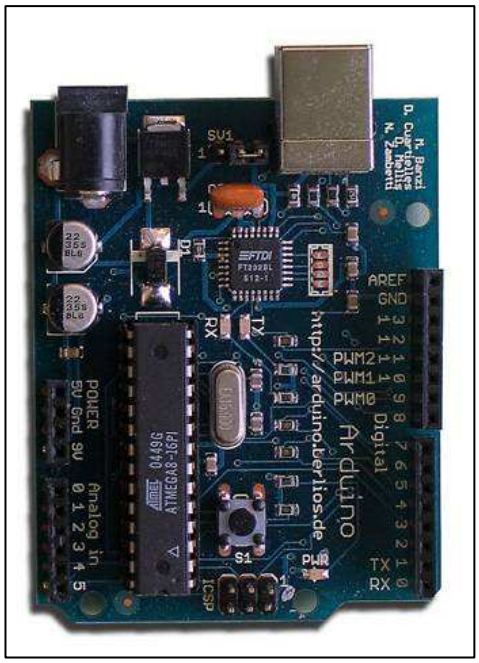
Einen gänzlich anderen Ansatz kann man für die Infrarotgesteuerten Modelle wählen. Während es bei Funksystemen in der Regel am einfachsten ist, eine bestehende Fernsteuerung für die Übertragung der Signale einzusetzen, kann man bei Infrarot-Modellen darauf verzichten und die Infrarot-Signale direkt mit dem Microcontroller erzeugen. Man benötigt hierfür nur entsprechende Infrarot-LEDs passendener Wellenlänge.
Ausser den LEDs benötigt man allerdings noch das Protokoll mit dem das zu steuernde Modell angesprochen wird. Teilweise finden sich Protokollbeschreibungen im Internet. Ansonsten kann man die jeweils benötigten Signale auch mit einem Microcontroller und Empfangsdioden sowie der originalen Infrarotfernsteuerung selbst auslesen.
Diese Möglichkeit wurde im Rahmen des Projekts allerdings nicht angewandt, da kein Infrarot-Modell benutzt wurde. Die spätere Einbindung dieser Möglichkeit dürfte allerdings keinen allzu großen Aufwand bedeuten, da nur das Programm auf dem Microcontroller entsprechend erweitert und angepasst werden müsste.

3. Microcontroller

Microcontroller sind Halbleiterchips, die Ein- und Ausgabe sowie Verarbeitungslogik auf einem Chip vereinen. Die Abgrenzung zu Microprozessoren fällt nicht immer ganz einfach, prinzipiell haben Microcontroller im Gegensatz zu Microprozessoren einen Teil der zum Betrieb nötigen Zusatzkomponenten, wie die bereits genannte Einagabeschnittstelle oder einen Taktgeber, bereits im Chip integriert.
Für ein solches Projekt stehen im Prinzip viele verschiedene Microcontroller zur Auswahl. Die Auswahl einer entsprechenden Plattform wurde an den Anforderungen und einem möglichst einfachen Einstieg in die Programmierung orientiert. Die Wahl fiel recht schnell auf die Arduino-Plattform.

3.1. Arduino-Plattform

Die Arduino-Plattform steht komplett unter einer OpenSource-Lizenz und verfügt über eine breite und aktive Community. Zur Plattform gehört neben dem physikalischen Teil auch eine Entwicklungsumgebung mit deren Hilfe der Microcontroller programmiert werden kann. Das bedeutet man hat eine Programmierumgebung, ähnlich wie Netbeans IDE oder Eclipse, die zusätzlich über eine Funktion verfügt, das entwickelte Programm in den Speicher des Microcontrollers zu schreiben.



Es existieren mehrere Varianten dieser Plattform. Einerseits werden immer wieder aktualisierte Versionen entwickelt, andererseits gibt es an speziellere Einsatzzwecke angepasste Modelle, zum Beispiel sehr kleine Varianten, die dafür gedacht sind, sogar in Kleidungsstücke eingenäht werden zu können, als auch Varianten die deutlich mehr Ein- und Ausgabeanschlüsse haben.
Diese Plattformen basieren allesamt auf verschiedenen Atmel AVR Controllern der megaAVR-Serie. Für dieses Projekt wurde ein Arduino Duemilanove verwendet, so ziemlich das Mainstream-Modell.
Es verfügt bereits über eine USB-Schnittstelle über die auch die Spannungsversorgung erfolgen kann.
Einer der großen Vorteile der Arduino-Plattform ist, dass sie alles was man benötigt bereits integriert hat und mitbringt. Da die gesamte Plattform als OpenSource Projekt angelegt ist, finden sich zahlreiche Anbieter individuell gestalteter Platinen mit unterschiedlicher Bestückung. Das bedeutet ebenso, man kann sich ein eigenes Board, das absolut nur die benötigten Komponenten enthält zusammenstellen und damit einen Teil der Kosten, die für ein komplettes Set anfallen, sparen. Damit ist man frei die Arduino Plattform für einen Prototyp zu verwenden und für eine spätere eventuelle Serienproduktion auf alle nicht benötigten Komponenten zu verzichten und damit den Preis für den endgültigen Microcontroller auf einen Bruchteil des Preises für ein komplettes Arduino-Set zu reduzieren.

3.2. Arduino-Programmierung

Programmiert wird die Arduino-Plattform mit Hilfe eines Java-Dialekts der sich Processing nennt. Die Entwicklungsumgebung selbst ist in Java geschrieben und daher relativ Plattformunabhängig. Processing selbst ist eine objektorientierte[23], stark typisierte[24] Programmiersprache und relativ einfach gehalten. Für die Arduino-Plattform wurde eine eigene Entwicklungsumgebung des quelloffenen Processing Projekts abgeleitet.
Sehr viele Funktionen für die Arduino-Plattform können mittels Libraries eingebunden werden, womit der Großteil der Programmierung für Standardbenutzer der Plattform darauf hinausläuft die sogenannten Sketches zu beschreiben.
Ein Sketch besteht mindestens aus der Funktion setup(), die beim Start des Microcontrollers einmal ausgeführt wird und für die Initialisierung genutzt werden kann, sowie der Funktion loop() die dauerhaft immer wieder aufgerufen wird, bis der Microcontroller ausgeschaltet wird.
Man kann beliebige weitere Funktionen und Objekte definieren.
Ein einfaches Beispiel Programm, das eine an Pin 13 angeschlossene LED im Sekundentakt ein- und ausschalten würde sieht folgendermaßen aus:

```
int ledPin = 13; // die LED ist an Pin 13 angeschlossen

void setup() {
    pinMode(ledPin, OUTPUT); // legt den LED-Pin als Ausgang fest
}

void loop() {
    digitalWrite(ledPin, HIGH); // LED an
    delay(1000); // 1 Sekunde warten
    digitalWrite(ledPin, LOW); // LED aus
    delay(1000); // 1 Sekunde warten
}
```

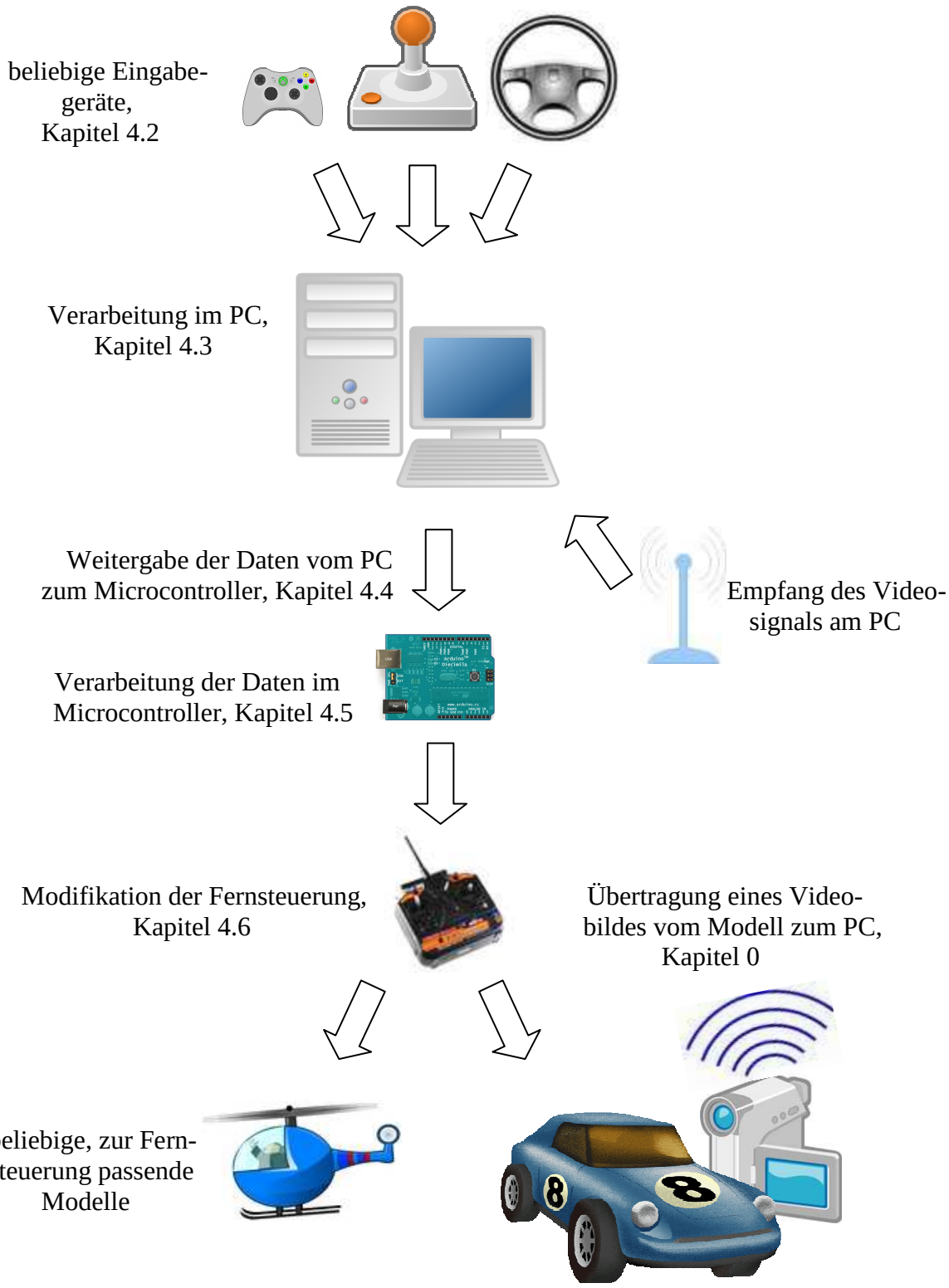
Es gibt sehr viele weitere Funktionen und Möglichkeiten, so kann man zum Beispiel Bitmuster in interne Register des Microcontrollers setzen um die Interrupts oder die Taktgeber zu verändern. Um tiefer in die Arduino-Programmierung einzusteigen findet sich im Anhang ein Link zum Arduino-Projekt.

[23] Programmierkonzept
[24] Typkonvertierungen müssen explizit durchgeführt werden

4. Gesamtübersicht des entwickelten Systems

Dieses Kapitel beschreibt die einzelnen Komponenten des entwickelten Systems und wie diese zusammen arbeiten. Ausserdem wird die an der Fernsteuerung vorgenommene Modifikation detailliert beschrieben. Eine Beschreibung des für den PC entwickelten Java-Programms folgt in Kapitel 5.

4.1. Gesamtschema



4.2. Eingabegeräte

Das System unterstützt beliebige Eingabegeräte, diese müssen nur vom Betriebsystem als Standardeingabegerät unterstützt werden. Das aktuell entwickelte System wurde bisher nur unter Windows getestet, da die Software jedoch in Java geschrieben wurde, sollte grundsätzlich die Möglichkeit bestehen das System auch auf Linux oder OS X zu portieren.
Im Projekt wurden 3 verschiedene Eingabegeräte verwendet, ein Lenkrad, ein Xbox360 Gamepad und ein Joystick. Da eine individuelle Konfiguration möglich sollte nahezu jedes Eingabegerät konfigurierbar sein. Ideal ist ein analoger Input, der auf die entsprechenden Achsen gemappt werden kann, es ist jedoch ebenso möglich einfache Tasten oder Buttons zuzuweisen. Eine genaue Steuerung ist damit allerdings nicht wirklich realisierbar.

4.3. Verarbeitung der Eingabe im PC

Die Daten werden zunächst als Rohwert vom Eingabegerät ausgelesen. Die meisten analogen Kanäle der gängigen Eingabegeräte haben einen Wertebereich von -128 bis +128 mit Neutralstellung bei 0. Dies ist allerdings nicht immer garantiert. So hat sich zum Beispiel gezeigt, dass das Bremspedal des Lenkrads nur einen Wertebereich von 0 bis 128 hat, das Gaspedal jedoch wiederum von -128 bis +128. Um dennoch eine funktionierende Konfiguration zu ermöglichen wurden mehrere Einstellmöglichkeiten implementiert.
Zum einen kann die Empfindlichkeit einzelner Achsen eingestellt werden. Umgesetzt wird dies intern dadurch, dass der gelesene Eingabewert mit einem Faktor multipliziert wird. Wählt man zum Beispiel eine Empfindlichkeit von 0,5 aus, so beträgt der Wertebereich nur noch -64 bis 64. Man braucht also am Eingabegerät den doppelten "Ausschlag" um den gleichen Wert auf die Ausgabeachse zu mappen. Dadurch dass der Wertebereich halbiert wird, kann zum Beispiel auch die maximal Geschwindigkeit auf die Hälfte reduziert werden.
Wählt man einen Faktor größer als 1, werden die Werte ausserhalb des gültigen Zielbereichs auf den maximal gültigen Wert "abgeschnitten". Damit wird verhindert, dass es zu Überläufen kommen kann, die beispielweise ein plötzliches Umspringen von Maximum auf Minimum bedeuten würden. Zusätzlich kann die Eingabe auch invertiert werden, eine Option die benötigt wird, da keine grundsätzliche Konvention besteht wie der minimale und der maximale Eingabewert zu interpretieren sind.
Eine weitere Einstellungsmöglichkeit für die Throttle-Kontrolle besteht darin, zwei Eingabekanäle gemeinsam einer Ausgabeachse zuzuweisen. Hierbei muss gegebenfalls etwas experimentiert werden, bis man die gewünschte Konfiguration gefunden hat.

Mit diesen Funktionen und der im folgenden beschriebenen Methode lässt es sich einrichten, dass das bereits erwähnte Problem des Rückwärtsfahrens gelöst werden kann.
Bei dem verwendeten Lenkrad musste zum Beispiel das Gaspedal nur auf halbe, das Bremspedal jedoch auf volle Empfindlichkeit eingestellt werden. Da sich dabei der Neutralpunkt, bei der aktuell verwendeten Berechnungsmethode, nicht mehr zwingend dort befindet wo er anfangs war, kann dies durch ein Offset ausgeglichen werden.

Um das Problem zu verdeutlichen wird es hier kurz durchgerechnet:

> Wird das Gaspedal nicht getreten liefert es den Wert -128.
> Das Bremspedal liefert ohne getreten zu sein den Wert 0.
> Das Resultat in diesem Zustand sollte sein dass, der Ausgabewert 0 ist, ebenfalls soll der Ausagebwert 0 sein, wenn beide Pedale voll durchgetreten werden. Voll durch getreten liefern beide Pedale den Wert 128.
> Bei Ausgabewerten größer als 0 fährt das Modell vorwärts, bei Werten kleiner als 0 fährt es rückwärts. Bei Wert 0 steht das Modell.
>
> Zum einen muss der Wertebereich des Gaspedals durch einen Empfindlichkeitsfaktor von 0,5 halbiert werden, da dessen ursprünglicher Zielbereich von 0 bis +128 geht. Damit beträgt der effektive Eingabewert des Gaspedals in der Grundstellung aber immer noch -64. Mit Offset +64 lässt sich dies zum gewünschten Neutralpunkt verschieben. Der Eingabewert des Brems- wird von dem des Gaspedals subtrahiert, damit ergeben sich nach folgender Formel verschiedene Fälle.
>
> Formel: (Gaspedal / 2) + Offset – Bremspedal = Ausgabewert
>
> | | |
> |---|---|
> | Wenn kein Pedal getreten wird: | (–128 / 2) + 64 – 0 = 0 |
> | Werden beide Pedale voll durchgetreten: | (128 / 2) + 64 – 128 = 0 |
> | Wird nur das Gaspedal durchgetreten: | (128 / 2) + 64 – 0 = 128 |
> | Wird nur das Bremspedal durchgetreten: | (–128 / 2) + 64 – 128 = – 128 |

Diese Lösung funktioniert, hat nur das Problem, das bei Anpassung einer der beiden Empfindlichkeiten das Offset angepasst werden muss. Praktisch lässt sich dies relativ leicht beheben, da ohnehin neben den bisher beschriebenen Funktionen auch noch möglich ist, im Betrieb den Neutralpunkt anzupassen. Dies wird normalerweise als Trimmen bezeichnet und wird auch von ziemlich jeder Funkfernsteuerung unterstützt. Wichtig ist diese Funktion vor allem weil manche elektrische Bauteile temperaturabhängig funktionieren und entsprechen schnell während dem Betrieb des Modells angepasst werden müssen. Am deutlichsten tritt dieser Effekt bei den Gyroskopen von Modellhelikoptern auf. Diese stabilisieren die Drehung um die Vertikalachse. So kann es sein dass kurz nach dem Einschalten des Modells, dieses leicht nach rechts dreht, während es sich, wenn man nicht über die Trimmung nachjustiert, nach einigen Minuten bereits deutlich nach links drehen kann.
Es gibt noch weitere Möglichkeiten diese Berechnung durchzuführen, für den ursprünglich geplanten Funktionsumfang ist die beschriebene Lösung allerdings durchaus ausreichend.
Im großen und ganzen sind die Funktionen zur Verarbeitung der Eingabe damit beschrieben. Auch das erwähnte kommerzielle System bietet ungefähr genau diese Funktionen an. Es gibt noch einige weitere die sich manche Modellbauer gelegentlich wünschen, zum Beispiel sollen bestimmte Achsen manchmal nicht linear gesteuert werden, sondern auf einer exponentiellen Kurve basieren. Gelegentlich sollen auch zwei Ausgabeachsen von einem Eingabekanal kontrolliert werden, dies ist mit dem entwickelten System auch möglich. Man kann ein und den selben Eingangskanal einfach mehrfach zuweisen und konfigurieren.
Die Berechnung der Ausgabewerte ist die eigentliche Hauptaufgabe des Programms, um die Justierung etwas einfacher zu gestalten werden darüberhinaus die berechneten Werte am Bildschirm dargestellt. Damit kann man die Steuerung im vorraus konfigurieren. Den Neutralpunkt am laufenden Modell einzustellen ist zumindest für die Throttle-Kontrolle nicht praktikabel.

4.4. Weitergabe der Daten an den Microcontroller

Bei der Planung des Systems war noch nicht klar wieviel Aufwand für eine saubere Kommunikation zwischen PC und Microcontroller erforderlich sein würde. Im Prinzip müssen nur die jeweiligen Ausgabewerte an den Microcontroller gesendet werden. Das bedeutet bei den verwendeten Komponenten, dass 4 8-Bit Zahlen für jeweils 256 Stufen ausreichen, da zum einen die Eingabegeräte keine höhrere Auflösung bieten und zum anderen der Chip in der Fernbedienung ebenfalls mit Werten von 0 bis 255 angesteruert wird.
Anfangs war ich skeptisch ob ein sehr naiver Ansatz ohne Bestätigungssignale oder Prüfsummen funktionieren würde. Um diesen Punkt allerdings nicht unnötig kompliziert zu gestalten, habe ich trotzdem zunächst mit einem sehr einfachen Ansatz begonnen.
Damit wenigstens ein halbwegs fester Bezugspunkt existiert und Anfang und Ende der 4 Zahlen unterschieden werden können habe ich mich kurzerhand beim ersten Versuch dafür entschieden, einfach den Buchstaben 'A' gefolgt von den 4 Zahlen an den Microcontroller zu senden. Da dies bisher sehr gut funktioniert und keinen einzigen Ausfall zur Folge hatte, blieb es bei diesem Ansatz. Tatsächlich ist es ein relativ gewagter Ansatz, da der Wert des ASCII[25]-Zeichens ebenfalls genau im Wertebereich der Zahlen sind, könnte im schlimmsten Fall eine der 4 Zahlen als 'A' interpretiert werden und damit das Auslesen der Werte an der falschen Stelle begonnen werden. Das Ergebnis wären falsch interpretierte Werte.
Da sich dieser Fehler allerdings nur dann fortsetzen kann, wenn ein Wert dauerhaft dem Wert des ASCII-Zeichens 'A' entspricht, tritt in der Praxis selbst mit diesem naiven und nicht hundertprozentigen Ansatz keine Beeinträchtigung auf. Sobald sich der falsch interpretierte Wert ändert synchronisiert sich das Auslesen wieder korrekt.
Für eine korrekte sicherere Kommunikation müsste sichergestellt werden, dass ein Wert oder Bitmuster zur Synchronisation verwendet wird, das nicht im zulässigen Wertebereich für die zu lesenden Zahlen liegt.
Eine Möglichkeit wäre zum Beispiel den Wertebereich der Zahlen auf 1 bis 254 zu begrenzen, damit verliert man zwar 2 Auflösungsstufen, was allerdings kaum ins Gewicht fällt, da bereits die Eingabegeräte oft nur ungenau bis an die maximalen Werte herankommen. Im Gegenzug erhielte man 2 Bitmuster die eine eindeutige Identifikation des Synchronisationspunktes erlauben. Praktisch würd es ausreichen das bisher verwendete 'A' durch einen der beiden erhaltenen Werte zu ersetzen. Für noch mehr Sicherheit kann man sich weitere Muster ausdenken.
Zusätzlich könnte man eine Prüfsumme mit übertragen um zu garantieren, dass keine Übertragungsfehler auftreten.
Da jedoch bereits der erste naive Ansatz sehr zuverlässig funktioniert wurden die genannten Verbesserungen bisher nur in der Theorie entwickelt.

Technisch betrachtet wird für diese Kommunikation ein virtueller serieller Kommunikations-Port verwendet, unter Windows zum Beispiel "COM 5" genannt. Für diese Ports muss man im Prinzip nur eine Geschwindigkeit und eine Verbindungskennung festlegen, die dann von Sender und Empfänger verwendet werden.
Dass hierbei ein virtueller serieller Kommunikations-Port verwendet wird, der in Wahrheit über einen USB-Port realisiert ist, bleibt für die Schnittstelle im Programm völlig transparent. Der virtuelle Port wird einfach über einen entsprechenden Treiber bereitgestellt.

[25] ASCII: 7-Bit Zeichenkodierung

4.5. Verarbeitung der Daten im Microcontroller

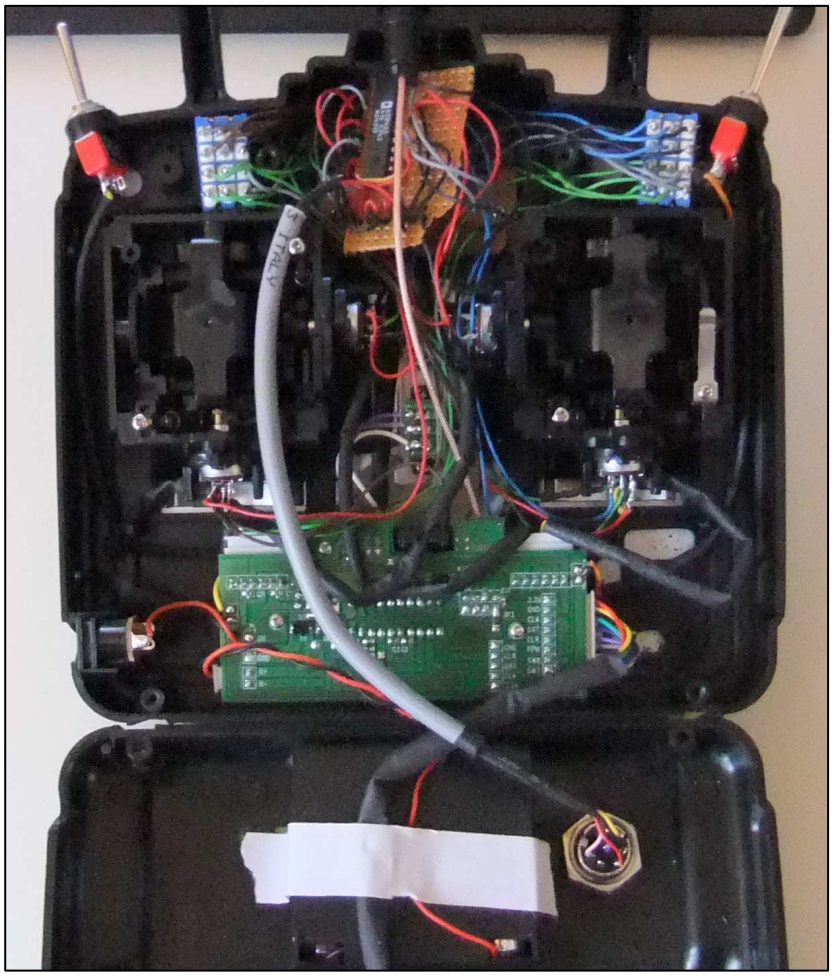
Da das im Microcontroller verwendete Programm relativ einfach und überschaubar ist wird es hier nur kurz beschrieben und der Quellcode im Anhang zugefügt.
Beim Starten des Microcontrollers initialisiert dieser eine serielle Kommunitkations-verbindung zum PC mit einer festgelegten Geschwindigkeit. Anschließend lauscht der Microcontroller nur noch auf das in Kapitel 4.4 beschriebene Muster und interpretiert daraus die 4 Werte. Diese Werte werden anschließend mit Hilfe einer frei verfügbaren Arduino Bibliothek in ein SPI-Signal umgewandelt und über 3 Leitungen zum Digitalpotentiometer übermittelt.

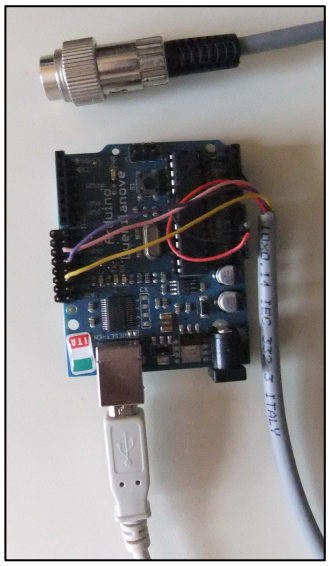
4.5.1. SPI Protokoll

Das Serial Peripheral Interface beschreibt einen von Motorola entwickelten synchronen seriellen Datenbus der digitale Schaltungen nach dem Master-Slave-Prinzip ermöglicht.
Da der Standard als Datenbus definiert ist, können mehrere Teilnehmer an die 3 gemeinsamen Leitungen gleichzeitig angeschlossen werden. Für welchen Teilnehmer ein Signal bestimmt ist, wird über eine zusätzliche Chip-Select-Leitung angezeigt. Für jeden Teilnehmer muss eine dedizierte Chip-Select-Leitung vorhanden sein.
Die 3 übrigen Leitung tragen die Bezeichnungen SDO (Serial Data Out) – für Signale vom Slave zum Master, SDI (Serial Data In) – für Signale vom Master zum Slave, und SCK (Serial Clock) – für den vom Master generierten gemeinsamen Takt.
Eine solche Verbindung ist Vollduplexfähig, es ist also gleichzeitige Kommunikation in beide Richtungen möglich. In der Praxis wird häufig jedoch nur die Leitung vom Master zum Slave verwendet. Antworten vom Slave zum Master könnten zum Beispiel Werte eines Sensors sein.
Das Einbinden dieses Protokoll in ein Arduino-Programm ist sehr einfach gestaltet und beschränkt sich im Großen und Ganzen darauf, die entsprechenede Bibliothek zu laden und sich eine Hilfsfunktion zu schreiben, die den gewünschten Teilnehmer mit Hilfe der Chip-Select-Leitung aktiviert, das entsprechende Signal ausgibt, und anschließend den Teilnehmer wieder deaktiviert. Wie die Signale für einzelne Teilnehmer aussieht muss der jeweiligen Dokumentation des Bauteils entnommen werden.

4.6. Modifikation der Fernsteuerung

An der Fernsteuerung wurden einige Modifikationen vorgenommen um einen komfortablen Anschluss an den Microcontroller zu ermöglichen.
Zum einen wurde an der Rückseite eine Steckerbuchse eingebaut, damit die Verbindung zum Microcontroller, nur bei Bedarf angeschlossen sein muss. Damit eine sichere Verbindung ohne Wackelkontakte garantiert ist, wurde ein verschraubbarer Stecker ausgewählt.
Um das Digitalpotentiometer schnell deaktivieren zu können wurden zwei Schalte eingebaut, die jeweils 2 Kanäle von den mechanischen auf die digitalen Potentiometer umschalten. Diese Schalter haben jeweils 4 Anschlüsse, die entweder von Pin 1 nach Pin 2 oder von Pin 3 nach Pin 1 leiten.
Ausserdem befindet sich das Digitalpotetiometer selbst in der Fernbedienung. Das bedeutet, die Fernbedienung kann wieder ganz normal verschlossen werden und die PC-Steuerung bequem dadurch aktiviert werden, dass der Microcontroller angeschlossen und die beiden Schalter umgelegt werden.





4.7. Funkkamera

Da mittlerweile kleine Funkkameras relativ günstig angeboten werden, und zumindest auf das Modellauto ohne weiteres montiert werden können, wollte ich auch eine Videofunktion integrieren. Die Bildqualität ist allerdings mit der aktuell verwendeten Kamera sehr bescheiden und eher als unbrauchbar einzustufen. Die momentan eingesetzt Kamera kann allerdings, wie bereits erwähnt, ohne weiteres durch eine bessere ersetzt werden, da zum einen die verwendete Library es erlaubt eine beliebige vom Betriebssystem unterstützte Kamera auszuwählen, und zum anderen die Kamera selbst auf analogem Wege angeschlossen ist. Dafür wird ein gängiger USB-Videograbber eingesetzt, der ein beliebiges Videosignal digitalisiert und dem PC zur Verfügung stellt.
Die getestete Kamera hat leider nur eine Reichweite von mehreren Metern. Da die Kamera im 2,4 GHz Frequenzband sendet wird ihr Signal leicht von WLAN-Signalen überlagert und die Sendeleistung reicht damit nicht mehr aus um ein brauchbares Bild zu liefern.
Da die üblicherweise mitgelieferten Linsen dieser billigen Kameras eine relative geringe Brennweite haben, ist normalerweise nur ein recht kleiner Winkel einsehbar. Dies kann entweder durch Zukaufen einer Linse mit höherer Brennweite oder den Einsatz von 2 Servomotoren, die es erlauben die Kamera in eine gwünschte Richtung zu drehen, verbessert werden.
Aufgrund der schlechten Bildqualität wurden bisher keine tiefergehenden Experimente mit montierter Kamera unternommen, das System an sich stellt die nötigen Funktionen jedoch allesamt zur Verfügung.

5. Entwicklung der Software

Dieses Kapitel beschreibt die Entwicklung der Software und stellt die verwendeten Libraries vor. Ausserdem werden verschiedene Entwicklungsstufen beschrieben und die Funktionen der grafischen Oberfläche dargestellt.

5.1. Anforderungen

Vor Beginn der konkreten Planung und Entwicklung des Programms, wurden verschiedene Anforderungen festgehalten.
Vor allem sollte es möglich sein beliebige Eingabegeräte zu verwenden und eine jeweils passende Konfigurationsmöglichkeit geboten werden. Hier wurde bereits klar, dass verschiedene Empfindlichkeiten und Korrekturmöglichkeiten nötig sein würden.
Auch sollte ein Videobild in die Hauptansicht des Programms integriert sein.
Ursprünglich war geplant, die komplette Konfiguration in der Hauptansicht vorzunehmen. Bei der Entwicklung zeigte sich dann allerdings, dass es praktikabler ist, bestimmte Einstellungen bei Programmstart vorzunehmen und diese danach nicht mehr zu verändern.

5.2. Entscheidung für Java

Als Programmiersprachen standen vor allem C++ und Java als Alternativen zu Auswahl. Da ich in den vergangenen Monaten eigentlich kaum mit C++ programmiert, dafür jedoch in Java mehrere Projekte bearbeitet hatte, entschied ich mich zunächst nach Java Libraries zu suchen und nur falls wichtige Funktionen nicht verfügbar wären zu C++ zu wechseln. Da ich entsprechende Libraries für alle Funktionen fand und diese auch gut funktionierten, war nach der Recherche zu den Libraries die Entscheidung für Java gefallen, ohne dass es wirkliche Gründe gegen C++ gegeben hätte.
Die gelegentlich anzutreffenden Vorbehalte gegenüber C++ teile ich eigentlich nicht, da Java jedoch in den letzten Jahren auch relativ stark und schnell geworden ist, gab es keine ausschlaggebenden Gründe gegen Java und die Wahl stand fest.

5.3. Verwendete Bibliotheken

Für dieses Projekt wurden neben gängigen Javakomponenten wie Swing für die grafische Oberfläche wurden 3 Libraries verwendet.
Für das Einbinden der Eingabegeräte habe ich jInput gewählt.
Um das Videobild auszulesen viel die Wahl auf JMF – das Java Media Framework.
Die dritte Library heisst Rxtx und ermöglicht es von Java aus bequem auf die seriellen Ports zuzugreifen.

5.3.1. jInput

Zum Zeitpunkt der Erstellung dieser Dokumentation war die Website zum jInput Projekt leider nicht verfügbar[26]. Daher konnte ich leider nur auf meine Erinnerung an die Beschreibungen auf dieser Website, sowie die Informationen die ich für den Einsatz dieser Library benötigte, zurückgreifen.
Grundsätzlich besteht mittels jInput die Möglichkeit plattformunabhängig zu arbeiten. Für Windows müssen zwei DLL-Dateien[27], unter Linux entsprechende SO-Dateien mit dem Programm verwendet werden. Auch für Mac OS X sollten vergleichbare kompilierte Bibliotheken zur Verfügung stehen. Als Schnittstelle zu Java dient eine Datei namens jInput.jar, welche alle nötigen Methoden bereitstellt.
jInput funktioniert nach dem Polling-Prinzip, das bedeutet es kann jederzeit vom Programm selbst kontrolliert werden, wann neue Daten vom Eingabegerät geholt werden. Diese Daten werden dann anschließend bequem ausgelesen und verarbeitet.
Um ein bestimmtes Eingabegerät auszuwählen, erhält man vom Framework eine Liste von allen verfügbaren Geräten. Daraus wählt man eines mit Hilfe des entsprechenden Index aus und kann dieses dann wie beschrieben "pollen" und über eine weitere Liste die jeweiligen Achsen und Knöpfe, components genannt, auslesen.
Die Verwendung von jInput ist ziemlich einfach und funktioniert zuverlässig ohne dass dadurch unkontrollierbarer (Rechen-)Aufwand erzeugt wird.

5.3.2. Java Media Framework

Das Java Media Framework ist ein ursprünglich von Sun, Intel und Silicon Graphics entwickeltes Framework um Mikrofone und Kameras in Java verfügbar zu machen. Das letzte Update von JMF ist bereits einige Jahre her, die letzte Version stammt aus dem Jahr 2004. Trotz des Alters hat sich JMF als brauchbar erwiesen und ließ sich gut einbinden.
Die Funktionsweise von JMF ist ähnlich wie die von jInput. Man kann sich eine Liste der verfügbaren Geräte liefern lassen und nachdem man einige Einstellung zum Videomodus und ähnlichem vorgenommen hat, in beliebiger Geschwindigkeit neue Bilder liefern lassen.
Wichtig ist nur, dass JMF vor der Verwendung von neuen Geräten, diese mit Hilfe eines Konfigurationsprogramms registrieren muss.

5.3.3. RxTx

Die Library für die Kommunikation mit dem Arduino-Microcontroller liegt der Arduino Entwicklungsumgebung direkt bei und lässt sich ebenfalls sehr einfach ansprechen. Auch hier gibt es eine Liste von verfügbaren Ports, von denen man einen auswählt, konfiguriert und sich einen Input- und einen Outputstream liefern lassen kann. Danach kann man auf dem seriellen Port fast wie auf einer Datei lesen und schreiben.

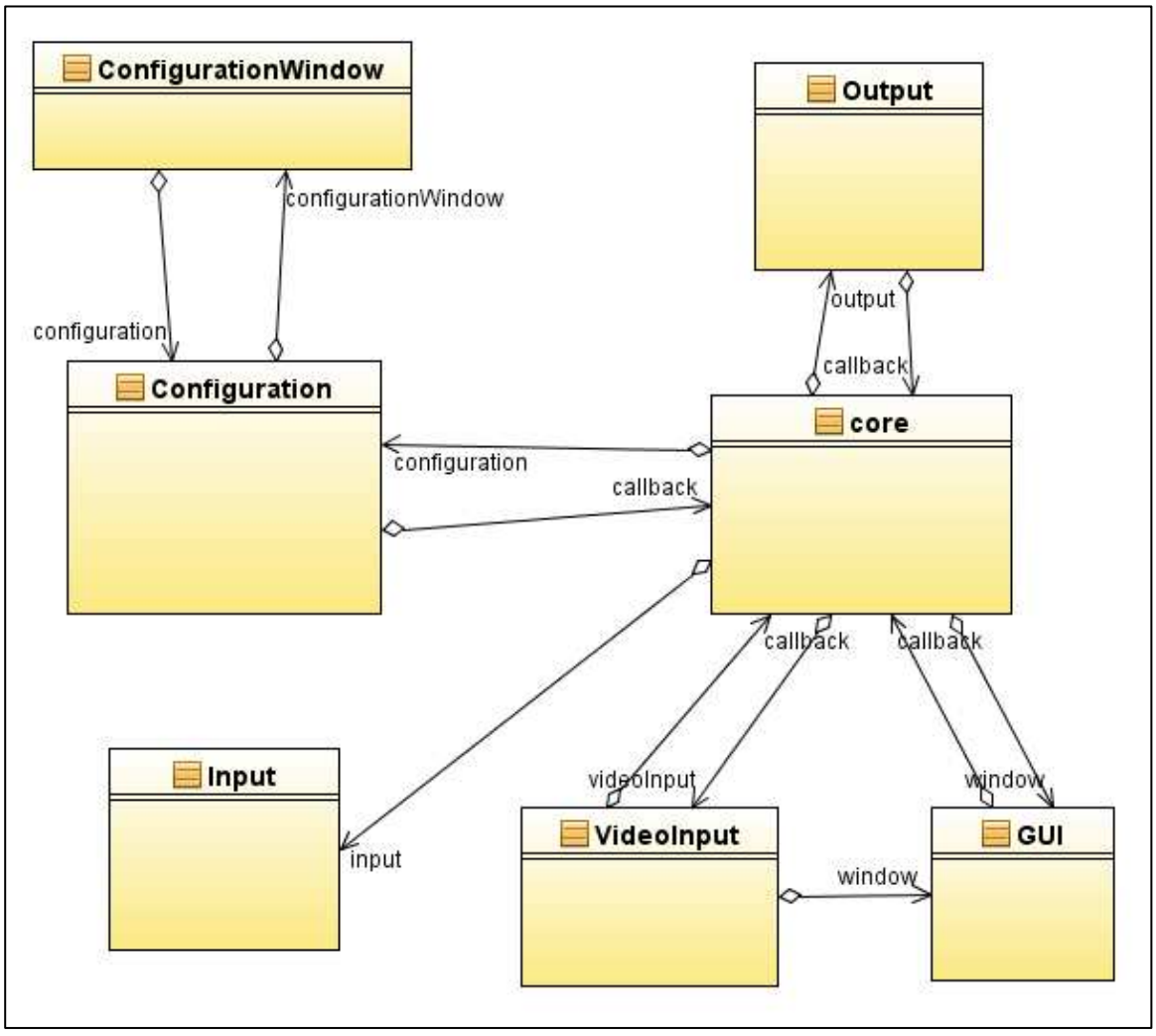
[26] https://jinput.dev.java.net/
[27] Dynamische Bibliotheken die in der Regel in kompilierter Form eingebunden werden müssen

5.4. Entwurf und Design der Software

Die einzelnen Funktionen der Software wurden anhand kleinerer Prototypen getestet und später im Programm dann zusammengebaut. Die meiste Zeit nahm das Planen der Konfigurationsmöglichkeit und das Zusammenstellen der grafischen Oberfläche in Anspruch. Die einzelnen Klassen und Komponenten waren relativ schnell entworfen.

5.4.1. Komponenten

Folgendes Diagramm zeigt ein stark vereinfachtes Klassendiagramm, um einen Überblick über die Komponenten und deren Zusammenhang zu bekommen.



Gestartet wird das Programm über die Klasse core, die alle restlichen benötigten Objekte erzeugt. Zunächste wird ein Objekt vom Typ Configuration angelegt und mit Hilfe von ConfigurationWindow konfiguriert. Anschließend werden Objekte vom Typ GUI, Input, VideoInput und Output erzeugt.
Die Klassen Output, Input und VideoInput sind von der Klasse Thread abgeleitet damit sie eigenständig parallel laufen können. VideoInput holt entsprechend der eingestellten Framerate neue Bilder und zeichnet diese auf die GUI. Ebenso liest das Input-Objekt entsprechend einem eingestellten Intervall neue Werte vom Eingabegerät und das Output-Objekt schreibt die berechneten Werte auf den seriellen Port.

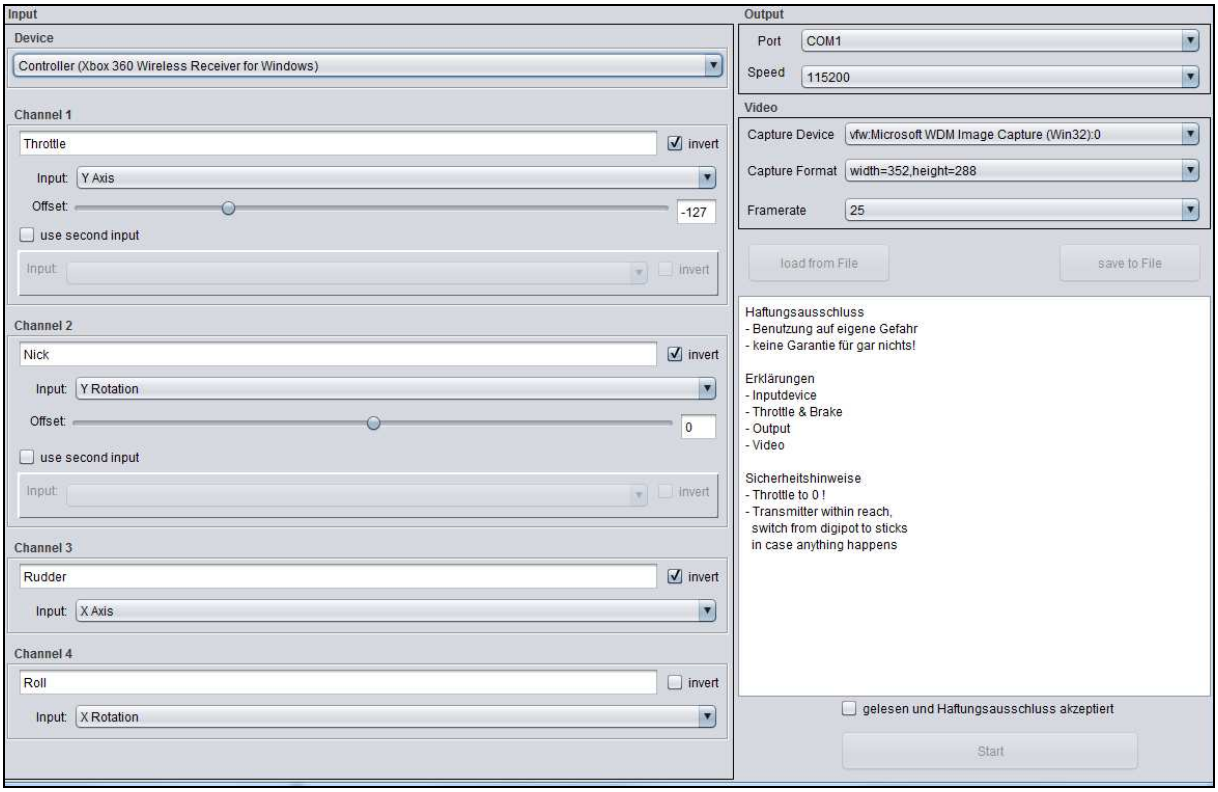
5.4.2. Grafische Oberfläche

Die grafische Oberfläche bietet auf der linken Seite unterschiedliche Einstellungsmöglichkeiten und zeigt die aktuellen Werte, wie sie berechnet werden, an.
Links oben können die einzelnen Achsen unterschiedlich nach justiert werden. Darunter kann man einstellen wie empfindlich die jeweiligen Achsen ansprechen sollen.
Auf der rechten Seite sieht man das Videobild, in diesem Beispiel war die Kamera allerdings nicht auf dem Modell montiert.

Mehr Einstellungen braucht man zur Laufzeit nicht mehr, der praktische Teil spielt sich ohnehin weniger auf dem Bildschirm ab solange keine Kamera auf dem Modell montiert ist.
Die Konfigurationsmöglichkeiten zu Beginn des Programms sind auf der folgenden Seite beschrieben.

5.4.3. Konfiguration und Einstellungsmöglichkeiten

Die Einstellungsmöglichkeiten zu Beginn sind deutlich komplexer als die, die man zur Laufzeit noch anpassen kann. So kann links oben das gewünschte Eingabegerät gewählt werden, darunter werden die einzelnen Eingabeachsen den jeweiligen Ausgabekanälen zugeordnet und bei Bedarf mit einem Offset angepasst.
Rechts oben kann man den Port für die Kommunikation mit dem Microcontroller sowie die Geschwindigkeit einstellen.
Direkt darunter wählt man eine der verfügbaren Kameras aus und gibt das Videoformat und die gewünschte Framerate vor.
Der Textframe ist vorbereitet um bei einer möglichen Veröffentlichung des Programms dort Sicherheitshinweise zu geben und auf Haftungsauschluss hinzuweisen.
Nachdem die Bedingungen akzeptiert wurden, kann der Start-Button rechts unten geklickt werden und das Programm startet.



5.5. Entwicklungsstufen

Während der Entwicklung gab es mehrere Zwischenstufen, der erste funktionierende Prototyp hatte keine vorgeschaltete Konfiguration und die meisten Einstellung direkt im Code. Auch die GUI war einfacher gestaltet und hatte im Prinzip nur Regler für die Trimmung sowie eine Einstellungsmöglichkeit für die Empfindlichkeit der Yaw-Achse (Drehen um die vertikale Achse).
Über mehrere Versionen hin wurde die GUI dann erweitert. Zwischendurch hatte ich geplant, die gesamte Konfiguration im Hauptfenster anzuzeigen, um mehr Übersichtlichkeit zu schaffen wurde ein großer Teil allerdings wie bereits beschrieben ausgelagert.
Das aktuelle Programm ist bereits relativ flexibel, es gibt jedoch durchaus noch einige Punkte die verbessert oder ausgefeilt werden können.
Um als vollständig abgeschlossen gelten zu können, sollte das Programm neben einigen Beispieleinstellungen auch die Möglichkeit bieten, Einstellungen zu speichern und zu laden.

6. Fazit & Ausblick

Um dieses Projekt im Nachhinein auszuwerten und die Erfahrungen, Erkenntnisse und Ideen zusammenzufassen enthält dieses Kapitel ein paar allgemeine und persönliche Reflektionen.

6.1. Bewertung der gefundenen Lösungen

Die für dieses Projekt eingesetzten Werkzeuge waren fast immer zuverlässig und brauchbar. Obwohl die Libraries nicht alle hochaktuell waren, resultierten daraus keinerlei Probleme oder mussten Einschränkungen in Kauf genommen werden.
Die Lösung um die Fernsteuerung zu kontrollieren, funktioniert sehr gut, ist nur leider aufgrund des nötigen Bastelaufwandes nicht ganz so flexibel und übertragbar wie es die anfangs geplante Nutzung des Trainerports erlaubt hätte. Da die Gründe hierfür jedoch im nicht standardkonformen Signal liegen, denke ich, die gefundene Lösung ist dem Problem absolut angemessen.
Die Konfiguration des Systems ist ganz brauchbar geworden, kann aber sicherlich noch etwas verbessert werden. Ich denke dafür wäre es allerdings sinnvoll das System einige Zeit praktisch zu testen und mit den entsprechenden Erfahrungen nach und nach zu verbessern. Ob eine in der Theorie schöne Alternative tatsächlich besser wäre ist nicht garantiert und rechtfertigt demnach, meiner Meinung nach im Moment nicht, sich noch weitere Stunden an der Konfiguration aufzuhalten.

6.2. Reflektion von Entwicklungsaufwand / -ablauf

Rückblickend bleibt die Erkenntnis, dass die Planung dieses Projekts sehr nahe am tatsächlichen Aufwand war. Durch umfangreiche Recherche im Vorfeld und hohe Motivation konnte das Projekt zügig und detailliert bearbeitet werden. Die eingeplanten Zeitpuffer für eventuelle Schwierigkeiten wurden zum Teil genutzt und bestätigen ebenfalls die Zeitplanung.

6.3. Soll / Ist – Vergleich

Betrachtet man die zu Beginn geplanten Funktionen und die Möglichkeiten die entwickelt wurden, zeigt sich, dass nahezu alles im Rahmen der technischen Möglichkeiten umgesetzt werden konnte.
Leider ist die Qualität der Funkkamera nicht wirklich ausreichend für dieses Projekt. Dies lässt sich allerdings durch Investition in einen bessere Kamera nachholen. Damit ergeben sich noch einige weitere sehr interessante Möglichkeiten, die allerdings den Zeitrahmen des Projekts ohnehin sprengen würden.
Alles in allem entspricht damit der Ist-Zustand des Systems ziemlich genau dem Soll-Zustand und es mussten keine gravierenden Abstriche gemacht werden.

6.4. Weitere Ideen für dieses System

Im Verlauf des Projekts entstanden einige weitere Ideen, die unter Umständen später tatsächlich realisiert werden können, teilweise jedoch selbst den Umfang eines eigenständigen Projektes annehmen würden.

6.4.1. Headtracking

Interessant wäre es neben der Steuerung des Modells auch eine Kamerasteuerung sehr intuitiv zu gestalten. Von dem eingangs beschriebenen kommerziellen System existiert eine Variante die ein Headtracking-System mit einer Videobrille verbindet. Dreht der Benutzer also den Kopf, so bewegt sich die Kamera im Modell automatisch entsprechend mit und erlaubt es dadurch sich auf natürliche Weise "umzusehen". Selbst ohne Videobrille wäre eine vergleichbare Möglichkeit denkbar, ob sich Headtracking allerdings dafür eignet um am fixen Monitor eingesetzt zu werden, kann ich mangels eigener Erfahrung mit solchen Systemen nicht wirklich einschätzen, wage dieses jedoch zu bezweifeln.
Es bestehen mehrere Möglichkeiten ein solches Headtracking zu realisieren. Entweder man setzt einfach ein kommerziel verfügbares System ein oder man entwickelt eine eigene Lösung. Mittels Marker die am Kopf getragen werden ist es durchaus möglich mit einer Kamera über dem Monitor, die Bewegungen zu verfolgen. Eine Alternative die auch in einigen Projekten in Internetforen beschrieben wird, besteht darin einen weiteren Microcontroller mit Bewegungssensoren zu bestücken und am Kopf zu tragen. Aus den Daten dieser Sensoren können dann die Bewegungen des Kopfes berechnet werden.

6.4.2. Einbinden eines Upstreams

Sehr interessant wäre es ebenfalls dem System einen Upstream hinzuzufügen, damit wäre es möglich Sensoren jeglicher Art auf dem Modell anzubringen und diese Daten am PC zu nutzen. Denkbar wären Abstandssensoren oder sogar GPS-Empfänger[28]. Ein Upstream wäre jedenfalls ein großer Schritt in Richtung autonomer Steuerung.
Für Microcontroller wie die Arduino-Plattform gibt es Funkmodule, die relativ einfach einzusetzen sind. Ein Beispiel für solche Module sind die sogenannten xBee-Shields[29], die mehr oder weniger einfach auf einen Arduinocontroller aufgesteckt werden können.
Ebenso gibt es Ethernet-Shields die es zum Beispiel mit Hilfe eines WLAN-Accesspoints oder WLAN-Bridges erlauben würden über ein vorhandenes WLAN mit dem PC zu kommunizieren und auf diesem Weg Daten zurückzuliefern.

[28] Global Positioning System, verwendet Satellitensignale und deren Laufzeitunterschiede um mittels Triangulierung eine weltweite Positionsbestimmung zu ermöglichen
[29] Shields sind fertige Platinen, die zur Erweiterung einfach auf andere Microcontroller aufgesetzt werden können

6.4.3. Auswerten des Kamerabildes zur Spur und Objekterkennung

Eine weitere interessante Idee erfordert zwar ein sehr gutes und zuverlässiges Kamerabild, bietet aber dann durchaus Potential um den Umfang eines eigenen Projekts zu erreichen. Man könnte durch Bildanalyse, angefangen von einfachem Motiontracking des Hintergrundes, bis hin zu Objekterkennung, die Steuerung des Modells berechnen. Eine definierte Linie zu verfolgen sollte mindestens möglich sein. Ähnliche Projekte aus verschiedenen Robotiklabors präsentieren immer wieder interessante Ergebnisse. Allerdings darf man die Anforderungen an Hard- und Software, sowie Programmierfähigkeiten nicht unterschätzen.

6.5. Persönliches Fazit

Persönlich hatte ich viel Spaß mit diesem Projekt und habe zu keinem Zeitpunkt bereut dieses Thema gewählt zu haben, im Gegenteil war ich sehr motiviert und kam meist recht gut voran.
Ich konnte einige Erfahrungen mit Microcontrollern sammeln und mir anschauen wie diese mit weiteren elektronischen integrierten Bauelementen zusammen eingesetzt werden können. Damit wurde ein Thema bearbeitet, das ich bereits seit einiger Zeit vor mir bearbeiten wollte.
Von programmiertechnischer Seite brachte mir dieses Projekt nicht unbedingt sehr viel neues, allerdings setzt eine gewisse Routine beim Entwurf von Programmen ein und es kristallisiert sich immer wieder heraus, wie man bestimmte Dinge praktisch umsetzen kann und wie man manches besser nicht löst. Erfahrung beim Entwurf von Software hat mir dieses Projekt mit Sicherheit gebracht.
Ausserdem bleiben noch einige interessante Punkte offen, unter anderem das Erweitern der Schnittstellen um Infrarot und den erwähnten Trainerport mittels PPM-Signal.

7. Anhang

An dieser Stelle finden sich weitere Informationen und zum Beispiel der Code mit dem der Microcontroller programmiert wurde.
Direkte Vorlagen oder Anleitungen wurden keine verwendet, allerdings waren unzählige Modellbauseiten und das Internet überhaupt unverzichtbar für viele der Recherchen zu diesem Projekt.

7.1. Informationsquellen

Endurance R/C, Overland Park, KS: http://www.endurance-rc.com/contact.html

Funkfrequenzen, z.Bsp unter: http://www.modellbausieghard.de/howto/funk-frequenzen/

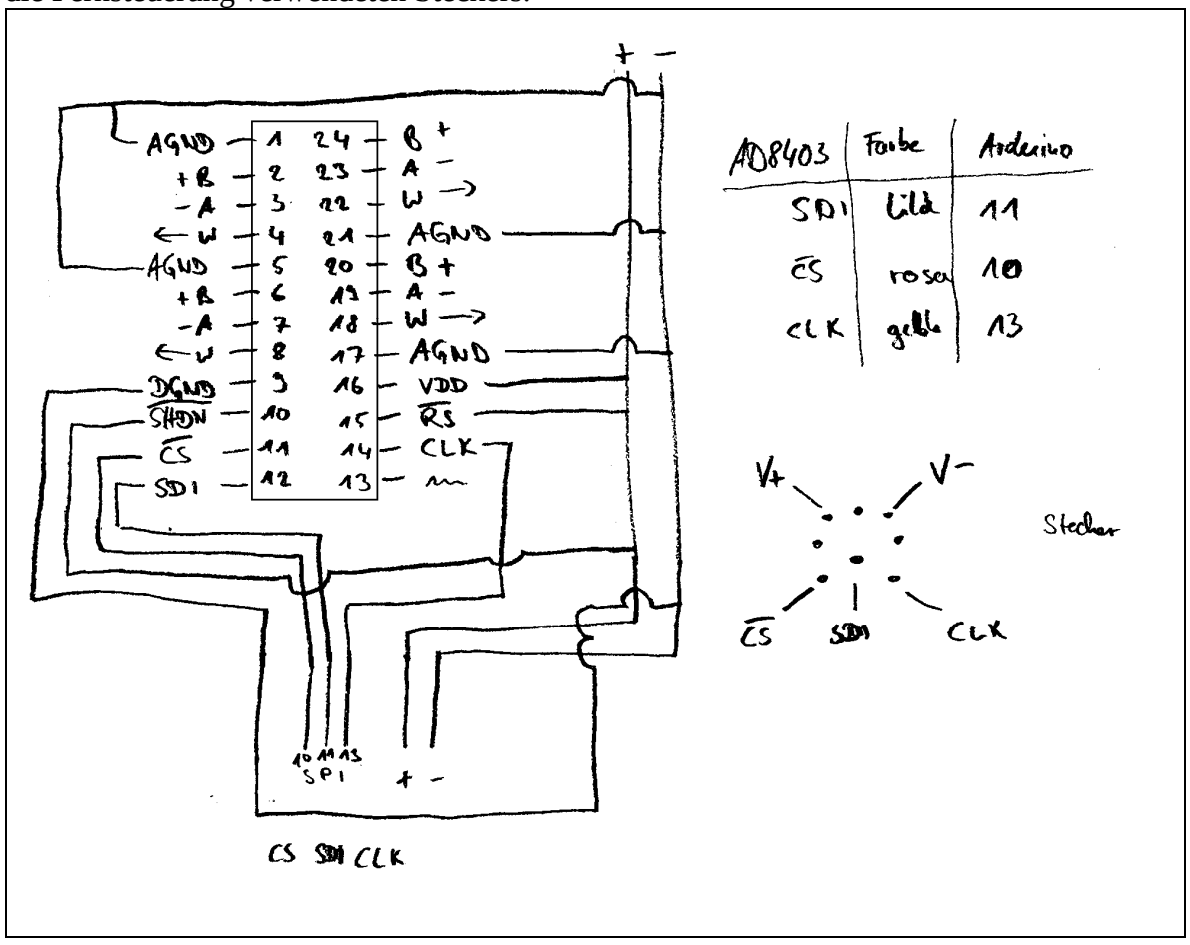
Arduino-Projekt, http://www.arduino.cc

7.2. Arduino-Code

```
// Library für SPI-Protokoll laden
#include <Spi.h>

const int slaveSelectPin = 10;
const int analogPin = 1;
int val = 0;

// Widerstände vorbelegen, 255 entspricht throttle low
int values[4];
byte throttle = 255;
byte gier = 127;
byte nick = 127;
byte roll = 127;
int incomingByte = 0;
byte counter = 4;

void setup(){
     Serial.begin(115200);
     pinMode(slaveSelectPin, OUTPUT);
// Initialisieren mit passenden Widerständen für die Fernsteuerung
     digitalPotWrite(0, 255);
     values[0] = 255;
     values[1] = 127;
     values[2] = 127;
     values[3] = 127;

}

void loop(){
     if (Serial.available() > 0) {
            incomingByte = Serial.read();
     if ((counter > 3) && (incomingByte == 'A')){
            counter = 0;
     }
     else{
            if (counter < 4){
// "flippen" des gewünschtem Wert nach benötigtem Widerstand
                  values[counter] = 255 - incomingByte;
                  counter++;
            }
     }
}

// Senden der 4 Werte
digitalPotWrite(0, values[0]);
digitalPotWrite(1, values[1]);
digitalPotWrite(2, values[2]);
digitalPotWrite(3, values[3]);

}

// Hilfsfunktion um das Digitalpotentiometer anzusprechen
void digitalPotWrite(byte address, byte value){
     digitalWrite(slaveSelectPin, LOW);
     Spi.transfer(address);
     Spi.transfer(value);
     digitalWrite(slaveSelectPin, HIGH);
}
```

7.3. Schaltplan des Digitalpotentiometers

Anhand dieser handschriftlichen Skizze wurde das Digitalpotentiometer verkabelt.
Ohne auf alle Pins genauer einzugehen, diese sind im Datenblatt detailliert beschrieben, ein paar Bemerkungen zu dieser Skizze.
Links in der Mitte ist der Chip mit seinen 24 Pins abgebildet. Die oberen 8 Pins auf beiden Seiten sind die Kontakte der 4 Potentiometer. Die Anschlüsse +B, -A und ←W entsprechen den 3 Kontakten bei mechanischen Potentiometern, wobei ←W den variablen Kontakt bereitstellt. Die unteren 4 Anschlüsse sind für die Spannnungsversorung des IC-Chips, sowie dessen Steuerung mittels SPI-Protokoll.
Unten im Schaltplan sieht man die 5 Leitungen die zum Arduino-Microcontroller führen, 3 Leitungen für den SPI-Anschluss, sowie eine Spannungsversorgung. Die Tabelle rechts beschreibt die jeweilige Kabelfarbe sowie den entsprechenden Pin an dem diese Leitung am Microcontroller angeschlossen werden muss. Darunter ist zusätzlich die Pinbelegung des für die Fernsteuerung verwendeten Steckers.



7.4. Datenblatt des Digitalpotentiometers

Das Datenblatt des Digitalpotentiometers ist mit 32 Seiten sehr umfangreich, der Vollständigkeit halber und um den Umfang der Recherche zu verdeutlichen, die für einen Laien notwendig ist um IC-Chips in eigene Schaltungen zu integrieren, diesem Dokument trotzdem beigelegt.

1-/2-/4-Channel Digital Potentiometers

AD8400/AD8402/AD8403

FEATURES

256-position variable resistance device
Replaces 1, 2, or 4 potentiometers
1 kΩ, 10 kΩ, 50 kΩ, 100 kΩ
Power shutdown—less than 5 µA
3-wire,SPI-compatible serial data input
10 MHz update data loading rate
2.7 V to 5.5 V single-supply operation

APPLICATIONS

Mechanical potentiometer replacement
Programmable filters, delays, time constants
Volume control, panning
Line impedance matching
Power supply adjustment

GENERAL DESCRIPTION

The AD8400/AD8402/AD8403 provide a single-, dual-, or quad-channel, 256-position, digitally controlled variable resistor (VR) device.[1] These devices perform the same electronic adjustment function as a mechanical potentiometer or variable resistor. The AD8400 contains a single variable resistor in the compact SOIC-8 package. The AD8402 contains two independent variable resistors in space-saving SOIC-14 surface-mount packages. The AD8403 contains four independent variable resistors in 24-lead PDIP, SOIC, and TSSOP packages. Each part contains a fixed resistor with a wiper contact that taps the fixed resistor value at a point determined by the digital code loaded into the controlling serial input register. The resistance between the wiper and either endpoint of the fixed resistor varies linearly with respect to the digital code transferred into the VR latch. Each variable resistor offers a completely programmable value of resistance between the A terminal and the wiper or the B terminal and the wiper. The fixed A-to-B terminal resistance of 1 kΩ, 10 kΩ, 50 kΩ, or 100 kΩ has a ±1% channel-to-channel matching tolerance with a nominal temperature coefficient of 500 ppm/°C. A unique switching circuit minimizes the high glitch inherent in traditional switched resistor designs, avoiding any make-before-break or break-before-make operation.

(continued on Page 3)

[1] The terms digital potentiometer, VR, and RDAC are used interchangeably.

FUNCTIONAL BLOCK DIAGRAM

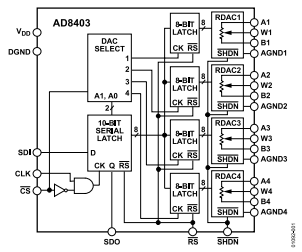


Figure 1.

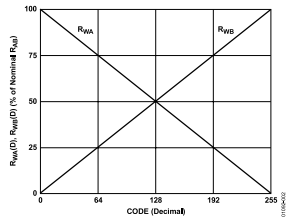


Figure 2. R_{WA} and R_{WB} vs. Code

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 **www.analog.com**
Fax: 781.461.3113

TABLE OF CONTENTS

REVISION HISTORY

10/05—Rev. C to Rev. D
Updated Format ... Universal
Changes to Features ... 1
Changes to Table 1 ... 4
Changes to Table 2 ... 6
Changes to Table 3 ... 8
Changes to Table 5 ... 11
Added Figure 36 ... 18
Replaced Figure 37 ... 19
Changes to Theory of Operation Section ... 20
Changes to Applications Section ... 24
Updated Outline Dimensions ... 26
Changes to Ordering Guide ... 28

11/01—Rev. B to Rev. C
Addition of new Figure ... 1
Edits to Specifications ... 2
Edits to Absolute Maximum Ratings ... 6
Edits to TPCs 1, 8, 12, 16, 20, 24, 35 ... 9
Edits to
the Programming the Variable Resistor Section ... 13

GENERAL DESCRIPTION

(continued from Page 1)

Each VR has its own VR latch that holds its programmed resistance value. These VR latches are updated from an SPI-compatible, serial-to-parallel shift register that is loaded from a standard 3-wire, serial-input digital interface. Ten data bits make up the data-word clocked into the serial input register.

The data-word is decoded where the first two bits determine the address of the VR latch to be loaded, and the last eight bits are the data. A serial data output pin at the opposite end of the serial register allows simple daisy chaining in multiple VR applications without additional external decoding logic.

The reset ($\overline{RS}$) pin forces the wiper to midscale by loading 80_H into the VR latch. The $\overline{SHDN}$ pin forces the resistor to an end-to-end open-circuit condition on the A terminal and shorts the wiper to the B terminal, achieving a microwatt power shutdown state. When $\overline{SHDN}$ is returned to logic high, the previous latch settings put the wiper in the same resistance setting prior to shutdown. The digital interface is still active in shutdown so that code changes can be made that will produce new wiper positions when the device is taken out of shutdown.

The AD8400 is available in the SOIC-8 surface mount. The AD8402 is available in both surface-mount (SOIC-14) and 14-lead PDIP packages, while the AD8403 is available in a narrow-body, 24-lead PDIP and a 24-lead, surface-mount package. The AD8402/AD8403 are also offered in the 1.1 mm thin TSSOP-14/TSSOP-24 packages for PCMCIA applications. All parts are guaranteed to operate over the extended industrial temperature range of −40°C to +125°C.

SPECIFICATIONS

ELECTRICAL CHARACTERISTICS—10 KΩ VERSION

V_{DD} = 3 V ± 10% or 5 V ± 10%, $V_A = V_{DD}$, V_B = 0 V, −40°C ≤ T_A ≤ +125°C, unless otherwise noted.

Table 1.

Parameter	Symbol	Conditions	Min	Typ[1]	Max	Unit
DC CHARACTERISTICS RHEOSTAT MODE (Specifications Apply to All VRs)						
Resistor Differential NL[2]	R-DNL	R_{WB}, V_A = no connect	−1	±1/4	+1	LSB
Resistor Nonlinearity[2]	R-INL	R_{WB}, V_A = no connect	−2	±1/2	+2	LSB
Nominal Resistance[3]	R_{AB}	T_A = 25°C, model: AD840XYY10	8	10	12	kΩ
Resistance Tempco	$\Delta R_{AB}/\Delta T$	$V_{AB} = V_{DD}$, wiper = no connect		500		ppm/°C
Wiper Resistance	R_W	V_{DD} = 5V, $I_W = V_{DD}/R_{AB}$		50	100	Ω
	R_W	V_{DD} = 3V, $I_W = V_{DD}/R_{AB}$		200		Ω
Nominal Resistance Match	$\Delta R/R_{AB}$	CH 1 to CH 2, CH 3, or CH 4, $V_{AB} = V_{DD}$, T_A = 25°C		0.2	1	%
DC CHARACTERISTICS POTENTIOMETER DIVIDER (Specifications Apply to All VRs)						
Resolution	N		8			Bits
Integral Nonlinearity[4]	INL		−2	±1/2	+2	LSB
Differential Nonlinearity[4]	DNL	V_{DD} = 5 V	−1	±1/4	+1	LSB
	DNL	V_{DD} = 3 V, T_A = 25°C	−1	±1/4	+1	LSB
	DNL	V_{DD} = 3 V, T_A = −40°C to +85°C	−1.5	±1/2	+1.5	LSB
Voltage Divider Tempco	$\Delta V_W/\Delta T$	Code = 80_H		15		ppm/°C
Full-Scale Error	V_{WFSE}	Code = FF_H	−4	−2.8	0	LSB
Zero-Scale Error	V_{WZSE}	Code = 00_H	0	1.3	2	LSB
RESISTOR TERMINALS						
Voltage Range[5]	$V_{A,B,W}$		0		V_{DD}	V
Capacitance[6] Ax, Capacitance Bx	$C_{A,B}$	f = 1 MHz, measured to GND, code = 80_H		75		pF
Capacitance[6] Wx	C_W	f = 1 MHz, measured to GND, code = 80_H		120		pF
Shutdown Current[7]	I_{A_SD}	$V_A = V_{DD}$, V_B = 0 V, $\overline{SHDN}$ = 0		0.01	5	µA
Shutdown Wiper Resistance	R_{W_SD}	$V_A = V_{DD}$, V_B = 0 V, $\overline{SHDN}$ = 0, V_{DD} = 5 V		100	200	Ω
DIGITAL INPUTS AND OUTPUTS						
Input Logic High	V_{IH}	V_{DD} = 5 V	2.4			V
Input Logic Low	V_{IL}	V_{DD} = 5 V			0.8	V
Input Logic High	V_{IH}	V_{DD} = 3 V	2.1			V
Input Logic Low	V_{IL}	V_{DD} = 3 V			0.6	V
Output Logic High	V_{OH}	R_L = 2.2 kΩ to V_{DD}	V_{DD} − 0.1			V
Output Logic Low	V_{OL}	I_{OL} = 1.6 mA, V_{DD} = 5 V			0.4	V
Input Current	I_I	V_{IN} = 0 V or 5 V, V_{DD} = 5 V			±1	µA
Input Capacitance[6]	C_{IL}			5		pF
POWER SUPPLIES						
Power Supply Range	V_{DD} range		2.7		5.5	V
Supply Current (CMOS)	I_{DD}	$V_{IH} = V_{DD}$ or V_{IL} = 0 V		0.01	5	µA
Supply Current (TTL)[8]	I_{DD}	V_{IH} = 2.4 V or 0.8 V, V_{DD} = 5.5 V		0.9	4	mA
Power Dissipation (CMOS)[9]	P_{DISS}	$V_{IH} = V_{DD}$ or V_{IL} = 0 V, V_{DD} = 5.5 V			27.5	µW
Power Supply Sensitivity	PSS	V_{DD} = 5 V ± 10%		0.0002	0.001	%/%
	PSS	V_{DD} = 3 V ± 10%		0.006	0.03	%/%

Parameter	Symbol	Conditions	Min	Typ[1]	Max	Unit
DYNAMIC CHARACTERISTICS[6, 10]						
Bandwidth −3 dB	BW_10 K	R = 10 kΩ		600		kHz
Total Harmonic Distortion	THD_W	V_A = 1 V rms + 2 V dc, V_B = 2 V dc, f = 1 kHz		0.003		%
V_W Settling Time	t_S	$V_A = V_{DD}$, V_B = 0 V, ±1% error band		2		µs
Resistor Noise Voltage	e_{NWB}	R_{WB} = 5 kΩ, f = 1 kHz, $\overline{RS}$ = 0		9		nV/√Hz
Crosstalk[11]	C_T	$V_A = V_{DD}$, V_B = 0 V		−65		dB

[1] Typical represents average readings at 25°C and V_{DD} = 5 V.
[2] Resistor position nonlinearity error R-INL is the deviation from an ideal value measured between the maximum resistance and the minimum resistance wiper positions. R-DNL measures the relative step change from ideal between successive tap positions. Parts are guaranteed monotonic. See the test circuit in Figure 38. I_W = 50 µA for V_{DD} = 3 V and I_W = 400 µA for V_{DD} = 5 V for the 10 kΩ versions.
[3] $V_{AB} = V_{DD}$, wiper (V_W) = no connect.
[4] INL and DNL are measured at V_W with the RDAC configured as a potentiometer divider similar to a voltage output D/A converter. $V_A = V_{DD}$ and V_B = 0 V. DNL specification limits of ±1 LSB maximum are guaranteed monotonic operating conditions. See the test circuit in Figure 37.
[5] Resistor Terminal A, Resistor Terminal B, and Resistor Terminal W have no limitations on polarity with respect to each other.
[6] Guaranteed by design and not subject to production test. Resistor-terminal capacitance tests are measured with 2.5 V bias on the measured terminal. The remaining resistor terminals are left open circuit.
[7] Measured at the Ax terminals. All Ax terminals are open-circuited in shutdown mode.
[8] Worst-case supply current is consumed when the input logic level is at 2.4 V, a standard characteristic of CMOS logic. See Figure 28 for a plot of I_{DD} vs. logic voltage.
[9] P_{DISS} is calculated from ($I_{DD} \times V_{DD}$). CMOS logic level inputs result in minimum power dissipation.
[10] All dynamic characteristics use V_{DD} = 5 V.
[11] Measured at a V_W pin where an adjacent V_W pin is making a full-scale voltage change.

ELECTRICAL CHARACTERISTICS—50 KΩ AND 100 KΩ VERSIONS

V_{DD} = 3 V ± 10% or 5 V ± 10%, $V_A = V_{DD}$, V_B = 0 V, −40°C ≤ T_A ≤ +125°C, unless otherwise noted.

Table 2.

Parameter	Symbol	Conditions	Min	Typ[1]	Max	Unit
DC CHARACTERISTICS RHEOSTAT MODE (Specifications Apply to All VRs)						
Resistor Differential NL[2]	R-DNL	R_{WB}, V_A = No Connect	−1	±1/4	+1	LSB
Resistor Nonlinearity[2]	R-INL	R_{WB}, V_A = No Connect	−2	±1/2	+2	LSB
Nominal Resistance[3]	R_{AB}	T_A = 25°C, Model: AD840XYY50	35	50	65	kΩ
	R_{AB}	T_A = 25°C, Model: AD840XYY100	70	100	130	kΩ
Resistance Tempco	$\Delta R_{AB}/\Delta T$	$V_{AB} = V_{DD}$, Wiper = No Connect		500		ppm/°C
Wiper Resistance	R_W	V_{DD} = 5V, $I_W = V_{DD}/R_{AB}$		50	100	Ω
	R_W	V_{DD} = 3V, $I_W = V_{DD}/R_{AB}$		200		Ω
Nominal Resistance Match	$\Delta R/R_{AB}$	CH 1 to CH 2, CH 3, or CH 4, $V_{AB} = V_{DD}$, T_A = 25°C		0.2	1	%
DC CHARACTERISTICS POTENTIOMETER DIVIDER (Specifications Apply to All VRs)						
Resolution	N		8			Bits
Integral Nonlinearity[4]	INL		−4	±1	+4	LSB
Differential Nonlinearity[4]	DNL	V_{DD} = 5 V	−1	±1/4	+1	LSB
	DNL	V_{DD} = 3 V, T_A = 25°C	−1	±1/4	+1	LSB
	DNL	V_{DD} = 3 V, T_A = −40°C to +85°C	−1.5	±1/2	+1.5	LSB
Voltage Divider Tempco	$\Delta V_W/\Delta T$	Code = 80_H		15		ppm/°C
Full-Scale Error	V_{WFSE}	Code = FF_H	−1	−0.25	0	LSB
Zero-Scale Error	V_{WZSE}	Code = 00_H	0	+0.1	+1	LSB
RESISTOR TERMINALS						
Voltage Range[5]	V_A, V_B, V_W		0		V_{DD}	V
Capacitance[6] Ax, Bx	C_A, C_B	f = 1 MHz, measured to GND, code = 80_H		15		pF
Capacitance[6] Wx	C_W	f = 1 MHz, measured to GND, code = 80_H		80		pF
Shutdown Current[7]	I_{A_SD}	$V_A = V_{DD}$, V_B = 0 V, $\overline{SHDN}$ = 0		0.01	5	µA
Shutdown Wiper Resistance	R_{W_SD}	$V_A = V_{DD}$, V_B = 0 V, $\overline{SHDN}$ = 0, V_{DD} = 5 V		100	200	Ω
DIGITAL INPUTS AND OUTPUTS						
Input Logic High	V_{IH}	V_{DD} = 5 V	2.4			V
Input Logic Low	V_{IL}	V_{DD} = 5 V			0.8	V
Input Logic High	V_{IH}	V_{DD} = 3 V	2.1			V
Input Logic Low	V_{IL}	V_{DD} = 3 V			0.6	V
Output Logic High	V_{OH}	R_L = 2.2 kΩ to V_{DD}	V_{DD} − 0.1			V
Output Logic Low	V_{OL}	I_{OL} = 1.6 mA, V_{DD} = 5 V			0.4	V
Input Current	I_{IL}	V_{IN} = 0 V or 5 V, V_{DD} = 5 V			±1	µA
Input Capacitance[6]	C_{IL}			5		pF
POWER SUPPLIES						
Power Supply Range	V_{DD} range		2.7		5.5	V
Supply Current (CMOS)	I_{DD}	$V_{IH} = V_{DD}$ or V_{IL} = 0 V		0.01	5	µA
Supply Current (TTL)[8]	I_{DD}	V_{IH} = 2.4 V or 0.8 V, V_{DD} = 5.5 V		0.9	4	mA
Power Dissipation (CMOS)[9]	P_{DISS}	$V_{IH} = V_{DD}$ or V_{IL} = 0 V, V_{DD} = 5.5 V			27.5	µW
Power Supply Sensitivity	PSS	V_{DD} = 5 V ± 10%		0.0002	0.001	%/%
	PSS	V_{DD} = 3 V ± 10%		0.006	0.03	%/%

Parameter	Symbol	Conditions	Min	Typ[1]	Max	Unit
DYNAMIC CHARACTERISTICS[6, 10]						
Bandwidth −3 dB	BW_50 K	R = 50 kΩ		125		kHz
	BW_100 K	R = 100 kΩ		71		kHz
Total Harmonic Distortion	THD_W	V_A = 1 V rms + 2 V dc, V_B = 2 V dc, f = 1 kHz		0.003		%
V_W Settling Time	t_S_50 K	$V_A = V_{DD}$, V_B = 0 V, ±1% error band		9		µs
	t_S_100 K	$V_A = V_{DD}$, V_B = 0 V, ±1% error band		18		µs
Resistor Noise Voltage	e_{NWB}_50 K	R_{WB} = 25 kΩ, f = 1 kHz, $\overline{RS}$ = 0		20		nV/√Hz
	e_{NWB}_100 K	R_{WB} = 50 kΩ, f = 1 kHz, $\overline{RS}$ = 0		29		nV/√Hz
Crosstalk[11]	C_T	$V_A = V_{DD}$, V_B = 0 V		−65		dB

[1] Typicals represent average readings at 25°C and V_{DD} = 5 V.
[2] Resistor position nonlinearity error R-INL is the deviation from an ideal value measured between the maximum resistance and the minimum resistance wiper positions. R-DNL measures the relative step change from ideal between successive tap positions. Parts are guaranteed monotonic. See the test circuit in Figure 38. $I_W = V_{DD}/R$ for V_{DD} = 3 V or 5 V for the 50 kΩ and 100 kΩ versions.
[3] $V_{AB} = V_{DD}$, wiper (V_W) = no connect.
[4] INL and DNL are measured at VW with the RDAC configured as a potentiometer divider similar to a voltage output D/A converter. $V_A = V_{DD}$ and V_B = 0 V. DNL specification limits of ±1 LSB maximum are guaranteed monotonic operating conditions. See the test circuit in Figure 37.
[5] Resistor Terminal A, Resistor Terminal B, and Resistor Terminal W have no limitations on polarity with respect to each other.
[6] Guaranteed by design and not subject to production test. Resistor-terminal capacitance tests are measured with 2.5 V bias on the measured terminal. The remaining resistor terminals are left open circuit.
[7] Measured at the Ax terminals. All Ax terminals are open-circuited in shutdown mode.
[8] Worst-case supply current consumed when input logic level at 2.4 V, standard characteristic of CMOS logic. See Figure 28 for a plot of I_{DD} vs. logic voltage.
[9] P_{DISS} is calculated from ($I_{DD} \times V_{DD}$). CMOS logic level inputs result in minimum power dissipation.
[10] All dynamic characteristics use V_{DD} = 5 V.
[11] Measured at a V_W pin where an adjacent V_W pin is making a full-scale voltage change.

ELECTRICAL CHARACTERISTICS—1 KΩ VERSION

V_{DD} = 3 V ± 10% or 5 V ± 10%, $V_A = V_{DD}$, V_B = 0 V, −40°C ≤ T_A ≤ +125°C, unless otherwise noted.

Table 3.

Parameter	Symbol	Conditions	Min	Typ[1]	Max	Unit
DC CHARACTERISTICS RHEOSTAT MODE (Specifications Apply to All VRs)						
Resistor Differential NL[2]	R-DNL	R_{WB}, V_A = no connect	−5	−1	+3	LSB
Resistor Nonlinearity[2]	R-INL	R_{WB}, V_A = no connect	−4	±1.5	+4	LSB
Nominal Resistance[3]	R_{AB}	T_A = 25°C, model: AD840XYY1	0.8	1.2	1.6	kΩ
Resistance Tempco	$\Delta R_{AB}/\Delta T$	$V_{AB} = V_{DD}$, wiper = no connect		700		ppm/°C
Wiper Resistance	R_W	V_{DD} = 5V, $I_W = V_{DD}/R_{AB}$		53	100	Ω
	R_W	V_{DD} = 3V, $I_W = V_{DD}/R_{AB}$		200		Ω
Nominal Resistance Match	$\Delta R/R_{AB}$	CH 1 to CH 2, $V_{AB} = V_{DD}$, T_A = 25°C		0.75	2	%
DC CHARACTERISTICS POTENTIOMETER DIVIDER (Specifications Apply to All VRs)						
Resolution	N		8			Bits
Integral Nonlinearity[4]	INL		−6	±2	+6	LSB
Differential Nonlinearity[4]	DNL	V_{DD} = 5 V	−4	−1.5	+2	LSB
	DNL	V_{DD} = 3 V, T_A = 25°C	−5	−2	+5	LSB
Voltage Divider Temperature Coefficient	$\Delta V_W/\Delta T$	Code = 80H		25		ppm/°C
Full-Scale Error	V_{WFSE}	Code = FF_H	−20	−12	0	LSB
Zero-Scale Error	V_{WZSE}	Code = 00_H	0	6	10	LSB
RESISTOR TERMINALS						
Voltage Range[5]	V_A, V_B, V_W		0		V_{DD}	V
Capacitance[6] Ax, Bx	C_A, C_B	f = 1 MHz, measured to GND, code = 80_H		75		pF
Capacitance[6] Wx	C_W	f = 1 MHz, measured to GND, code = 80_H		120		pF
Shutdown Supply Current[7]	I_{A_SD}	$V_A = V_{DD}$, V_B = 0 V, $\overline{SHDN}$ = 0		0.01	5	µA
Shutdown Wiper Resistance	R_{W_SD}	$V_A = V_{DD}$, V_B = 0 V, $\overline{SHDN}$ = 0, V_{DD} = 5 V		50	100	Ω
DIGITAL INPUTS AND OUTPUTS						
Input Logic High	V_{IH}	V_{DD} = 5 V	2.4			V
Input Logic Low	V_{IL}	V_{DD} = 5 V			0.8	V
Input Logic High	V_{IH}	V_{DD} = 3 V	2.1			V
Input Logic Low	V_{IL}	V_{DD} = 3 V			0.6	V
Output Logic High	V_{OH}	R_L = 2.2 kΩ to V_{DD}	V_{DD} − 0.1			V
Output Logic Low	V_{OL}	I_{OL} = 1.6 mA, V_{DD} = 5 V			0.4	V
Input Current	I_{IL}	V_{IN} = 0 V or 5 V, V_{DD} = 5 V			±1	µA
Input Capacitance[6]	C_{IL}			5		pF
POWER SUPPLIES						
Power Supply Range	V_{DD} range		2.7		5.5	V
Supply Current (CMOS)	I_{DD}	$V_{IH} = V_{DD}$ or V_{IL} = 0 V		0.01	5	µA
Supply Current (TTL)[8]	I_{DD}	V_{IH} = 2.4 V or 0.8 V, V_{DD} = 5.5 V		0.9	4	mA
Power Dissipation (CMOS)[9]	P_{DISS}	$V_{IH} = V_{DD}$ or V_{IL} = 0 V, V_{DD} = 5.5 V			27.5	µW
Power Supply Sensitivity	PSS	ΔV_{DD} = 5 V ± 10%		0.0035	0.008	%/%
	PSS	ΔV_{DD} = 3 V ± 10%		0.05	0.13	%/%

Parameter	Symbol	Conditions	Min	Typ[1]	Max	Unit
DYNAMIC CHARACTERISTICS[6, 10]						
Bandwidth −3 dB	BW_1 K	R = 1 kΩ		5,000		kHz
Total Harmonic Distortion	THD_W	V_A = 1 V rms + 2 V dc, V_B = 2 V dc, f = 1 kHz		0.015		%
V_W Settling Time	t_S	$V_A = V_{DD}$, V_B = 0 V, ±1% error band		0.5		µs
Resistor Noise Voltage	e_{NWB}	R_{WB} = 500 Ω, f = 1 kHz, $\overline{RS}$ = 0		3		nV/√Hz
Crosstalk[11]	C_T	$V_A = V_{DD}$, V_B = 0 V		−65		dB

[1] Typicals represent average readings at 25°C and V_{DD} = 5 V.
[2] Resistor position nonlinearity error R-INL is the deviation from an ideal value measured between the maximum resistance and the minimum resistance wiper positions. R-DNL measures the relative step change from ideal between successive tap positions. See the test circuit in Figure 38. I_W = 500 µA for V_{DD} = 3 V and I_W = 2.5 mA for V_{DD} = 5 V for 1 kΩ version.
[3] $V_{AB} = V_{DD}$, wiper (V_W) = no connect.
[4] INL and DNL are measured at VW with the RDAC configured as a potentiometer divider similar to a voltage output D/A converter. $V_A = V_{DD}$ and V_B = 0 V. DNL specification limits of ±1 LSB maximum are guaranteed monotonic operating conditions. See the test circuit in Figure 37.
[5] Resistor Terminal A, Resistor Terminal B, and Resistor Terminal W have no limitations on polarity with respect to each other.
[6] Guaranteed by design and not subject to production test. Resistor-terminal capacitance tests are measured with 2.5 V bias on the measured terminal. The remaining resistor terminals are left open circuit.
[7] Measured at the Ax terminals. All Ax terminals are open-circuited in shutdown mode.
[8] Worst-case supply current is consumed when the input logic level is at 2.4 V, a standard characteristic of CMOS logic. See Figure 28 for a plot of I_{DD} vs. logic voltage.
[9] P_{DIS} is calculated from ($I_{DD} \times V_{DD}$). CMOS logic level inputs result in minimum power dissipation.
[10] All dynamic characteristics use V_{DD} = 5 V.
[11] Measured at a V_W pin where an adjacent V_W pin is making a full-scale voltage change.

ELECTRICAL CHARACTERISTICS—ALL VERSIONS

V_{DD} = 3 V ± 10% or 5 V ± 10%, $V_A = V_{DD}$, V_B = 0 V, −40°C ≤ T_A ≤ +125°C, unless otherwise noted.

Table 4.

Parameter	Symbol	Conditions	Min	Typ[1]	Max	Unit
SWITCHING CHARACTERISTICS[2, 3]						
Input Clock Pulse Width	t_{CH}, t_{CL}	Clock level high or low	10			ns
Data Setup Time	t_{DS}		5			ns
Data Hold Time	t_{DH}		5			ns
CLK to SDO Propagation Delay[4]	t_{PD}	R_L = 1 kΩ to 5 V, C_L ≤ 20 pF	1		25	ns
$\overline{CS}$ Setup Time	t_{CSS}		10			ns
$\overline{CS}$ High Pulse Width	t_{CSW}		10			ns
Reset Pulse Width	t_{RS}		50			ns
CLK Fall to $\overline{CS}$ Rise Hold Time	t_{CSH}		0			ns
$\overline{CS}$ Rise to Clock Rise Setup	t_{CS1}		10			ns

[1] Typicals represent average readings at 25°C and V_{DD} = 5 V.
[2] Guaranteed by design and not subject to production test. Resistor-terminal capacitance tests are measured with 2.5 V bias on the measured terminal. The remaining resistor terminals are left open circuit.
[3] See the timing diagram in Figure 3 for location of measured values. All input control voltages are specified with $t_R = t_F$ = 1 ns (10% to 90% of V_{DD}) and timed from a voltage level of 1.6 V. Switching characteristics are measured using V_{DD} = 3 V or 5 V. To avoid false clocking, a minimum input logic slew rate of 1 V/µs should be maintained.
[4] Propagation delay depends on the value of V_{DD}, R_L, and C_L (see the Applications section).

TIMING DIAGRAMS

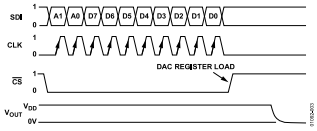


Figure 3. Timing Diagram

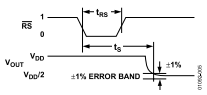


Figure 5. Reset Timing Diagram

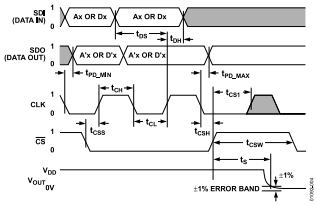


Figure 4. Detailed Timing Diagram

ABSOLUTE MAXIMUM RATINGS

T_A = 25°C, unless otherwise noted.

Table 5.

Parameter	Rating
V_{DD} to GND	−0.3 V, +8 V
V_A, V_B, V_W to GND	0 V, V_{DD}
Maximum Current	
I_{WB}, I_{WA} Pulsed	±20 mA
I_{WB} Continuous (R_{WB} ≤ 1 kΩ, A Open)[1]	±5 mA
I_{WA} Continuous (R_{WA} ≤ 1 kΩ, B Open)[1]	±5 mA
I_{AB} Continuous (R_{AB} = 1 kΩ/10 kΩ/50 kΩ/100 kΩ)[1]	±5 mA/±500 µA/±100 µA/±50 µA
Digital Input and Output Voltage to GND	0 V, 7 V
Operating Temperature Range	−40°C to +125°C
Maximum Junction Temperature (T_J Maximum)	150°C
Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec)	300°C
Package Power Dissipation	(T_J max − T_A)/θ_{JA}
Thermal Resistance (θ_{JA})	
SOIC (R-8)	158°C/W
PDIP (N-14)	83°C/W
PDIP (N-24)	63°C/W
SOIC (R-14)	120°C/W
SOIC (R-24)	70°C/W
TSSOP-14 (RU-14)	180°C/W
TSSOP-24 (RU-24)	143°C/W

[1] Maximum terminal current is bounded by the maximum applied voltage across any two of the A, B, and W terminals at a given resistance, the maximum current handling of the switches, and the maximum power dissipation of the package; VDD = 5 V.

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

SERIAL DATA-WORD FORMAT

Table 6.

ADDR		DATA							
B9	**B8**	**B7**	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**
A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
MSB	LSB	MSB							LSB
2^9	2^8	2^7							2^0

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although this product features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

PIN CONFIGURATIONS AND FUNCTION DESCRIPTIONS

Figure 6. AD8400 Pin Configuration

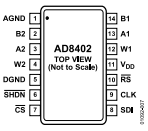


Figure 7. AD8402 Pin Configuration

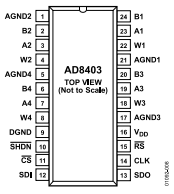


Figure 8. AD8403 Pin Configuration

Table 7. AD8400 Pin Function Descriptions

Pin No.	Mnemonic	Description
1	B1	Terminal B RDAC.
2	GND	Ground.
3	$\overline{CS}$	Chip Select Input, Active Low. When $\overline{CS}$ returns high, data in the serial input register is decoded, based on the address bits, and loaded into the target DAC register.
4	SDI	Serial Data Input.
5	CLK	Serial Clock Input, Positive Edge Triggered.
6	V_{DD}	Positive Power Supply. Specified for operation at both 3 V and 5 V.
7	W1	Wiper RDAC, Addr = 00_2.
8	A1	Terminal A RDAC.

Table 8. AD8402 Pin Function Descriptions

Pin No.	Mnemonic	Description
1	AGND	Analog Ground.[1]
2	B2	Terminal B RDAC 2.
3	A2	Terminal A RDAC 2.
4	W2	Wiper RDAC 2, Addr = 01_2.
5	DGND	Digital Ground.[1]
6	$\overline{SHDN}$	Terminal A Open Circuit. Shutdown controls Variable Resistor 1 and Variable Resistor 2.
7	$\overline{CS}$	Chip Select Input, Active Low. When $\overline{CS}$ returns high, data in the serial input register is decoded, based on the address bits, and loaded into the target DAC register.
8	SDI	Serial Data Input.
9	CLK	Serial Clock Input, Positive Edge Triggered.
10	$\overline{RS}$	Active Low Reset to Midscale. Sets RDAC registers to 80_H.
11	V_{DD}	Positive Power Supply. Specified for operation at both 3 V and 5 V
12	W1	Wiper RDAC 1, Addr = 00_2.
13	A1	Terminal A RDAC 1.
14	B1	Terminal B RDAC 1.

[1] All AGND pins must be connected to DGND.

Table 9. AD8403 Pin Function Descriptions

Pin No.	Mnemonic	Description
1	AGND2	Analog Ground 2.[1]
2	B2	Terminal B RDAC 2.
3	A2	Terminal A RDAC 2.
4	W2	Wiper RDAC 2, Addr = 01_2.
5	AGND4	Analog Ground 4.[1]
6	B4	Terminal B RDAC 4.
7	A4	Terminal A RDAC 4.
8	W4	Wiper RDAC 4, Addr = 11_2.
9	DGND	Digital Ground.[1]
10	$\overline{SHDN}$	Active Low Input. Terminal A open circuit. Shutdown controls Variable Resistor 1 through Variable Resistor 4.
11	$\overline{CS}$	Chip Select Input, Active Low. When $\overline{CS}$ returns high, data in the serial input register is decoded, based on the address bits, and loaded into the target DAC register.
12	SDI	Serial Data Input.
13	SDO	Serial Data Output. Open drain transistor requires a pull-up resistor.
14	CLK	Serial Clock Input, Positive Edge Triggered.
15	$\overline{RS}$	Active Low Reset to Midscale. Sets RDAC registers to 80_H.
16	V_{DD}	Positive Power Supply. Specified for operation at both 3 V and 5 V.
17	AGND3	Analog Ground 3.[1]
18	W3	Wiper RDAC 3, Addr = 10_2.
19	A3	Terminal A RDAC 3.
20	B3	Terminal B RDAC 3.
21	AGND1	Analog Ground 1.[1]
22	W1	Wiper RDAC 1, Addr = 00_2.
23	A1	Terminal A RDAC 1.
24	B1	Terminal B RDAC 1.

[1] All AGND pins must be connected to DGND.

TYPICAL PERFORMANCE CHARACTERISTICS

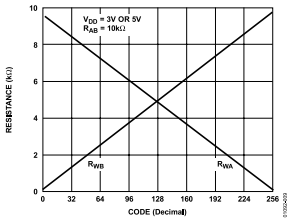


Figure 9. Wiper to End Terminal Resistance vs. Code

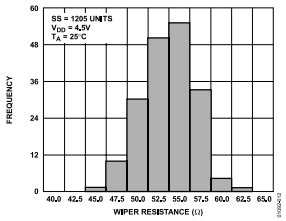


Figure 12. 10 kΩ Wiper-Contact-Resistance Histogram

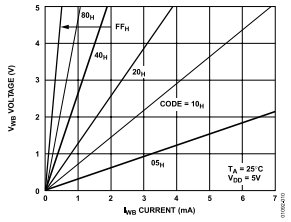


Figure 10. Resistance Linearity vs. Conduction Current

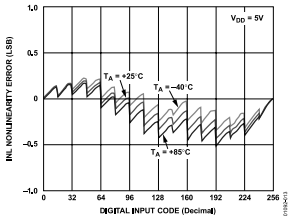


Figure 13. Potentiometer Divider Nonlinearity Error vs. Code

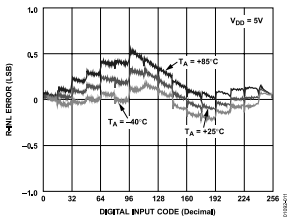


Figure 11. Resistance Step Position Nonlinearity Error vs. Code

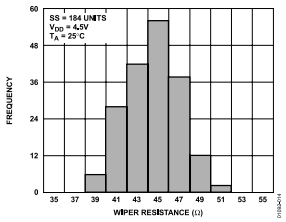


Figure 14. 50 kΩ Wiper-Contact-Resistance Histogram

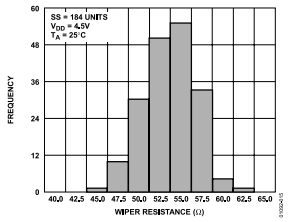


Figure 15. 100 kΩ Wiper-Contact-Resistance Histogram

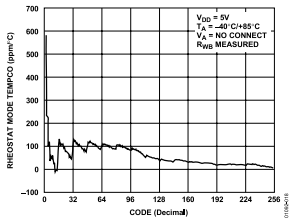


Figure 18. $\Delta R_{WB}/\Delta T$ Rheostat Mode Tempco

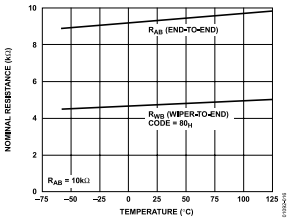


Figure 16. Nominal Resistance vs. Temperature

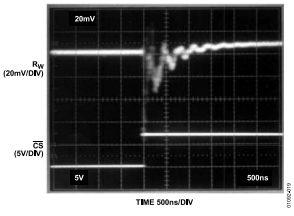


Figure 19. One Position Step Change at Half-Scale (Code $7F_H$ to 80_H)

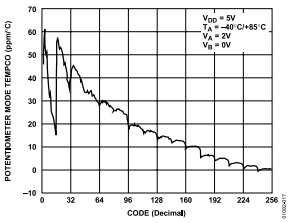


Figure 17. $\Delta V_{WB}/\Delta T$ Potentiometer Mode Tempco

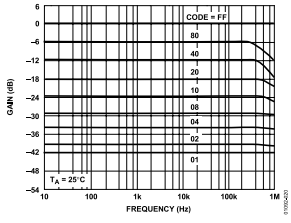


Figure 20. 10 kΩ Gain vs. Frequency vs. Code
(See Figure 43)

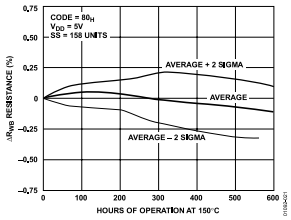


Figure 21. Long-Term Drift Accelerated by Burn-In

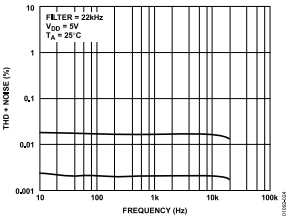


Figure 24. Total Harmonic Distortion Plus Noise vs. Frequency
(See Figure 41 and Figure 42)

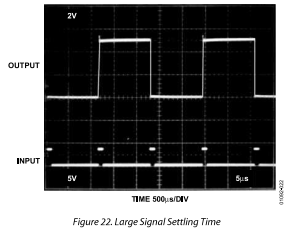


Figure 22. Large Signal Settling Time

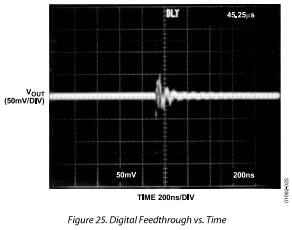


Figure 25. Digital Feedthrough vs. Time

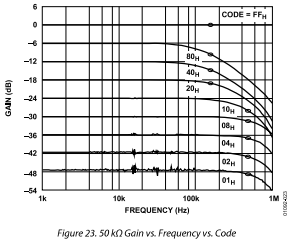


Figure 23. 50 kΩ Gain vs. Frequency vs. Code

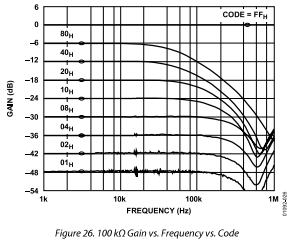


Figure 26. 100 kΩ Gain vs. Frequency vs. Code

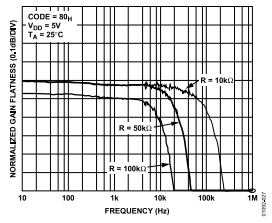


Figure 27. Normalized Gain Flatness vs. Frequency (See Figure 43)

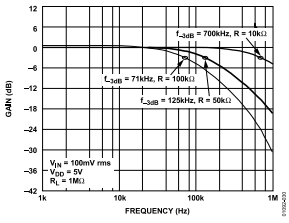


Figure 30. –3 dB Bandwidths

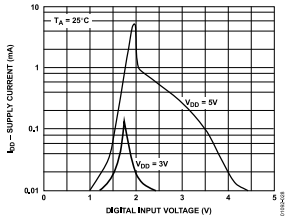


Figure 28. Supply Current vs. Digital Input Voltage

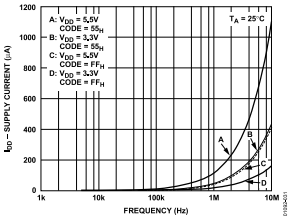


Figure 31. Supply Current vs. Clock Frequency

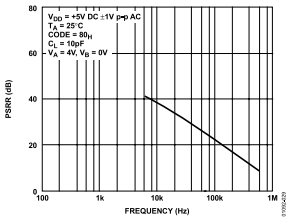


Figure 29. Power Supply Rejection Ratio vs. Frequency (See Figure 40)

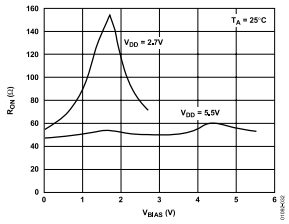


Figure 32. AD8403 Incremental Wiper On Resistance vs. V_{DD} (See Figure 39)

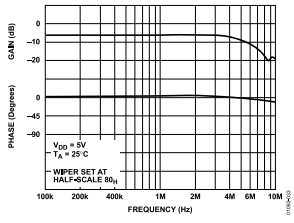


Figure 33. 1 kΩ Gain and Phase vs. Frequency

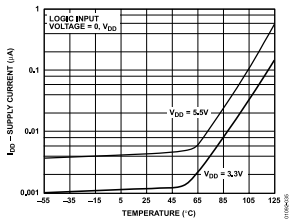


Figure 35. Supply Current vs. Temperature

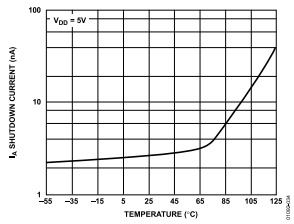


Figure 34. Shutdown Current vs. Temperature

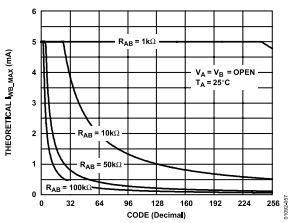


Figure 36. I_{WB_MAX} vs. Code

TEST CIRCUITS

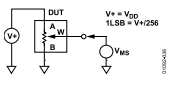


Figure 37. Potentiometer Divider Nonlinearity Error (INL, DNL)

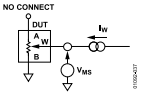


Figure 38. Resistor Position Nonlinearity Error (Rheostat Operations; R-INL, R-DNL)

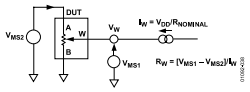


Figure 39. Wiper Resistance

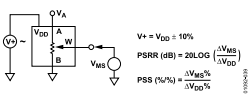


Figure 40. Power Supply Sensitivity (PSS, PSRR)

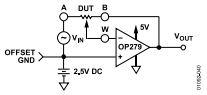


Figure 41. Inverting Programmable Gain

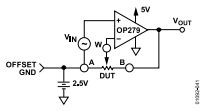


Figure 42. Noninverting Programmable Gain

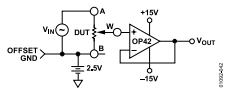


Figure 43. Gain vs. Frequency

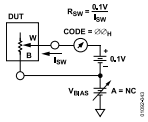


Figure 44. Incremental On Resistance

THEORY OF OPERATION

The AD8400/AD8402/AD8403 provide a single, dual, and quad channel, 256-position, digitally controlled variable resistor (VR) device. Changing the programmed VR setting is accomplished by clocking in a 10-bit serial data-word into the SDI (Serial Data Input) pin. The format of this data-word is two address bits, MSB first, followed by eight data bits, also MSB first. Table 6 provides the serial register data-word format. The AD8400/AD8402/AD8403 have the following address assignments for the ADDR decoder, which determines the location of the VR latch receiving the serial register data in Bit B7 to Bit B0:

$$VR\# = A1 \times 2 + A0 + 1 \quad (1)$$

The single-channel AD8400 requires A1 = A0 = 0. The dual-channel AD8402 requires A1 = 0. VR settings can be changed one at a time in random sequence. A serial clock running at 10 MHz makes it possible to load all four VRs under 4 µs (10 × 4 × 100 ns) for AD8403. The exact timing requirements are shown in Figure 3, Figure 4, and Figure 5.

The AD8400/AD8402/AD8403 do not have power-on midscale preset, so the wiper can be at any random position at power-up. However, the AD8402/AD8403 can be reset to midscale by asserting the $\overline{RS}$ pin, simplifying initial conditions at power-up. Both parts have a power shutdown $\overline{SHDN}$ pin that places the VR in a zero-power-consumption state where Terminal Ax is open-circuited and the Wiper Wx is connected to Terminal Bx, resulting in the consumption of only the leakage current in the VR. In shutdown mode, the VR latch settings are maintained so that upon returning to the operational mode, the VR settings return to the previous resistance values. The digital interface is still active in shutdown, except that SDO is deactivated. Code changes in the registers can be made during shutdown that will produce new wiper positions when the device is taken out of shutdown.

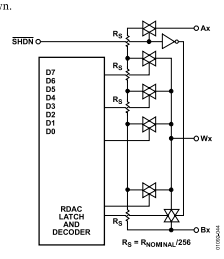


Figure 45. AD8402/AD8403 Equivalent VR (RDAC) Circuit

PROGRAMMING THE VARIABLE RESISTOR

Rheostat Operation

The nominal resistance of the VR (RDAC) between Terminal A and Terminal B is available with values of 1 kΩ, 10 kΩ, 50 kΩ, and 100 kΩ. The final digits of the part number determine the nominal resistance value; that is, 10 kΩ = 10; 100 kΩ = 100. The nominal resistance (R_{AB}) of the VR has 256 contact points accessible by the wiper terminal, and the resulting resistance can be measured either across the wiper and B terminals (R_{WB}) or across the wiper and A terminals (R_{WA}). The 8-bit data-word loaded into the RDAC latch is decoded to select one of the 256 possible settings. The wiper's first connection starts at the B terminal for data 00_H. This B terminal connection has a wiper contact resistance of 50 Ω. The second connection (for the 10 kΩ part) is the first tap point located at 89 Ω = [R_{AB} (nominal resistance) + R_W = 39 Ω + 50 Ω] for data 01_H. The third connection is the next tap point representing 78 Ω + 50 Ω = 128 Ω for data 02_H. Each LSB data value increase moves the wiper up the resistor ladder until the last tap point is reached at 10,011 Ω. Note that the wiper does not directly connect to the B terminal even for data 00_H. See Figure 45 for a simplified diagram of the equivalent RDAC circuit.

The AD8400 contains one RDAC, the AD8402 contains two independent RDACs, and the AD8403 contains four independent RDACs. The general transfer equation that determines the digitally programmed output resistance between Wx and Bx is

$$R_{WB}(D) = \frac{D}{256} \times R_{AB} + R_W \quad (2)$$

where D, in decimal, is the data loaded into the 8-bit RDAC# latch, and R_{AB} is the nominal end-to-end resistance.

For example, when the A terminal is either open-circuited or tied to the Wiper W, the following RDAC latch codes result in the following R_{WB} (for the 10 kΩ version):

Table 10.

D (Dec)	R_{WB} (Ω)	Output State
255	10,011	Full scale
128	5,050	Midscale ($\overline{RS}$ = 0 condition)
1	89	1 LSB
0	50	Zero-scale (wiper contact resistance)

Note that in the zero-scale condition, a finite wiper resistance of 50 Ω is present. Care should be taken to limit the current flow between W and B in this state to a maximum value of 5 mA to avoid degradation or possible destruction of the internal switch contact.

Like a mechanical potentiometer, RDAC is symmetrical. The resistance between the Wiper W and Terminal A also produces a digitally controlled complementary resistance, R_{WA}. When these terminals are used, the B terminal can be tied to the wiper or left floating. R_{WA} starts at the maximum and decreases as the data loaded into the RDAC latch increases. The general transfer equation for this R_{WA} is

$$R_{WA}(D) = \frac{256 - D}{256} \times R_{AB} + R_W \quad (3)$$

where D is the data loaded into the 8-bit RDAC# latch, and R_{AB} is the nominal end-to-end resistance.

For example, when the B terminal is either open-circuited or tied to the Wiper W, the following RDAC latch codes result in the following R_{WA} (for the 10 kΩ version):

Table 11.

D (Dec)	R_{WA} (Ω)	Output State
255	89	Full-Scale
128	5,050	Midscale ($\overline{RS}$ = 0 Condition)
1	10,011	1 LSB
0	10,050	Zero-Scale

The typical distribution of RAB from channel to channel matches within ±1%. However, device-to-device matching is process lot dependent and has a ±20% variation. The temperature coefficient, or the change in R_{AB} with temperature, is 500 ppm/°C.

The wiper-to-end-terminal resistance temperature coefficient has the best performance over the 10% to 100% of adjustment range where the internal wiper contact switches do not contribute any significant temperature related errors. The graph in Figure 18 shows the performance of R_{WB} tempco vs. code. Using the potentiometer with codes below 32 results in the larger temperature coefficients plotted.

PROGRAMMING THE POTENTIOMETER DIVIDER

Voltage Output Operation

The digital potentiometer easily generates an output voltage proportional to the input voltage applied to a given terminal.

For example, connecting the A terminal to 5 V and the B terminal to ground produces an output voltage at the wiper starting at 0 V up to 1 LSB less than 5 V. Each LSB is equal to the voltage applied across the A to B terminals divided by the 256-position resolution of the potentiometer divider. The general equation defining the output voltage with respect to ground for any given input voltage applied to the A to B terminals is

$$V_W = \frac{D}{256} \times V_{AB} + V_B \quad (4)$$

Operation of the digital potentiometer in the voltage divider mode results in more accurate operation over temperature. Here the output voltage is dependent on the ratio of the internal resistors, not the absolute value; therefore, the temperature drift improves to 15 ppm/°C.

At the lower wiper position settings, the potentiometer divider temperature coefficient increases because the contribution of the CMOS switch wiper resistance becomes an appreciable portion of the total resistance from the B terminal to the Wiper W. See Figure 17 for a plot of potentiometer tempco performance vs. code setting.

DIGITAL INTERFACING

The AD8400/AD8402/AD8403 contain a standard SPI-compatible, 3-wire, serial input control interface. The three inputs are clock (CLK), chip select ($\overline{CS}$), and serial data input (SDI). The positive-edge sensitive CLK input requires clean transitions to avoid clocking incorrect data into the serial input register. For the best result, use logic transitions faster than 1 V/µs. Standard logic families work well. If mechanical switches are used for product evaluation, they should be debounced by a flip-flop or other suitable means. The block diagrams in Figure 46, Figure 47, and Figure 48 show the internal digital circuitry in more detail. When $\overline{CS}$ is taken active low, the clock loads data into the 10-bit serial register on each positive clock edge (see Table 12).

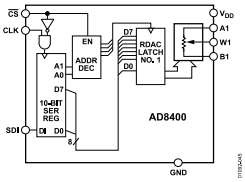


Figure 46. AD8400 Block Diagram

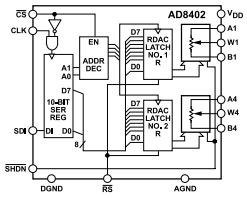


Figure 47. AD8402 Block Diagram

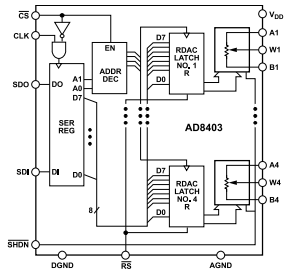


Figure 48. AD8403 Block Diagram

Table 12. Input Logic Control Truth Table[1]

CLK	$\overline{CS}$	$\overline{RS}$	$\overline{SHDN}$	Register Activity
L	L	H	H	No SR effect; enables SDO pin
P	L	H	H	Shift one bit in from the SDI pin. The 10th previously entered bit is shifted out of the SDO pin.
X	P	H	H	Load SR data into RDAC latch based on A1, A0 decode (Table 13).
X	H	H	H	No operation
X	X	L	H	Sets all RDAC latches to midscale, wiper centered, and SDO latch cleared
X	H	P	H	Latches all RDAC latches to 80_H
X	H	H	L	Open-circuits all Resistor A terminals, connects W to B, turns off SDO output transistor.

[1] P = positive edge, X = don't care, SR = shift register

The serial data output (SDO) pin, which exists only on the AD8403 and not on the AD8400 or AD8402, contains an open-drain, n-channel FET that requires a pull-up resistor to transfer data to the SDI pin of the next package. The pull-up resistor termination voltage may be larger than the V_{DD} supply (but less than the max V_{DD} of 8 V) of the AD8403 SDO output device. For example, the AD8403 could operate at V_{DD} = 3.3 V, and the pull-up for interface to the next device could be set at 5 V. This allows for daisy-chaining several RDACs from a single processor serial data line. The clock period needs to be increased when using a pull-up resistor to the SDI pin of the following device in the series. Capacitive loading at the daisy-chain node SDO to SDI between devices must be accounted for in order to transfer data successfully. When daisy chain is used, $\overline{CS}$ should be kept low until all the bits of every package are clocked into their respective serial registers and the address and data bits are in the proper decoding location.

If two AD8403 RDACs are daisy-chained, it requires 20 bits of address and data in the format shown in Table 6. During shutdown ($\overline{SHDN}$ = logic low), the SDO output pin is forced to the off (logic high) state to disable power dissipation in the pull-up resistor. See Figure 50 for equivalent SDO output circuit schematic.

The data setup and hold times in the specification table determine the data valid time requirements. The last 10 bits of the data-word entered into the serial register are held when $\overline{CS}$ returns high. At the same time $\overline{CS}$ goes high it gates the address decoder, which enables one of the two (AD8402) or four (AD8403) positive edge-triggered RDAC latches. See Figure 49 and Table 13.

Table 13. Address Decode Table

A1	A0	Latch Decoded
0	0	RDAC#1
0	1	RDAC#2
1	0	RDAC#3 AD8403 Only
1	1	RDAC#4 AD8403 Only

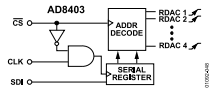


Figure 49. Equivalent Input Control Logic

The target RDAC latch is loaded with the last eight bits of the serial data-word completing one RDAC update. In the case of AD8403, four separate 10-bit data-words must be clocked in to change all four VR settings.

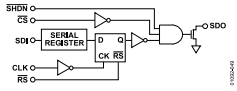


Figure 50. Detailed SDO Output Schematic of the AD8403

All digital pins are protected with a series input resistor and parallel Zener ESD structure shown in Figure 51. This structure applies to digital pins $\overline{CS}$, SDI, SDO, $\overline{RS}$, $\overline{SHDN}$, and CLK. The digital input ESD protection allows for mixed power supply applications where 5 V CMOS logic can be used to drive an AD8400, AD8402, or AD8403 operating from a 3 V power supply. Analog Pin A, Pin B, and Pin W are protected with a 20 Ω series resistor and parallel Zener diode (see Figure 52).

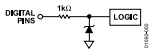


Figure 51. Equivalent ESD Protection Circuits

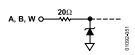


Figure 52. Equivalent ESD Protection Circuit (Analog Pins)

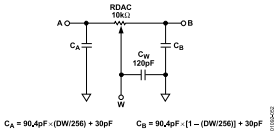


Figure 53. RDAC Circuit Simulation Model for RDAC = 10 kΩ

The AC characteristics of the RDAC are dominated by the internal parasitic capacitances and the external capacitive loads. The −3 dB bandwidth of the AD8403AN10 (10 kΩ resistor) measures 600 kHz at half scale as a potentiometer divider. Figure 30 provides the large signal Bode plot characteristics of the three available resistor versions 10 kΩ, 50 kΩ, and 100 kΩ. The gain flatness vs. frequency graph of the 1 kΩ version predicts filter applications performance (see Figure 33). A parasitic simulation model has been developed and is shown in Figure 53. Listing I provides a macro model net list for the 10 kΩ RDAC.

Listing I. Macro Model Net List for RDAC

```
.PARAM DW=255, RDAC=10E3
*
.SUBCKT DPOT (A,W,)
*
CA A 0 {DW/256*90.4E-12+30E-12}
RAW A W {(1-DW/256)*RDAC+50}
CW W 0 120E-12
RBW W B {DW/256*RDAC+50}
CB B 0 {(1-DW/256)*90.4E-12+30E-12}
*
.ENDS DPOT
```

The total harmonic distortion plus noise (THD + N), shown in Figure 41, is measured at 0.003% in an inverting op amp circuit using an offset ground and a rail-to-rail OP279 amplifier. Thermal noise is primarily Johnson noise, typically 9 nV/√Hz for the 10 kΩ version at f = 1 kHz. For the 100 kΩ device, thermal noise becomes 29 nV/√Hz. Channel-to-channel crosstalk measures less than −65 dB at f = 100 kHz. To achieve this isolation, the extra ground pins provided on the package to segregate the individual RDACs must be connected to circuit ground. AGND and DGND pins should be at the same voltage potential. Any unused potentiometers in a package should be connected to ground. Power supply rejection is typically −35 dB at 10 kHz. Care is needed to minimize power supply ripple in high accuracy applications.

APPLICATIONS

The digital potentiometer (RDAC) allows many of the applications of a mechanical potentiometer to be replaced by a solid-state solution offering compact size and freedom from vibration, shock, and open contact problems encountered in hostile environments. A major advantage of the digital potentiometer is its programmability. Any settings can be saved for later recall in system memory.

The two major configurations of the RDAC include the potentiometer divider (basic 3-terminal application) and the rheostat (2-terminal configuration) connections shown in Figure 37 and Figure 38.

Certain boundary conditions must be satisfied for proper AD8400/AD8402/AD8403 operation. First, all analog signals must remain within the GND to V_{DD} range used to operate the single-supply AD8400/AD8402/AD8403. For standard potentiometer divider applications, the wiper output can be used directly. For low resistance loads, buffer the wiper with a suitable rail-to-rail op amp such as the OP291 or the OP279. Second, for ac signals and bipolar dc adjustment applications, a virtual ground is generally needed. Whichever method is used to create the virtual ground, the result must provide the necessary sink and source current for all connected loads, including adequate bypass capacitance. Figure 41 shows one channel of the AD8402 connected in an inverting programmable gain amplifier circuit. The virtual ground is set at 2.5 V, which allows the circuit output to span a ±2.5 V range with respect to virtual ground. The rail-to-rail amplifier capability is necessary for the widest output swing. As the wiper is adjusted from its midscale reset position (80_H) toward the A terminal (code FF_H), the voltage gain of the circuit is increased in successively larger increments. Alternatively, as the wiper is adjusted toward the B terminal (code 00_H), the signal becomes attenuated. The plot in Figure 54 shows the wiper settings for a 100:1 range of voltage gain (V/V). Note the ±10 dB of pseudologarithmic gain around 0 dB (1 V/V). This circuit is mainly useful for gain adjustments in the range of 0.14 V/V to 4 V/V; beyond this range the step sizes become very large, and the resistance of the driving circuit can become a significant term in the gain equation.

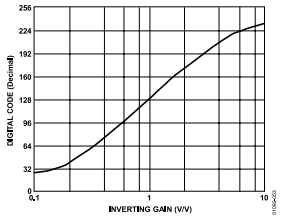


Figure 54. Inverting Programmable Gain Plot

ACTIVE FILTER

The state variable active filter is one of the standard circuits used to generate a low-pass, high-pass, or band-pass filter. The digital potentiometer allows full programmability of the frequency, gain, and Q of the filter outputs. Figure 55 shows the filter circuit using a 2.5 V virtual ground, which allows a ±2.5 V_P input and output swing. RDAC2 and RDAC3 set the LP, HP, and BP cutoff and center frequencies, respectively. These variable resistors should be programmed with the same data (as with ganged potentiometers) to maintain the best Circuit Q. Figure 56 shows the measured filter response at the band-pass output as a function of the RDAC2 and RDAC3 settings that produce a range of center frequencies from 2 kHz to 20 kHz. The filter gain response at the band-pass output is shown in Figure 57. At a center frequency of 2 kHz, the gain is adjusted over a −20 dB to +20 dB range determined by RDAC1. Circuit Q is adjusted by RDAC4. For more detailed reading on the state variable active filter, see Analog Devices' application note AN-318.

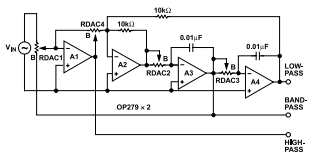


Figure 55. Programmable State Variable Active Filter

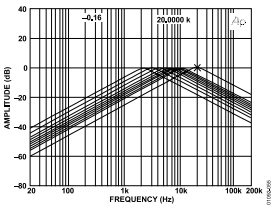


Figure 56. Programmed Center Frequency Band-Pass Response

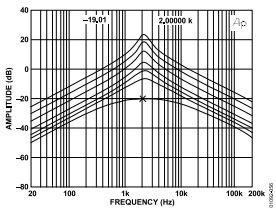


Figure 57. Programmed Amplitude Band-Pass Response

OUTLINE DIMENSIONS

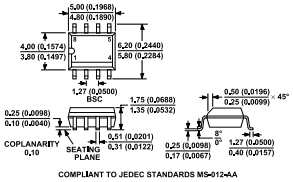


Figure 58. 8-Lead Standard Small outline package [SOIC]
Narrow Body (R-8)
Dimensions shown in millimeters and (inches)

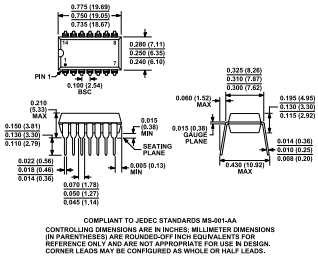


Figure 59. 14-Lead Plastic Dual-In-Line Package [PDIP]
Narrow Body (N-14)
Dimensions shown in inches and (millimeters)

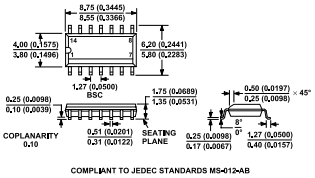


Figure 60. 14-Lead Standard Small Outline Package [SOIC]
Narrow Body (R-14)
Dimensions shown in millimeters and (inches)

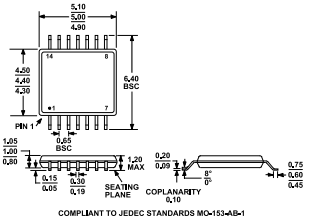


Figure 61. 14-Lead Thin Shrink Small Outline Package [TSSOP]
(RU-14)
Dimensions shown in millimeters

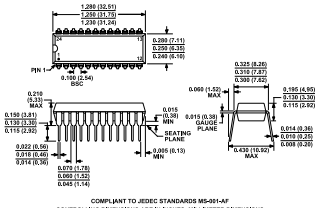


Figure 62. 24-Lead Plastic Dual-In-Line Package [PDIP]
Narrow Body (N-24-1)
Dimensions shown in inches and (millimeters)

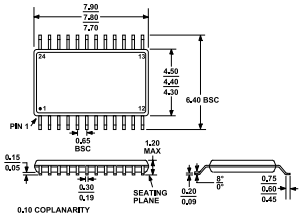


Figure 64. 24-Lead Thin Shrink Small Outline Package [TSSOP]
(RU-24)
Dimensions shown in millimeters

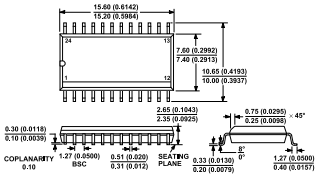


Figure 63. 24-Lead Standard Small Outline Package [SOIC]
Wide Body (R-24)
Dimensions shown in millimeters and (inches)

ORDERING GUIDE

Model[1]	Number of Channels	End-to-End R_{AB} (kΩ)	Temperature Range (°C)	Package Description	Package Option	Ordering Quantity	Branding Information
AD8400AR10	1	10	–40 to +125	8-Lead SOIC	R-8	98	AD8400A10
AD8400AR10-REEL	1	10	–40 to +125	8-Lead SOIC	R-8	2,500	AD8400A10
AD8400ARZ10[2]	1	10	–40 to +125	8-Lead SOIC	R-8	98	AD8400A10
AD8400ARZ10-REEL[2]	1	10	–40 to +125	8-Lead SOIC	R-8	2,500	AD8400A10
AD8400AR50	1	50	–40 to +125	8-Lead SOIC	R-8	98	AD8400A50
AD8400AR50-REEL	1	50	–40 to +125	8-Lead SOIC	R-8	2,500	AD8400A50
AD8400ARZ50[2]	1	50	–40 to +125	8-Lead SOIC	R-8	98	AD8400A50
AD8400ARZ50-REEL[2]	1	50	–40 to +125	8-Lead SOIC	R-8	2,500	AD8400A50
AD8400AR100	1	100	–40 to +125	8-Lead SOIC	R-8	98	AD8400AC
AD8400AR100-REEL	1	100	–40 to +125	8-Lead SOIC	R-8	2,500	AD8400AC
AD8400ARZ100[2]	1	100	–40 to +125	8-Lead SOIC	R-8	98	AD8400AC
AD8400ARZ100-REEL[2]	1	100	–40 to +125	8-Lead SOIC	R-8	2,500	AD8400AC
AD8400AR1	1	1	–40 to +125	8-Lead SOIC	R-8	98	AD8400A1
AD8400AR1-REEL	1	1	–40 to +125	8-Lead SOIC	R-8	2,500	AD8400A1
AD8400ARZ1[2]	1	1	–40 to +125	8-Lead SOIC	R-8	98	AD8400A1
AD8400ARZ1-REEL[2]	1	1	–40 to +125	8-Lead SOIC	R-8	2,500	AD8400A1
AD8402AN10	2	10	–40 to +125	14-Lead PDIP	N-14	25	AD8402A10
AD8402AR10	2	10	–40 to +125	14-Lead SOIC	R-14	56	AD8402A10
AD8402AR10-REEL	2	10	–40 to +125	14-Lead SOIC	R-14	2,500	AD8402A10
AD8402ARU10	2	10	–40 to +125	14-Lead TSSOP	RU-14	96	8402A10
AD8402ARU10-REEL	2	10	–40 to +125	14-Lead TSSOP	RU-14	2,500	8402A10
AD8402ARUZ10[2]	2	10	–40 to +125	14-Lead TSSOP	RU-14	96	8402A10
AD8402ARUZ10-REEL[2]	2	10	–40 to +125	14-Lead TSSOP	RU-14	2,500	8402A10
AD8402ARZ10[2]	2	10	–40 to +125	14-Lead SOIC	R-14	96	AD8402A10
AD8402ARZ10-REEL[2]	2	10	–40 to +125	14-Lead SOIC	R-14	2,500	AD8402A10
AD8402AR50	2	50	–40 to +125	14-Lead SOIC	R-14	56	AD8402A50
AD8402AR50-REEL	2	50	–40 to +125	14-Lead SOIC	R-14	2,500	AD8402A50
AD8402ARU50	2	50	–40 to +125	14-Lead TSSOP	RU-14	96	8402A50
AD8402ARU50-REEL	2	50	–40 to +125	14-Lead TSSOP	RU-14	2,500	8402A50
AD8402ARUZ50[2]	2	50	–40 to +125	14-Lead TSSOP	RU-14	96	8402A50
AD8402ARUZ50-REEL[2]	2	50	–40 to +125	14-Lead TSSOP	RU-14	2,500	8402A50
AD8402ARZ50[2]	2	50	–40 to +125	14-Lead SOIC	R-14	96	AD8402A50
AD8402ARZ50-REEL[2]	2	50	–40 to +125	14-Lead SOIC	R-14	2,500	AD8402A50
AD8402AR100	2	100	–40 to +125	14-Lead SOIC	R-14	56	AD8402AC
AD8402AR100-REEL	2	100	–40 to +125	14-Lead SOIC	R-14	2,500	AD8402AC
AD8402ARU100	2	100	–40 to +125	14-Lead TSSOP	RU-14	96	8402A-C
AD8402ARU100-REEL	2	100	–40 to +125	14-Lead TSSOP	RU-14	2,500	8402A-C
AD8402ARUZ100[2]	2	100	–40 to +125	14-Lead TSSOP	RU-14	96	8402A-C
AD8402ARUZ100-REEL[2]	2	100	–40 to +125	14-Lead TSSOP	RU-14	2,500	8402A-C
AD8402ARZ100[2]	2	100	–40 to +125	14-Lead SOIC	R-14	96	AD8402AC
AD8402ARZ100-REEL[2]	2	100	–40 to +125	14-Lead SOIC	R-14	2,500	AD8402AC
AD8402AR1	2	1	–40 to +125	14-Lead SOIC	R-14	56	AD8402A1
AD8402AR1-REEL	2	1	–40 to +125	14-Lead SOIC	R-14	2,500	AD8402A1
AD8402ARU1	2	1	–40 to +125	14-Lead TSSOP	RU-14	96	8402A1
AD8402ARU1-REEL	2	1	–40 to +125	14-Lead TSSOP	RU-14	2,500	8402A1
AD8402ARUZ1[2]	2	1	–40 to +125	14-Lead TSSOP	RU-14		AD8402A1
AD8402ARUZ1-REEL[2]	2	1	–40 to +125	14-Lead TSSOP	RU-14	2,500	AD8402A1
AD8402ARZ1[2]	2	1	–40 to +125	14-Lead SOIC	R14		AD8402A1
AD8402ARZ1-REEL[2]	2	1	–40 to +125	14-Lead SOIC	R-14	2,500	AD8402A1

Model[1]	Number of Channels	End-to-End R_{AB} (kΩ)	Temperature Range (°C)	Package Description	Package Option	Ordering Quantity	Branding Information
AD8403AN10	4	10	−40 to +125	24-Lead PDIP	N-24-1	15	AD8403A10
AD8403AR10	4	10	−40 to +125	24-Lead SOIC	R-24	31	AD8403A10
AD8403AR10-REEL	4	10	−40 to +125	24-Lead SOIC	R-24	1,000	AD8403A10
AD8403ARU10	4	10	−40 to +125	24-Lead TSSOP	RU-24	63	8403A10
AD8403ARU10-REEL	4	10	−40 to +125	24-Lead TSSOP	RU-24	2,500	8403A10
AD8403ARUZ10[2]	4	10	−40 to +125	24-Lead TSSOP	RU-24	63	8403A10
AD8403ARUZ10-REEL[2]	4	10	−40 to +125	24-Lead TSSOP	RU-24	2,500	8403A10
AD8403ARZ10[2]	4	10	−40 to +125	24-Lead SOIC	R-24	63	AD8403A10
AD8403ARZ10-REEL[2]	4	10	−40 to +125	24-Lead SOIC	R-24	2,500	AD8403A10
AD8403AN50	4	50	−40 to +125	24-Lead PDIP	N-24-1	15	AD8403A50
AD8403AR50	4	50	−40 to +125	24-Lead SOIC	R-24	31	AD8403A50
AD8403AR50-REEL	4	50	−40 to +125	24-Lead SOIC	R-24	1,000	AD8403A50
AD8403ARU50	4	50	−40 to +125	24-Lead TSSOP	RU-24	63	8403A50
AD8403ARUZ50[2]	4	50	−40 to +125	24-Lead TSSOP	RU-24	2,500	8403A50
AD8403ARZ50[2]	4	50	−40 to +125	24-Lead SOIC	R-24	63	AD8403A50
AD8403ARZ50-REEL[2]	4	50	−40 to +125	24-Lead SOIC	R-24	2,500	AD8403A50
AD8403AR100	4	100	−40 to +125	24-Lead SOIC	R-24	31	AD8403A100
AD8403AR100-REEL	4	100	−40 to +125	24-Lead SOIC	R-24	1,000	AD8403A100
AD8403ARU100	4	100	−40 to +125	24-Lead TSSOP	RU-24	63	8403A100
AD8403ARU100-REEL	4	100	−40 to +125	24-Lead TSSOP	RU-24	2,500	8403A100
AD8403ARUZ100[2]	4	100	−40 to +125	24-Lead TSSOP	RU-24	63	8403A100
AD8403ARUZ100-REEL[2]	4	100	−40 to +125	24-Lead TSSOP	RU-24	2,500	8403A100
AD8403ARZ100[2]	4	100	−40 to +125	24-Lead SOIC	R-24	63	AD8403A100
AD8403ARZ100-REEL[2]	4	100	−40 to +125	24-Lead SOIC	R-24	2,500	AD8403A100
AD8403AR1	4	1	−40 to +125	24-Lead SOIC	R-24	31	AD8403A1
AD8403AR1-REEL	4	1	−40 to +125	24-Lead SOIC	R-24	1,000	AD8403A1
AD8403ARU1	4	1	−40 to +125	24-Lead TSSOP	RU-24	63	8403A1
AD8403ARU1-REEL	4	1	−40 to +125	24-Lead TSSOP	RU-24	2,500	8403A1
AD8403ARUZ1[2]	4	1	−40 to +125	24-Lead TSSOP	RU-24	63	8403A1
AD8403ARUZ1-REEL[2]	4	1	−40 to +125	24-Lead TSSOP	RU-24	2,500	8403A1
AD8403ARZ1[2]	4	1	−40 to +125	24-Lead SOIC	R-24	63	AD8403A1
AD8403ARZ1-REEL[2]	4	1	−40 to +125	24-Lead SOIC	R-24	2,500	AD8403A1
AD8403EVAL				Evaluation Board			

[1] Non-lead-free parts have date codes in the format of either YWW or YYWW, and lead-free parts have date codes in the format of #YWW, where Y/YY is the year of production and WW is the work week. For example, a non-lead-free part manufactured in the 30th work week of 2005 has the date code of either 530 or 0530, while a lead-free part has the date code of #530.
[2] Z = Pb-free part.

NOTES

NOTES

NOTES