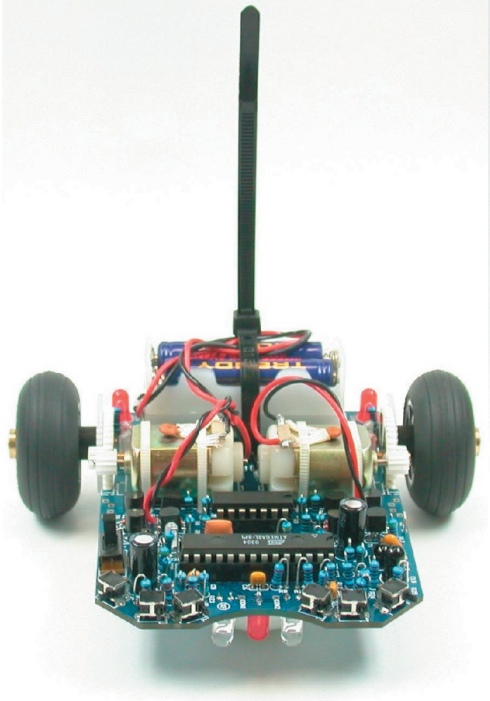
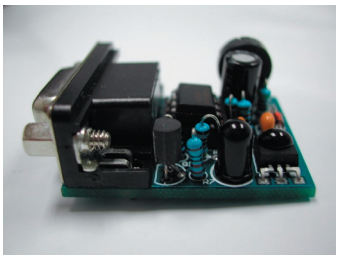
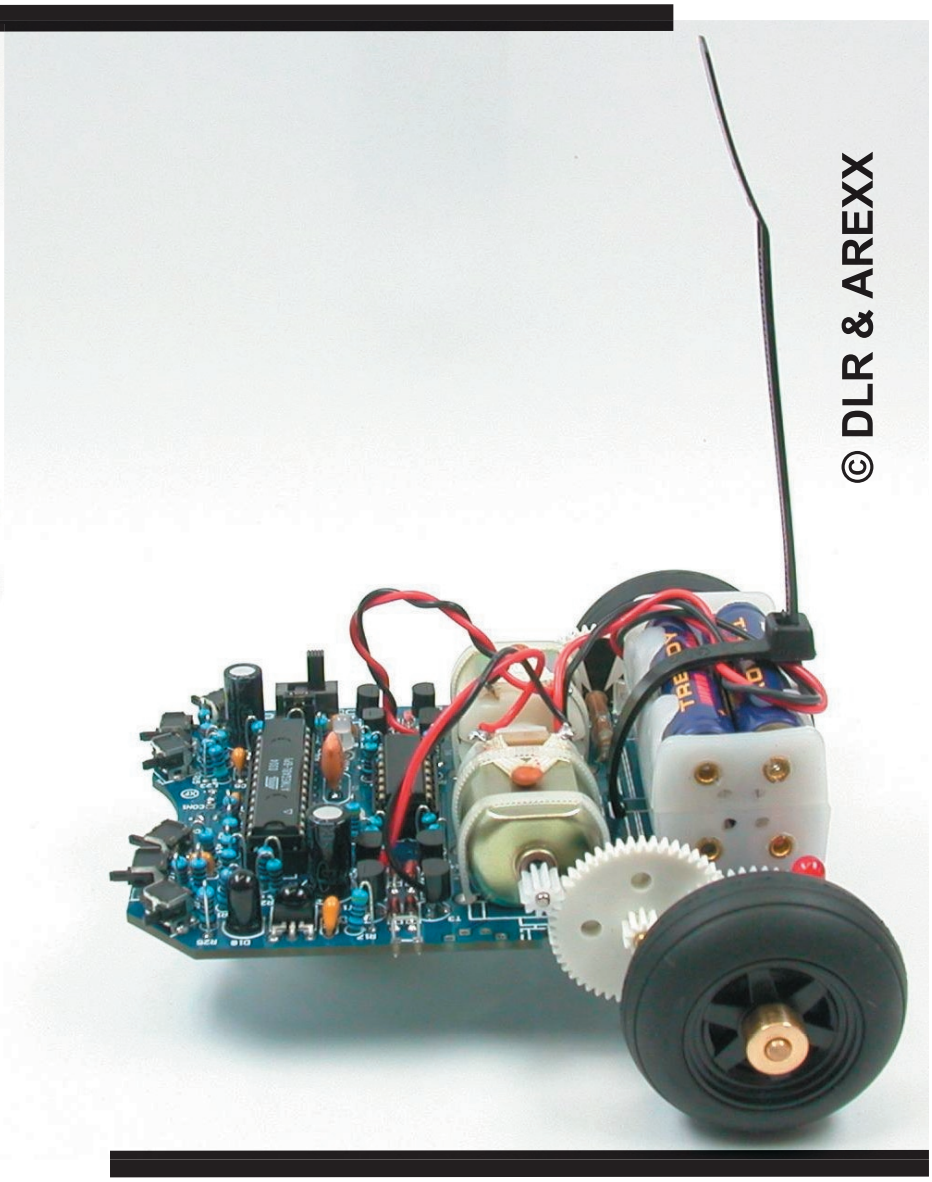
Licence by DLR

# ASURO ROBOTERBAUSATZ









## BAU- UND BEDIENUNGSANLEITUNG

**Modell ARX-03**

**Hersteller**

**AREXX, Zwolle - NIEDERLANDE**
**JAMA, Taichung - TAIWAN**

**www.arexx.com**

# Einleitung

ASURO ist ein kleiner, frei in C programmierbarer mobiler Roboter, welcher am Deutschen Zentrum für Luft- und Raumfahrt (DLR) im Institut für Robotik und Mechatronik für die Lehre entwickelt wurde. Der Aufbau ist für den erfahrenen Elektroniker ein Kinderspiel und für den Elektronikeinsteiger ohne weiteres durchzuführen. Es werden - bis auf die Platinen - nur handelsübliche, mit normaler Feinmotorik handhabbare und leicht zu beschaffende Bauteile verwendet. Ebenso erfolgt die Programmierung ausschließlich mit Freeware-Tools. ASURO eignet sich daher ausgezeichnet für Hobbybastler, welche den Einstieg in prozessorgesteuerte Schaltungen wagen wollen, für Schüler und Studentenprojekte, Fortbildungen oder Volkshochschulkurse. Aufgrund der Tatsache, dass auch die gesamte Entwicklung der Elektronik mit (für den privaten Bereich) Freeware-Werkzeugen möglich ist, tritt ASURO nebenbei auch noch den Beweis an, dass man es auch ohne aufwändige und teure Software, Technik und Werkzeugmaschinen in der Lage ist, funktionsfähige Roboter zu konstruieren.

ASURO besitzt neben seinem RISC-Prozessor, zwei Motoren, die unabhängig voneinander angesteuert werden können, eine optische Linienfolgeeinheit, sechs Kollisionstaster, zwei Drehzahlsensoren für die Räder, drei optische Anzeigen und eine Infrarot-Kommunikationseinheit, welche die Programmierung und auch eine Fernsteuerung über einen PC ermöglicht (siehe Abb.0.1).

Das Achtung-Symbol  weist auf Abschnitte hin, die sehr sorgfältig beachtet werden müssen, da Fehler zu zerstörter Hardware oder Gesundheit führen können.

Es sei auch noch darauf hingewiesen, dass ASURO natürlich kein Spielzeug und für Kinder unter drei Jahren nicht geeignet ist, da hier dutzende Kleinteile verschluckt werden können.
Jetzt nur noch Batterien oder Akkus bereitlegen und los geht's!

Ach ja, ASURO steht übrigens für "Another Small and Unique Robot from Oberpfaffenhofen"!

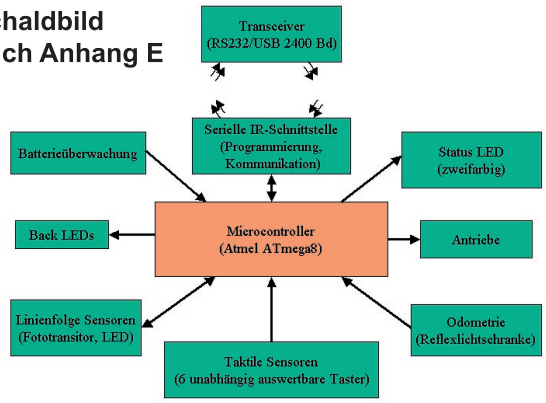


**Oberpfaffenhofen 2004**

Jan Grewe
Robin Gruber

WWW.DLR.DE

**Abb.0.1 Blockschaldbild**
**Sehe auch Anhang E**




## WARNUNG

* Mit dem Öffnen der Plastikbeutel mit Komponenten und Bauteilen erlischt das Rückgaberecht.
* Lies vor dem Bauen zuerst die Bedienungsanleitung aufmerksam durch.
* Sei vorsichtig beim Handhaben der Werkzeuge.
* Baue nicht im Beisein kleiner Kinder. Die Kinder können sich an den Werkzeugen verletzen oder kleine Komponenten und Teile in den Mund stecken.
* Achte auf die Polung der Batterien.
* Sorge dafür, dass die Batterien und die Batteriehalter trocken bleiben.
* Falls ASURO nass wird, entferne die Batterien.
* Entferne die Batterien, wenn der Roboter mehr als eine Woche ruht.

# Inhaltsverzeichnis

# Teil I. Mechanik

## 1. Erforderliches Werkzeug

Um ASURO vernünftig zusammenbauen zu können, wird - außer den Bausatzteilen - folgendes Werkzeug und Verbrauchsmaterial benötigt:

kleiner Schraubstock oder Dritte Hand: nicht immer sind zwei Hände ausreichend
Teppichmesser oder Säge
feine Zange
Seitenschneider: ein kleiner für Elektronik
evtl. Abisolierzange
Lötkolben: Hier sollte ein Elektronik-Lötkolben (ca. 20W bis 40W) oder gleich eine Lötstation (mind. 50W) verwendet werden.
Lötzinn: 1 mm dickes Elektroniklot, ggf. auch bleifrei
Entlötlitze: ca. 2-3mm breit, falls mal was nicht da landet, wo es hingehört
Schleifpapier mit feiner Körnung
Sekunden-, Zweikomponenten- oder Heißkleber
Evt. kleiner Hammer
Evtl. Multimeter
Computer: Laptop oder PC mit Windows oder Linux


**Abbildung 1.1.: Erforderliches Werkzeug**

# 2. Mechanische Vorarbeiten

Bevor die ersten Basteleien beginnen können, sollte unbedingt überprüft werden, ob alle erforderlichen Teile vorhanden sind. Das kann am einfachsten anhand der Stückliste in Anhang A geschehen. Bevor es an die Elektronik gehen kann, müssen noch ein paar mechanische Tätigkeiten durchgeführt werden.

## 2.1. Motorritzel

Damit die Motoren ihre Kraft auf das Getriebe übertragen können, müssen die Motorritzel (das sind die kleinen Zahnräder mit der 1,9mm-Bohrung und den 10 Zähnen) auf der Motorachse montiert sein. Falls die gelieferten Motoren diese Ritzel noch nicht auf ihrer Achse haben, muss man sie aufpressen. Dazu steckt man, ohne viel Kraft aufzubringen, auf die Achse jedes Motors ein Ritzel. Es muss erstmal nur steckenbleiben. Den Motor hält man dann Ritzel nach unten auf eine nicht zu harte Unterlage (Plastik, Karton, o.ä.) und klopft zart mit einem kleinen Hammer auf die aus der Rückseite des Motors etwas herausragende Achse, bis die Motorachse vollständig im Ritzel steckt (siehe Abb.2.1). Alternativ kann man das von Hand aufgesteckte Ritzel mit einem Schraubstock auf die Motorachse drücken. Dabei darf aber nur Kraft auf die durchgehende Motorwelle, keinesfalls auf das Gehäuse oder die Lager ausgeübt werden.

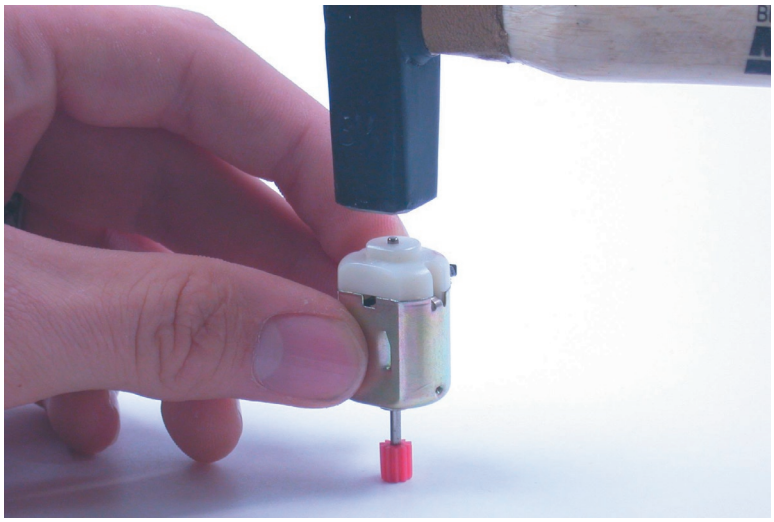


**Abbildung 2.1.: Motorritzelmontage**

## 2.2. Tischtennisball

ASURO soll später auf einem halben Tischtennisball gleiten. Dieser ist herzustellen. Am besten nimmt man einen ganzen Tischtennisball und sägt oder schneidet diesen mit einem Teppichmesser (den Ball, nicht die Finger) auseinander. Die Schnittkanten dann noch mit einer Feile oder Schleifpapier entgraten.

**Abbildung 2.2.: Zersägter Tischtennisball**

***Elektrowerkzeug darf wegen der Brandgefahr nicht verwendet werden!***

## 2.3. Radsensoren

Die Leuchtdiode und der Fototransistor (Reflexlichtschranke für die Odometrie), welche sich so vertrauensvoll Richtung erstem Getriebezahnrad wenden werden, wollen später nicht enttäuscht werden. Daher bringt man auf das jeweils erste Getriebezahnrad (das mit mit den 50 und 10 Zähnen) auf der Seite ohne Abtriebsritzel noch die selbstklebenden Musterscheiben an (siehe Abb.2.3.).

Abbildung 2.3.: Montage der Radsensormuster

Je mehr Segmente das Muster besitzt, umso genauer lässt sich die Drehzahl des Zahnrades und damit ASUROs Geschwindigkeit auflösen, allerdings wird dann auch der gemessene Unterschied zwischen hell und dunkel geringer.

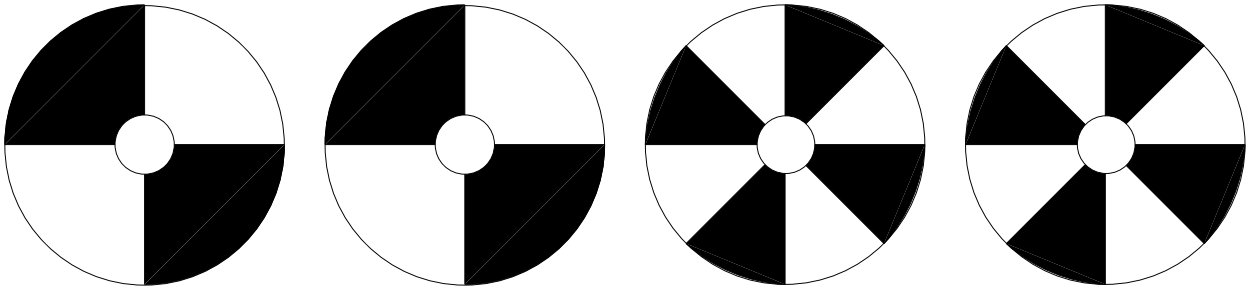


Abbildung 2.4.: Beispiel für Radsensoren

Damit ist der erste Teil geschafft. Die mechanischen Komponenten sind soweit alle fertig vorbereitet.

***Kurze Pause...***

# Jetzt geht es weiter mit der Elektronik.

# Teil II. Elektronik

## 3. Kleine Lötfibel

Obwohl ASURO vollständig mit bedrahteten Bauteilen aufgebaut ist und damit - im Gegensatz zu oberflächenmontierten SMD-Bauteilen (Abb. 3.1 zeigt den Vergleich zwischen dem kleinsten und unserem Gehäuse, in dem ASUROs Prozessor erhältlich ist. Der eigentliche Siliziumchip ist in beiden Gehäusen der gleiche!) - geradezu hervorragen von Hand zu bestücken ist, sollten vor allem vom untrainierten Löter einige Hinweise beachtet werden.

*Selbstverständlich ist auch, dass die zu bestückende Platine in jedem Fall stromlos sein muss. Ausschalten läuft nicht unter stromlos! Batterien herausnehmen!*

### 3.1. Spitze, Zinn und Temperatur

Abbildung 3.2 zeigt die für's Löten wichtigste Grundlage!

Das gefährliche Ende sollte zum Löten eine Temperatur von ca. 360ºC bei bleihaltigem, ca. 390ºC bei bleifreiem Lötzinn haben, zum Einlöten der Achsen kann sie etwas erhöht werden (420ºC). Für Elektronikbestückung wie diese sollte eine bleistiftspitze Spitze verwendet werden, für die Achsen kann man ggf. eine breitere Spitze nehmen.

Auch ist das Lötschwämmchen noch anzufeuchten (es darf nicht triefen) und die Spitze des Lötkolbens mit etwas Lötzinn zu benetzen. Kurz bevor man den Lötkolben nach einer Pause oder am Beginn des Lötens an die Lötstelle hält, oder wenn verschlackte Zinnrückstände am Lötkolben stören, wischt man diese einfach am Lötschwämmchen ab.

Als Lötzinn sollte Elektroniklot mit 0,8 oder 1mm Durchmesser benutzt werden.

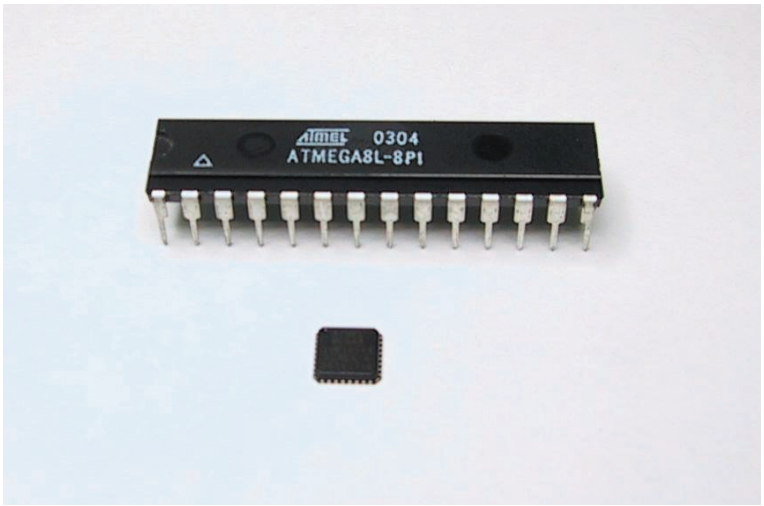



**Abbildung 3.1.: Vergleich der größten und kleinsten Bauform in der der ATmega8L erhältlich ist.**


**Abbildung 3.2.: Grundlage des Lötens**

*Dämpfe, welche während des Lötprozesses entstehen sind langfristig eher ungesund und sollten nach Möglichkeit nicht eingeatmet, besser noch abgesaugt werden!*
*Anderes Lötzinn als Elektroniklot oder säurehaltiges Flussmittel können die Schaltung zerstören!*

## 3.2. Vorbereiten der Bauteile

Jeder, der schon mal Elektronik gelötet hat, kennt das Problem: man hat eigentlich immer eine Hand zuwenig. Daher gibt's ein paar Tricks, wie man die verschiedenen Bauteile bändigt, bis man mit Lötkolben und Lötzinn gut hinkommt.

ASUROs Transistoren, Leuchtdioden, Fototransistoren, ICs, Taster, Schalter, Kondensatoren und Jumper haben bereits die Anschlussenden in eine Richtung. Bei den Dioden und Widerständen muss das erst noch erzeugt werden.

Alle Widerstände bei ASURO werden aus Platzgründen stehend eingelötet. Das heißt, ein Beinchen bleibt wie es ist, das andere wird um 180º gebogen. Die Biegung sollte einen Durchmesser von 2,5mm haben und erst in ein paar Millimeter Abstand vom Widerstandskörper liegen, damit dieser nicht mechanisch belastet wird, was eine einwandfreie Funktion beeinträchtigen könnte.

Beim späteren Einlöten gibt ein Kreis im Bestückungsdruck an, über welchem Loch der Widerstandkörper zu liegen kommt und ein kleiner Stich in welches Loch der umgebogene Draht gesteckt werden muss.

Die Dioden werden liegend eingebaut, d.h. beide Beinchen müssen (am besten mit einer feinen Zange) um 90º abgewinkelt werden, und das in einem Abstand, damit sie leicht in die entsprechenden Bohrungen in der Leiterplatte passen.

*Beim Prozessor IC1 ATmega8, dem Gatterbaustein IC3 CD4081 und dem IR-Empfänger IC2 SFH5110-36 handelt es sich um elektrostatisch gefährdete Bauteile.*
*Das bedeutet, dass sie bereits durch bloßes Anfassen zerstört werden können, sofern man vorher elektrisch geladen war, was beispielsweise durch Laufen über Teppiche passieren kann. Vor dem Handhaben dieser Bauteile ist es ratsam, sich mit einem Erdungsband zu erden oder zumindest ein Metallgehäuse eines Gerätes oder die Heizung anzufassen.*

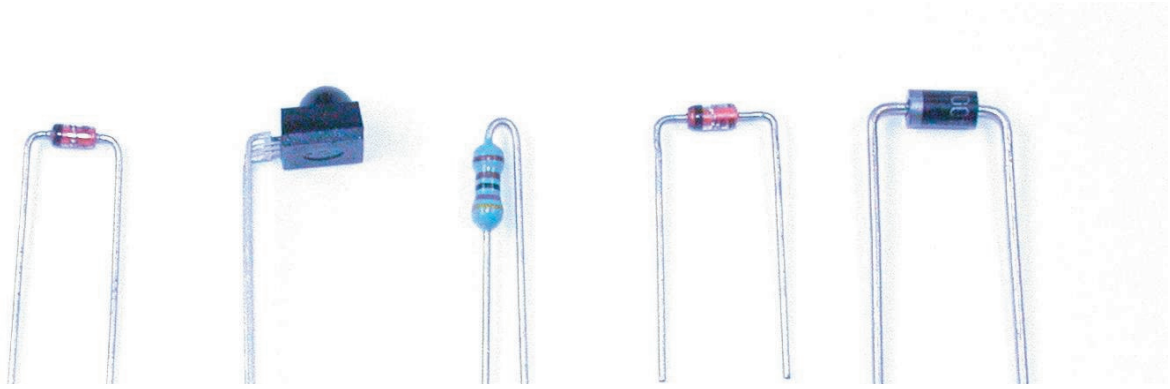


**Abbildung 3.3.: Bauteile mit passend gebogenen Beinchen**

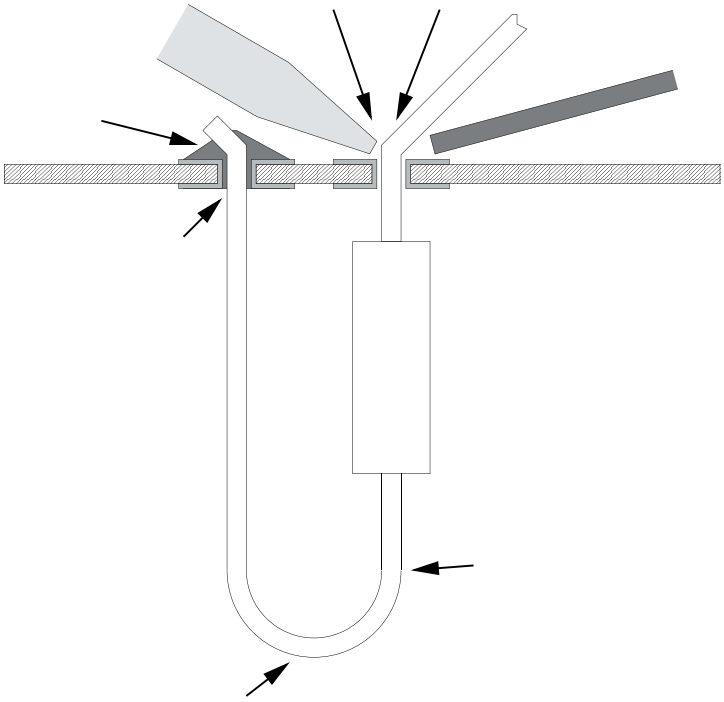


**Abbildung 3.4.:**
**Herstellen einer sauberen Lötstelle**

## 3.3. Einlöten der Bauteile

Sind die Bauteile vorbereitet, werden sie durch die durchkontaktierten Bohrungen in der Leiterplatte gesteckt und - bei Teilen mit nur zwei oder drei Beinchen - diese auf der Unterseite der Platine direkt an der Bohrung etwas auseinandergebogen (ca. 30º bis 40º sind ausreichend), so dass sie nicht mehr herausfallen können.

Bei Bauteilen mit mehr Beinchen - wie den Sockeln für die ICs - reicht es, zwei diagonal gegenüberliegende Beinchen nach außen zu biegen. Weiter als 45º weit biegen ist eher unpraktisch, da man ein Teil - so man es doch einmal verkehrt eingebaut haben sollte - sonst kaum mehr aus der Platine bekommt.

Sitzt das Bauteil fest, so erhitzt man mit der Lötkolbenspitze das Beinchen und das Lötauge gleichzeitig und gibt an diese Stelle etwas Lötzinn zu. Dieses schmilzt nun und läuft in die Bohrung. Es wird solange Zinn zugegeben, bis die Bohrung vollständig ausgefüllt ist (siehe Abb.3.4). Danach nimmt man erst das Zinn und anschließend den Lötkolben weg und lässt die Lötstelle erkalten. Keinesfalls darf das Bauteil während des Erkaltens bewegt werden, das würde zu kalten Lötstellen und damit zu Wackelkontakten führen.

Bei Lötaugen, die an der Kupferfläche an Ober- und Unterseite angeschlossen sind, kann etwas hartnäckigere Wärmezufuhr erforderlich sein, bis das Zinn in die Bohrung gelaufen ist.

Schlechte Lötstellen zeichnen sich durch kugelförmige Lötzinnansammlungen am Lötauge oder eine matte (bei bleifreiem Lötzinn: sehr matte) Oberfläche aus. Hier muss nachgebessert werden. Um die Sockel oder andere Teile, welche flach auf der Platine aufliegen sollen zu montieren, kann folgender Trick verwendet werden: Das Bauteil wird zunächst an einem Beinchen angelötet. Danach drückt man mit den Fingern von oben leicht auf das Teil und erwärmt nochmal die Lötstelle

***(Achtung: Das Bauteil kann dabei sehr heiß werden)***, sodass das Teil auf der Platine aufsetzten kann. Anschließend werden die anderen Beine verlötet und die erste Lötstelle nochmal zusammen mit etwas Zinn verflüssigt. Ist ein Teil festgelötet, werden die überstehenden Drahtenden mit einem Seitenschneider knapp über der Platine abgezwickt, ohne dabei am Beinchen zu ziehen.

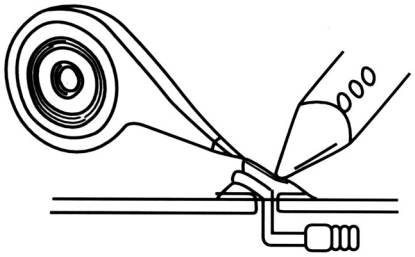
*Beim Abzwicken muss darauf geachtet werden, dass die eventuell wegfliegenden Drahtstücke niemanden gefährden können.*
*Die auf der Oberseite der Platine stehenden Teile dürfen sich natürlich keinesfalls mit den Anschlüssen berühren. Eventuell müssen diese geradegebogen werden.*

## 3.4. Auslöten falsch eingebauter Teile

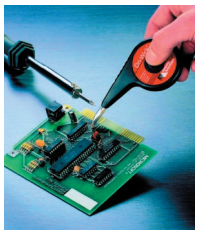
Sollte es doch mal vorkommen, dass ein Bauteil nicht da gelandet ist, wo es hingehört, dann muss es wieder raus. Da ASURO - wie schon erwähnt - eine doppelseitige Platine mit durchkontaktierten Löchern hat, wird das meist etwas kniffelig.

Folgendes Vorgehen hat sich bewährt:
Zunächst verflüssigt man alle Lötstellen des betreffenden Bauteile (gegebenenfalls mit Zugabe von etwas weiterem Lötzinn) gleichzeitig und zieht das Bauteil mit einer Zange aus der Platine. Anschließend befreit man die Löcher mit Hilfe von Entlötlitze vom übrigen Zinn.

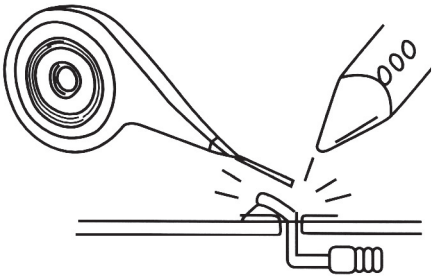
Dazu legt man die Entlötlitze auf die Lötstelle und erhitzt beides gleichzeitig, bis die Litze das Lötzinn aufgesaugt hat. Anschließend nimmt man Lötkolben und Lötzinn wieder weg. Eventuell kann es hilfreich sein, auch nochmal von der Oberseite aus das Lötzinn abzusaugen.



**1.**
**Die Entlötlitze auf die Lötverbindung legen. Danach Litze und Lötverbindung gleichzeitig erhitzen.**



***professionelle Entlötsauglitze***



**2.**
**Den Lötkolben und die Litze wenehmen, sobald die Litze das Lötzinn aufgesaugt hat.**

# 4. Bestückung

## *Lötfibel gelesen? Wirklich? Na gut, los geht's!*

### 4.1. Bestückung des RS232-Infrarot-Transceivers

• IC1: Hier wird zunächst nur der 8-polige Sockel eingelötet. Dieser besitzt eine Richtungsmarkierung, welche mit der Markierung auf der Platine übereinstimmen muss.

• D1, D2, D3: 1N4148, auf richtige Polung achten! Nicht verwechseln mit ZPD5.1 oder BZX55-C5V1 (Aufdruck)!

•D4: ZPD5.1 oder BZX55-C5V1, auf richtige Polung achten! Nicht verwechseln mit 1N4148 (Aufdruck)!

• C2, C4: 100nF keramisch, Aufdruck: 104

• C3: 680pF keramisch, Aufdruck: 681

• Q1: BC547 (A,B oder C) oder BC548 (A,B oder C)

• R1, R5: 20k Ohm, 5% (rot, schwarz, orange, gold)

• R2: 4.7k Ohm, 5% (gelb, violett, rot, gold)

• R3: 470 Ohm, 5% (gelb, violett, braun, gold)

• R6: 10k Ohm, 5% (braun, schwarz, orange, gold)

• R7: 220 Ohm (rot, rot, braun, gold)

• C1: 100µF/ mindestens 16V, auf richtige Polung achten!

• TR1: 10k Ohm Trimmer

• D5: SFH 415-U IR-LED (schwarzes Gehäuse), auf richtige Polung achten! Gehäuse sollte auf der Platine aufliegen!

• IC2: SFH5110-36 Infrarot-Empfänger-IC, Beinchen mit Zange abwinkeln! Auf richtige Polung achten (Seite mit Wölbung muss nach oben weisen), Achtung: elektrostatisch gefährdet und - Hinweis für die Hobbyschweißer - hitzeempfindlich!

• X1: 9pol. SUB-D Buchse, Gehäuse muss auf der Platine aufliegen, auch die Befestigungslaschen müssen angelötet werden!

• IC1: NE555P einstecken, Richtungsmarkierung (Nase oder Kreis) beachten!

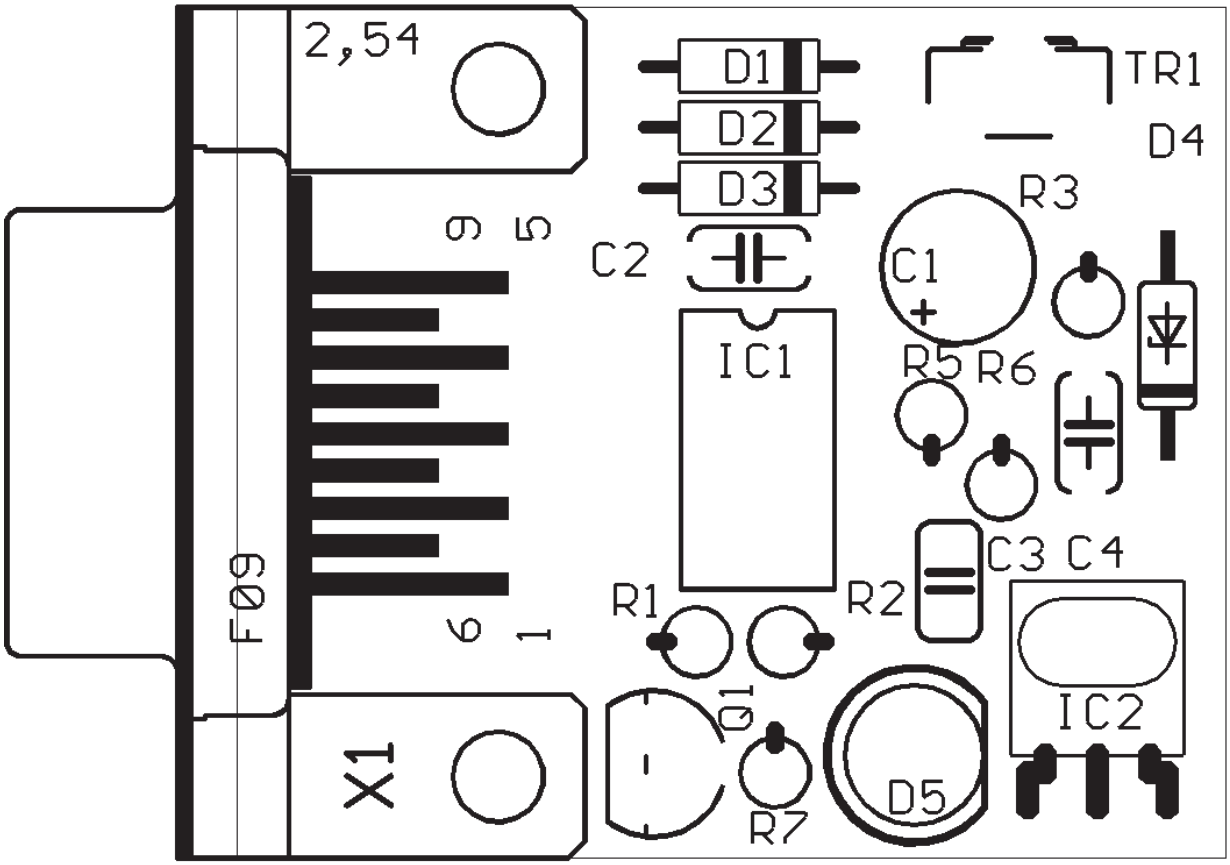



**Abbildung 4.1.: Bestückung des RS232- Infrarot-Transceivers**

Nun nochmal mit kritischem Blick die Lötstellen auf gute Verbindung oder Kurzschlüsse überprüfen und ggf. nachbessern.

# Fertig!

## 4.2. Fertiggerät USB-Infrarot-Transceiver

Optional ist ein USB-IR-Transceiver als Fertiggerät lieferbar.

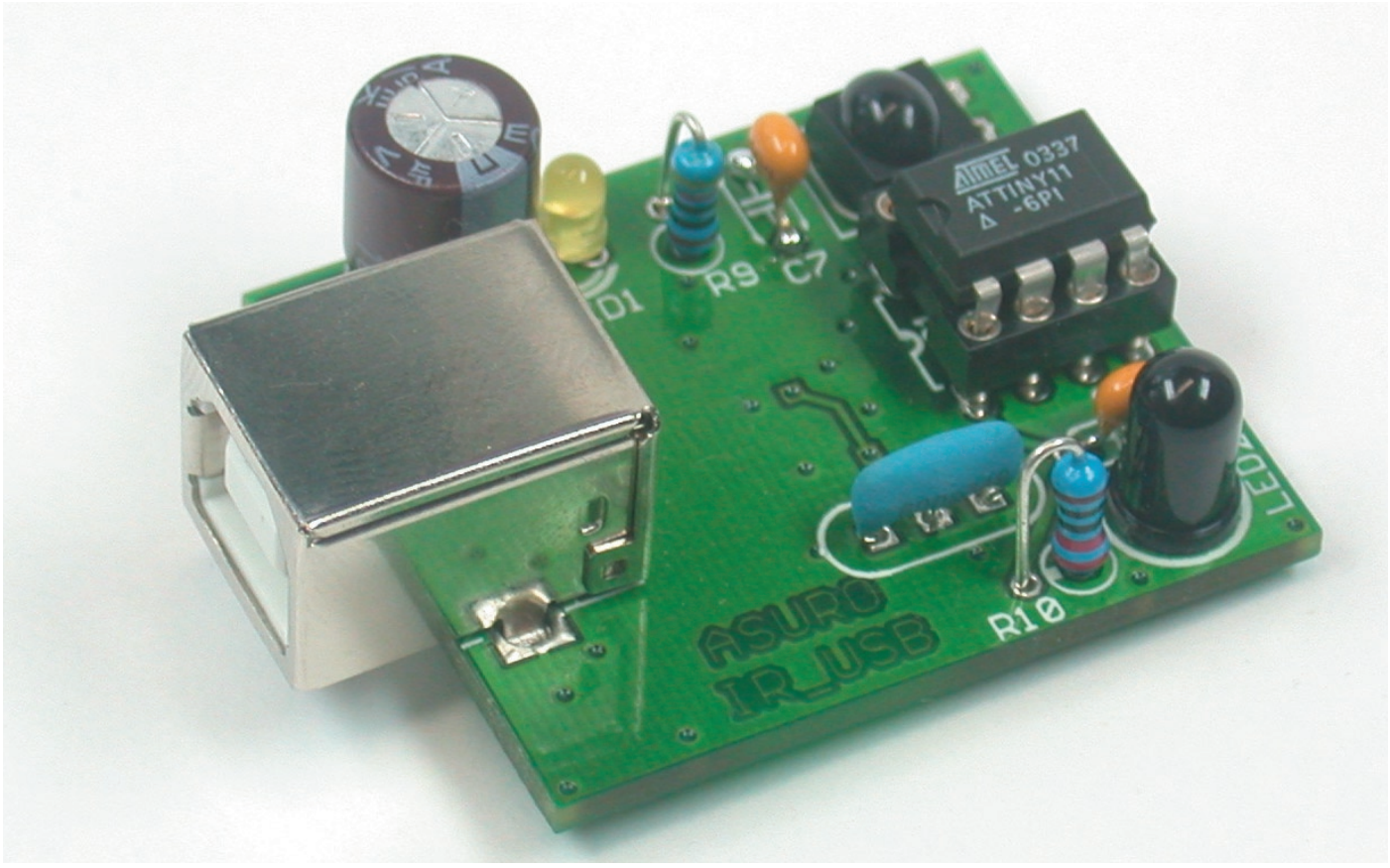



**Abbildung 4.2.: USB Infrarot-Transceivers**

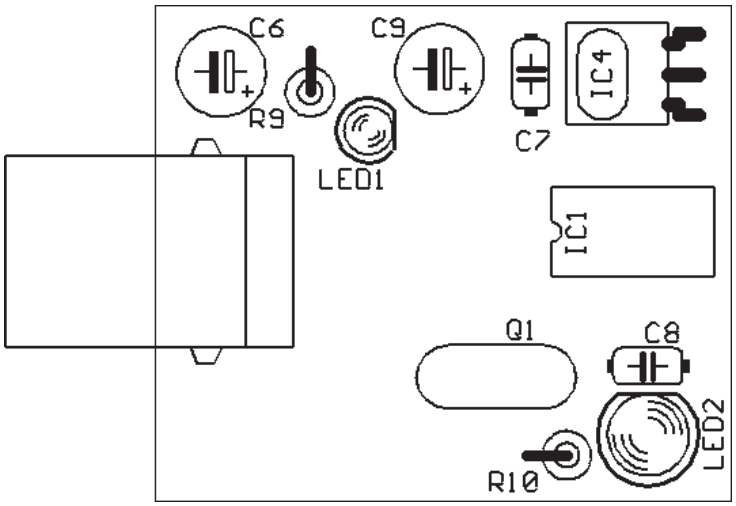



**Abbildung 4.3.: Oberseite des USB Infrarot-Transceivers**

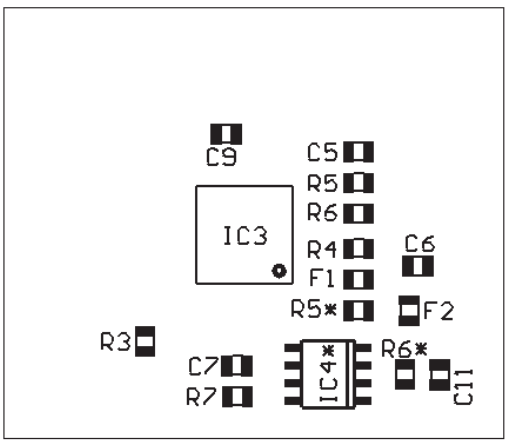



**Abbildung 4.4.: Unterseite des USB Infrarot-Transceivers**

## 4.3. Bestückung der ASURO-Platine

Die zwei längeren Achsen, die für die zweite Getriebestufe erforderlich sind, werden auf der Unterseite angelötet oder geklebt. Löten ist praktischer, denn Korrekturen sind einfacher möglich und abkühlen lassen geht schneller als warten, bis der Kleber fest ist.

Die beiden kürzeren liegen auf der Oberseite und weiter Richtung Platinenmitte. Vor dem Einbau kann man die Achsen an der zu lötenden oder zu klebenden Stelle (nicht auf der Lauffläche) noch mit einem sehr feinen Schleifpapier (240er Körnung oder mehr) säubern, dann nehmen diese das Zinn oder den Kleber besser an. Entscheidet man sich für's Löten empfiehlt sich folgendes Vorgehen:

Zuerst werden die längeren Achsen befestigt. Dazu legt man die Platine auf die Oberseite, legt die entsprechende Achse bis zum Anschlag in den ausgefrästen Schlitz. Dabei muss die Achse auf der ganzen Länge aufliegen. Dann verzinnt man die Lötkolbenspitze und drückt damit die Achse auf die Platine. Sobald die Achse heiß ist, gibt man Lötzinn an den Auflagestellen dazu und verbindet so Achse und Platine. Ist die Achse rundum angelötet, drückt man die Achse mit einem Schraubendreher weiter auf die Platine und nimmt den Lötkolben weg. Das Einlöten der Achsen klappt am besten, wenn man die Löttemperatur erhöht (ca. 420°C) und eine breite (ca. 3mm) Spitze nimmt. Für die Elektronikkomponenten muss man die Temperatur natürlich wieder auf ca. 360°C senken.

Ist das Ganze erkaltet, lötet man die zweite lange Achse auf der Platinenunterseite fest, danach kommen nach gleichem Prinzip die Achsen auf der Oberseite dran. Abbildung 4.5. zeigt die Platine mit montierten Achsen.

Sind die Achsen abgekühlt, werden die Getrieberäder aufgesteckt. Die Zähne der Getrieberäder müssen gut ineinandergreifen und die Räder müssen sich leicht drehen lassen. Ist dies nicht der Fall sind entweder die Achsen schief eingelötet und müssen neu ausgerichtet werden (nicht zur Strafe, nur zur Übung), oder auf den Achsen befindet sich im Bereich außerhalb der Platine Lötzinnreste, welche entfernt werden müssen.

Dies geschieht am besten mit einer feinen Feile oder Schmirgelpapier.
Passt alles, werden die Getrieberäder erst mal wieder beiseite gelegt und die restlichen Bauteile bekommen auf der Platine ihren zukünftigen Arbeitsplatz zugewiesen.

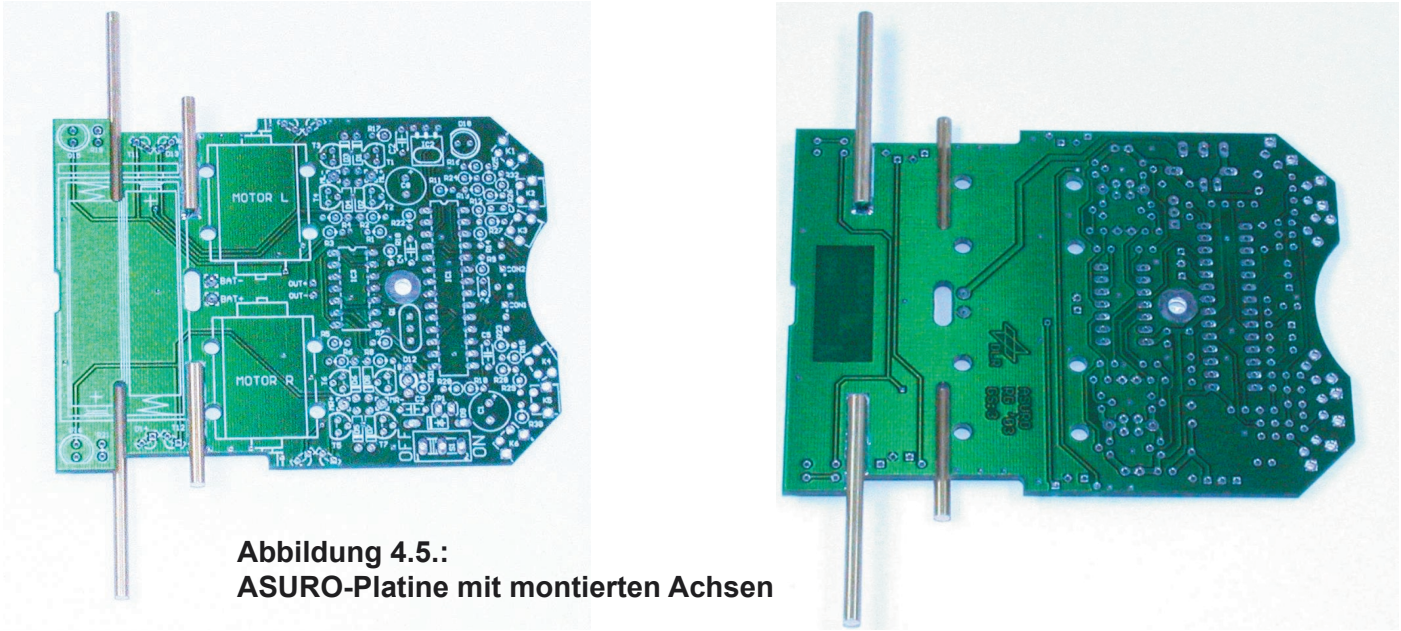



**Abbildung 4.5.:**
**ASURO-Platine mit montierten Achsen**

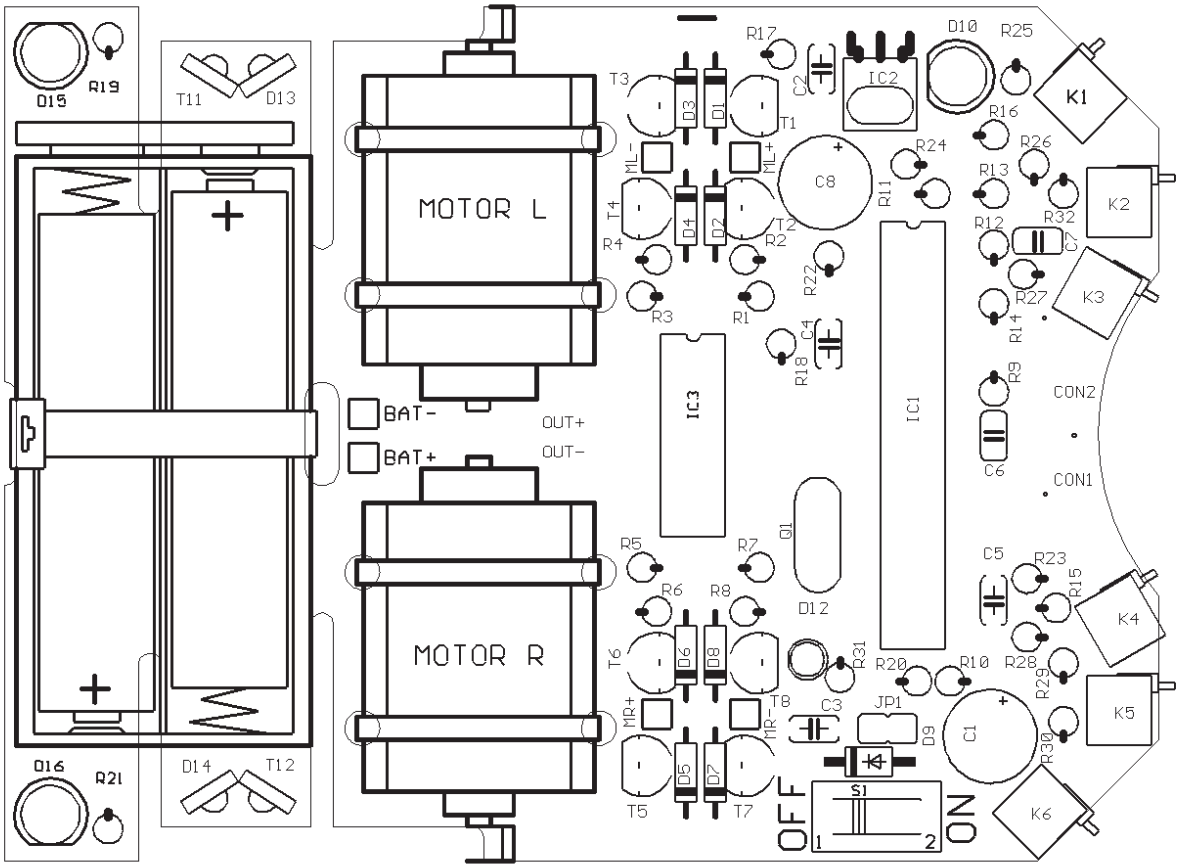



**Abbildung 4.6.: Bestückung ASURO Hauptplatine auf der Oberseite**

## In folgender Reihenfolge wird bestückt:

• IC1: erstmal nur der Sockel, hier werden entweder ein 28poliger (wenn vorhanden) oder zwei 14polige hintereinander eingebaut; auf richtige Polung achten (Kerbe zeigt in Richtung der Kerbe vom Bestückungsdruck auf der Platine)!

• IC3: auch nur der Sockel, 14polig; auf richtige Polung achten (Kerbe zeigt in Richtung der Kerbe vom Bestückungsdruck auf der Platine)!

• K1, K2, K3, K4, K5, K6: Sensortaster; müssen möglichst gerade auf der Platine aufliegen!

• Q1; Schwinger 8MHz

• D1, D2, D3, D4, D5, D6, D7, D8: 1N4148; auf richtige **Polung** achten!

• D9: 1N4001; auf richtige **Polung** achten!

• JP1: zweipoliger Pfostenstecker; die kurzen Pins werden eingelötet, der zugehörige Jumper wird noch nicht aufgesteckt!

• D12: zweifarbige LED, 3mm Durchmesser, drei Anschlussbeinchen, auf **Polung** achten (Markierung kann unterschiedlich sein, in jedem Fall: kürzestes Beinchen muss in quadratisches Pad)!

• C2, C3, C4, C5: 100nF keramisch; Aufdruck: 104

• C6, C7: 4,7nF keramisch; Aufdruck: 472

• T1, T3, T5, T7: BC327-40 oder BC328-40

• T2, T4, T6, T8: BC337-40 oder BC338-40

• R1, R2, R3, R4, R5, R6, R7, R8, R19, R21, R24: 1k Ohm, 5% (braun, schwarz, rot, gold)

• R9, R16: 220 Ohm, 5% (rot, rot, braun, gold)

• R10, R17, R22, R31: 470 Ohm, 5% (gelb, violett, braun, gold)

• R11: 100 Ohm, 5% (braun, schwarz, schwarz, gold)

• R12: 12k Ohm, 1% (braun, rot, schwarz, rot, braun)

• R13: 10k Ohm, 1% (braun, schwarz, schwarz, rot, braun)

• R14, R15: 20k Ohm, 5% (rot, schwarz, orange, gold)

• R18, R20: 4,7k Ohm, 5% (gelb, violett, rot, gold)

• R23: 1M Ohm, 5% (braun, schwarz, grün, gold)

• R25, R26, R32: 2k Ohm, 1% (rot, schwarz, schwarz, braun, braun)

• R27: 8,2k Ohm, 1% (grau, rot, schwarz, braun, braun)

• R28: 16k Ohm, 1% (braun, blau, schwarz, rot, braun)

• R29: 33kΩ Ohm, 1% (orange, orange, schwarz, rot, braun)

• R30: 68kΩ Ohm, 1% (blau, grau, schwarz, rot, braun)

• C1, C8: Elko 220F 10V oder mehr, auf richtige Polung achten!

• IC2: SFH5110-36 Infrarot-Empfänger-IC, Beinchen mit Zange abwinkeln! Auf richtige Polung achten (Seite mit Wölbung muss nach oben weisen) Achtung: elektrostatisch gefährdet und - wieder der Hinweis für die Hobbyschweißer - hitzeempfindlich!

• D10: SFH 415-U IR-LED 5mm; schwarzes Gehäuse; auf richtige Polung achten! Gehäuse sollte auf der Platine aufliegen!

• T11, T12: LPT80A, Fototransistor, farbloses
Gehäuse muss auf Platine aufliegen, auf richtige Polung achten!

• D13, D14: IRL80A, IR-LED, rosafarbenes, Gehäuse
muss auf Platine aufliegen, auf richtige Polung achten!

• D15, D16: LED 5mm rot, rotes Gehäuse, auf richtige Polung achten
(kurzes Bein an markierte Seite)!

• S1: Ein-Aus-Schalter

Drei weitere Bauteile werden noch benötigt (sie ermöglichen das Folgen einer Linie), allerdings werden sie auf der Unterseite der Platine angebracht und von oben her eingelötet (siehe Abb.4.7.):

• T9, T10: SFH300, Fototransistor 5mm, auf richtige Polung achten!
Diese stehen leicht von der Platine ab.

• D11: LED 5mm rot, rotes Gehäuse, auf richtige Polung achten
(kurzes Bein an markierte Seite)!

Abbildung 4.8. zeigt die soweit bestückte Platine von oben und unten.

***Das war's! Mehr elektronische Bauteile werden nicht benötigt.***
***Als nächstes werden die elektromechanischen und mechanischen Komponenten montiert.***

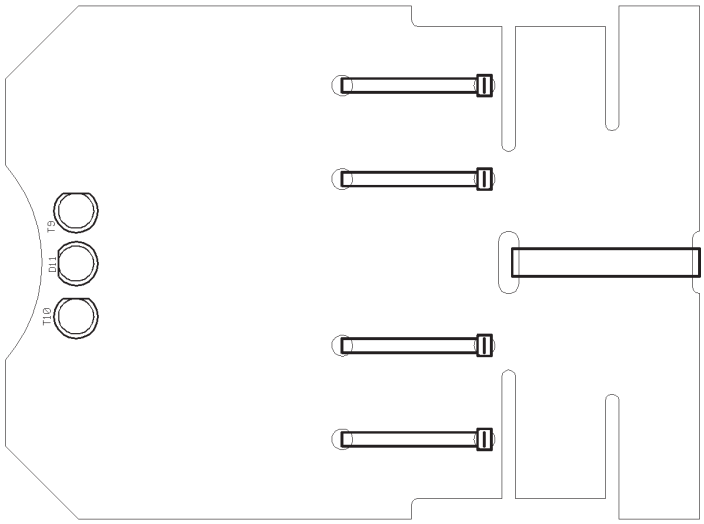



**Abbildung 4.7.:**
**Bestückung der ASURO-Platine auf der Unterseite**

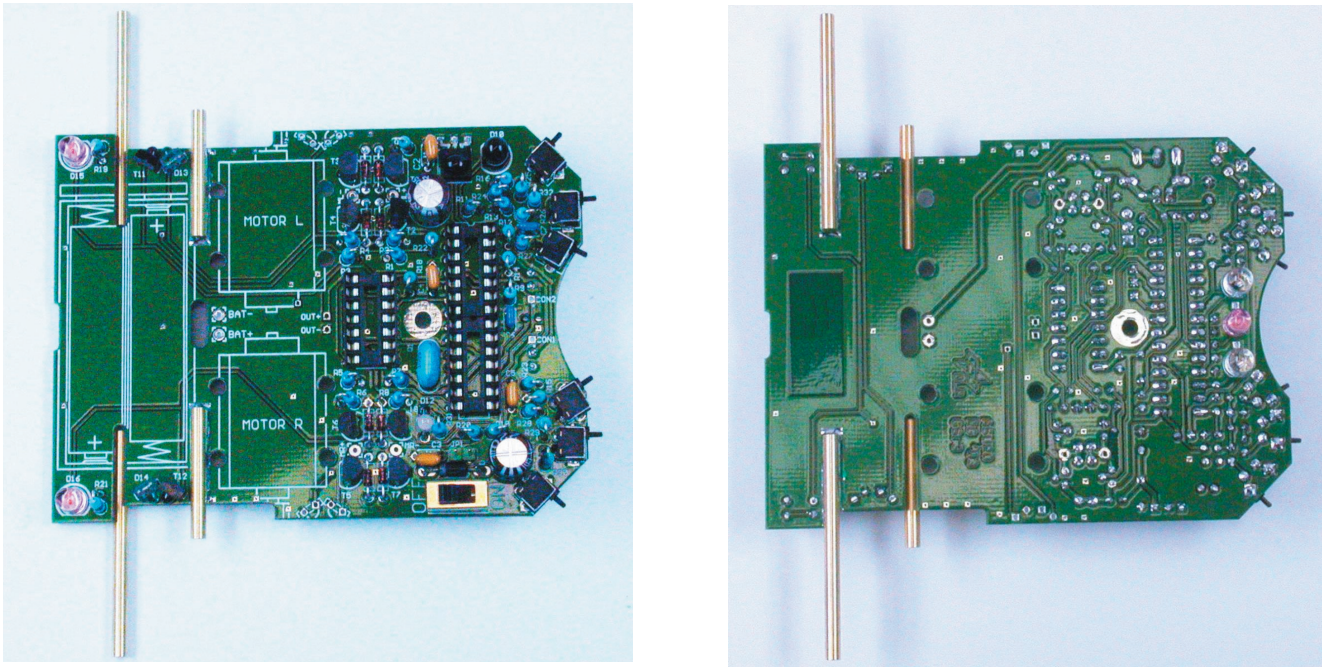



**Abbildung 4.8.: ASURO bestückt von oben und unten**

## 4.4. Motormontage

Wenn die Bestückung der ASURO-Platine abgeschlossen ist, müssen nur noch die Motoren mit Kabeln versehen und provisorisch befestigt werden.

Zum Anschluss der Motoren wird je ein schwarzes und rotes ca. 70mm langes Kabel mit abisolierten und verzinnten Enden benötigt. Sind die beiliegenden Kabel noch nicht passend vorbereitet, so isoliert man die Enden ca. 4mm lang ab, verdrillt sie und verzinnt sie anschließend, indem man sie zusammen mit etwas Lötzinn an die Lötkolbenspitze hält. Falls störende Lötzinnreste am Kabelende übrigbleiben, können diese mit einem Seitenschneider abgeschnitten werden. Das rote Kabel wird nun an den mit einem roten Punkt oder Pluszeichen markierten Motoranschluss gelötet, das schwarze an den anderen.

Die Motoranschlusskabel jedes Motors werden noch verflochten (muss nicht sein, bringt aber Vorteile bei der elektromagnetischen Verträglichkeit und schaut auch erheblich besser aus...).

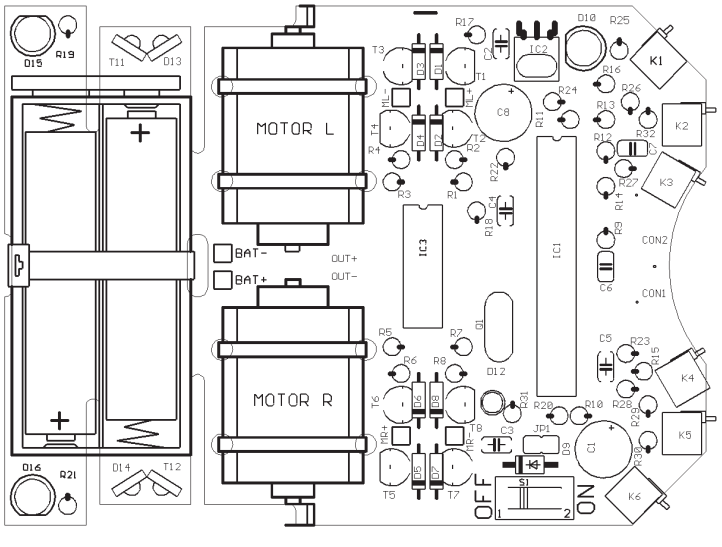
Das rote Anschlusskabel des linken Motors wird in “ML+” und das schwarze in “ML-”, das rote des rechten Motors in ”MR+” und das schwarze in ”MR-” eingelötet.

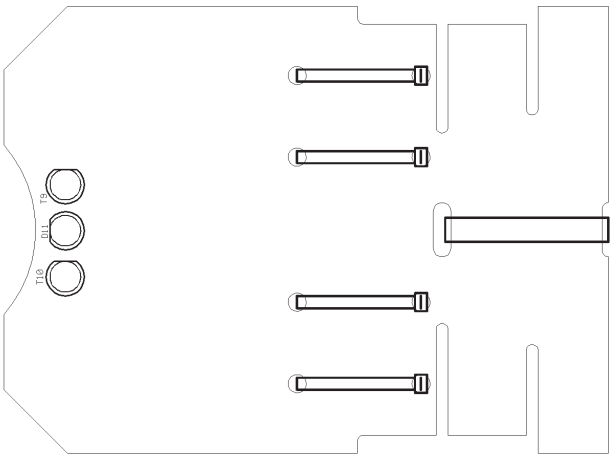
Jetzt müssen noch die Motoren provisorisch auf der Platine befestigt werden. Dazu zieht man die Kabelbinder durch die Löcher in der Platine neben den Motoren, sodass die Köpfe der Kabelbinder auf der Platinenunterseite bleiben und umschlingt die Motoren.

## 4.5. Stromversorgung

*Wenn ASURO mit Batterien versorgt werden soll, ist der Jumper JP1 unbedingt zu öffnen! Werden Akkus verwendet, so ist er zu schließen. Eine Falschpolung der Akkus bei geschlossenem Jumper führt zur Zerstörung der Elektronik!*

Der Batteriehalter wird (ohne die Batterien) mit seinem roten Kabel in BAT+ und seinem schwarzen in BAT- festgelötet. Danach wird sichergestellt, dass der Schalter auf OFF steht und die vier Batterien bzw. Akkus polungsrichtig in den Batteriehalter eingelegt. Der Batteriehalter wird jetzt gleich oder nach der Inbetriebnahme mit dem größeren (wieder lösbaren) Kabelbinder durch das Loch in der Platine befestigt.



# 5. Inbetriebnahme und Test

Endlich ist alles zusammengebaut und der Fahrspaß kann beginnen. Zunächst müssen aber noch die zuvor eingebauten Fehler gesucht, gefunden und beseitigt werden, ohne dabei allzu großen Schaden anzurichten.

## 5.1. RS232-Infrarot-Transceiver

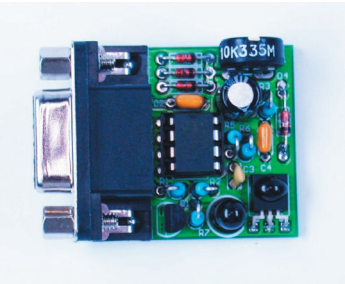
Diese Inbetriebnahme gilt nur für den RS232-IR-Transceiver.
Als erstes sollte der RS232-IR-Transceiver auf seine volle Funktionsfähigkeit überprüft werden, da dieser später für den Selbsttest des Fahrzeugs benötigt wird.
Dazu wird der RS232-IR-Transceiver über das mitgelieferte serielle Verlängerungskabel an eine freie serielle Schnittstelle angeschlossen.

Danach startet man das Terminal-Programm von Windows “Hyperterminal” (unter Linux beispielsweise “Minicom”). Normalerweise findet man es unter Start --> Programme --> Zubehör --> Kommunikation --> Hyperterminal. Falls es nicht vorhanden ist, muss man es von der Windows-CD nachinstallieren.

Terminalprogramme stammen eigentlich noch aus der Zeit des Modems und davor, als man häufiger über die serielle Schnittstelle mit anderen Rechnern kommunizierte. Heutzutage werden sie vorwiegend dann benutzt, wenn man sich über das Internet rein textbasiert auf einem anderen Rechner einloggen will.

Nach dem Start von Hyperterminal fragt das Programm nach einem Namen für die neue Verbindung. Hier kann man ASURO eingeben und ein beliebiges Symbol auswählen. Im nächsten Fenster wählt man bei “Verbinden über:” die COM-Schnittstelle aus, an der der Transceiver angeschlossen worden ist. Nach Drücken auf “OK” wählt man

- Bits pro Sekunde: 2400
- Datenbits: 8
- Parität: keine
- Stoppbits: 1
- Flusssteuerung: kein




Danach wieder bestätigen mit “OK”.

Nun den IR-Transceiver ca. 10 cm über ein weißes Blatt Papier halten. Die Bauteile zeigen zum Papier. Nun munter ein paar Tasten auf der PC-Tastatur gedrückt und das Terminal-Programm sollte diese Tasten anzeigen. Der IR-Transceiver sendet dabei den Tastendruck über die IR-Diode (D5), das am Papier reflektierte Signal trifft auf das Empfänger-IC (IC2) und wird zum PC übertragen. Kommen gar keine oder falsche Zeichen an, so kann man mit einem kleinen Schraubenzieher vorsichtig den Trimmer zwischen seinem linken und rechten Anschlag verdrehen und wieder ein paar Tasten drücken, bis korrekte Zeichen erscheinen.

Das Ganze funktioniert nicht so wie beschrieben? Schade, hier muss wohl ein Fehler vorliegen, der behoben werden sollte (siehe Abschnitt 6.1).

Zur Sicherheit kann man abschließend den IR-Transceiver wieder abstecken und nochmal ein paar Tasten drücken. Nun dürfen keine Zeichen mehr erscheinen.

## 5.2. USB-Infrarot-Tranceiver

Diese Inbetriebnahme gilt nur für den USB-Infrarot-Transceiver.
Achtung! Der ungehäuste USB-Infrarot-Transceiver ist empfindlich gegen elektrostatische Entladungen. Vor der Benutzung muss man sich an einem metallischen Körper (Heizung, Computergehäuse) entladen um Schäden zu vermeiden. Alternativ kann man den Transceiver auch in ein für Infrarotlich durchsichtiges Gehäuse einbauen.

### 5.2.1 Windows

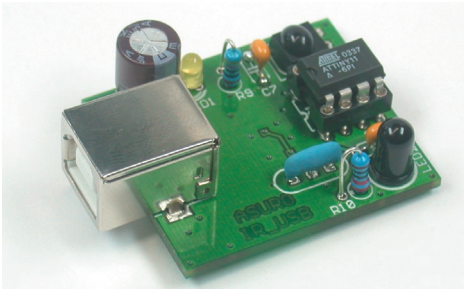
Der USB-Transceiver wird mit dem USB-Kabel an einer freien USB-Buchse angeschlossen. Es erscheint die Meldung:

“Neue Hardware gefunden: ***AREXX ASURO USB-IR-Transceiver”***

Danach den USB-Treiber von der ASURO-CD installieren. Sollte der Treiber nicht automatisch gefunden werden, das Verzeichnis (D: steht hier für das CD-ROM-Laufwerk) D:\Windows\USB_ Driver” auswählen. Eventuell sind hierzu Administratorrechte erforderlich. Dann abmelden und als Administrator erneu anmelden. Es wird nun ein Treiber installiert, damit man unter Windows den USB-Transceiver wie eine normale serielle Schnittstelle ansprechen kann.

Hat dies Fehlerfrei geklappt, so startet man zum Ausprobieren auch hier das Terminalprogramm “Hyperterminal”, gibt ASUROUSB für den Verbindungsnamen an und wählt ein beliebiges Symbol aus. Beim nächsten Fenster “Verbinden über:” wählt man die letzte verfügbare COM-Schnittstelle aus. Nach Drücken auf “OK” wählt man:

- Bits pro Sekunde: 2400
- Datenbits: 8
- Parität: keine
- Stoppbits: 1
- Flusssteuerung: kein



Danach wieder bestätigen mit “OK”.

Nun hält man den Transceiver mit der Seite der Leuchtdiode nach unten ca. 10cm über ein weißes Blatt Papier. Falls der Transceiver ohne Gehäuse betrieben wird, darf die Platine nur am Stecker oder am Platinenrand gehalten werden, um die Schaltung nicht zu stören. Nun drückt man am Terminalprogramm einige Tasten. Dabei muss die gelbe Leuchtdiode auf der Platine blinken und die Tastendrücke auf dem Bildschirm erscheinen.

Funktioniert das nicht, bei 6.2 weiterlesen.

Hat alles geklappt, kann man mit der Inbetriebnahme der ASURO-Platine weitermachen.

### 5.2.2 Linux

Der USB-Transceiver wird mit dem USB-Kabel an einer freien USB-Buchse angeschlossen. Ein kurzes “Piep” ertönt, wenn Linux den Transceiver erkannt hat. Um zu überprüfen, ob das Gerät korrekt erkannt wurde, kann man sich den entsprechenden Eintrag im proc-Verzeichnis anschauen:

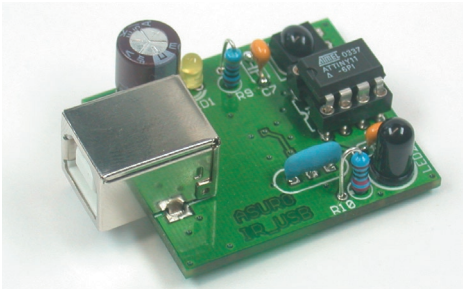
foo@bar:/>cat /proc/tty/driver/usb-serial

Was eine Ausgabe produzieren muss, die mindestens folgende Einträge aufweist (statt der “0:” kann auch “1:”, “2:” usw. stehen):

usbserinfo:1.0 driver:v1.4
0: module:ftdi_sio name:”FTDI 8U232AM Compatible” vendor:0403 product:6001
num_ports:1 port:1 path:usb-00:11.2-1

Zum Ausprobieren konfiguriert man Minicom auf die Schnittstelle /dev/ttyUSB0 (oder 1, 2 usw...) und folgede Parameter:

- Bits pro Sekunde: 2400
- Datenbits: 8
- Parität: keine
- Stoppbits: 1
- Flusssteuerung: kein



Danach wieder bestätigen mit “OK”.

Dazu sind eventuell root-Rechte erforderlich.

Eventuell muss man dem gewünschten User oder der gewünschten Gruppe noch Lese- und Schreibrechte auf dem Device /dev/ttyUSB? einräumen. Das kann mit einem chmod u+rw /dev/ttyUSB0 (oder 1, 2...) oder chmod g+rw /dev/ttyUSB0 (auch mit root-Rechten) erfolgen.

Nun hält man den Transceiver mit der Seite der Leuchtdiode nach unten ca. 10cm über ein weißes Blatt Papier. Falls der Transceiver ohne Gehäuse betrieben wird, darf die Platine nur am Stecker oder am Platinenrand gehalten werden, um die Schaltung nicht zu stören.
Nun drückt man am Terminalprogramm einige Tasten. Dabei muss die gelbe Leuchtdiode auf der Platine blinken und die Tastendrücke auf dem Bildschirm erscheinen.

Funktioniert das nicht, bei 6.2 weiterlesen.

Hat alles geklappt, kann man mit der Inbetriebnahme der ASURO-Platine weitermachen.

## 5.3. Inbetriebnahme der ASURO-Platine

***Der Prozessor (IC1) ist zu diesem Zeitpunkt noch nicht eingebaut!***

Jetzt die Nackenhaare sträuben, und den Schalter auf ON stellen. Die beiden Back-LEDs (D15, D16) sollten nun beide leicht glimmen. Ist dies nicht der Fall den Hauptschalter sofort auf OFF stellen und in Abschnitt 6.3 weiterlesen. Klappt's? Dann Schalter auf OFF stellen und IC1 (Prozessor) sowie IC3 (AND Gatter) einsetzen (siehe Abb. 5.1).

Evtl. müssen die Beinchen der ICs noch vorsichtig gebogen werden, damit alle Beinchen die vorgesehene angestammte Position im Sockel einnehmen können. Das klappt am besten, wenn man das IC seitlich nimmt und die Beinchen leicht gegen eine Tischkante drückt.

*Beim Prozessor IC1 ATmega8 und dem Gatterbaustein IC3 CD4081 handelt es sich um elektrostatisch gefährdete Bauteile. Das bedeutet, dass sie bereits durch bloßes Anfassen zerstört werden können, sofern man vorher elektrisch geladen war, was beispielsweise durch Laufen über Teppiche passieren kann. Vor dem Handhaben dieser Bauteile ist es ratsam, sich mit einem Erdungsband zu erden oder zumindest das Metallgehäuse eines Gerätes oder die Heizung anzufassen.*

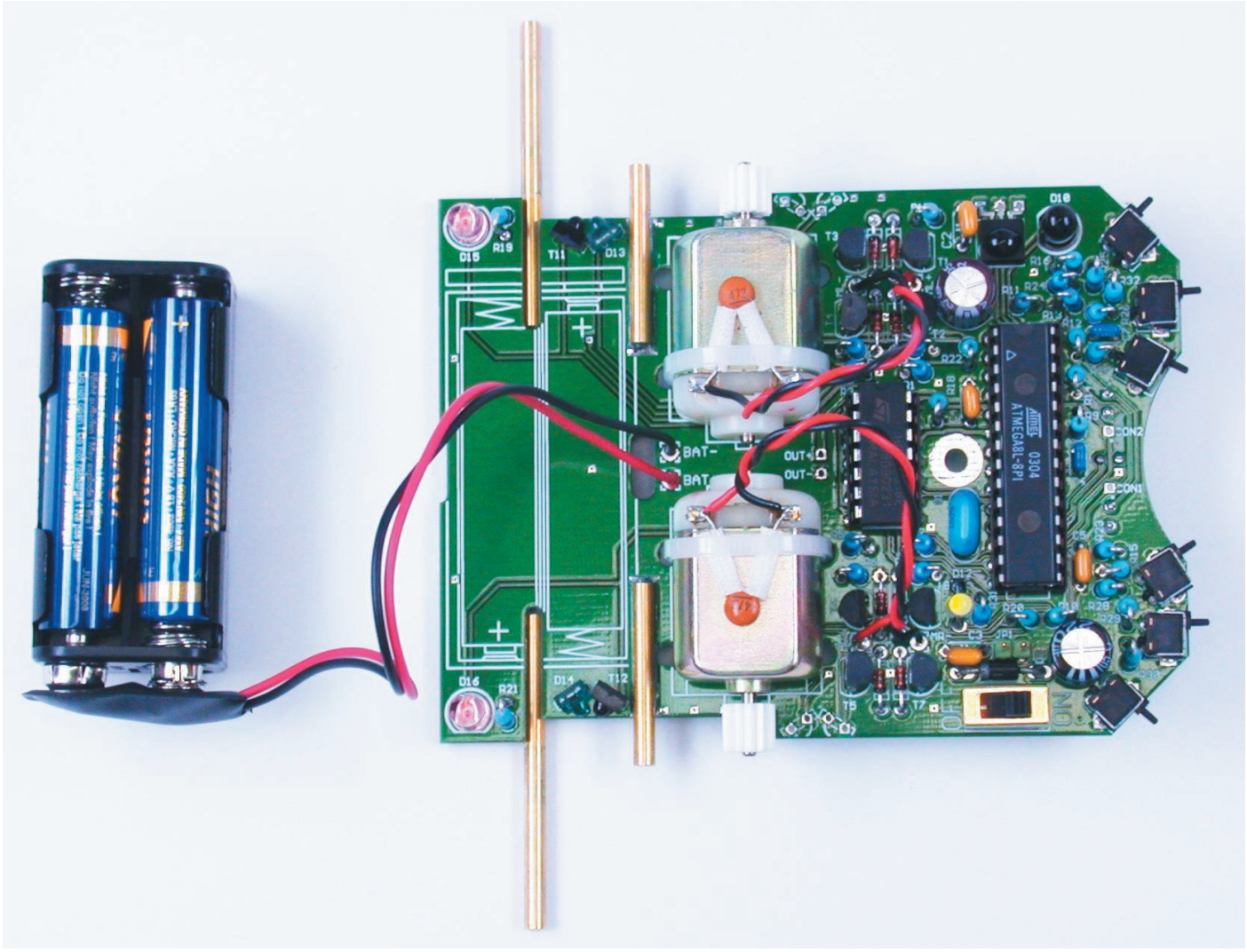



**Abbildung 5.1.: ASURO nach Einbau der ICs**

Nur bei Akku-Betrieb Jumper (J1) stecken. Die Kerben der ICs müssen mit der Kerbe der hoffentlich richtig eingesetzten Sockel übereinstimmen. Der Prozessor ist bereits werksseitig mit einem Selbsttest vorprogrammiert und wird nach dem Einschalten alle Komponenten überprüfen. Damit es hierbei gleich keine Schwierigkeiten gibt, sollte man den folgenden Abschnitt einmal komplett lesen bevor man einschaltet und anschließend wieder hierher zurückkehren.
Jetzt geht es los, Schalter auf ON stellen und ASURO nicht mehr aus dem Auge lassen.

***Wenn Hyperterminal (Windows) oder minicom (Linux) läuft, der IR-Transceiver eingesteckt ist und Sichtkontakt zu ASURO besteht, kann man den Selbsttest am Bildschirm mitverfolgen.***

## 5.3.1. Anzeigenelemente

Die Status-LED (D12) leuchtet kurz orange auf und die “Back-LEDs” (D15, D16) glimmen ebenfalls, allerdings nicht besonders hell. Falls nicht, den Schalter sofort auf OFF stellen und Fehler beheben (siehe Abschnitt 6.3.3). Dies war die Boot-Phase von ASURO. Nun werden nacheinander alle Anzeigenelemente für ca. 3 Sekunden in folgender Reihenfolge einzeln geprüft :

- Status-LED (D12) grün
- Status-LED (D12) rot
- Front-LED (D11) auf der Unterseite von ASURO
- Back-LED (D15) links
- Back-LED (D16) rechts
- Alle Anzeigenelemente auf einmal

Sollte wider Erwarten ein Fehler aufgetreten sein, muss ASURO sofort ausgeschaltet und der Fehler behoben werden (siehe Abschnitt 6.4), da alle eben getesteten Anzeigenelemente für die weiteren Tests unbedingt notwendig sind.

### Fototransistoren (T9, T10)

Nach dem Anzeigentest sollte die Status-LED (D12) grün aufleuchten. Dies ist eine deutliches Zeichen, dass nun die Fototransistoren auf der Unterseite von ASURO, welche für die Linienverfolgung notwendig sind geprüft werden (ca. 10 Sekunden). Werden die Fototransistoren (T9, T10) beleuchtet, sollte die dazugehörige Back-LED (D15, D16) aufleuchten und beim Abdunkeln wieder ausgehen. (rechter Fototransistor(T10) --> rechte Back-LED (D16); linker Fototransistor(T9) --> linke Back-LED (D15)). Es ist möglich, dass die entsprechende Back-LED im abgeschalteten Zustand nicht vollständig erlischt, sondern noch ein wenig glimmt, das ist normal. Bei einem Fehler kann mit dem Selbsttest fortgefahren und Fehlerbehebung später durchgeführt werden.

## Schalter

ASURO steht still, alle Anzeigenelemente sind aus. Ein gutes Zeichen! Die Schalter werden nun überprüft (ca 15sec). Einfach mal ein bisschen rumdrücken, irgendetwas wird hoffentlich passieren.
Die Zuordnung sollte wie folgt aussehen :

K1 --> Status-LED (D12) leuchtet grün
K2 --> Status-LED (D12) leuchtet rot
K3 --> Front-LED auf der Unterseite leuchtet auf (D11)
K4 --> Back-LED links (D15)
K5 --> Back-LED rechts (D16)
K6 --> Motor links dreht sich (Sollte sich der Motor nicht drehen kann im Selbsttest fortgefahren werden. Die Antriebe werden separat getestet hier kann dann ein evtl. vorliegender Fehler in der Motoransteuerung (siehe Abschnitt 6.8) behoben werden.)

Das Drücken von mehreren Tastern führt zur entsprechenden Kombination der Signale. Bei einem Fehler kann mit dem Selbsttest fortgefahren werden. Die Fehlerbehebung kann später stattfinden.

## Reflexlichtschranke (Odometrie)

Die Linienfolge-LED (D11) auf der Unterseite von ASURO leuchtet auf. Der nächste Test steht an (ca. 15sec). Die Reflexlichtschranken für die Odometrie werden überprüft. Ein weißes Stück Papier vor die Lichtschranke gehalten, sorgt für ein Aufleuchten der Status-LED. Papier wird vor die linke Lichtschranke gehalten (T11) --> Status-LED (D12) leuchtet grün auf. Bei der rechten Lichtschranke (T12) sollte die Status-LED (D12) rot aufleuchten. Papier weg und die entsprechende Farbe der Status-LED erlischt. Ein Hell-Dunkel-Übergang kann also detektiert werden. Die Odometrie funktioniert. Bei einem Fehler kann mit dem Selbsttest fortgefahren werden. Die Fehlerbehebung kann später stattfinden.

## Antriebe

Beide Back-LEDs (D15, D16) leuchten hell auf. Der vorletzte Test steht an (ca. 15sec). Die Antriebe werden auf Herz und Nieren geprüft. Der linke Motor wird in Vorwärtsrichtung von Stillstand auf maximale Drehzahl und wieder zurück zum Stillstand gebracht. Die Drehrichtung wird geändert und wieder von Stillstand auf Maximum zu Stillstand durchgefahren. Die selbe Prozedur muss der rechte Motor über sich ergehen lassen. Danach werden beide Motoren gleichzeitig betrieben. Zum letzten Mal die altbekannte Bemerkung: Bei einem Fehler kann mit dem Selbsttest fortgefahren werden. Die Fehlerbehebung kann später stattfinden.

## IR-Transceiver

Wenn die Status-LED geblich flackert, ist der letzte Test in vollem Gang (ca. 15sec). Die IR-Transceiver schickt bzw. empfängt Daten. Um diese empfangen zu können, ist der fertig zusammengebaute IR-Tranceiver an den PC anzuschließen und ein Terminalprogramm, wie das Windows Terminalprogramm “Hyperterminal” zu verwenden. Die Konfiguration ist die gleiche, wie beim Test des IR-Tranceivers .

Auf empfangenen Zeichen antwortet ASURO mit dem im Alfabet folgenden Zeichen. Kommen in einem fest eingestellten Zeitrahmen keine Daten an, sendet ASURO ’T’. Bei jedem gesendeten Zeichen wird zur grünen Status-LED die rote Status-LED dazugeschaltet, daher das gelbliche Flackern.

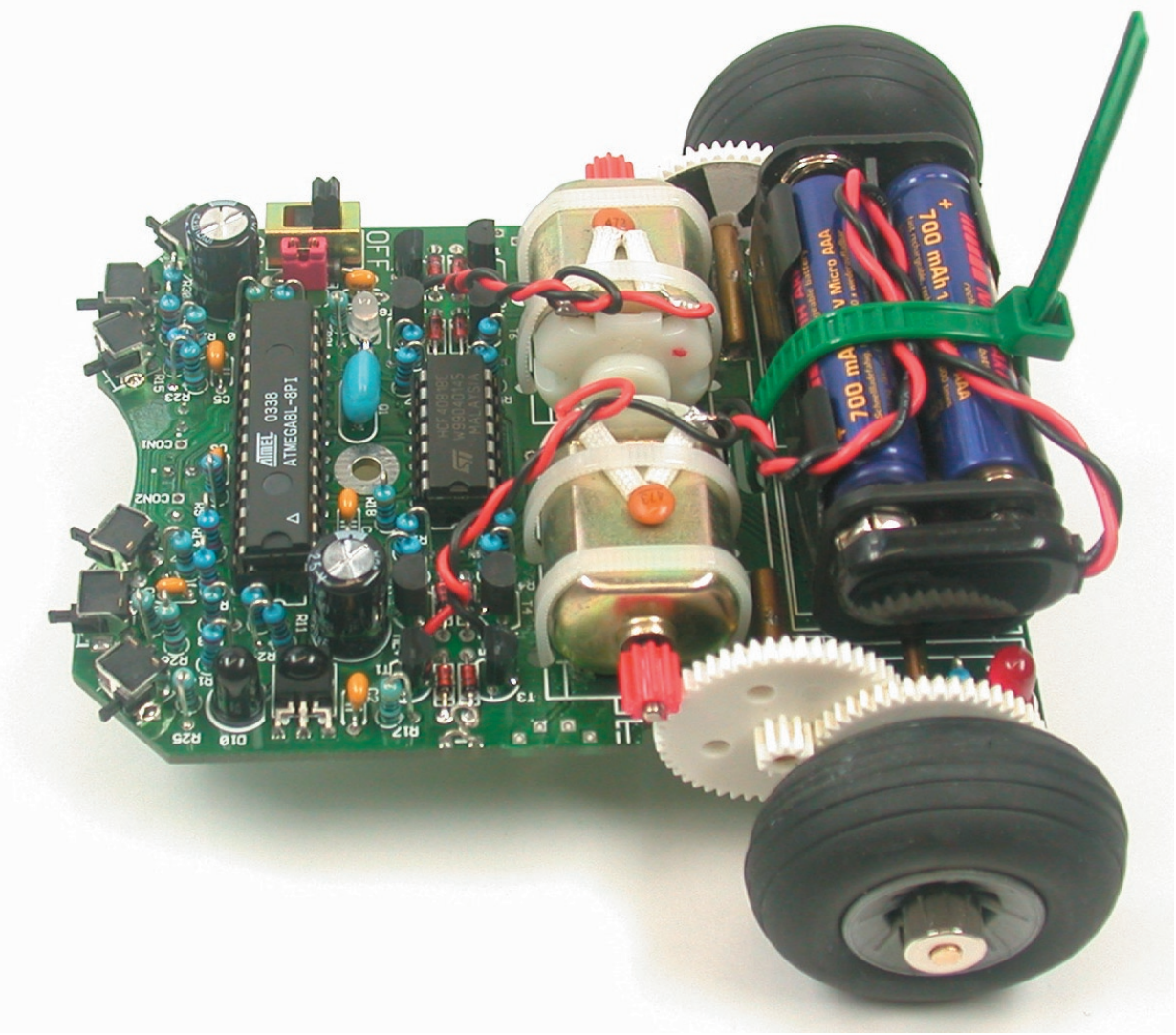
Besteht zwischen ASURO und dem Infrarot-Transceiver Sichtkontakt (ca. 50cm Abstand) sollte im Terminal-Programm regelmäßig ein 'T' erscheinen bzw. die am PC gedrückte Taste erscheint einmal (vom Transceiver gesendetes und reflektiertes Signal) gefolgt von dem im Alfabet folgenden (von ASURO gesendeten) Zeichen. z.B.:

*Taste "e" wird gedrückt => Terminal-Programm zeigt "ef"*
*Taste "j" wird gedrückt => Terminal-Programm zeigt "jk"*
*Taste "3" wird gedrückt => Terminal-Programm zeigt "34"*

Im Fehlerfall unter Abschnitt 6.9 nachsehen.

## 5.3.3. Fertig?

Ist ein Fehler aufgetreten, so muss die Batterie abgeklemmt und der Fehler mit Hilfe des Kapitels 6 behoben und - sofern keines der Bauteile Schaden genommen hat - anschließend der Selbsttest wiederholt werden. Erst dann ist ASURO voll einsatzfähig und es ist sichergestellt, dass in Zukunft die Fehler in der Software zu suchen sind und nicht in der Hardware. Soll später einmal ein Selbsttest durchgeführt werden, nachdem schon eigene Programme auf ASURO gelaufen sind, so ist ASURO mit der Datei "SelfTest.hex" von der CD zu programmieren.



# 6. Fehlersuche

## 6.1. RS232-IR-Transceiver geht nicht !

### 6.1.1. Tastendruck und Zeichenausgabe stimmen nicht überein

Solange am Trimmer TR1 drehen bis Tastendruck und Zeichenausgabe übereinstimmen.

### 6.1.2. Das Terminal-Programm gibt keine Zeichen aus

Ist das Timer-IC (IC1) montiert bzw. richtig herum (Kerbe zeigt in Richtung der drei Dioden) eingesteckt? Eine Infrarot-Fernbedienung eines beliebigen Gerätes (Videorecorder, Fernseher, Tuner, ...) nehmen, diese in Richtung des IR-Transceivers halten und einige Tasten drücken. Passiert gar nichts, nochmal die COM-Port-Einstellungen kontrollieren. Gibt das Terminal-Programm wirre Zeichen aus, funktioniert der Empfangsteil (IC2, R3, C4, D4, T1). Alle anderen Teile sind zu überprüfen.

### 6.1.3. Es geht immer noch nicht

Alle Bauteile auf richtige Einbaulage (Polung) und korrekten Wert prüfen (siehe Abb.4.1). Lötstellen auf Kurzschlüsse bzw. kalte Lötstellen nachsehen. Hat sich irgendwo ein Lötauge gelöst? Sind diese Kontrollen durchgeführt ohne einen Fehler zu finden, muss mit Hilfe des Schaltplanes (siehe Abschnitt B) und einem geeigneten Messgerät (Multimeter bzw. Oszilloskop) das defekte Bauteil gesucht werden. (IC1, IC2, Q1, D4 sind die wahrscheinlichsten Kandidaten für einen Defekt)

## 6.2. USB-Infrarot-Transceiver funktioniert nicht

### 6.2.1 Windows

Ist der Treiber ordungsgemäß installiert? Teilweise werden andere COM-Port-Nummern zugewiesen als die letzte. Hier hilft im Hyperterminal auch mal andere Ports einzustellen und den Test zu wiederholen. Ggf. kann man auch in der Systemsteuerung nachsehen, welcher Port zugewiesen wurde.

### 6.2.2 Linux

Hier kann es helfen, den USB-Transceiver abzustecken und nach etwas Warten wieder zu kontaktieren. Taucht dann der Eintrag im proc-Verzeichnis immer noch nicht auf, kann es nützen, einen neueren Kernel

## 6.3. Back-LEDs (D15,D16) glimmen nach dem Einschalten nicht!

### 6.3.1. Keine der beiden Back-LEDs glimmt auf

Ganz genau hinsehen und den Raum evtl. etwas abdunkeln. Ist immer noch nichts zu sehen, so sind folgende Dinge zu überprüfen:

- Sind alle 4 Batterien eingelegt und voll oder Akkus geladen?
- Ist das Batteriekabel polungsrichtig eingelötet? (rot Bat+ / schwarz Bat-)
- Ist Diode D9 (1N4001) polungsrichtig eingebaut?
- Hat R22 den richtigen Wert? 470Ω (ge,vio,br,gld)
- Eventuell auch

  R18 4,7KΩ (ge,vio,ro,gld)
  R19 1KΩ (br,sw,ro,gld)
  R20 4,7KΩ (ge,vio.ro,gld)
  R21 1KΩ (br,sw,ro,gld)

  überprüfen.

### 6.3.2. Nur eine der beiden LED's glimmt auf

Sind die Dioden (rosafarbenes Gehäuse) D13 (links) ,D14 (rechts), sowie die Fototransistoren (klares Gehäuse) T11 (links), T12 (rechts) an der richtigen Stelle (siehe Abb.4.3) und Polungsrichtig eingebaut?

Haben folgende Widerstände den richtigen Wert R18, R19 (links) und R20, R21 (rechts)?
4,7KΩ (ge,vio,ro,gld)
1KΩ (br,sw,ro,gld)

Sind die Bauteile alle an der richtigen Stelle eingebaut? (Bei den Widerständen auf den Platinenaufdruck achten!)

### 6.3.3. Status-LED (D12) leuchtet nach dem Start nicht zweifarbig auf

Status-LED leuchtet gar nicht => siehe 6.4!

Status-LED flackert => Batteriespannung zu niedrig => Batterien tauschen. Sind die Batterien frisch, sollten die Widerstände R12 und R13 überprüft werden.
12KΩ (br,ro,sw,ro,br)
10KΩ (br,sw,sw,ro,br)

## 6.4. Ein Anzeigenelement geht nicht

Prozessor richtig eingebaut? (Polung!)

### 6.4.1. Status-LED D12 geht nicht

Polung der LED D12 überprüfen.

Widerstand R10, R31 prüfen.
470Ω (ge,vio,br,gld)

Ein einfacher Test besteht darin, den Prozessor (IC1) zu entfernen und eine Verbindung zwischen Pin7* (VCC) und Pin14 (Status-LED leuchtet grün) bzw. Pin4 (Status-LED leuchtet rot) herzustellen. Ist dieser Test erfolgreich, liegt ein Defekt im Prozessor oder Schwinger vor bzw. ist eine Leiterbahn unterbrochen.

**** Pin1 ist links oben, dann wird links runter und rechts hoch gezählt***

### 6.4.2. Front-LED D11 geht nicht

Polung D11 prüfen.

Widerstand R9 prüfen.
220Ω (ro,ro,br,gld)

Ein einfacher Test besteht darin, den Prozessor (IC1) zu entfernen und eine Verbindung zwischen Pin7* (VCC) und Pin12 Front-LED leuchtet rot herzustellen. Ist dieser Test erfolgreich, liegt ein Defekt im Prozessor oder Schwinger vor bzw. ist eine Leiterbahn unterbrochen.

**** Pin1 ist links oben, dann wird links runter und rechts hoch gezählt***

### 6.4.3. Linke Back-LED D15 geht nicht

Polung D15 prüfen.

Widerstände R19, R18 überprüfen.
1KΩ (br,sw,or,gld)
4,7KΩ (ge,vio,ro,gld)

Ein einfacher Test besteht darin, den Prozessor (IC1) zu entfernen und eine Verbindung zwischen Pin7 (VCC) und Pin24 herzustellen, die linke Back-LED leuchtet rot. Ist dieser Test erfolgreich, liegt ein Defekt im Prozessor oder Schwinger vor, bzw. ist eine Leiterbahn unterbrochen.

### 6.4.4. Rechte Back-LED D16 geht nicht

Polung D16 prüfen.

Widerstände R21, R20 überprüfen.
1KΩ (br,sw,ro,gld)
4,7KΩ (ge,vio,ro,gld)

Ein einfacher Test besteht darin, den Prozessor (IC1) vorsichtig zu entfernen und eine Verbindung zwischen Pin7 (VCC) und Pin23 herzustellen, rechte Back-LED leuchtet dann rot. Ist dieser Test erfolgreich, liegt ein Defekt im Prozessor oder Schwinger vor, bzw. ist eine Leiterbahn unterbrochen.

## 6.5. Linienfolgesensor (T9, T10) reagiert nicht

Polung T9, T10 überprüfen.

Widerstand R14 20KΩ (ro,sw,or,gld) , R15 20KΩ (ro,sw,or,gld) überprüfen.

Auch darauf achten, dass R15 nicht mit R23 oder R28 zusammen verdreht eingebaut wurden!
Mit einem Multimeter kann an Pin 25 bzw. Pin 26 das Sensorsignal bei ausgebautem Prozessor nachgemessen werden. (dunkel 0V, hell VCC).

## 6.6. Ein Schalter funktioniert nicht richtig

### 6.6.1. Angeblich ist eine Kombination aus Schaltern gedrückt worden

R12 12KΩ (br,ro,sw,ro,br) und R13 10KΩ (br,sw,sw,ro,br) überprüfen.

Zu kontrollieren sind , R25, R26, R27, R28, R29, R30, R32!

| | | |
|---|---|---|
| R24 | 1KΩ | (br,sw,sw,br,br) |
| R25 | 2KΩ | (ro,sw,sw,br,br) |
| R26 | 2KΩ | (ro,sw,sw,br,br) |
| R27 | 8,2KΩ | (gra,ro,sw,br,br) |
| R28 | 16KΩ | (br,bl,sw,ro,br) |
| R29 | 33KΩ | (or,or,sw,ro,br) |
| R30 | 68KΩ | (bl,gr,sw,ro,br) |
| R32 | 2KΩ | (ro,sw,sw,br,br) |

In seltenen Fällen kann es vorkommen, dass durch Bauteiletoleranzen ein falsches Muster erkannt wird. Das kann aber später softwareseitig behoben werden.”

### 6.6.2. Die Anzeige verhält sich so, als seien Schalter vertauscht worden

Die Widerstände der jeweiligen Schalter sind vertauscht.
Zu kontrollieren sind

| | | |
|---|---|---|
| R24 | 1KΩ | (br,sw,sw,br,br) |
| R25 | 2KΩ | (ro,sw,sw,br,br) |
| R26 | 2KΩ | (ro,sw,sw,br,br) |
| R27 | 8,2KΩ | (gra,ro,sw,br,br) |
| R28 | 16KΩ | (br,bl,sw,ro,br) |
| R29 | 33KΩ | (or,or,sw,ro,br) |
| R30 | 68KΩ | (bl,gr,sw,ro,br) |
| R32 | 2KΩ | (ro,sw,sw,br,br) |

### 6.6.3. Irgendwie funktioniert es immer noch nicht so richtig

Widerstände R23 1MΩ (br,sw,gr,gld), R24 1KΩ (br,sw,ro,gld) nachsehen,
R12 12KΩ (br,ro,sw,ro,br), R13 10KΩ (br,sw,sw,ro,gld) kontrollieren und C7 220F/10V überprüfen!

## 6.7. Eine Reflexlichtschranke funktioniert nicht

### 6.7.1. Keine der Reflexlichtschranken funktioniert

Widerstand R22 prüfen!
470 Ω (ge,vio,br,gld)

Einbaulage von D13 und D14 nachsehen.
D13 und D14 sind die rosa zweipoligen Gebilde mit dem kleinen Pickel auf einer Seite. Dieser Pickel muss zur Außenseite der Platine zeigen.

### 6.7.2. Die linke Reflexlichtschranke funktioniert nicht

Widerstand R18 prüfen!
4,7KΩ (ge,vio,ro,gld)

Einbaulage von T11 prüfen. T11 ist das klare zweipolige Gebilde mit einem kleinen Pickel, der zur Außenseite der Platine zeigen muss.

### 6.7.3. Die rechte Reflexlichtschranken funktioniert nicht

Widerstand R20 prüfen!
4,7 KΩ (ge,vio,ro,gld)

Einbaulage von T12 prüfen. T12 ist das klare oder schwarze zweipolige Gebilde mit einem kleinen Pickel, der zur Außenseite der Platine zeigen sollte.

Mit einem Multimeter kann an Pin24 bzw. Pin23 das Sensorsignal bei ausgebautem Prozessor nachgemessen werden. (dunkel VCC, hell 0V)

## 6.8. Ein Antrieb geht nicht

### 6.8.1. Kein Antrieb bewegt sich

Polung und Einbaulage IC3 überprüfen.

### 6.8.2. Der linke Motor bewegt sich nicht bzw. nur in eine Richtung

Hier ist die komplette Motorbrücke, bestehend aus Transistoren T1, T2, T3, T4 (sind die richtigen Transistoren an der richtigen Stelle eingebaut), den Dioden D1, D2, D3, D4 (Polung !) und den Widerständen R1, R2, R3, R4 zu kontrollieren.

T1, T3 (BC327-40 oder BC328-40), T2, T4 (BC337-40 oder BC338-40)

R1, R2, R3, R4  1KΩ (br,sw,ro,gld)

### 6.8.3. Der rechte Motor bewegt sich nicht bzw. nur in eine Richtung

Hier ist die komplette Motorbrücke, bestehend aus Transistoren T5, T6, T7, T8 (sind die richtigen Transistoren an der richtigen Stelle eingebaut?), den Dioden D5, D6, D7, D8 (Polung!) und den Widerständen R5, R6, R7, R8 zu kontrollieren.

T5, T7 (BC327-40 oder BC328-40), T6, T8 (BC337-40 oder BC338-40)

R5, R6, R7, R8  1KΩ (br,sw,ro,gld)

### 6.8.4. Ein Motor dreht in die falsche Richtung

Die beiden Anschlusskabel am Motor, der in die verkehrte Richtung dreht müssen vertauscht werden.

## 6.9. IR-Schnittstelle

### 6.9.1. ASURO sendet keine Zeichen

Polung der IR-Diode D10 prüfen.
Widerstand R16 richtig?
220 Ω  (ro,ro,br,gld)

### 6.9.2. ASURO empfängt keine Zeichen

Zwischen IR-Transceiver und ASURO muss Sichtverbindung bestehen (Abstand ca. 50cm) und der IR-Transceiver muss voll funktionsfähig sein (siehe Abschnitt 6.1).

IC2 richtig eingebaut?
Widerstand R17 und C2 überprüfen!
100 Ω  (br,sw,br,gld)
100nF  (Aufdruck104)

Wer den Fehler bisher noch nicht gefunden hat, der möge sich überlegen, ob er das IC2 eingelötet oder “eingeschweißt” hat. IC2 ist ein wenig hitzeempfindlich und ist evtl. beim Einbauen kaputt gegangen, dann neues IC (SFH 5110-36) besorgen und einbauen.

## 6.9.3. Es geht immer noch nicht so richtig

C8 polungsrichtig eingebaut ?
220F/ mindestens 10V

Kommt es bei der Übertragung von Daten vom PC zu ASURO immer wieder zu Schwierigkeiten, so muss ein wenig am Trimmer TR1 des Transceivers gedreht werden.


***Zweikomponentenkleber steht im Verdacht sensibilisierend zu wirken, was soviel heißt wie: man wird gegen immer mehr Zeug allergisch, er darf also keinesfalls mit der Haut in Berührung kommen. Vinylhandschuhe können hierbei sehr hilfreich sein. Falls doch was passiert, sofort und gründlich mit Seife abwaschen! Sekundenkleber wurde ursprünglich für die Chirurgie entwickelt. Man merkt das dann, wenn er innerhalb von Sekunden Hautteile zusammenklebt. Sofern das mit Fingern passiert, kann man mit warmen Wasser, Seife und etwas Geduld die Verbindung wieder lösen. Keinesfalls darf das mit Lippen oder Augenliedern passieren! Irgendwelche Gesichtskratz- oder Augenwischreflexe sind beim Arbeiten mit Sekundenkleber in jedem Fall zu unterdrücken!***

# 7. Letzte Einstellarbeiten

Die Achsen werden leicht eingefettet, das Getriebezahnrad mit dem schönen Schwarz-Weiß-Muster wird auf die kurze Achse gesteckt. Der Reifen wird auf das Getrieberad mit den 50 und 12 Zähnen gesteckt, dann zusammen auf die hintere Achse gesteckt und mit einem Stellring fixiert, sodass es sich noch leicht drehen kann. Der provisorisch befestigte Motor wird vorsichtig solange verschoben, bis er gerade ausgerichtet ist, das Motorritzel auf der gesamten Breite des ersten Getriebezahnrades eingreift und sich Motorritzel und Getrieberad leicht drehen lassen. (Evtl. nochmals den kompletten Selbsttest durchlaufen lassen um zu sehen ob sich beim Motortest alles dreht.)
Ist die Position für gut befunden, werden Motor und Platine krampfhaft festgehalten und mit einem Tropfen Sekundenkleber, welcher seitlich auf den Spalt zwischen Motor und Platine getropft wird fixiert. Hier ist zu beachten, dass Sekundenkleber auch mal ein paar Minuten brauchen kann, bis er fest wird.

Der halbe Tischtennisball wird nun noch mit zwei gegenüberliegenden Tropfen Sekundenkleber auf die Unterseite der Platine direkt hinter die Linienfolge-Bauteile (siehe Abb.7.1) geklebt und trocknen gelassen.

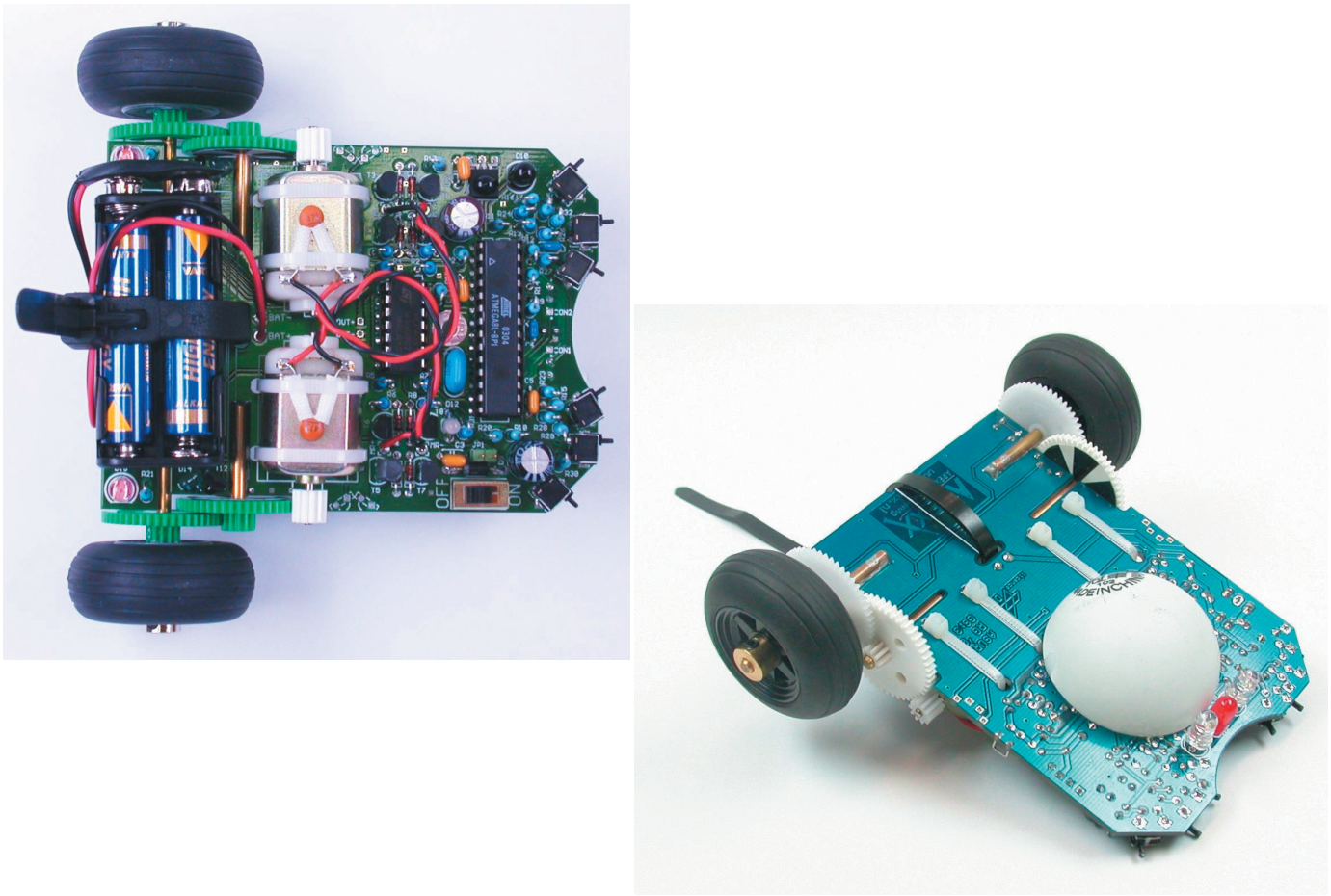


**Abbildung 7.1.: ASURO fertig aufgebaut**

# Teil III. Informatik

## 8. Installation der Software und erste Schritte

Die ASURO-CD ins CDROM-Laufwerk einlegen, sie ist autostartfähig. Falls Autostart deaktiviert ist, kann man die CD auch mit dem Windows-Explorer öffnen. Nach der Auswahl der Sprache findet sich im Abschnitt “Software” alles, was für den Betrieb von ASURO erforderlich ist. Diese Programme müssen zunächst installiert werden. Für die Installation des Compilers sind Administrator-Rechte erforderlich. Falls der aktuelle Benutzer diese Rechte nicht besitzt, abmelden und als Administrator wieder anmelden.

Während der Softwareinstallation werden folgende Schritte durchgeführt:

1. Das Flash-Tool zum Übertragen der eigenen Programme auf ASURO wird installiert
2. Ein Programmeditor (Programmers Notepad 2, PN2) und ein Compilierer (WinAVR) wird installiert.
3. Ein Beispielprogramm wird von CDROM auf Festplatte kopiert.
4. Im Programmditor (PN2) wird je ein Menüeintrag für MAKE und für CLEAN eingerichtet.

### 8.1 Windows 

### 8.1.1 Flash-Tool 

Das Flash-Tool kann entweder in ein Verzeichnis auf der Festplatte kopiert (z.B.:C:\Programme\Flash) oder später direkt von CD ausgeführt werden. In jedem Fall ist es hilfreich, wenn man sich einen Link auf den Desktop einrichtet, um das Flash-Tool einfach starten zu können.

Klick auf [Save]


### 8.1.2 Installation des Programmeditors und des Compilers

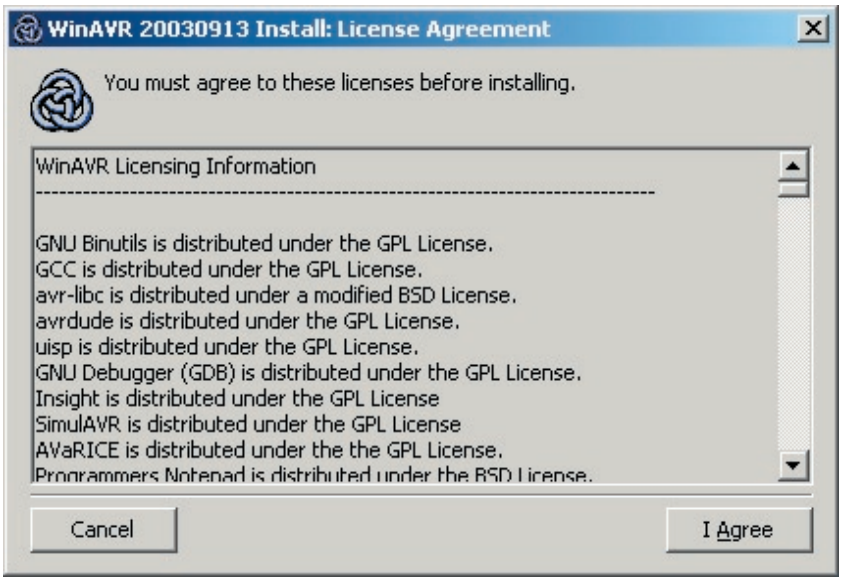
Für die Installation des Compilers sind Administratorrechte erforderlich (weil bei der Installation die Registry geändert wird). Sollte der aktuelle Benutzer diese nicht besitzen, abmelden und als Administrator wieder anmelden!

Klick auf [Install]

**COMPILER WinAVR (20030913)**

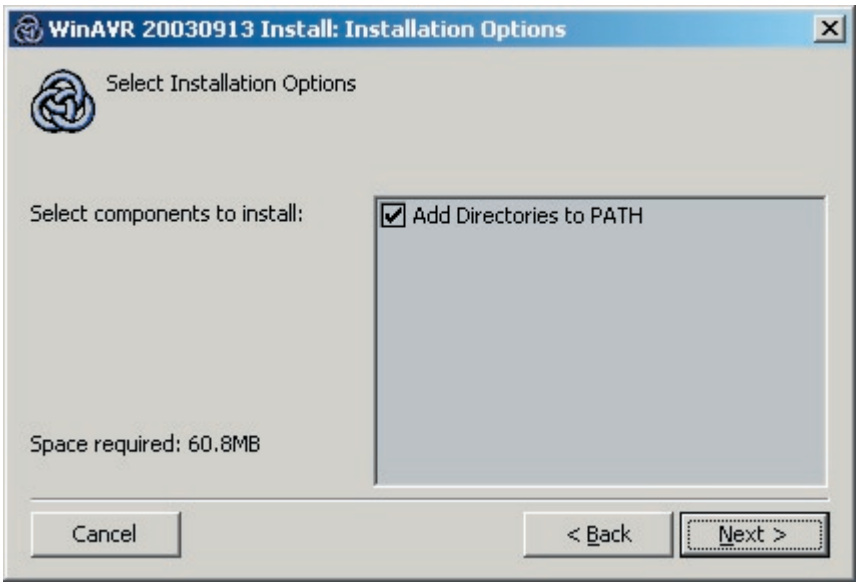
***WINDOWS is the trade mark of Microsoft Corporation.***

Folgendes Fenster erscheint:



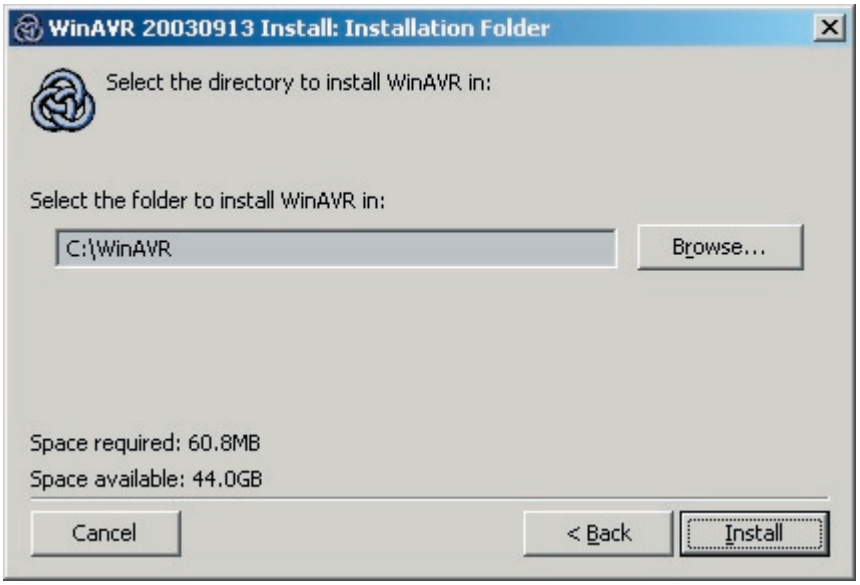

Klick auf [I Agree]

Folgendes Fenster erscheint:



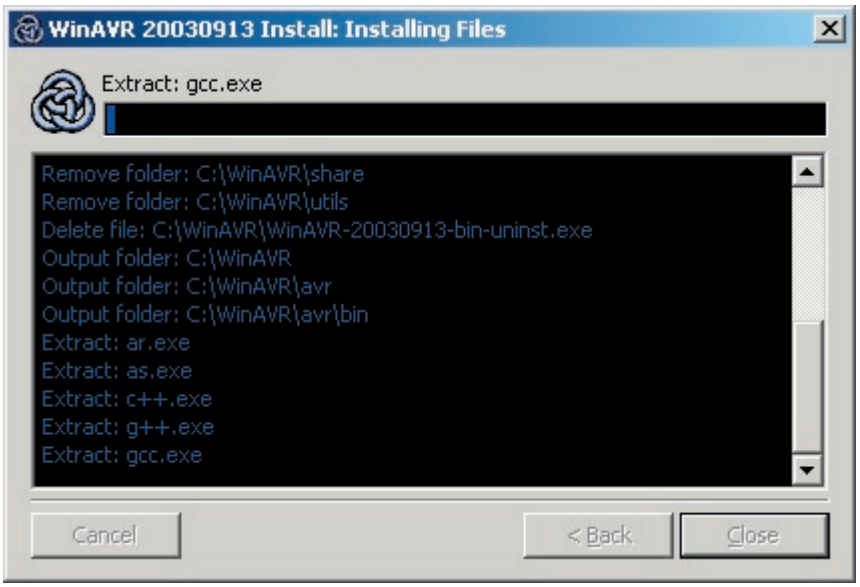

Klick auf [Next]

Folgendes Fenster erscheint:



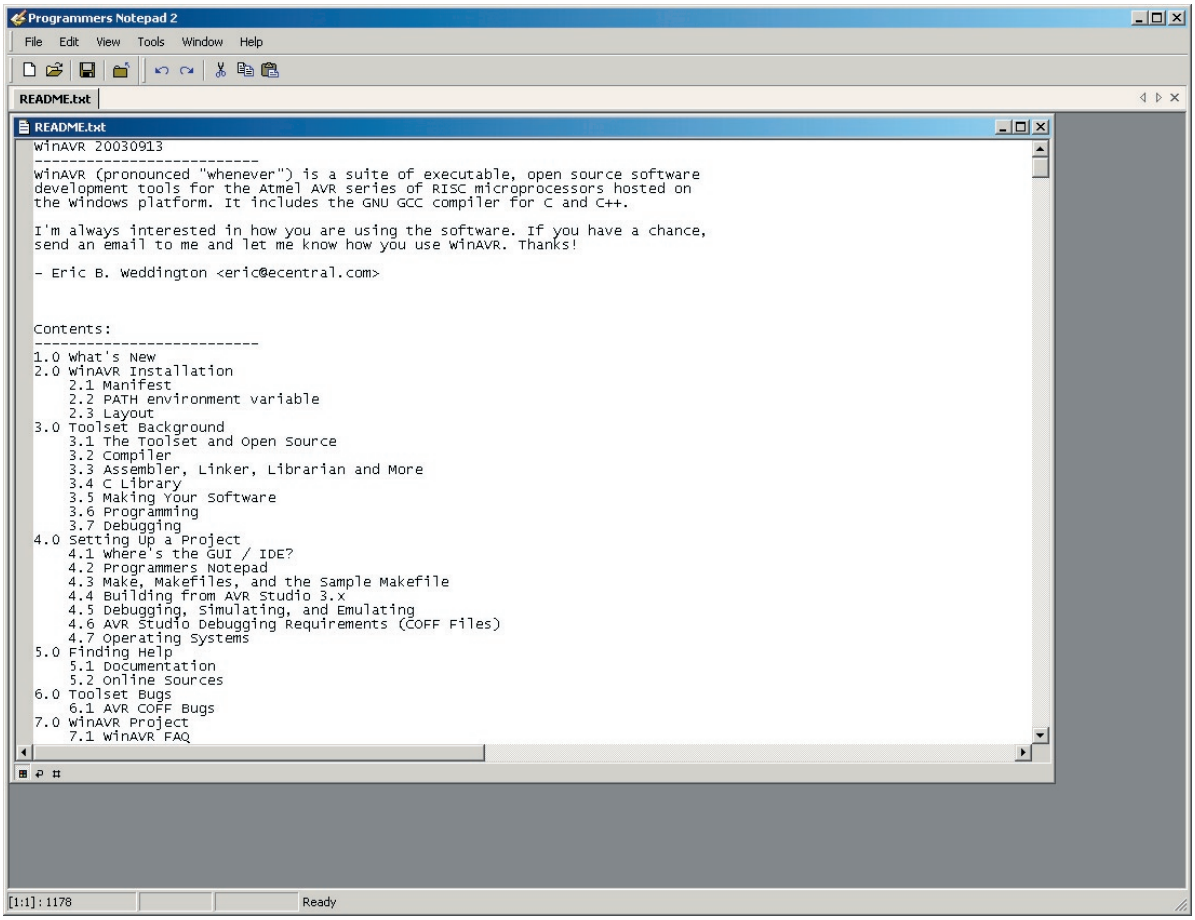

Klick auf [Install]

Folgendes Fenster erscheint:




Warten…

…bis der Programmers Notepad 2 (PN2) Editor mit der Datei README.txt erscheint.




Nun das Fenster ‘Programmers Notepad 2’ schließen.

Auf dem Desktop erscheint das ‘Programmers Notepad 2’ Symbol:


Der Programmeditor und der Compiler sind jetzt installiert.

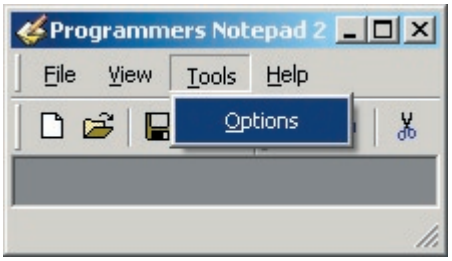
## 8.1.3. Beispielprogramme von CDROM auf die Festplatte kopieren.

Von der CD den Ordner ‘ASURO_src’ in einen beliebigen Ordner (Z.B. ‘C:\ASURO_src’) auf der Festplatte kopieren.

Durch Markieren der kopierten Dateien im Zielverzeichnis, Klicken mit der rechten Maustaste und Auswahl von Eigenschaften sicherstellen, dass der Schreibschutz deaktiviert ist.

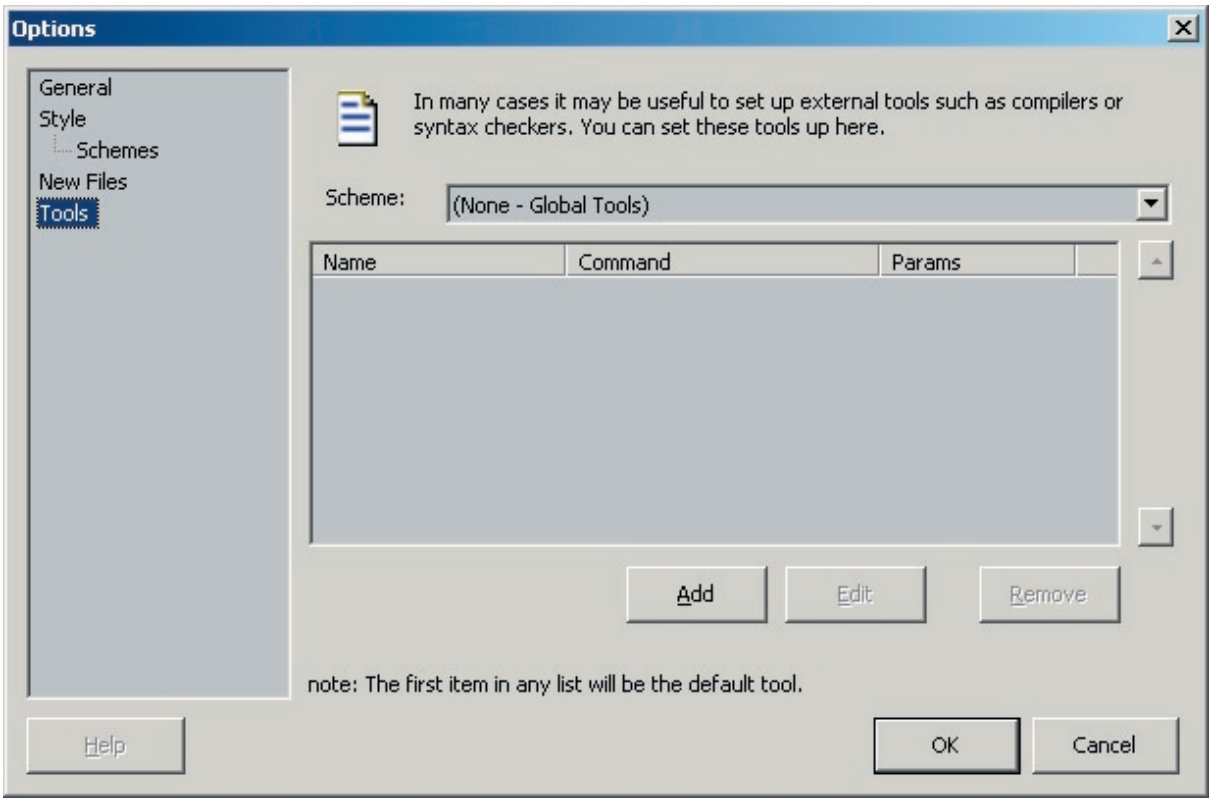
**Im Programmeditor einen Menüeintrag zum Compilieren einrichten**

‘Programmers Notepad 2’ durch Doppellklick auf das Symbol auf dem Desktop ‘Programmers Notepad’ öffnen:



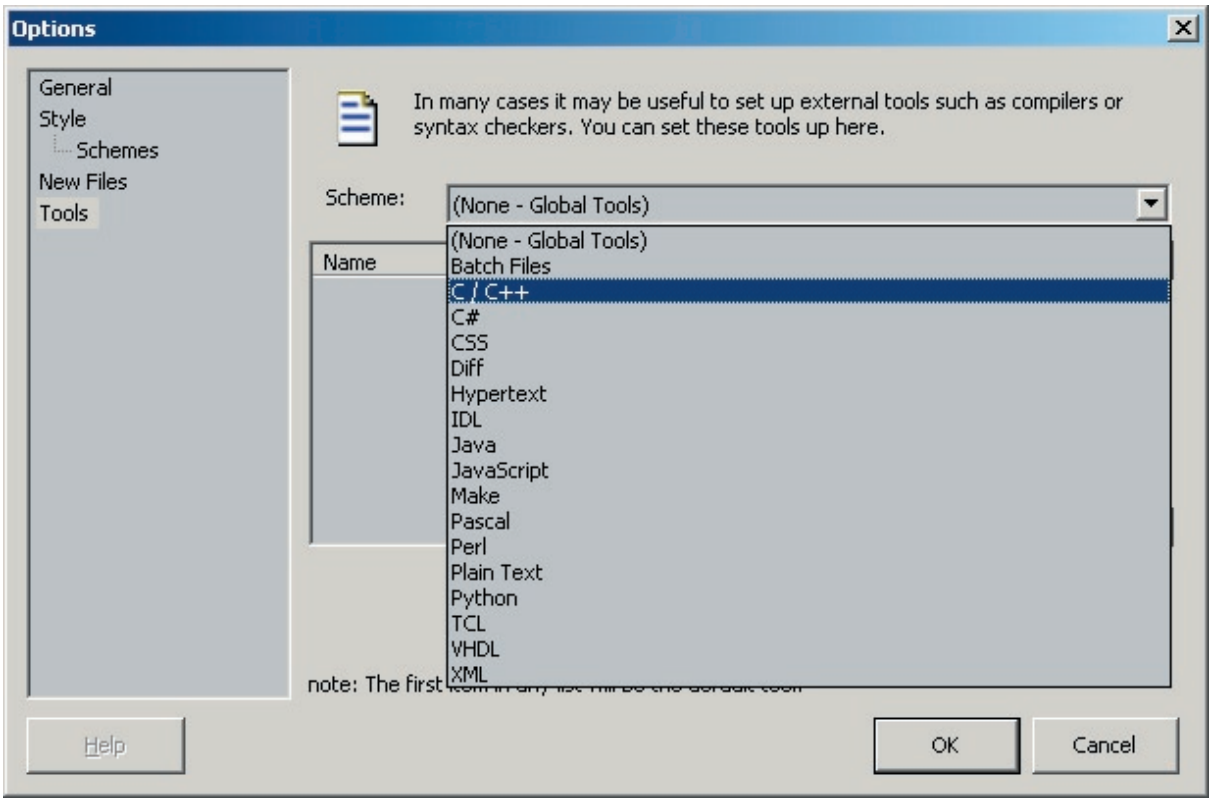

Im Menü Tools | Options auswählen.

Das Options-Fenster erscheint.

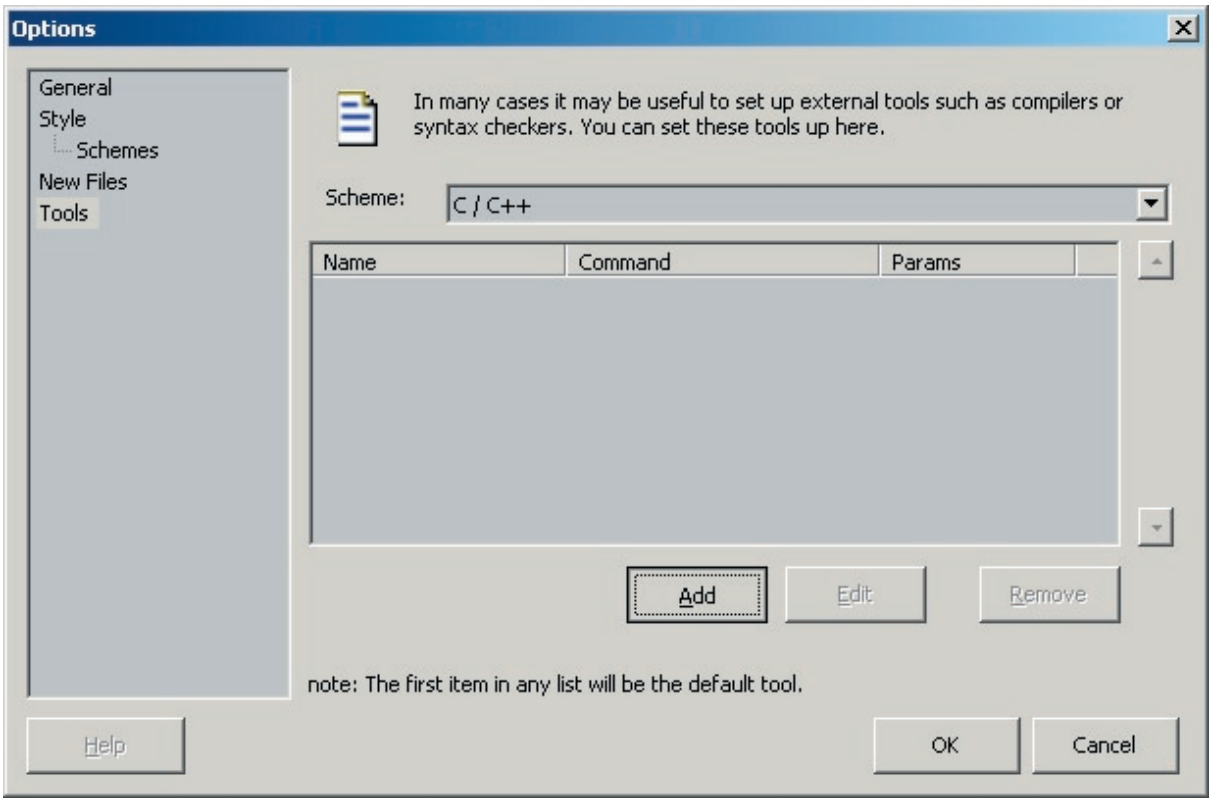



Jetzt Tools auswählen.

Auf der rechten Seite ‘C/C++’-Scheme auswählen.

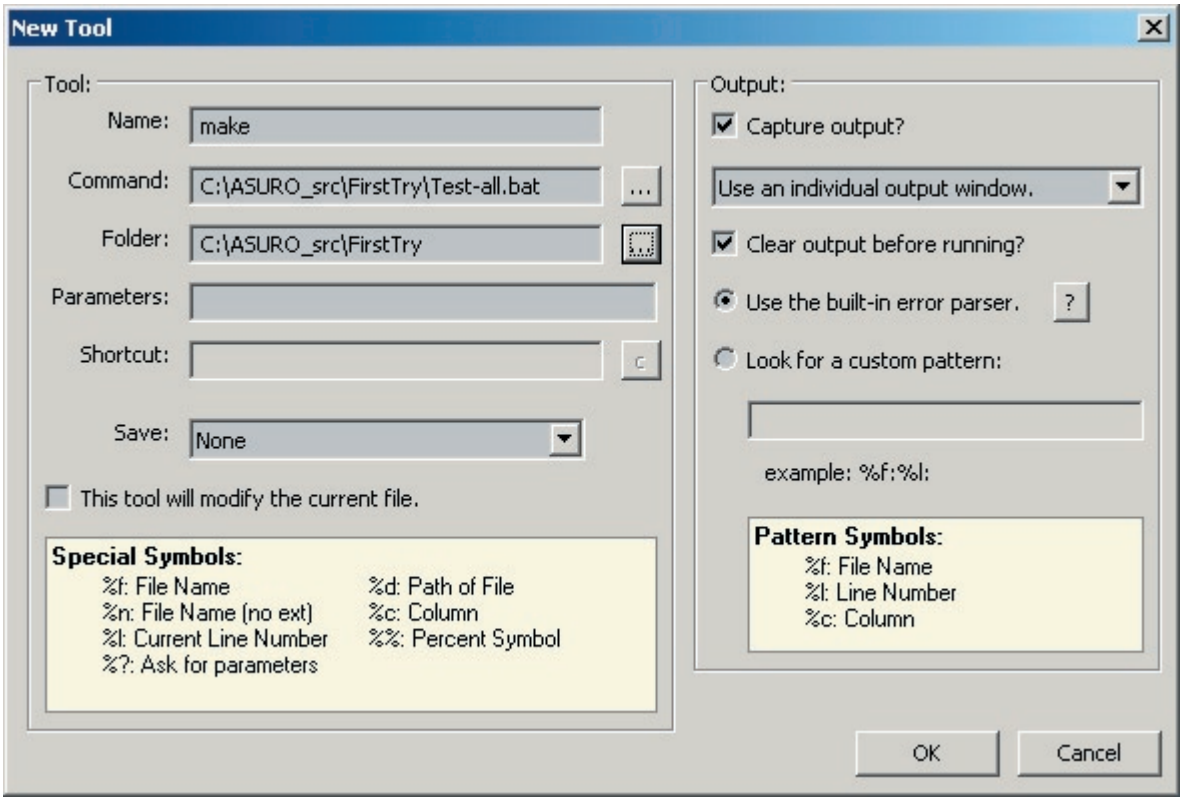



‘C/C++’-Scheme ist ausgewählt.




Auf [Add] klicken (…um ein neues Tool hinzuzufügen)

Das Fenster ‘New Tool’ erscheint.



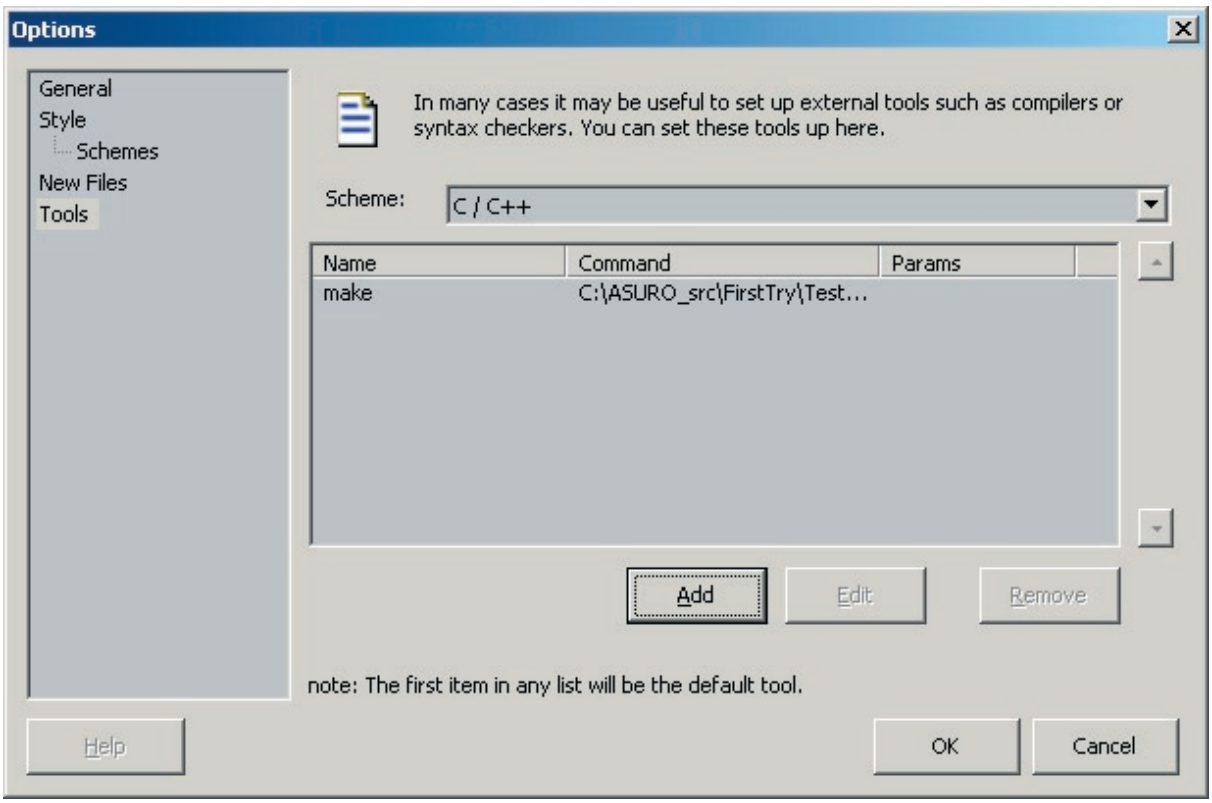

Folgende Einstellungen eintippen oder mit der Browse-Taste … auswählen:
Name: make
Command: C:\ASURO_src\FirstTry\Test-all.bat
Folder: C:\ASURO_src\FirstTry

Klick auf [OK]

Ein neues PN-Tool mit Namen make ist ab sofort im Tools-Hauptmenü verfügbar. (Wird das Tool aktiviert, so wird eine Batch-Datei mit Namen Test-clean.bat ausgeführt, welche das Programm test.c - zusammen mit asuro.c - compiliert und eine datei text.hex erzeugt)

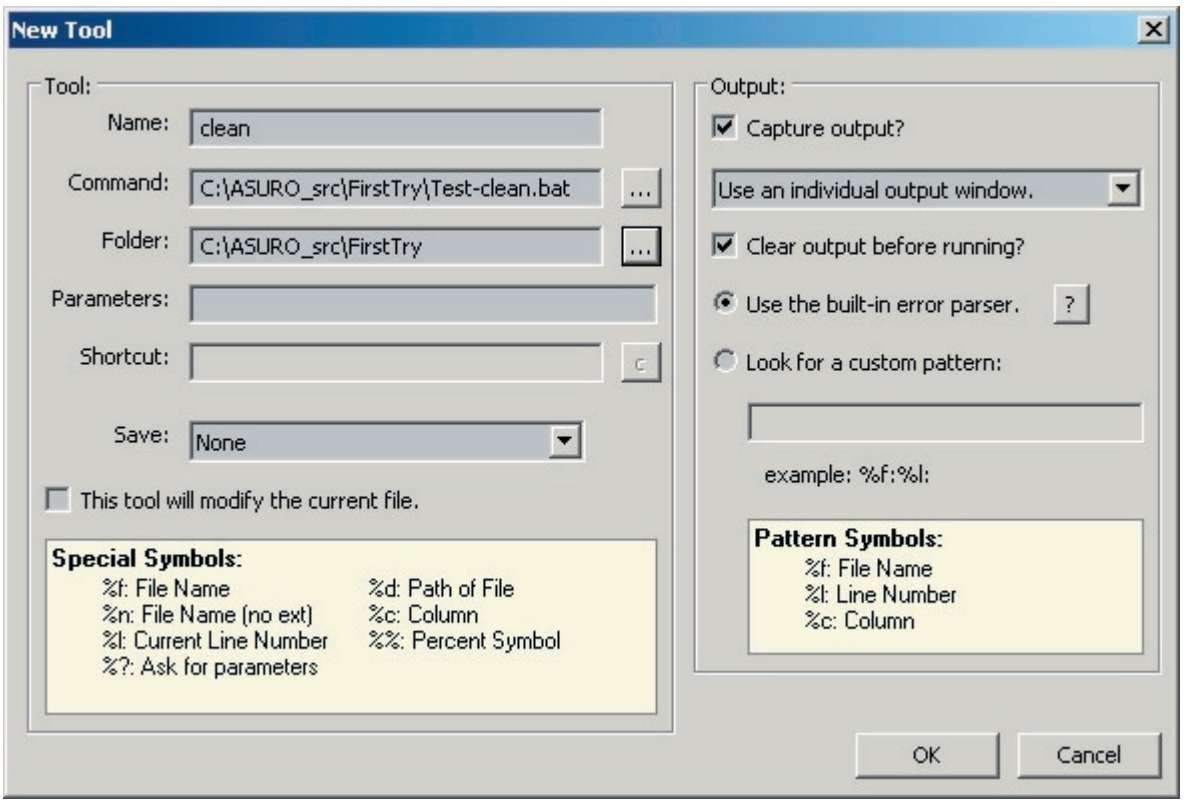
## Im Programmeditor einen Menüeintrag zum Aufräumen einrichten.

Im Hauptmenü “Tools” wieder “Options” und dort wieder “C/C++-Scheme” auswählen:




Auf [Add] klicken um ein weiteres Tool hinzuzufügen:

Das Fenster ‘New Tool’ erscheint.

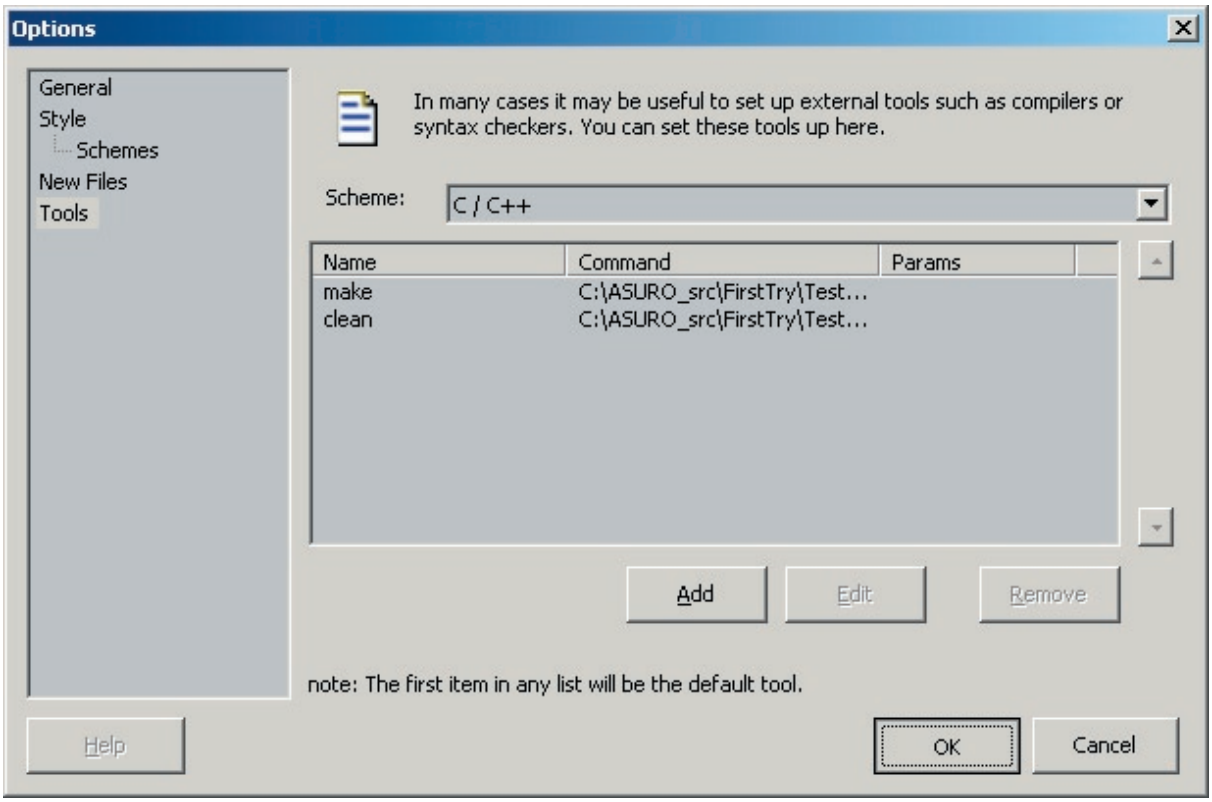



Folgende Einstellungen eintippen oder mit der Browse-Taste ... auswählen:
Name: clean
Command: C:\ASURO_src\FirstTry\Test-clean.bat
Folder: C:\ASURO_src\FirstTry

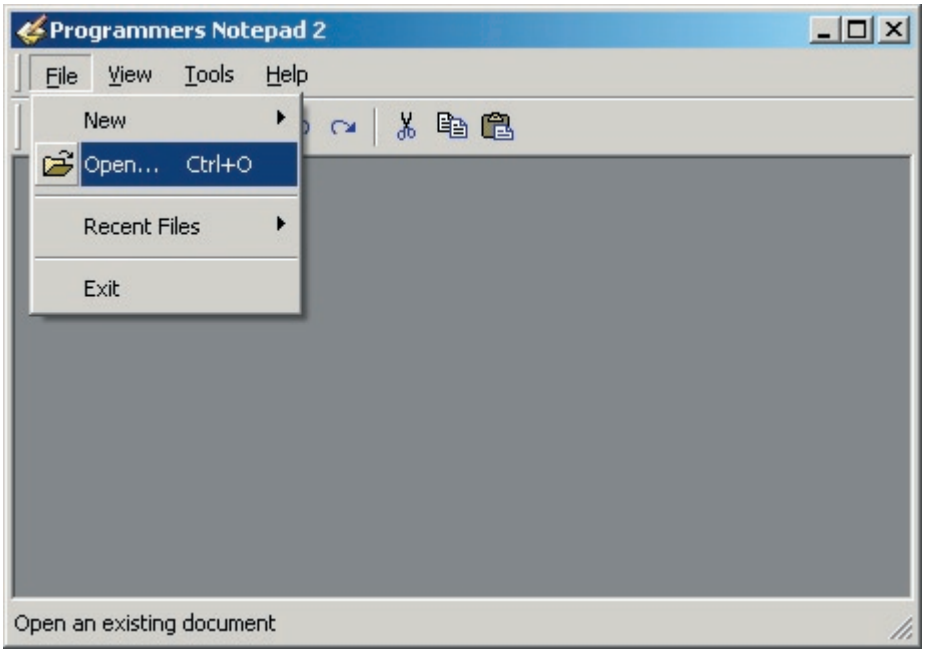
Auf [OK] klicken

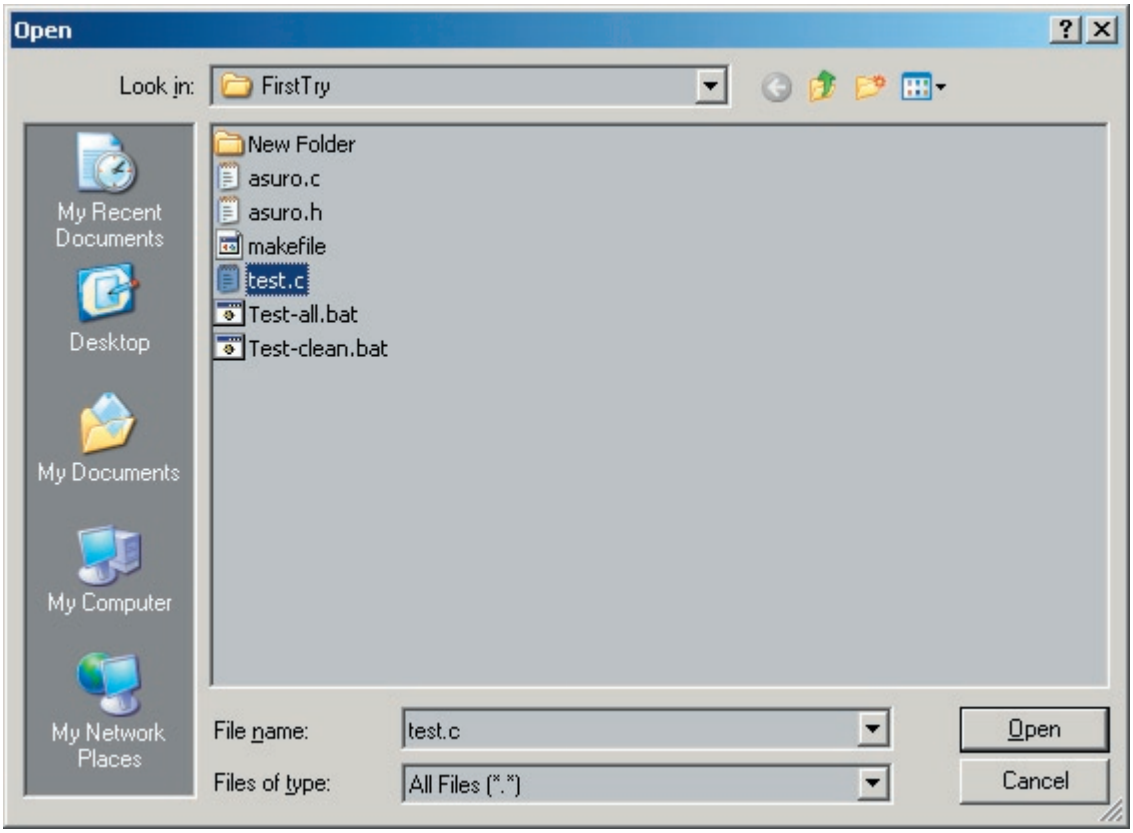
Ein neues PN-Tool mit Namen clean ist ab sofort im Tools-Hauptmenü verfügbar. (Wird das tool aktiviert, so wird eine Batch-Datei mit Namen Test-clean.bat ausgefuhrt, welche temporäre Dateien im Ordner C:\ASURO_src\FirstTry löscht.

Im Options-Fenster müssen jetzt die beiden Einträge für die Tools ‘make’ und ‘clean’ sichtbar sein.




Wieder auf [OK] klicken.

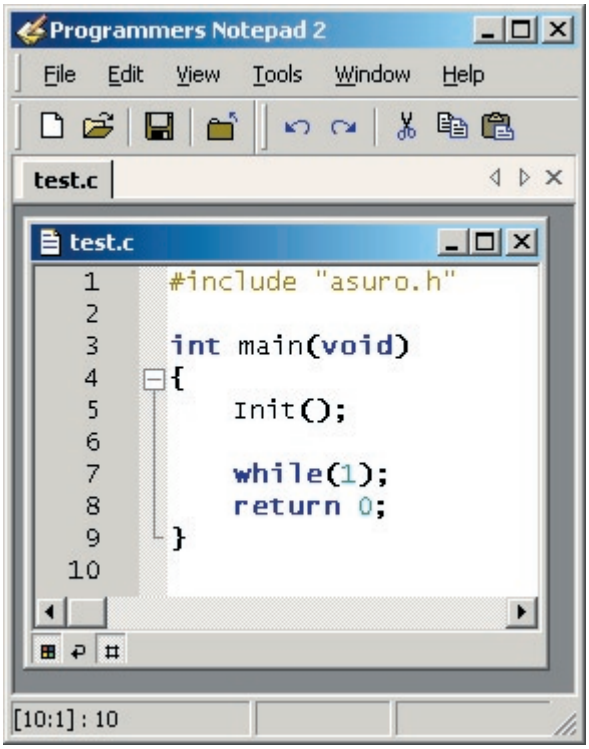




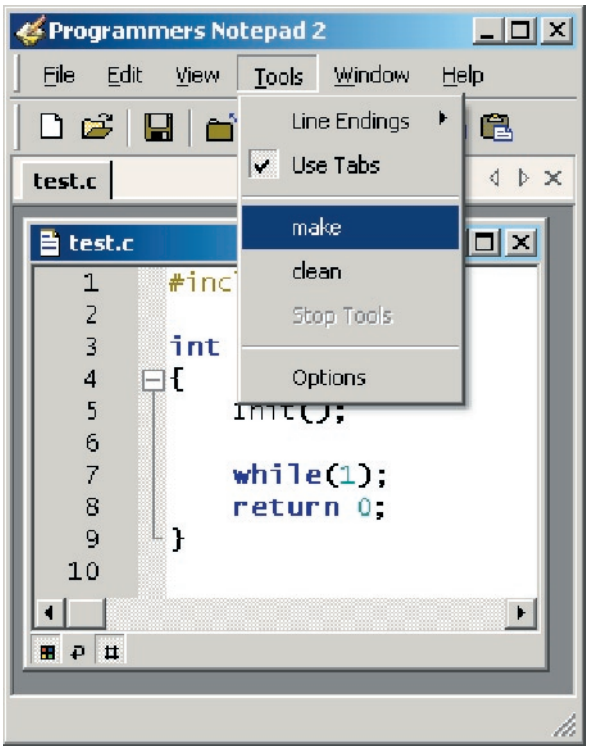

Auf [Open] klicken.

Nun zum Ausprobieren die Datei Datei 'C:\ASURO_src\FirstTry\test.c' öffnen:

Datei test.c wird geöffnet.



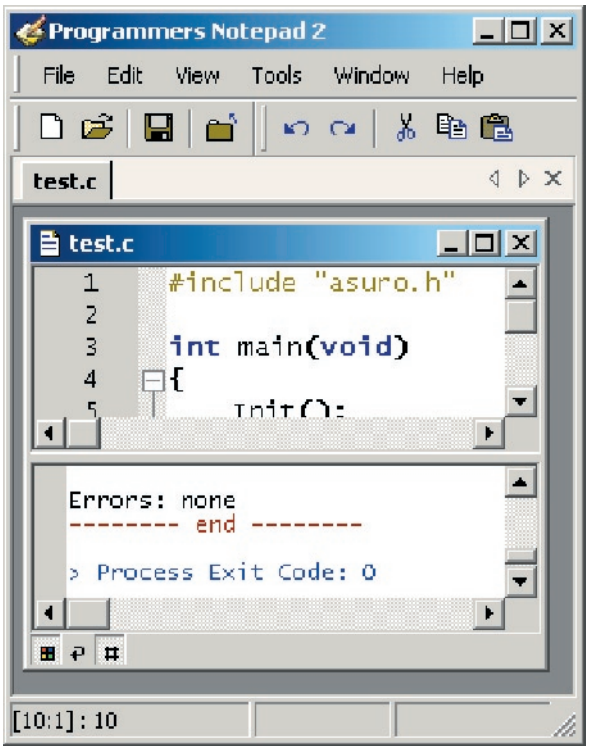

Wenn man Tools auswählt…




…sieht man die neuen Tools make und clean als Menüeintrag.

Nun make anklicken.

Die Dateien test.c (zusammen mit asuro.c) wird nun compiliert…




…und wenn das Programm keine Fehler enthält (was zu erwaren ist, da gerade das Beispielprogramm geladen ist), erscheint unten die Meldung: Errors: none.

Was ist passiert?
Aus der Datei test.c (und asuro.c) ist eine Datei test.hex generiert worden. Diese Datei stellt das in Maschinencode übersetzte Programm dar, welches nun in ASUROs Speicher geladen (geflasht) werden kann. Das Programm selbst macht noch gar nichts, wir werden es später aber zum Ausprobieren des Flash-Tools benötigen.

Wie hat's funktioniert?
Der Menüeintrag make ruft die Batch-Datei Test-all.bat auf (eine Batch-Datei enthält eine Liste mit Kommandozeilenbefehlen, die der Reihe nach ausgeführt werden).
In Test-all.bat wird der Befehl 'make all' ausgeführt. 'make' führt immer ein makefile aus, das sich (bei der ASURO-Programmierung) im gleichen Ordner befinden muss, wie Test-all.bat .

Ein makefile ist eine Textdatei, die festlegt, wie ein oder mehrere Programm compiliert werden müssen. Bei Programmen, die nur aus einer Datei übersetzt werden ist das noch recht übersichtlich. Nachdem aber ganze Betriebssysteme in C geschrieben werden und der Code auf mehrere Dateien aufgeteilt ist, die alle in einer bestimmten Reihenfolge übersetzt und zusammengebunden (gelinkt) werden müssen, kann auch ein makefile sehr aufwändig werden. Das 'all' ruft den Eintrag im makefile namens 'all' auf, was bedeutet, dass ein komplettes Projekt und nicht nur einzelne Teile übersetzt werden sollen.

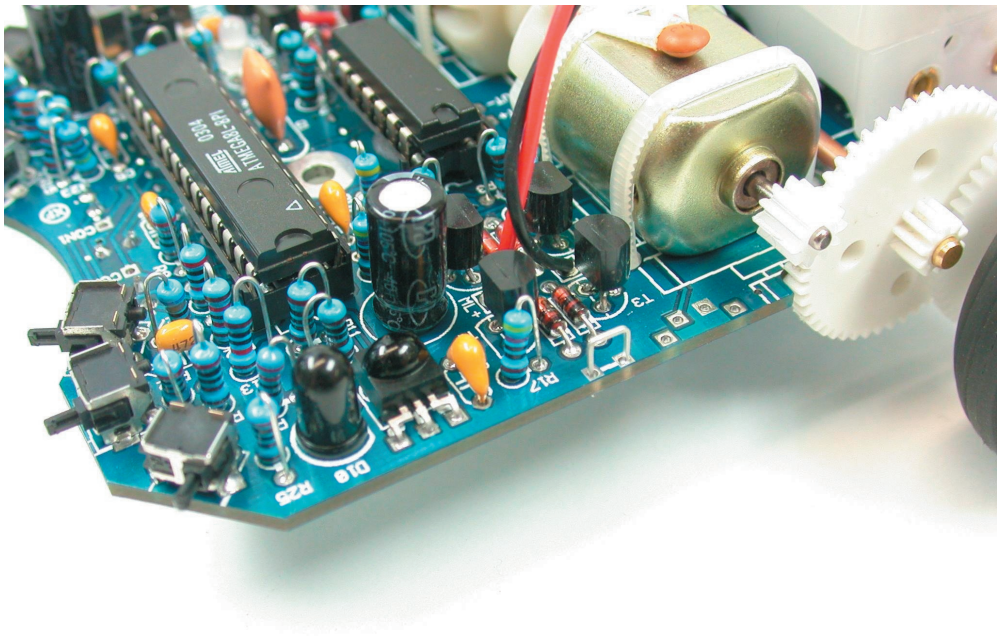
Das zu unserem Beispielprogramm gehörige makefile ist so geschrieben, dass es eine Datei mit Namen test.c zusammen mit asuro.c (das einige vordefinierte  Funktionen enthält) compiliert und eine .hex-Datei erzeugt, die so auf ASURO geladen werden kann.

***Achtung! Das heißt auch, dass – solange das makefile nicht geändert, sondern nur kopiert wird – das das eigene Programm immer test.c heißen muss.***

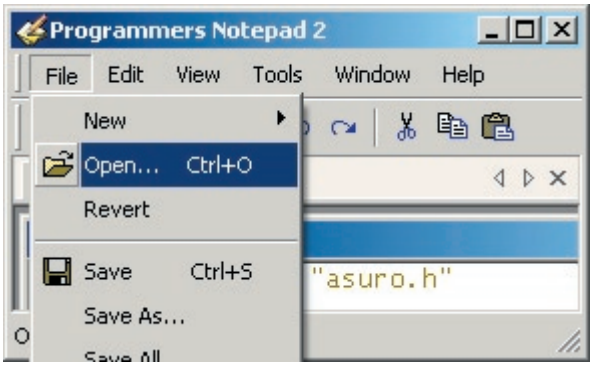
Wer makefiles komplett verstehen will (was aber für erste Schritte nicht erforerlich ist) kann beispielsweise unter http://www.gnu.org/directory/make.html die Dokumentationzu Rate ziehen.

**Die Grundlagen der ASURO-Programmierung werden in Kapitel 9 erklärt.**

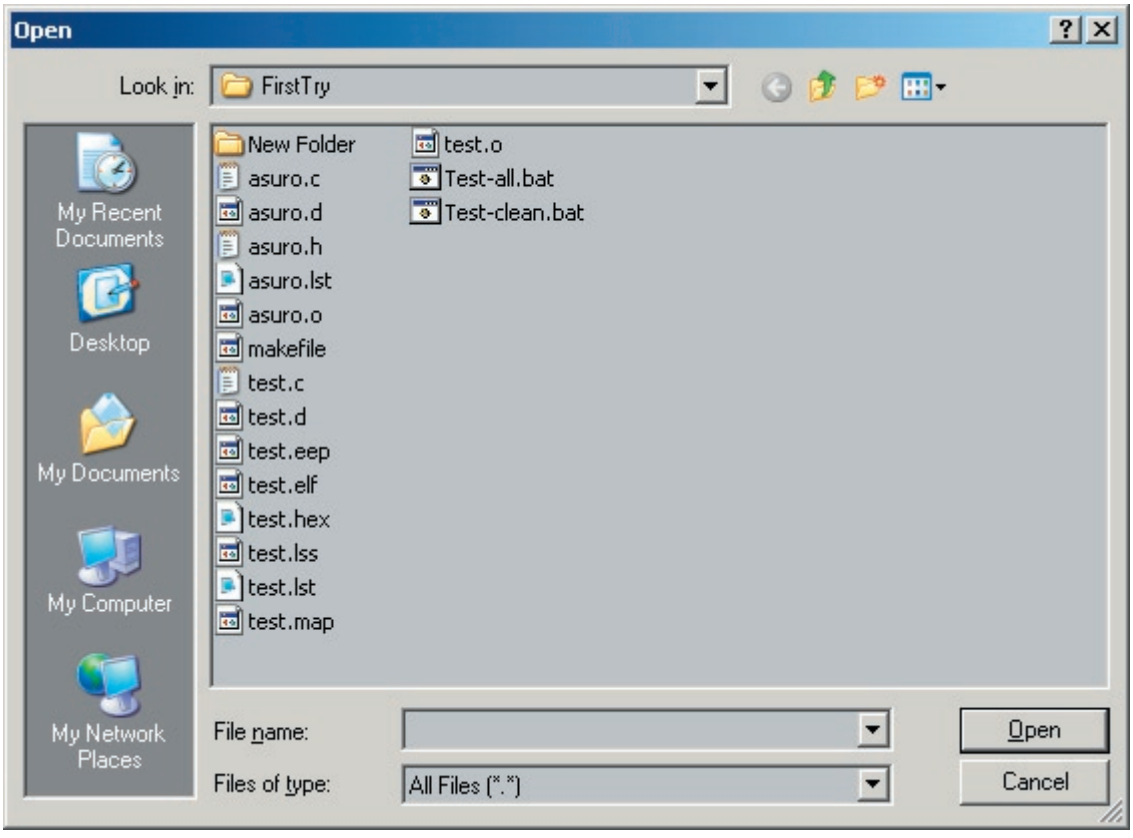
Beim Compilieren eines Programms werden einige “Nebendateien” erzeugt, die nur während der Übersetzung benötigt werden und später überflüssig sind. Diese können mit dem neu eingerichteten ‘clean’-Tool gelöscht werden.



Beim Öffnen…

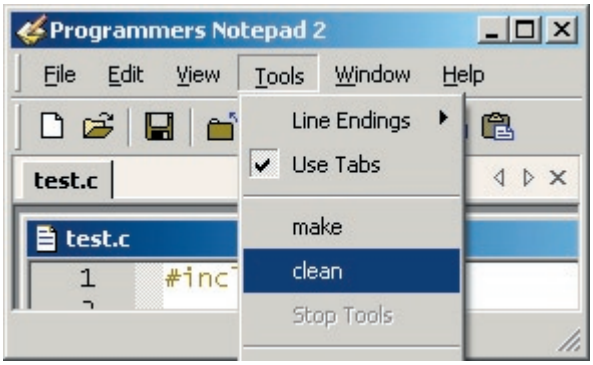



…kann man jetzt die generierten Dateien sehen…

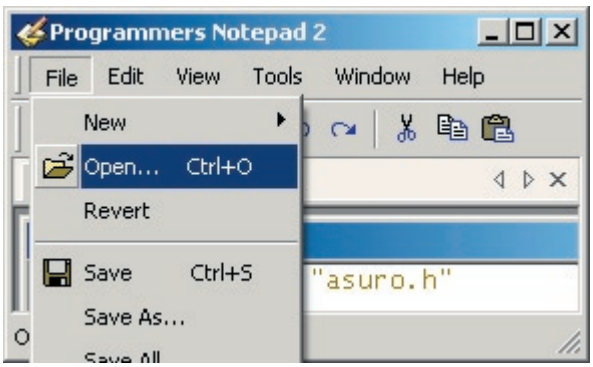



(Klick auf [Cancel])

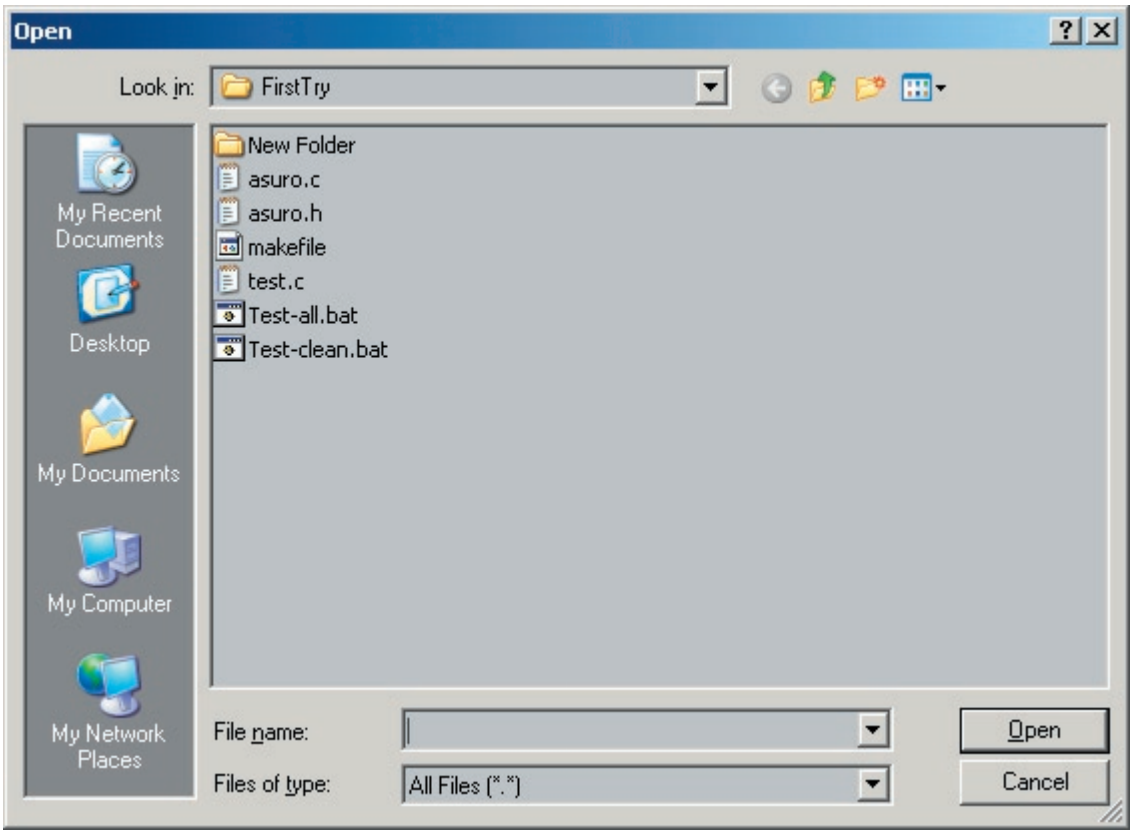
…und nach dem Ausführen von ‘clean’…



…kann man sehen…




…dass die generierten Dateien wieder gelöscht worden sind sind




Wie hat's funktioniert?
Der Menüeintrag 'clean' hat die Batch-Datei Test-clean.bat aufgerufen, welche make mit dem Parameter 'clean' gestartet hat. Dadurch wird der Eintrag im makefile namens clean ausgeführt und die ganzen inzwischen überflüssigen Dateien wieder gelöscht.

## 8.2. LINUX

Für die Installation der Software sind root-Rechte erforderlich. Entweder ausloggen und als root einloggen oder eine shell öffnen und mit “su” root-Rechte erlangen.

### 8.2.1 Flash-Tool

Die ASURO-CDROM einlegen, ggf. mounten und die beiden Flash-Tools “asuroflash” und “asurocon” aus dem Verzeichnis “/Linux/Tools/” in das Verzeichnis “/usr/local/bin” kopieren. Danach noch das Ausführen mit “chmod a+x /usr/local/bin asurocon asuroflash” erlauben.

Wird ein in einer Shell eingetipptes “asuroflash” nicht gefunden, muss der Pfad “/usr/local/bin” noch der %PATH-Variable hinzugefügt oder das Programm mit vollem Pfad aufgerufen werden.

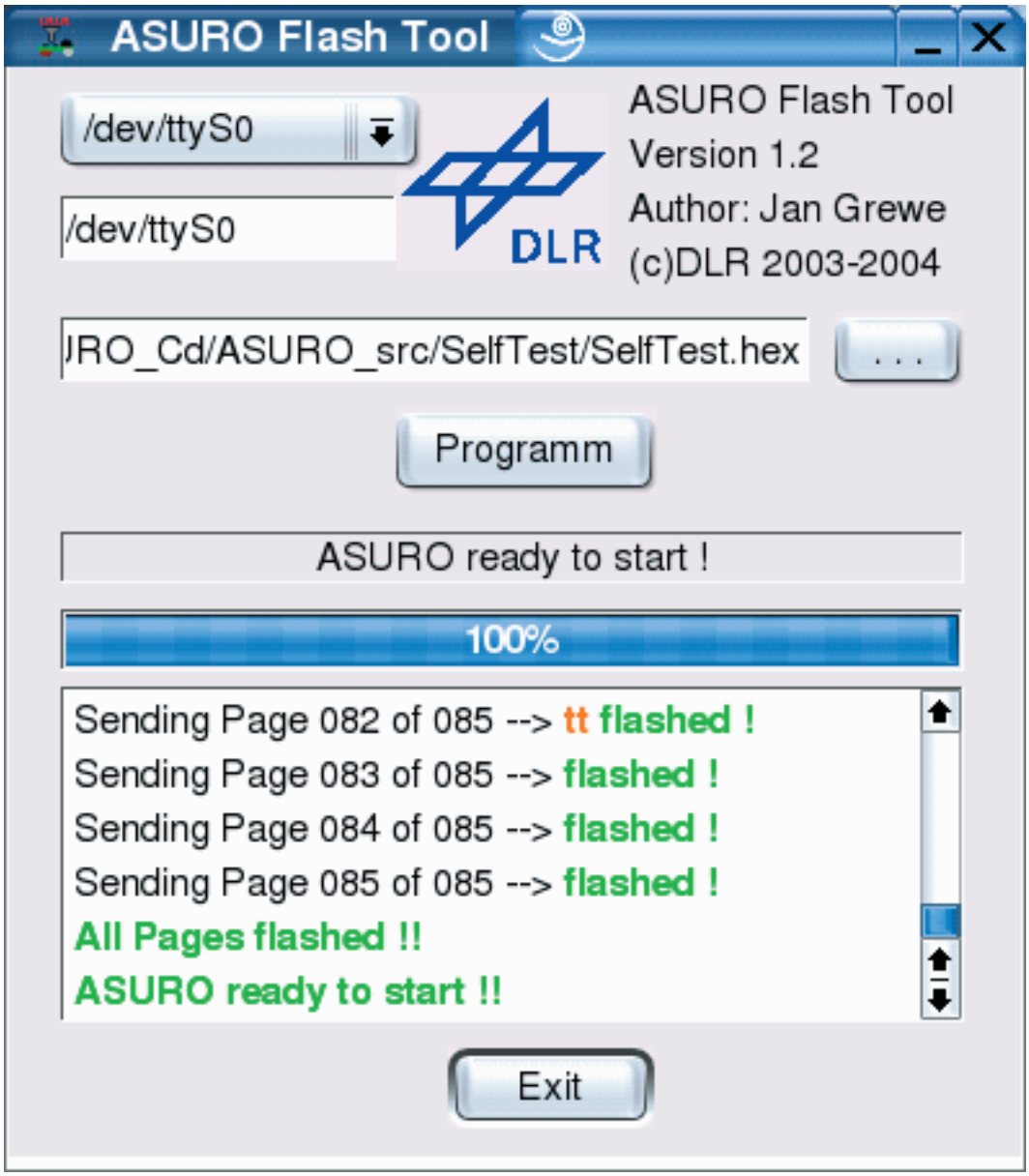



Abbildung 8.1.: Flash tool

### 8.2.2 Compiler

Zur Installation des Gnu-Compilers für AVR-Prozessoren die ASURO-CDROM einlegen und aus dem Verzeichnis “/Linux/Compiler/” mindestens die folgenden Pakete in der angegebenen Reihenfolge installieren:

**1. avr-binutils-... .rpm**
**2. avr-gcc-... .rpm**
**3. avr-libc-... .rpm**

Die Installation ist denkbar einfach!
Einfach in der Konsole mit root-Rechten den Befehl : rpm -i <paket>.rpm  eingeben.

***Fertig!***

Als Editoren eignen sich zum Beispiel Exmacs, Kate oder Kedit. Zum Ausprobieren kopiert man sich (als normaler User) die Beispieldateien von der CD aus dem Verzeichnis “/ASURO_src/FirstTry/” ins home-Verzeichnis beispielsweise unter “~/ASURO/”.
Danach öffnet man eine Shell, wechselt in obiges Verzeichnis und gibt “make” ein. Ist alles richtig installiert, ergibt sich etwa folgendes Bild: (siehe Abb. 8.2)

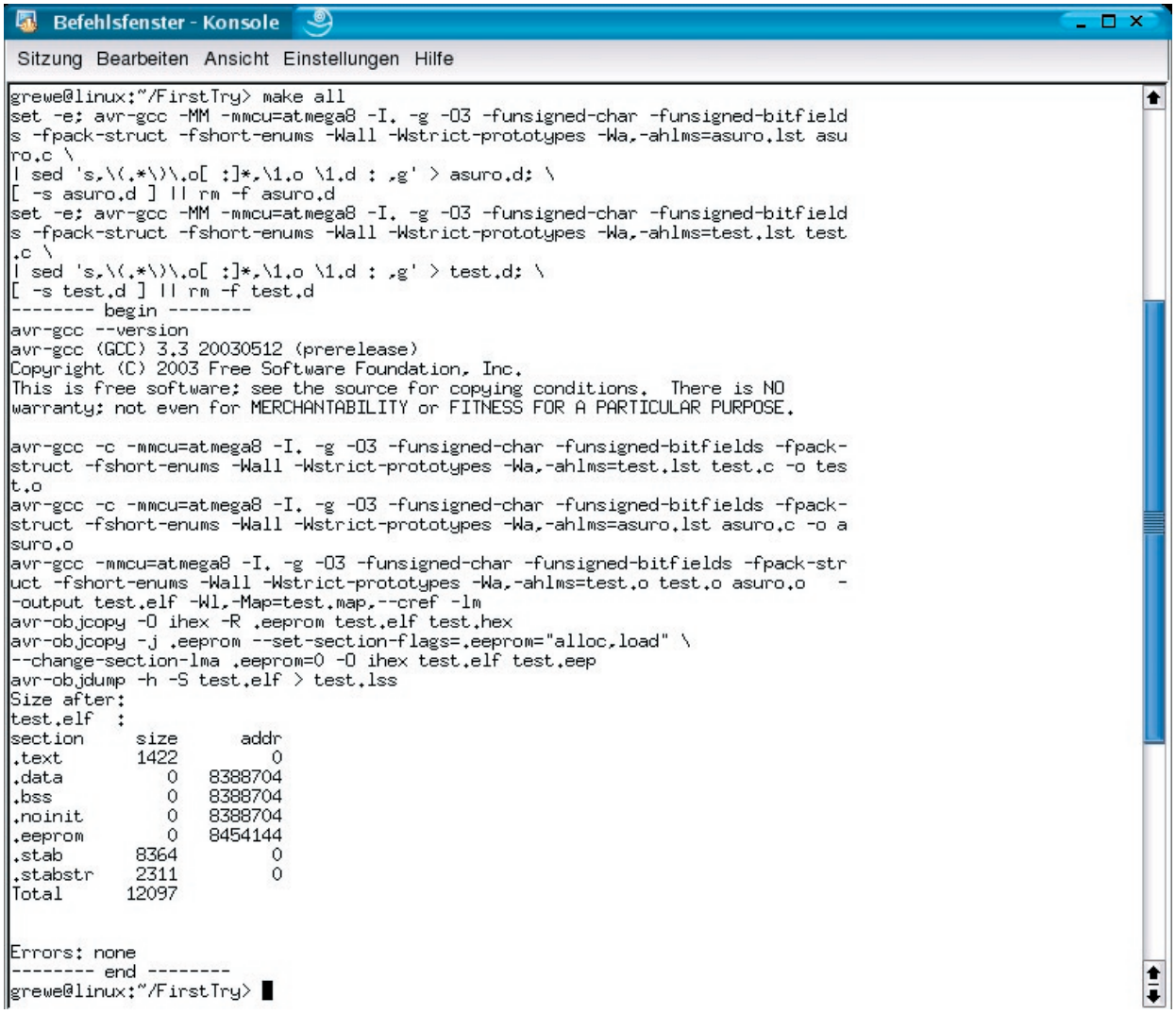



**Abbildung 8.2.: Make all**

## 8.3. Flash - das ASURO-Programmier-Tool

Hierzu wird das Programm Flash (siehe Abb. 8.3) benötigt.

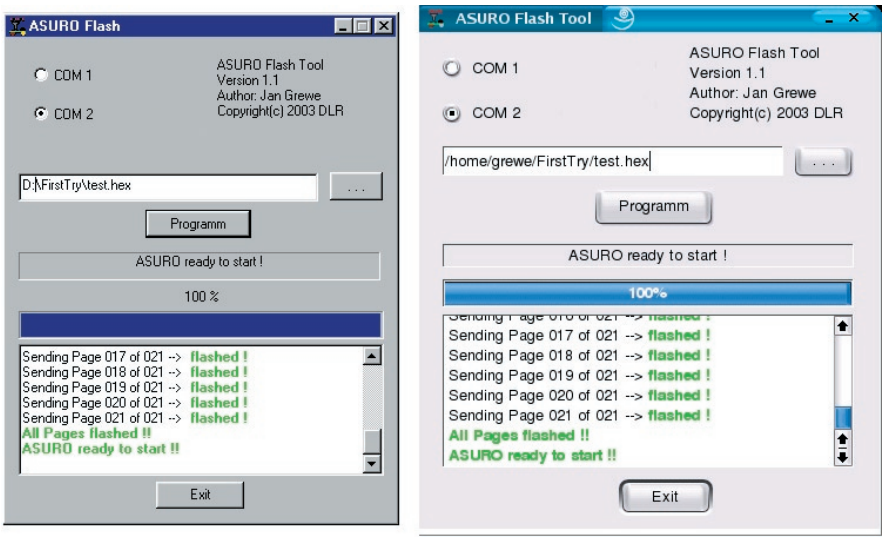




**Abbildung 8.3.: Flash-Tools für Windows und LINUX**

Damit das Flashen auch richtig gut klappt, muss natürlich der RS232- oder USB-IR-Transceiver angeschlossen werden. Danach das Programm starten und die Schnittstelle auswählen, welche auch bei der Inbetriebnahme funktioniert hat.

Die Datei “Test.hex” aus dem Verzeichnis “C:\ASURO_src\FirstTry” (bzw. ~/ASURO/) auswählen.

Den fertig zusammengebauten und getesteten ASURO bereit halten und beim Flash-Tool den Knopf Programm anklicken. ASURO muss Sichtkontakt (ca. 50cm Abstand zwischen IR-Transceiver und ASURO, beide Bestückungsseiten zeigen zueinander und nichts ist im Lichtweg) zum IR-Transceiver haben und wird jetzt eingeschaltet (S1 auf ON), bevor der Statusbalken ganz rechts angekommen ist.

Sollte man mal nicht schnell genug gewesen sein oder sollte die Kontaktaufnahme nicht geklappt haben, einfach ASURO ausschalten, erneut Programm drücken und ASURO einschalten.

Hat der Verbindungsaufbau geklappt, sieht man an der Statusanzeige und dem Protokollfenster, dass nun die Datei Test.hex zu ASURO übertragen wird. Dort wird das Programm im Flash-Speicher des Prozessors abgelegt, sodass das Programm auch nach einem Abschalten der Versorgungsspannung immer wieder zur Verfügung steht.

War der Vorgang erfolgreich, muss ASURO aus- und erneut eingeschaltet werden, um das Programm zu starten. Das vor kurzem geschriebene Programm wird ausgeführt und die grüne LED erstrahlt in hellem Glanz.

### 8.3.1. Wie funktioniert das Flashen?

Sobald das Programm Flash ausgeführt wird, versucht der Rechner 10 Sekunden lang eine Verbindung mit ASURO herzustellen. Schaltet man ASURO ein, leuchtet die Status-LED für ca. 1 Sekunde zweifarbig auf. Dies ist die “Boot-Phase”. ASURO schaut nach, ob der PC neue Software für ihn bereitstellt. Diese wird gegebenenfalls geladen. Nach Aus- und Wiedereinschalten wird diese dann gestartet.

## 8.4. Flash Fehler

Folgende Fehler können während des Flashens auftreten:

- “c” Checksum Error. Es sind andere als die vom PC geschickten Daten bei ASURO angekommen. Das kann durch Störlicht (wie Leuchtstofflampen) kurze Unterbrechungen in der Sichtverbindung oder ähnliches passieren.

- “t” Timeout. Die Sichtverbindung zu ASURO ist abgerissen.

* “v” Verify Error. ASURO hat falsche Daten in seinen Flashspeicher geschrieben. Das sollte normalerweise nicht passieren und ist ein Zeichen dafür, dass der nichtflüchtige Programmspeicher (Flash-EPROM) das Ende seiner Lebenszeit erreicht hat, was typischerweise erst nach 10.000 Programmierungen passiert.

Es wird bis zu zehnmal versucht den Fehler zu beheben. Gelingt dies nicht, wird der Flashvorgang abgebrochen.

***Treten beim Flashen gehäuft Checksum Errors auf, hilft es oftmals das Raumlicht abzuschalten bzw. etwas abzuschatten, besonders wenn es sich um Leuchtstofflampen handelt.***

***Immer erst den Programm-Knopf drücken, dann ASURO einschalten, sonst ist kein Softwaredownload möglich!***

## 8.5. Erstes eigenes Programm

Bevor wir zu einer Kurzfassung der C-Programmierung kommen, erstmal ein kleines eigenes Programm. Dazu laden wir wieder mit dem Programmers Notepad (Linux: oder einem anderen Editor) die Datei test.c aus dem Verzeichnis C:\Eigene Dateien\ASURO_src\FirstTry:

```
#include “asuro.h”
int main(void) {
Init();
while(1);
return 0;
}
```

Für die ersten Versuche ist es erforderlich, dass das Programm immer den Dateinamen test.c trägt, weil das mitgelieferte Beispiel-makefile (eine Datei, die beschreibt, wie ein Programm zu übersetzen ist) darauf ausgelegt ist. Auch ist es wohl am einfachsten, von einem vorgegebenen Beispiel aus weiterzuschreiben. Später können dann eigene Programme erstellt und auch eigene makefiles geschrieben werden.

Das geladene Programm wird wie folgt abgeändert (Achtung, auf exakte Schreibweise auch Groß-/Kleinschreibung achten):

```
#include “asuro.h”
int main(void) {
Init();
StatusLED(RED);
while(1);
return 0;
}
```

Danach wieder im Menü Tools -> make auswählen (unter Linux: in einer Shell im Verzeichnis “~/ASURO/” “make” eintippen oder den Editor passend konfigurieren und die Compilierung abwarten, bis keine neuen Meldungen mehr im Statusfenster erscheinen).

Überprüfen, ob Process Exit Code: 0 unten im Statusfenster steht, damit ist das Programm vom Compiler verstanden und übersetzt worden.

Steht ein anderer Code da, muss anhand der Fehlermeldungen der Fehler gesucht werden. Meistens hilft es, in der Zeile zu suchen anzufangen, wo - laut Statusfenster - der erste Fehler gefunden wurde. Im Editor steht die Zeilennummer, in der der Cursor gerade positioniert ist ganz links unten.

Hat das Compilieren fehlerfrei funktioniert, kann das neue Programm geflasht werden. Dazu wird wieder der IR-Transceiver angeschlossen, das Flash-Tool gestartet, die Datei test.hex und die korrekte COM-Schnittstelle ausgewählt, ASURO in Sichtkontakt zum IR-Transceiver gebracht, Programm angeklickt, ASURO eingeschaltet und die Übertragung des Programms abgewartet.

Hat auch die Übertragung - laut Statusfenster - einwandfrei geklappt, ASURO ausschalten, ASURO einschalten, eine Sekunde Spannung und (*Tusch*) die Statusleuchtdiode leuchtet rot. Um zu vermeiden, dass weiterhin Programmzeilen ohne Programmierkenntnisse geschrieben werden müssen, sei vor weiteren Experimenten das folgende Kapitel empfohlen.

# 9. C für ASURO

Dieser Abschnitt beschäftigt sich mit der Programmiersprache C. Der Leser wird hierbei nur die für die Programmierung von ASURO notwendigen Bestandteile von C erklärt bekommen. Dies stellt also keinesfalls eine komplette Einführung in C dar. Hierfür gibt es bessere Bücher[1].

C wurde als Sprache gewählt, da dieser Standard weit verbreitet ist und für fast jeden Prozessor, zumindest ein C-Compiler existiert. Bei ASURO findet der Gnu-C-Compiler Verwendung, da er ein frei erhältliches Freeware-Programm ist und trotzdem gut optimierten Code für den ATmega8, ASUROs Prozessor erzeugt.

Wer schon C Programmieren kann, der lese einfach bei Abschnitt 9.2 weiter. Der Rest ist für diese Lesergruppe absolut uninterssant. Es werden wirklich nur die allernotwendigsten Sprachelemente erklärt, um auf möglichst einfache Weise das Wissen zu vermitteln, welches für den Betrieb von ASURO unbedingt notwendig ist.

Und keine Sorge, wenn man brav an seine Klammern und Strichpunkte denkt, ist C gar nicht so schwer. Na und schließlich ist das auch kein Spielzeugroboter für den Kindergarten!

## 9.1. Grundlagen der C-Programmierung

### 9.1.1. Allgemeines

Prinzipiell wird ein C-Programm Anweisung nach Anweisung von oben nach unten vom Prozessor abgearbeitet[2]. Gleichzeitige Ausführung von zwei Befehlen gibt es nicht, zumindest nicht bei ASUROs Prozessor. Dementsprechend muss man denken: Ein Arbeitsauftrag nach dem anderen.

Die Leerzeichen am Anfang der Zeilen in den Beispielen sind nicht unbedingt erforderlich. Die Methode des Einrückens ist aber sehr hilfreich, wenn es darum geht, auch längere Programme überschaubar zu halten.

Jede Anweisung wird in C mit einem “;” abgeschlossen. Damit kann der Compiler die einzelnen Anweisungen voneinander unterscheiden.

Sollen mehrere Anweisungen zusammengefasst werden, wie es für Funktionen, Schleifen oder Bedingungen (dazu später) erforderlich ist, so wird dieser Anweisungsblock mit geschweiften Klammern (“{”, “}”) eingeklammert.

**Beispiel:**

```
#include “asuro.h”
int main (void) {
/* Alles, was hier steht, gehört in einen Block */
}
```

[1] Beispielsweise: Brian W. Kerninghan, Dennis M. Ritchie: “Programmieren in C”, Hanser Verlag, ISBN 3-446-15497-3

[2] Methoden, welche in den sequentiellen Ablauf von Befehlen eingreifen, werden als Flusssteuerung bezeichnet und später im Kapitel erklärt.

Will man einige Zeilen aus seinem Code auskommentieren so beginnt der Kommentarblock mit “/*” und endet mit “*/ ”. Um nur eine Zeile auzukommentieren genügt ein “//“ vor der betreffenden Zeile[3]. Das Auskommentieren dient dazu, dass Textstücke vom Compiler nicht beachtet werden. Damit kann man Kommentare in das Programm einfügen, ohne dass beim Übersetzten des Programms dadurch Probleme entstehen.

## 9.1.2. Variablen und Datentypen

Variablen sind “Behälter” für Daten. Im Laufe eines Programms können diese beschrieben, ausgelesen oder geändert werden. Um eine Variable nutzen zu können, muss sie zunächst deklariert werden. Hierbei wird festgelegt welchen Typ diese Variable besitzen und evtl. auch welchen Anfangswert sie erhalten soll. Der Typ legt fest, welche Art von Zahlen man in der Variable speichern kann (ganze Zahlen, positive ganze Zahlen, Dezimalbrüche...).

Der Name einer Variablen muss dabei mit einem Buchstaben beginnen (“_” zählt ebenfalls als Buchstabe) und darf auch Zahlen, aber keine Sonderzeichen enthalten. Groß- und Kleinschreibung werden unterschieden; somit sind x und X verschiedene Variablen. Traditionellerweise werden Kleinbuchstaben für Variablennamen verwendet. Folgende Bezeichnungen sind bereits reserviert und können nicht als Variablennamen verwendet werden :

| | | | | | |
|---|---|---|---|---|---|
| ***auto*** | ***default*** | ***float*** | ***long*** | ***sizeof*** | ***union*** |
| ***break*** | ***do*** | ***for*** | ***register*** | ***static*** | ***unsigned*** |
| ***case*** | ***double*** | ***goto*** | ***return*** | ***struct*** | ***void*** |
| ***char*** | ***else*** | ***if*** | ***short*** | ***switch*** | ***volatile*** |
| ***const*** | ***enum*** | ***int*** | ***signed*** | ***typedef*** | ***while*** |
| ***continue*** | ***extern*** | | | | |

Folgende Datentypen sind für das Programmieren von ASURO von Interesse :

| **Typ** | **Wertebereich** | **Bemerkung** |
|---|---|---|
| | | |
| char | -128 ... +127 | ein Byte-Wert; kann ein Zeichen aus dem Zeichensatz aufnehmen |
| unsigned char | 0 ... 255 | vorzeichenloser char |
| int | -32768 .. +32767 | zwei Byte Wert |
| unsigned int | 0 ... 65535 | vorzeichenloser int |
| float | | einfach genauer Gleitpunktwert |

3 “//” ist ein Kommentarzeichen nach C++ Standard. Da der hier verwendete Compiler eigentlich ein C++-Compiler ist,funktioniert dies, kann aber bei anderen Compilern zu Fehlermeldungen führen.

Die Deklaration erfolgt entweder “außerhalb” der main()-Funktion als globale Variable, (das heißt, die Variable ist für das gesamte Programm verfügbar), innerhalb der main()-Funktion, (dann gilt sie nur für Programmcode, der in der main()-Funktion steht) oder innerhalb einer eigenen Funktion (dann ist sie nur hier gültig).

Was nützt die schönste Variable, wenn man nicht weiß, wie man Daten rein oder raus kriegt. Daten rein klappt mit einer Zuweisung:

```
a=17; // a hat jetzt den Wert 17
```

oder auch als Rechnung:

```
a=17+23; // a ist jetzt 40
b=a+3; // b ist jetzt 43
b=b*2; // b ist jetzt 86
```

Und jetzt im ganzen Programm:

```
#include “asuro.h”
int main(void) {
	int i; // i kann Zahlen zwischen -32768 und 32767 aufnehmen
	char zeichen; // zeichen kann ASCII-Zeichen oder Zahlen zwischen -128 und 127
				// aufnehmen
	i=3;
	zeichen=17+i; // zeichen ist jetzt 20
	i=i/2; // Division durch 2, es wird immer abgerundet i ist demnach 1!
	return 0;
}
```

Ein paar praktische Abkürzungen gibt’s noch. Für

```
i=i+1;
```

kann man auch schreiben als:

```
i++;
```

Und

```
i=i-1;
```

entspricht:

```
i--;
```

## 9.1.3. Compilerdirektiven

Sicher hat das #include “asuro.h” schon für Verwirrung gesorgt. Die #include-Direktive heißt nichts anderes, als dass Text, welcher in der angegebenen Datei steht in das Programm eingebunden und beim Compilieren mitübersetzt wird. Im vorliegenden Fall werden einige Routinen, welche für den Betrieb des Roboters erforderlich sind, verfügbar gemacht.

Eine weitere wichtige Direktive (und es gibt noch einige andere, welche den Rahmen dieses Werkes sprengen würden) ist der so genannte Textersatz. Dieser hat die Form

```
#define NAME ersatztext
```

und wird vorwiegend zur Definition von Konstanten verwendet.

Tritt das Symbol NAME im Code auf, wird dies automatisch durch ersatztext ersetzt. Der NAME bei #define hat dabei die gleiche Form wie Variablennamen. Es hat sich bei C-Programmieren durchgesetzt, dass bei #define für NAME nur Großbuchstaben verwendet werden.

***Beispiel:***

```
#include “asuro.h”
#define STARTWERT 33
int main(void) {
        int i;
        i=STARTWERT; // i ist jetzt 33
        return 0;
}
```

Hinter Compiler-Direktiven steht übrigens kein Strichpunkt!

## 9.1.4. Bedingungen

Oftmals ist es erforderlich Anweisungen nur dann ausführen zu lassen, wenn bestimmte Vorraussetzungen erfüllt sind. Hierfür benötigt man Kontrollstrukturen. Die einfachste, mit der Entscheidungen formuliert werden können, ist die “if-else” -Anweisung.

Formal gilt folgende Syntax:

```
if (Bedingung)
        Anweisungsblock 1
else
        Anweisungsblock 2
```

Die Bedingung wird auf ihren Warheitsgehalt hin überprüft. Ist diese wahr (also ungleich 0), so wird der Anweisungsblock 1 ausgeführt, ansonsten der optionale Anweisungsblock 2.

Will man eine Entscheidung unter mehreren Alternativen treffen, so kann man mehrere “else if”-Anweisungen verwenden.

```
if (Bedingung 1)
        Anweisungsblock 1
else if (Bedingung 2)
        Anweisungsblock 2
else if (Bedingung 3)
        Anweisungsblock 3
else if (Bedingung 4)
        Anweisungsblock 4
else
        Anweisungsblock 5
```

Folgende Bedingungen sind möglich:

| Operator | Bedeutung |
|---|---|
| == | logischer Vergleich auf **gleich** |
| != | logischer Vergleich auf **nicht gleich** |
| < | logischer Vergleich auf **kleiner** |
| > | logischer Vergleich auf **größer** |
| <= | logischer Vergleich auf **kleiner gleich** |
| >= | logischer Vergleich auf **größer gleich** |

***Beispiel:***

```
#include “asuro.h”
int main(void) {
        Init ()
        while (1) {
        if (PollSwitch()>0) {StatusLED (RED);}
        else {StatusLED (GREEN);}
        }
}
```

Wenn einer der Kollisionstaster gedrückt wird, leuchtet die Statusleuchtdiode rot, ansonsten grün. Die restlichen verwendeten Elemente werden später erklärt.

In C steht “1” für wahr und “0” für falsch. Die Bedingung

```
if (0) {StatusLED(RED);}
```

führt also dazu, dass StatusLED (RED) nie ausgeführt wird.

## 9.1.5. Schleifen

Schleifen dienen dazu, Anweisungen mehrmals auszuführen.
In der “while”-Schleife wird eine Bedingung ausgewertet. Ist die Bedingung wahr, so wird der Anweisungsblock ausgeführt und die Bedingung erneut geprüft, bis diese falsch wird. Dannach wird das Programm hinter dem Anweisungsblock fortgesetzt.

```
while( Bedingung)
  Anweisungsblock
```

***Beispiel:***

```
#include “asuro.h”
int main(void) {
        Init ()
        MotorDir(FWD,FWD);              // Beide Motoren auf vorwärts
        MotorSpeed(120,120);            // Beide Motoren etwa halbe Kraft voraus
        StatusLED(GREEN);               // Status-Leuchtdiode auf grün schalten
        while (PollSwitch()==0) {       // Solange keine Kollision erfolgte...
                SerWrite(“Alles OK!\n”,10); // ... Euphorie verbreiten
        }
        MotorSpeed(0,0);                // Kollision! Sofort anhalten!
        StatusLED(RED);                 // Statusleuchtdiode rot schalten
        while (1) {
                SerWrite(“Aua!\n”,5);   // und weinen!
        }
}
```

Die “for (expr1, epr2, expr3)”-Anweisung ist äquivalent zu :

```
expr1;
while (expr2) {
        Anweisungsblock
        expr3;
}
```

Die “for”-Schleife wird normalerweise als Zählschleife verwendet.

```
for (i = 0; i < n; i++)
. . .
```

***Beispiel:***

```
#include “asuro.h”
int main(void) {
        Init ()
        int zaehler;                                    // Variable für’s Zählen deklarieren
        for (zaehler=0; zaehler<10; zaehler++) {        // zehnmal wiederholen:
                SerWrite(“Los geht’s!\n”,12);           // “Los geht’s”  schicken
        }
        MotorDir(FWD,FWD);              // Beide Motoren auf vorwärts
        MotorSpeed(120,120);            // Beide Motoren etwa  halbe Kraft voraus
        while (1) {                     // Danach nichts mehr machen!
        }       }
```

“while(1)” äquivalent zu “for(;;)” ist eine Endlosschleife, die niemals verlassen wird, da die Bedingung für den Abbruch niemals falsch (also 0) wird.

Als weiteres Schleifekonstrukt gibt es die “do”-Schleife

```
do
  Anweisungsblock
while( Bedingung);
```

Im Gegensatz zur “while”-Schleife wird hier die Bedingung am Ende des Anweisungsblockes auf Ihren Wahrheitgehalt hin untersucht. Diese Schleife wird auf alle Fälle wenigstens einmal durchlaufen.

### 9.1.6. Funktionen

Funktionsdefinitionen haben immer die folgende Form:

```
Funktionstyp FunktionsName (ParameterTyp 1 ParameterName 1,
ParameterTyp 2 ParameterName 2, ...)
```

Toll, Funktionsdefinitionen! Und wozu das Ganze??? Ist sehr praktisch, aber etwas komplizierter und kann man auch ein wenig später lesen...

Häufig kommt es vor, dass Programmteile an verschiedenen Stellen im Programm immer wieder benötigt werden. Dann kann man sie entweder jedes Mal wieder schreiben (super lästig und total unübersichtlich) oder einmal eine Funktion deklarieren.

Oft möchte man einer Fuktion auch einen oder mehrere Werte übergeben. Beispielsweise macht eine (selbstgeschriebene) FahreEinStueckVorwaerts()-Funktion einfach mehr Spaß, wenn man ihr die Geschwindigkeit, die Dauer oder die Strecke sagen könnte. Das wird mit den Parametern gemacht.

Gelegentlich kommt es auch vor, dass eine Funktion einen Wert zurückliefert. Leicht nachzuvollziehen an einer *WievieleTasterSindGedrueckt()* -Funktion. Das geschieht über den Rückgabewert der Funktion, welcher irgendwie und irgendwo innerhalb der Funktion erzeugt und mit der *return*-Anweisung zurückgegeben wird. Daher endet jede Funktion auch mit *return; oder return ZAHL;.*

Eine besondere Funktion stellt die main *()*-Funktion dar. Diese ist der Einsprungpunkt in ein Programm. Bei ASURO wird diese Funktion nach dem Einschalten ausgeführt. Die main *()*-Funktion **muss** in jedem Programm vorhanden sein!

Nachdem die Datentypen bekannt sind und ein klein wenig über Funktionen geredet wurde, versuchen wir uns an einer kleinen Beispielfunktion, welche zwei 8-Bit-Zahlen multiplizieren und das Ergebnis zurückgeben soll.

```
int Mult(char a, char b)
/* Funktion liefert einen int-Wert zurück, hat den Namen Mult, und bekommt zwei char als
   Parameter übergeben */
{                       // Beginn der Funktion
        int c;          // Variable c wird als int deklariert
        c = a * b;      // berechne c
        return c;       // gib c zurück
}                       // Ende der Funktion
```

Nun noch eine kleine Routine, welche die eben definierte Funktion ausführt :

```
int main (void)                 // Funktion main liefert immer einen int zurück,
                                // und bekommt keine Parameter übergeben
{                               // Begin der Funktion
        char mult1,mult2;       // Definiton zweier char-Variablen
        int erg;                // Definition einer int-Variable, die das Ergebnis aus der
                                // Multiplikation der Variablen mult1 und mult2
                                // enthalten soll
        mult1 = 2;              // Zuweisung
        mult2 = 10;             // Zuweisung
        erg = Mult(mult1,mult2); // Aufruf der vorher definierten Funktion Mult
        return 0;
}                               // Ende
```

## 9.1.7. Zeiger und Vektoren

Zeiger und Vektoren werden hier nur soweit behandelt, wie diese zum Betreiben von ASURO notwendig sind.

Werden die Linienfolgesensoren, bzw. die Sensoren der Odometrie ausgelesen, benötigt man Vektoren. Deren Deklaration ist denkbar einfach:

```
int lData[2];
int oData[2];
```

Wie man erkennt werden für die Liniensensoren bzw. für die Odometrie zwei Vektoren (l Data, o Data) mit 2 Elemeneten angelegt. In Element [0] steht, nach dem Aufruf der passenden ASURO Funktion (LineData(), OdometrieData ()), der Wert des linken Sensors, in Element [1] der Wert des rechten Sensors.

***Hierzu ein kleines Beispiel :***
Ist von den beiden Liniensensoren der rechte heller als der linke beleuchtet, soll Anweisung 1 ausgeführt werden, ansonsten Anweisung 2.

```
int lData[2];                   // Speicher für die Messwerte zur Verfügung stellen
LineData(lData);                // Einlesen der Messwerte
if (lData[1] > lData[0])
        Anweisung1;
else
        Anweisung2;
```

Um die seriellen Schnittstellenfunktionen (SerWrite(), SerRead() ) benutzen zu können, werden Zeichenketten benötigt. Diese werden wie folgt deklariert:

```
char message [] = “Hier steht ein Text”;
```

Um eine Zeichenkette zu senden ist bei ASURO nur die Funktion SerWrite() mit den entsprechenden Parametern aufzurufen. Der erste Parameter gibt den Text bzw. die Zeichenkettenvariable an, der zweite Parameter gibt an, wieviele Zeichen der Zeichenkette übertragen werden sollen.

```
SerWrite(message,20);
```

bzw.

```
SerWrite(“Hier steht ein Text”,20);
```

senden über die IR-Schnittstelle “Hier steht ein Text”.

Will man Zeichen empfangen, ist für ASURO die Funktion *SerRead* () definiert. Der erste Parameter enthält die Zeichenkettenvariable, in welcher empfangen Zeichen abgespeichert werden, der zweite Parameter gibt an, wieviele Zeichen empfangen werden sollen, der dritte stellt ein Timeout dar. Werden innerhalb der eingestellten Zeit (Prozessortakte) keine Daten empfangen, bricht die Funktion ab. Wird hier “0” eingestellt, wartete die Funktion bis alle Zeichen empfangen wurden.

***Auch hierzu ein kleines Beispiel:***
ASURO soll “Hallo hier bin ich” über die IR-Schnittstelle empfangen:

```
char message [] = ”01234567890123456789”;
```

Platz für den zu empfangenden Text ist geschaffen worden. Die eben bereitgestellte Zeichenkette muss groß genug sein, den zu empfangenden Text aufzunehmen.

```
SerRead(message,18,0);
```

Lies 18 Zeichen ein und warte dabei solange, bis alle 18 Zeichen eingetroffen sind. Wir gehen jetzt mal davon aus, dass die Zeichenkette “Hallo hier bin ich” gesendet wird. Die zuvor definierte Zeichenkette message sieht jetzt wie folgt aus:

```
Hallo hier bin ich89
```

Die ersten 18 Zeichen von message wurden mit den empfangenen Zeichen überschrieben.

## 9.2. Beschreibung der ASURO-Funktionen

Um die Programmierung von ASURO möglichst einfach zu gestalten, gibt es einige vorgefertigte Funktionen. Diese stellen nicht zwingend das Optimum dar, für einige Anwendungen ist es sicher besser, eigene Funktionen zu schreiben.

Die Funktionen sind klassisch im Stil ihrer Deklaration dargestellt. Wer damit nichts anfangen kann, schaut sich am besten jeweils die Beispiele an.

Um Missverständnissen vorzubeugen: Funktionen, die etwas steuern, wie die Antriebs-Funktionen oder die Funktionen für die Anzeigenelemente legen Einstellungen fest, die solange gültig sind, bis sie wieder geändert werden. Also eine grüne Status-Leuchtdiode bleibt solange grün, bis sie auf eine andere Farbe gesetzt oder ausgeschaltet wird.

## 9.2.1. void Init(void)

Der Mikrocontroller wird in seinen Grundzustand gebracht. Diese Funktion muss immer am Anfang eines Programms aufgerufen werden. Ohne den Aufruf dieser Funktion am Anfang weiß der Prozessor nicht mal, was er mit seinen Beinchen machen soll.

***Ein Programm für ASURO muss mindestens so ausschauen:***

```
#include “asuro.h”
int main(void) {
                        // hier werden die benötigten Variablen deklariert
        Init();
                        // hier stehen dann die eigenen Programmideen
        while(1);       // Endlosschleife
        return 0;       // wird nicht mehr ausgeführt
}
```

Warum die Endlosschleife am Ende der *main* () -Funktion? Normalerweise bedeutet ein Beenden der *main* () -Funktion mit *return 0*; ein Ende des Programms. Bei ASURO kann aber passieren, dass dann Teile früher geflashter Programme ausgeführt werden oder das Programm neu startet, was zu seltsamen Effekten führt. Um das zu vermeiden, wird das Programm - nachdem es seine Aufgaben abgearbeitet hat - in einer Endlosschleife “gefangen”, was ein definiertes Ende des Programms darstellt.

## 9.2.2. void StatusLED(unsigned char color)

Die Status-LED (D12) kann zum Aufleuchten gebracht werden. Mögliche Übergabeparameter sind OFF, GREEN, RED oder YELLOW

***Beispiel:***
Die Status-LED soll rot leuchten:

```
StatusLED(RED);
```

Okay, okay, nochmal im ganzen Programm:

```
#include “asuro.h”
        int main(void) {
        Init();
        StatusLED(YELLOW);
        while(1); // Endlosschleife
        return 0;
}
```

### 9.2.3. void FrontLED(unsigned char status)

Die Front-LED (D11) kann ein- bzw. ausgeschaltet werden. Mögliche Übergabeparameter sind ON bzw. OFF.

***Beispiel:***
Die Front-LED soll leuchten:

```
FrontLED(ON);
```

### 9.2.4. void BackLED(unsigned char left, unsigned char right)

Die Back-LEDs (D15 und D16) können ein- bzw. ausgeschaltet werden. Der erste Parameter beschreibt den Zustand der linken Back-LED (D15), der zweite Parameter den der rechten Back-LED (D16). Mögliche Zustände sind ON bzw. OFF.

***Beispiel:***
Die rechte BackLED(D16) soll ein- und die linke (D15) ausgeschaltet werden:

```
BackLED(OFF,ON);
```

### 9.2.5. void Sleep(unsigned char time72kHz)

Diese Funktion lässt den Prozessor für eine einstellbare Zeit warten. Damit kann man prima Verzögerungen programmieren. Diese Funktion basiert auf einem 72kHz-Timer und kann als Parameter maximal den Wert 255 übergeben bekommen (unsigned char)[4].

***Beispiel:***
Der Prozessor soll für ca. 3ms warten ==> $\frac{0{,}003s}{\frac{1}{72KHz}} = 216$. Die Funktion Sleep () wird für 3ms langes Warten wie folgt aufgerufen:

```
Sleep (216) ;
```

### 9.2.6. void MotorDir(unsigned char left_dir, unsigned char right_dir)

Mit dieser Funktion wird die Drehrichtung der beiden Motoren festgelegt. Sie sollte vor der Geschwindigkeitseinstellung aufgerufen werden. Mögliche Parameter sind FWD (Vorwärtsrichtung), RWD (Rückwärtsrichtung), BREAK (Bremsen bzw. Stehenbleiben, hierbei werden die Motoren über die Transistorbrücken kurzgeschlossen) und FREE (Freilauf).

***Beispiel:***
Der linke Motor soll sich vorwärts drehen, während der rechte Motor stillstehen soll.

```
MotorDir(FWD,BREAK);
```

[4] Das ist böswille Absicht der Autoren und soll zum Nachdenken zwingen!

## 9.2.7. void MotorSpeed(unsigned char left_speed, unsigned char right_speed)

Hier wird die Geschwindigkeit der Antriebsmotoren vorgegeben. Der maximal mögliche Geschwindigkeitswert ist 255 (unsigned char). Der Motor fängt sich erst ab einem Wert von ca. 60 zu drehen an. (Hängt stark vom mechanischen Zusammenbau ab.) Der eingestellte Wert gibt eigentlich nur an, welche elektrische Leistung die Motoren erhalten sollen. Welche tatsächliche Drehzahl resultiert, hängt auch noch von anderen Faktoren wie der Reibung oder der Steigung ab.

***Sobald diese Funktion benutzt wird, kann ASURO losfahren. Manchmal ist das Ergebnis der Programmierung nicht das beabsichtigte, sodass dafür gesorgt werden muss, das ASURO durch plötzliche Fahrmanöver weder sich noch andere in Gefahr bringen kann.***

**Beispiel:**
Der linke Motor soll sich mit maximaler Geschwindigkeit drehen, der rechte Motor gar nicht. Die Drehrichtung ist mit der Funktion MotorDir() schon vorgegeben.

```
MotorSpeed (255,0) ;
```

## 9.2.8. void SerWrite(unsigned char *data, unsigned char length)

Mit dieser Funktion werden Daten von ASURO über die serielle IR-Schnittstelle mit 2400Bit/s, No-Parity, 1 StopBit, NoFlowControl ausgegeben. Das ist die gleiche Einstellung, wie sie auch beim Testen des IR-Transceivers benutzt wird (wer hätte das gedacht). Im ersten Parameter wird die Adresse auf die zu sendenden Daten übergeben. Der zweite Parameter gibt an, wieviele Bytes übertragen werden sollen.

***Beispiel:***
Die Zeichenkette „Hallo Du Da !“ soll über die serielle IR-Schnittstelle gesendet werden.

```
SerWrite(“Hallo Du Da!”,12);
```

## 9.2.9. void SerRead(unsigned char *data, unsigned char length, unsigned int timeout)

Wenn man schon Daten über die serielle IR-Schnittstelle senden kann, so möchte man vielleicht auch mal welche empfangen. Dazu gibt es diese Funktion. Der erste Parameter ist der Zeiger auf die Speicherstelle, in der die empfangenen Daten abgelegt werden sollen. Der zweite Parameter gibt an, wieviele Datenbytes erwartet werden. Der dritte und letzte Parameter stellt ein Timeout dar. Hiermit kann dafür gesorgt werden, dass die Funktion nicht unendlich lange auf Daten wartet. Wird nach einer gewissen Zeitspanne kein Zeichen mehr empfangen, wird diese Funktion einfach abgebrochen. Das allererste Zeichen in den empfangen Daten wird dabei mit ’T’ (Timeout) überschrieben. Trägt man als dritten Parameter ’0’ ein, wartet die Funktion solange, bis die Anzahl der im zweiten Parameter eingestellten Bytes empfangen wurde.

***Beispiel:***
Die Zeichenkette „Fahr los“ soll empfangen werden. Dabei soll sichergestellt sein, dass alle Zeichen bei ASURO eingetroffen sind, bevor weitergemacht werden kann. ***Anmerkung: Es werden nur acht Zeichen empfangen. Eine Überprüfung ob auch wirklich ‘Fahr los’ angekommen ist, findet nicht statt.***

```
#include “asuro.h”
int main(void) {
        char daten[8]; //Speicher bereitstellen
        Init();
        SerRead(daten,8,0); // Daten einlesen
        MotorDir(FWD,FWD);
        MotorSpeed(120,120);
        while(1); // Endlosschleife
        return 0;
}
.
.
```

## 9.2.10. void LineData(unsigned int *data)

Hiermit können die Fototransistoren auf der Unterseite von ASURO ausgelesen werden. Die Adresse auf einen Speicherbereich, der zwei Integerwerte aufnehmen kann muss übergeben werden. Diese Funktion füllt dann den Inhalt der Adresse mit den A/D-Wandler-Werten der beiden Fototransistoren. Der erste Integerwert enthält den Wandler-Wert des linken (T9), der zweite Integerwert den des rechten Fototransistors (T10). Maximale Helligkeit entspricht einem Wert von ’1023’ dunkel entspricht einem Wert von ’0’. Die beiden Extremwerte werden normalerweise nicht erreicht, der Messwert bewegt sich irgendwo dazwischen.

***Beispiel:***
Auslesen der Fototransistoren (T9, T10)

```
unsigned int data[2];          //Speicher bereitstellen
.
.
LineData(data);
```

data[0] enthält den Wert des linken Fototransistors (T9).

data[1] enthält den Wert des rechten Fototransistors (T10).

Jaja, diesmal wieder im ganzen Programm:

```
#include “asuro.h”                    // Linienverfolgung auf die einfachste Art
int main(void) {
        unsigned int data[2];          //Speicher bereitstellen
        Init();
        FrontLED(ON);                  // Linienbeleuchtung einschalten
        MotorDir(FWD,FWD);             // Beide Motoren auf vorwärts
        while(1){                      // Endlosschleife, ASURO soll beliebig
                                       // lang einer Linie nachfahren
                LineData(data);        // aktuelle Helligkeitswerte der
                                       // Fototransistoren einlesen
                if (data [0] > data [1] )   // links heller als rechts...
                {MotorSpeed(200,150);}      // ... dann links mehr Gas geben...
        else
                {MotorSpeed(150,200);}      // ... sonst rechts mehr Gas geben!
        }
        return 0;
}
```

## 9.2.11. void OdometrieData(unsigned int *data)

Die Reflexlichtschranke wird ausgewertet. Die Leuchtdioden (D13, D14) werden aktiviert und die A/D-Wandler-Werte der Fototransistoren (T11, T12) zurückgegeben. Wie in der Funktion LineData () muss ein Speicherbereich mit zwei Integerwerten übergeben werden, der dann von der Funktion gefüllt wird. Der erste Integerwert enthält den Wandler-Wert des linken (T11), der zweite Integerwert den des rechten Fototransistors (T12). Maximale Helligkeit entspricht einem Wert von ’0’ dunkel entspricht einem Wert von ’1023’ [5]. Die beiden Extremwerte werden normalerweise nicht erreicht, der Messwert bewegt sich irgendwo dazwischen.

***Beispiel:***
Auslesen der Reflexlichtschranke

```
unsigned int data[2];          //Speicher bereitstellen
.
.
OdometrieData(data);
```

data[0] enthält den Wert vom linken Fototransistor (T11)

data[1] enthält den Wert vom rechten Fototransistor (T12)

Um Missverständnissen vorzubeugen: OdomertieData() liest nicht die Drehzahl direkt aus, sondern nur die aktuelle Helligkeit der Geberscheibe an der Lichtschranke. Eine Auswertung der Helligkeitswerte, ein Zählen der Hell-Dunkel-Übergänge und die Bestimmung der Drehzahl des Rades daraus bleibt dem Programmierer überlassen!

[5] Dass die Wertigkeiten genau andersrum sind, wie bei den Fototransistoren der Linienverfolgung ist schaltungstechnisch bedingt und ein Zugeständnis an die Einfachheit der Schaltung.

### 9.2.12. unsigned char PollSwitch(void)

Die Taster (K1-K6) werden ausgewertet. Diese Funktion liefert ein Byte. In diesem Byte ist die Information enthalten, welche Taster gedrückt wurden. Dabei setzt Taster 1 das 5. Bit, Taster 6 das 0.

Bit.
Bit0 (1) -> K6
Bit1 (2) -> K5
Bit2 (4) -> K4
Bit3 (8) -> K3
Bit4 (16) -> K2
Bit5 (32) -> K1

Wären also die Tasten 1,3 und 5 gedrückt, so würde die Funktion 32 + 8 + 2 = 42 zurückliefern.

Evtl. muss diese Funktion mehrfach hintereinander aufgerufen werden, um das „richtige“ Ergebnis zu erhalten. Der geladene Kondensator C7 muss sich erst entladen. Dies kann ein wenig dauern. Wenn der A/D-Wandler zu früh abtastet, können verschiedenste Spannungswerte gemessen werden.

***Beispiel:***

```
unsigned char taste;
.
.
taste = PollSwitch();
if (taste>0) {MotorSpeed(0,0);}
```

***So, das war’s. Jetzt darf man etwas eigene Kreativität walten lassen.***

# Teil IV. Anhänge

## A. Stückliste

Zusätzlich zu einem Tischtennisball werden zum Aufbau eines ASUROs noch folgende Teile benötigt:

1x Platine ASURO
2x Motoren Typ Igarashi 2025-02
1x Diode 1N4001
8x Dioden 1N4148
4x Transistoren BC 327/40 oder BC 328/40
4x Transistoren BC 337/40 oder BC 338/40
1x Integrierter Schaltkreis CD 4081BE
1x Prozessor ATmega 8L-8PC (vorprogrammiert)
1x IR-Empfänger SFH 5110-36
1x IR-LED SFH415-U
2x Fototransistoren SFH300
3x LEDs 5mm rot hell diffus
1x Duo-LED 3mm rot/grün
2x Side-Fototransistoren LPT80A
2x Side-LEDs IRL80A
1x Schwinger 8MHz
2x Elko 220F mind. 10V RM 3,5/10
4x keramische Kondensatoren 100nF RM 5,08
2x keramische Kondensatoren 4,7nF RM 2,54
1x 100 1/4 W 5%
2x 220 1/4 W 5%
4x 470 1/4 W 5%
11x 1k 1/4 W 5%
3x 2k 1/4 W 1%
2x 4,7k1/4 W 5%
1x 8,2k 1/4 W 1%
1x 10k 1/4 W 1%
1x 12k 1/4 W 1%
1x 16k 1/4 W 1%
1x 20k 1/4 W 5%
1x 33k 1/4 W 1%
1x 68k 1/4 W 1%
1x 1M 1/4 W 5%
3x Sockel 14 pol.
6x Detektor-Taster
1x Schalter
1x Batteriehalter
1x Jumper
1x Stiftleiste 2pol RM 2.5
2x Getriebezahnräder 10/50 Zähne; 3,1mm Bohrung Modul 0,5
2x Getriebezahnräder 12/50 Zähne; 3,1mm Bohrung Modul 0,5
2x Motorritzel 10 Zähne
2x Stellring für 3mm-Achse

# A. Stückliste

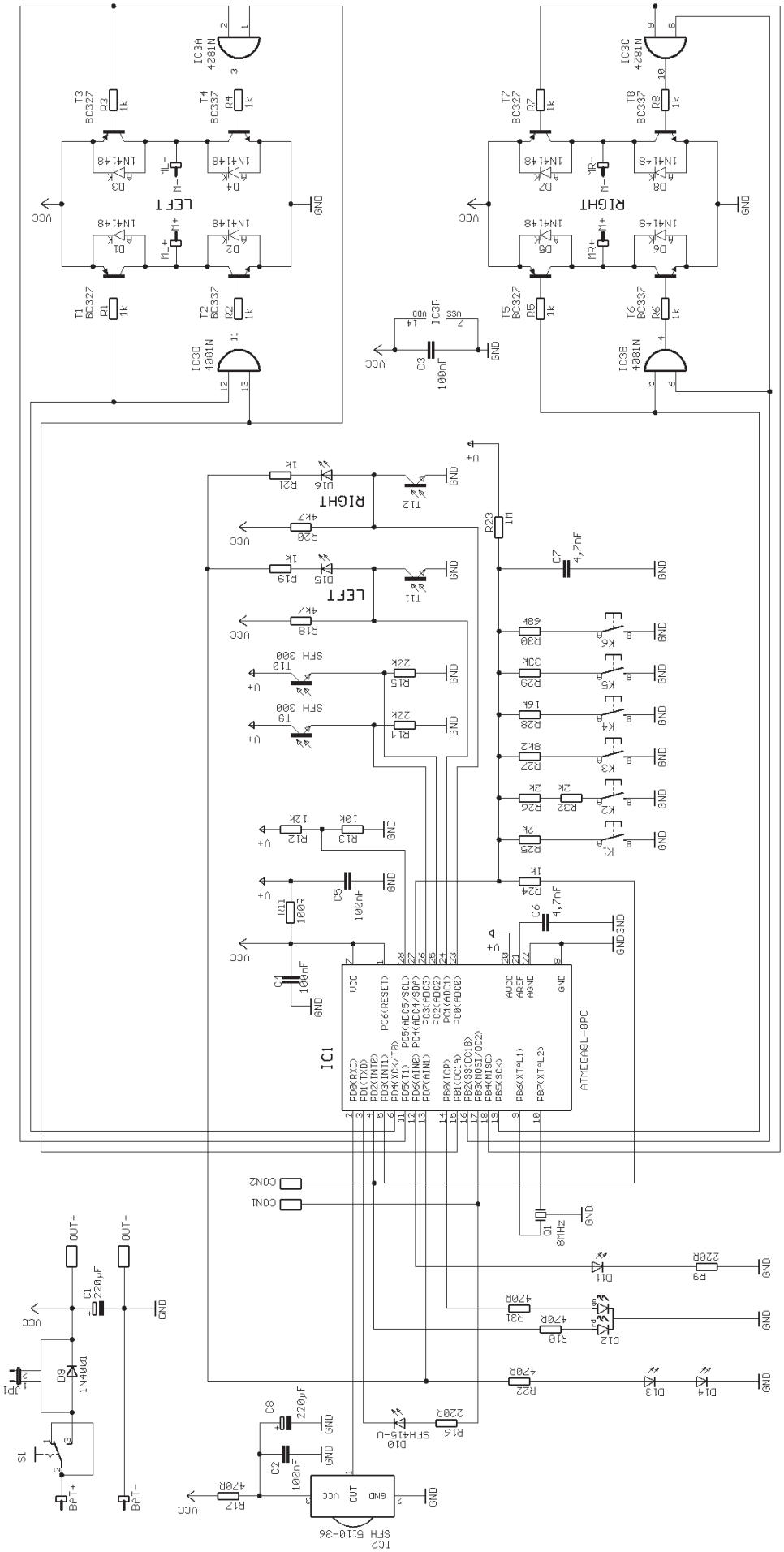
4x Kabelbinder
1x Kabelbinder lösbar
2x Gummireifen 38mm
2x Messingstab 42mm lang, 3mm Durchmesser
2x Messingstab 24,5mm lang, 3mm Durchmesser
ca. 15cm Schaltlitze rot 0,14mm
ca. 15cm Schaltlitze schwarz 0,14mm
2x Encoderscheiben (siehe 2.4)

## Für den zugehörigen RS-232-IR-Transceiver braucht man folgende Bauteile:

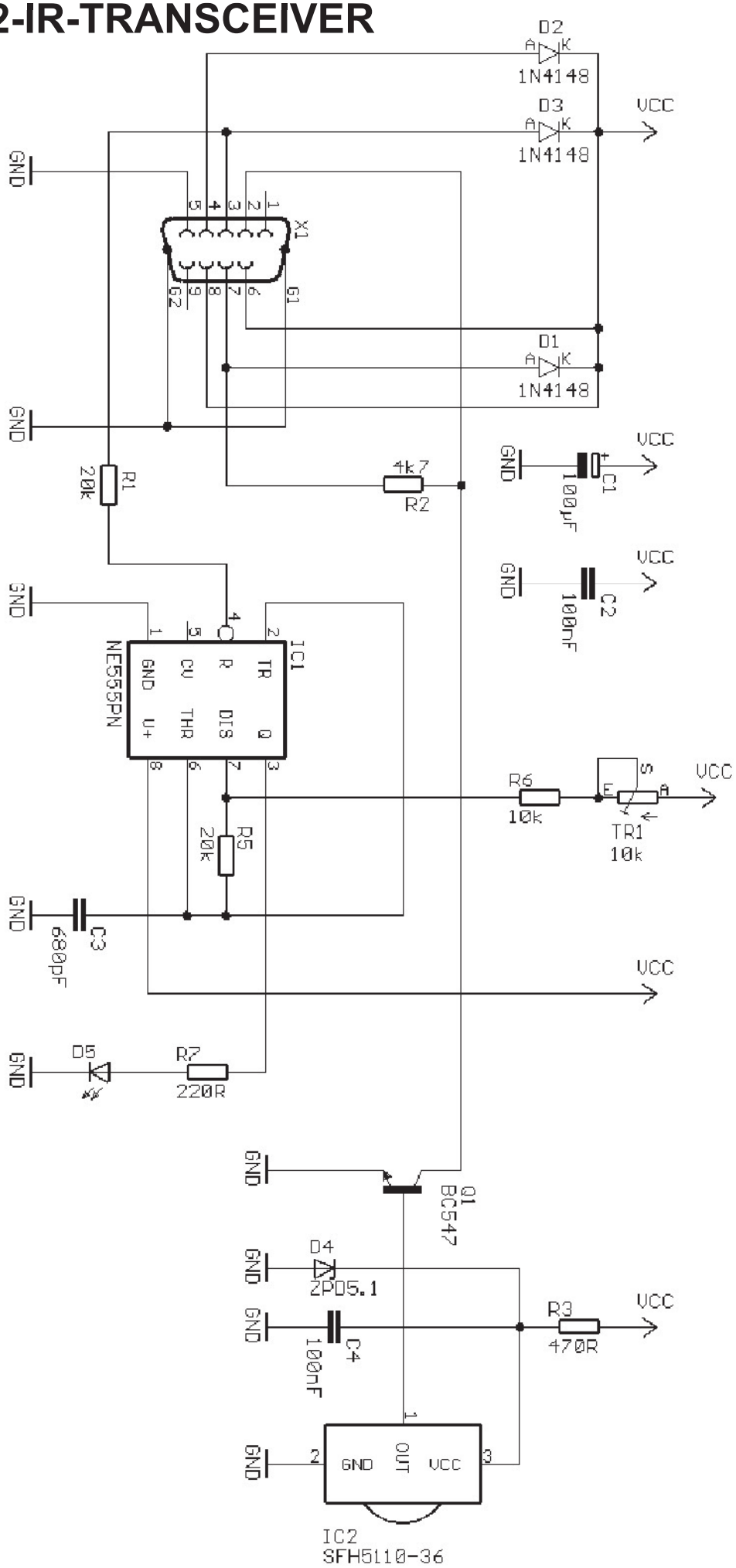
1x Platine IR-RS232-Transceiver
3x Dioden 1N4148
1x Zenerdiode ZPD5.1
1x Transistor BC547 A,B oder C oder BC548 A,B oder C
1x Integrierter Schaltkreis NE555N
1x IR-Empfänger SFH 5110-36
1x IR-LED SFH415-U
1x Elko 100F mind. 16V RM 2,5/6
2x keramische Kondensatoren 100nF RM 5,08
1x keramischer Kondensator 680pF RM 2,54
1x 220 1/4 W 5% oder besser
1x 470 1/4 W 5% oder besser
1x 4,7k1/4 W 5% oder besser
1x 10k 1/4 W 1%
2x 20k 1/4 W 5% oder besser
1x Trimmer 10k stehend RM 2,5/5
1x Sockel 8 pol.
1x 9-pol. SUB-D-Buchse

## Alternativ kann man den USB-Tranceiver benutzen:
## Der USB-IR-Transceiver wird als fertiggerät geliefert!

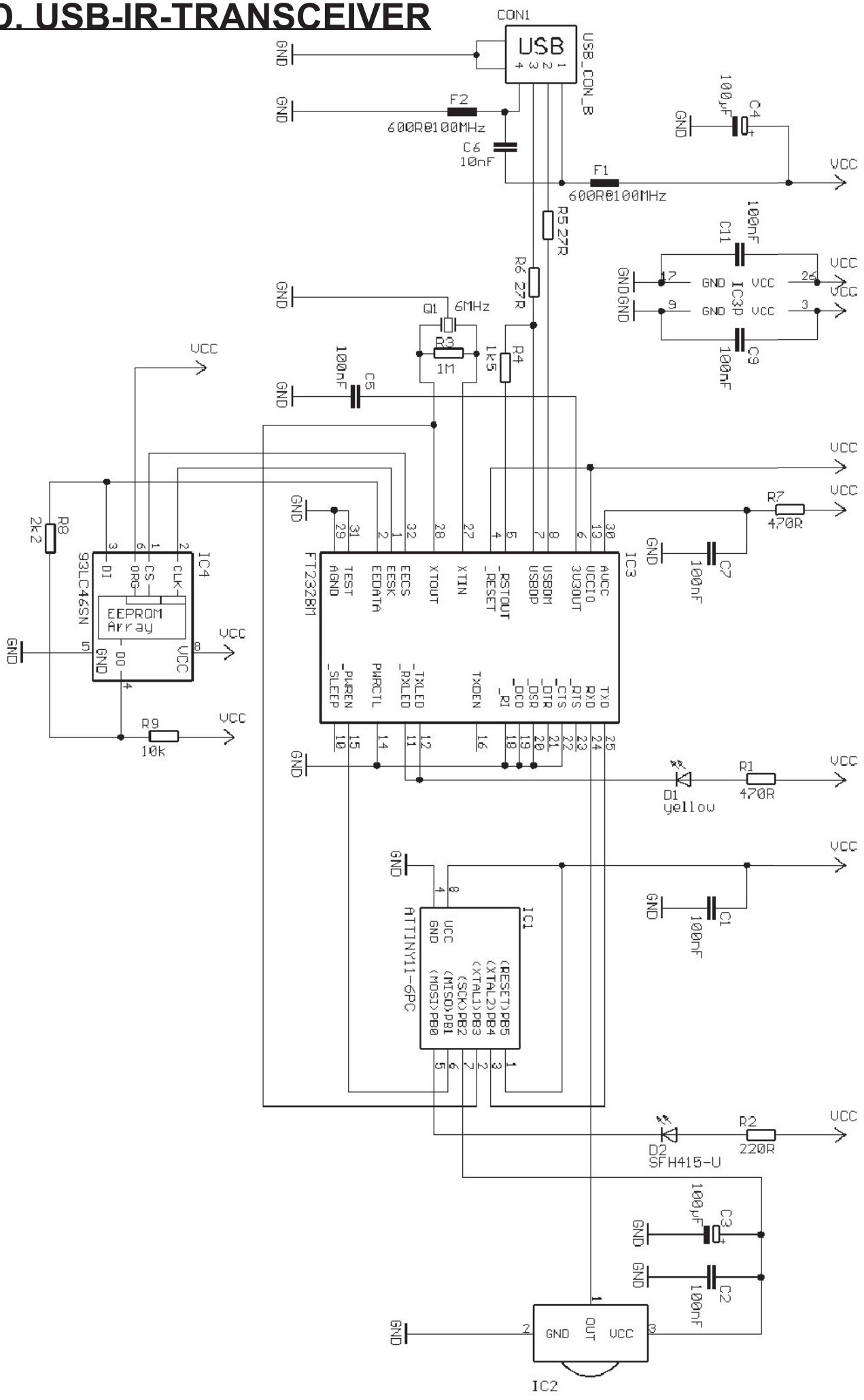
# B. Schaltpläne ASURO



# C. RS232-IR-TRANSCEIVER



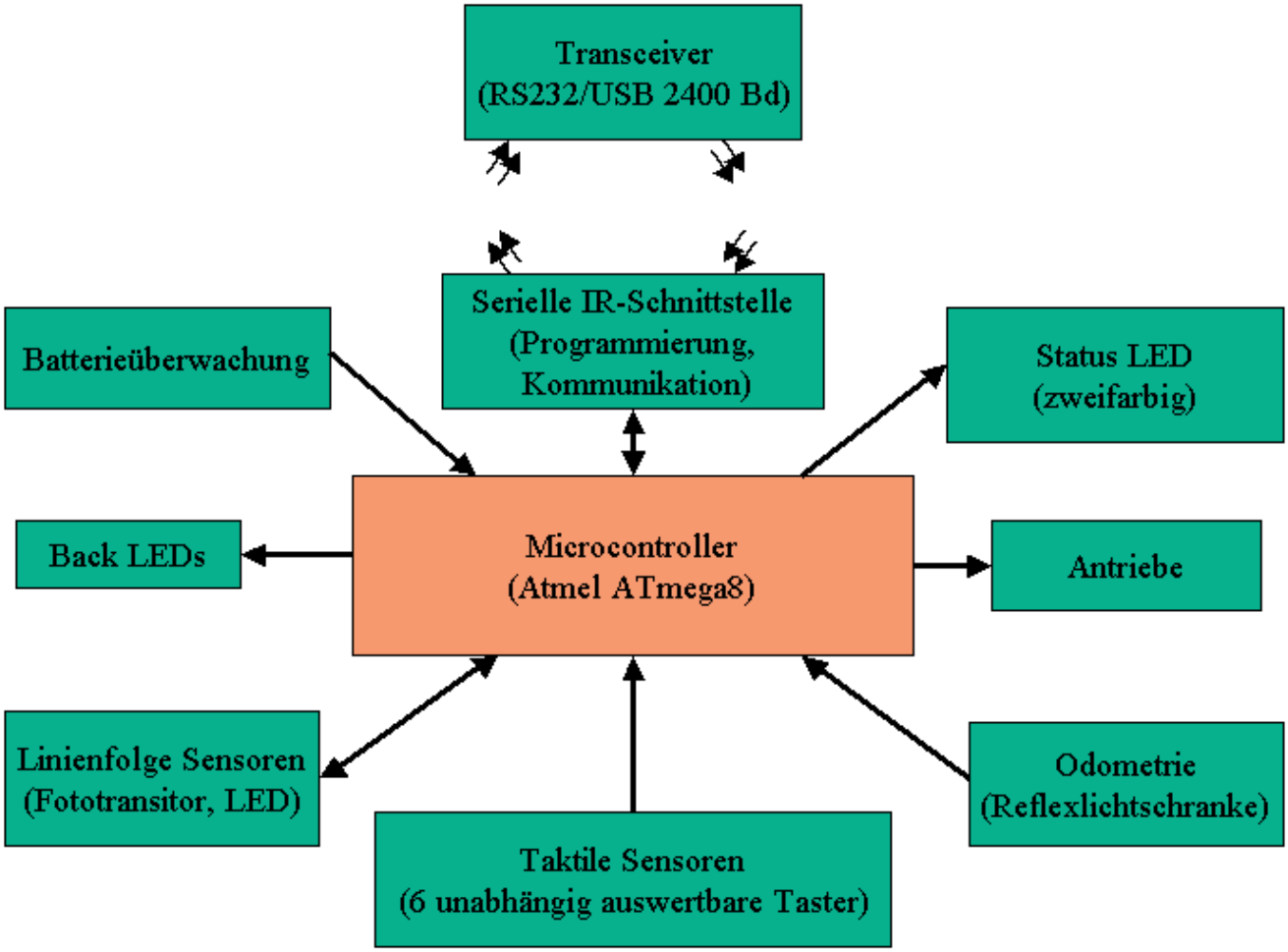
# D. USB-IR-TRANSCEIVER

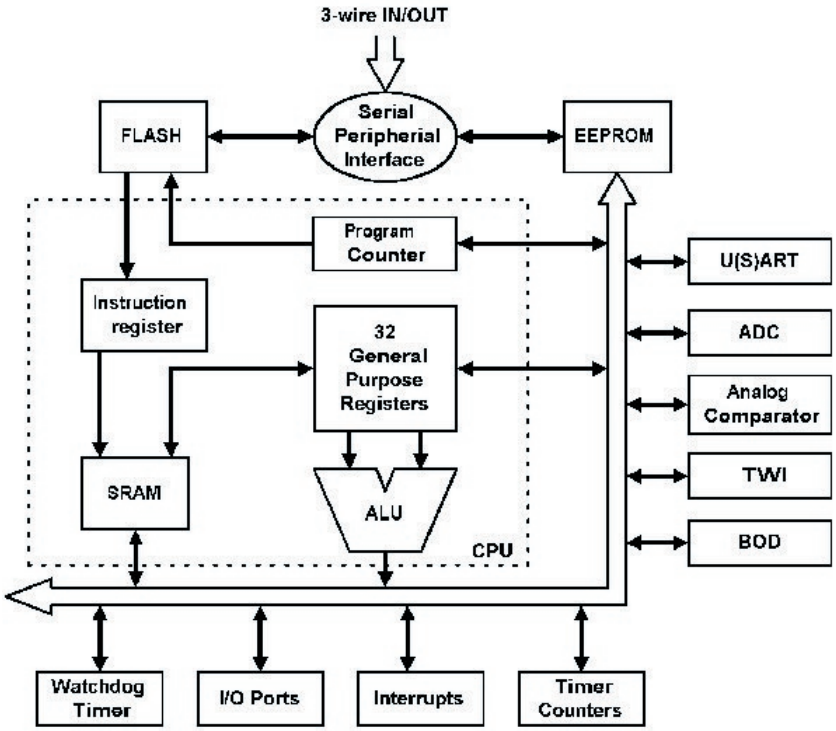





SFH5110

# E. BLOCKSCHALTBILD ASURO




# F. BLOCKSCHALTBILD AVR PROCESSOR

# G. LIEFERUMFANG ASURO