ParkHilfe

Auto 1, Kratzer 0.

Von **Terry Hinrich** (USA)

Parken in Garagen kann sich als schwierig erweisen, da man nicht so leicht erkennen kann, wo das Auto eigentlich aufhört. Ein Tennisball an einer Schnur oder eine Markierung am Boden oder an der Wand kann helfen. Aber da der Tennisball nicht wirklich sicher richtig hängt bzw. die Peilung per Markierung so eindeutig nicht ist, sind solche Tricks wenig zuverlässig. Daher wurde eine elektronische Hilfe auf Ultraschallbasis entwickelt. Die „ParkHilfe“ löst das Einparkproblem tatsächlich einfach, zuverlässig und preiswert. Nur rund 30 € genügen für die Elektronik samt Gehäuse und Netzteil.

Nomen est omen

ParkHilfe setzt auf ein Ultraschall-Transceiver-Modul. Dabei handelt es sich um ein Stück Elektronik, das Audiosignale mit 40 kHz sendet und empfängt. Diese Frequenz ist weit jenseits des menschlichen Hörvermögens, das üblicherweise bei unter 20 kHz endet. Selbst Hunde können so hohe Töne nicht hören. Der Sender schickt eine kurze Impulsfolge los und der Empfänger wartet auf die Echos – Reflektionen von Gegenständen in der Nähe. Ein Auto wäre z.B. solch ein Reflektor.

Das Modul erledigt den größten Teil der Signalverarbeitung. Es benötigt lediglich einen Trigger-Impuls, um eine Messung auszulösen. An seinem Ausgang stellt es dann die empfangenen Echos zur Verfügung. Durch die Messung der Zeit zwischen Impuls und Echo kann man sehr einfach die Entfernung berechnen. Falls kein reflektierendes Objekt im Ultraschallbereich ist, wird ein Impuls für die maximale Distanz erzeugt. Mit einem ein-

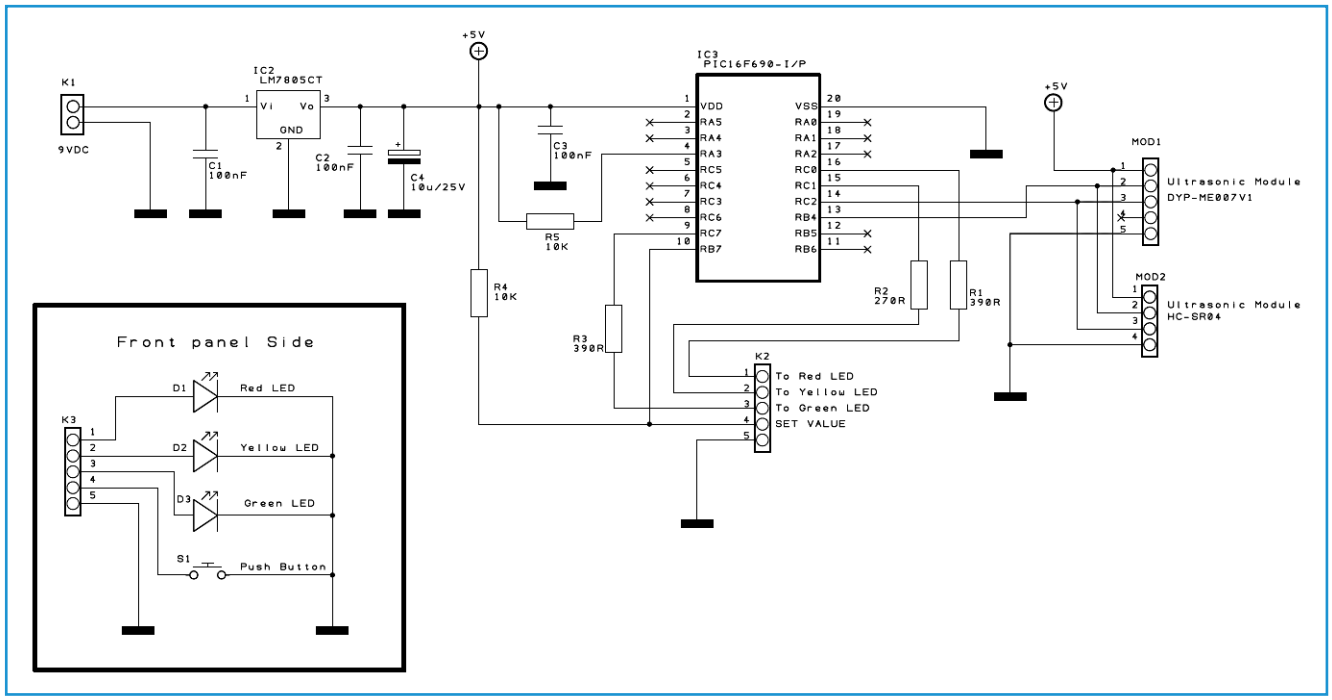


Bild 1.
Schaltung der ParkHilfe.

fachen Mikrocontroller kann man die Messung starten und eine Zeitmessung vornehmen.

Aufgrund amerikanischer Gepflogenheiten wurde die Auflösung vom Autor auf 1“ = 2,54 cm festgelegt. Einem Zoll hin und zurück entspricht auf Meereshöhe eine Laufzeit von etwa 148 µs. Nah dem Trigger zählt das Programm daher Intervalle zu 148 µs, um die Entfernung reflektierender Objekte zu bestimmen. Für den Einsatzzweck ist das genau genug. Der Schall-„Strahl“ ist natürlich kegelförmig und fächert mit zunehmender Entfernung auf. Irgendwann erhält man daher Reflektionen „ungültiger“ Objekte. Diese Tatsache begrenzt die nutzbare Entfernung auf etwa 3 m. In der Praxis ist das kein Problem, da man das Auto kaum mehr als 3 m von der Wand parken wird. Sowohl der Autor als auch seine Frau konnten dank dieser Technik zuverlässig und präzise einparken.

Für dieses Projekt ist es gut zu wissen, wie ParkHilfe eingesetzt werden soll. Von daher folgt nun eine Tour mit den Stationen Display (die LEDs), Mikrocontroller und Firmware, bevor die eigentliche Schaltung erläutert wird. Letztere ist in **Bild 1** zu finden. Die passenden Platinendateien können als DesignSpark-Projekt von der Elektor.LABS-Seite zu diesem Projekt [1] heruntergeladen werden.

Grün – gelb – rot: ein universaler Code

Die Schaltung kommt mit drei LEDs zur Anzeige der Distanz aus. Die grüne LED signalisiert, dass das Auto noch zu weit weg von der Wand ist. Die gelbe LED zeigt die richtige Stoppposition an. Leuchtet die rote LED, ist man schon zu nahe an der Wand und sollte besser ein Stückchen vorziehen.
Insgesamt werden von der ParkHilfe sieben Entfernungen mit drei LEDs signalisiert:

- Grünes Dauerleuchten signalisiert eine große Entfernung und bedeutet, dass man näherkommen kann.
- Die grüne LED blinkt, wenn man näher als 12“ (ca. 30 cm) ist.
- Die grüne und die gelbe LED leuchten dauernd, wenn man 1“ von der Zielposition entfernt ist. Man sollte langsam anhalten.
- Das Erreichen der Zielposition wird durch gelbes Dauerlicht signalisiert. Die Zielzone ist etwa 1“ groß.
- Leuchten die gelbe und die rote LED, verlässt man die Zielposition. Man sollte dann etwas zurücksetzen.
- Leuchtet die rote LED alleine, ist man schon über die Zielposition hinaus. Das sollte man korrigieren.
- Eine blinkende rote LED bedeutet, dass

man schon viel zu nahe an der Wand steht. Man sollte sofort anhalten und zurücksetzen oder man zerstört das Gerät und mehr.

Mit Hilfe dieser Ampel kann man sehr einfach replizierbar und genau einparken. Dies zu erreichen ist mit Software sehr einfach. Rein in Hardware wäre das sehr viel schwieriger. Außerdem können dank Software die Entfernungen sehr leicht geändert werden und per geänderter Firmware in den Mikrocontroller übertragen werden.

Nach dem Einschalten leuchten alle drei LEDs kurz auf und zeigen so, dass die Elektronik betriebsbereit ist. Die Elektronik kann jederzeit stromlos gemacht werden. Eingeschaltet ist sie sehr schnell wieder betriebsbereit.
Bild 2 zeigt den fertigen Aufbau des Autors. Oben sind die LEDs. Bei den zwei großen runden Dingen unter den LEDs handelt es sich um Ultraschall-Sender und -Empfänger. Links sieht man einen Taster und rechts befindet sich die Buchse für ein Steckernetzteil.

PIC als Rechenknecht

Beim verwendeten Mikrocontroller handelt es sich um den Typ PIC16F690 [2] von Microchip. Er hat 20 Pins und mehr Funktionen als hier nötig. Der Autor verwendete diesen Chip, weil er dem PICKit-2-Programmer von Microchip als kostenloses Muster beilag. Er ist schnell genug, hat viele I/O-Pins, genug Speicher sowie EEPROM und ist mit etwa 1,50 € sehr preiswert. Die Anpassung des Codes für andere PICs oder gar Controller anderer Hersteller sollte keine Probleme bereiten.
Der Mikrocontroller wird mit seinem internen Takt von 8 MHz betrieben. Eine einzige Interrupt-Routine lässt die LEDs im 250-ms-Rhythmus blinken. Andere Zeiten werden per Warteschleifen produziert. Der Taster wird von der Software entprellt.

Ihr Wunsch ist mir Befehl

Der Code wurde mit dem HiTech PICC C Lite Compiler [3] generiert. Dabei handelt es sich um eine kostenlose Version, der die Optimierungsmöglichkeiten der Professional-Version fehlen. Für gelegentliche Nutzung ist das aber ein guter Kompromiss. Der Code ist logisch genug, um ihn einfach portieren zu können. Der Source-Code ist online via Elektor.LABS [1] erhältlich. Man versteht die Firmware am besten, wenn man den Code ändert und schaut, was passiert. Falls Sie keinen Compiler und nur einen Programmer haben, können Sie auch die zugehörige Hex-Datei downloaden. Der Source-Code ist etwas ungewöhnlich formatiert. Die Hauptfunktion steckt am Ende der Datei. Die einzelnen Funktionen kommen also vorher. Auf diese Weise braucht man keine Funktions-Prototypen. Außerdem sind die einzelnen Funktionen nach Funktion gruppiert und mit einem entsprechenden alphanumerischen Präfix versehen, damit sie sich alphabetisch leicht sortieren und finden lassen. Dieses Prinzip ist vor allem bei größeren Projekten nützlich.

Bild 2.
Der fertige Prototyp des Autors sieht im Gehäuse ganz passabel aus.

„Sehen“ per Ultraschall-Modul

Im Prototypen des Autors steckt ein Modul des Typs TS601P01. Es hat drei Anschlüsse: 5 V als Eingang, Masse und einen bidirektionalen Daten-Pin.
Als wir aber für Elektor.LABS einen Prototypen bauen wollten, zeigte sich, dass ein TS601P01 nur schwer erhältlich ist. Dank Internet fanden wir aber mit dem DYP-ME007 und dem HC-SR04 schnell andere brauchbare und gut erhältliche Module. Beide Module nutzen vier Anschlüsse, auch wenn das erste fünf Anschluss-Pins hat. Der MOD1-Anschluss von Bild 1 ist für das Modul DYP-ME007 gedacht – MOD2 entsprechend für den Typ HC-SR04. Technische Feinheiten der Module finden sich in den zugehörigen Datenblättern, die ebenfalls via Elektor.LABS [1] erhältlich sind.
Die Software wurde für die Verwendung von Modulen mit 4-Pin-Anschluss adaptiert.

Energie

Der eingesetzte Spannungsregler 7805 benötigt keinen Kühlkörper. Mit einer LED fließt ein Strom von etwa 15 mA. Mit drei leuchtenden LEDs steigt der Strom auf etwa 28 mA. Hierfür reicht sogar ein 78L05. Da die Schaltung ja dauernd in Betrieb sein sollte und sie sowieso kaum Strom verbraucht, wurde bewusst auf einen Netzschalter verzichtet. Zur Versorgung eignet sich praktisch jedes kleine Netzteil, das eine Gleichspannung von 9 V bis 15 V liefert.

Platine(n)

Der Prototyp des Autors wurde auf einer einseitigen Experimentierplatine (Radio Shack 276-170) aufgebaut. Diese Platine misst 5 x 15 cm und eignet sich für ICs und Punkt-zu-Punkt-Verdrahtung. **Bild 3** zeigt den kompletten Aufbau. Die Platine wurde mit 15 mm langen Abstandshaltern auf dem (herausnehmbaren) Boden des Gehäuses befestigt. Die LEDs sitzen auf der Oberseite.

Im Elektor-Labor wurden zwei Platinen entworfen, die den Nachbau vereinfachen (siehe **Bild 4**). Ultraschall-Modul, Mikrocontroller, Spannungsregler und die meisten Bauteile sitzen auf der Hauptplatine. Die LEDs und der Taster sitzen auf einer Tochterplatine. Wie schon die Schaltung können auch die Layout-Dateien im DesignSpark-Format von Elektor.LABS [1] heruntergeladen werden.

Einstellung der Zielposition

Notwendig ist die Einstellung bei der ersten Inbetriebnahme und wenn man ein neues Auto kauft. Beim ersten Einschalten sind Standardwerte enthalten, die in den seltensten Fällen genau passen dürften.

Mit Betätigung von S1 versetzt man die Park-Hilfe in den Programmier-Modus. Die rote und die gelbe LED blinken und zeigen so an, dass man die Zielposition einstellt. Bevor man S1 betätigt, sollte man sein Auto schon etwas näher als ideal geparkt haben. Wenn nun S1 zweimal innerhalb einer Sekunde betätigt wird, wird diese Entfernung als wandnahe Grenze der Zielposition (gelbe LED) abgespeichert. Zur Bestätigung blinken die rote und grüne LED dreimal. Die Grenzen für den roten und grünen Bereich werden automatisch daran angepasst. Nun kann die entferntere Grenze der Zielposition eingestellt werden.

Die entfernte Grenze der Zielposition entspricht dem Punkt, an dem die grüne LED zu blinken beginnt. Wenn man die wandnahe Grenze der Zielposition neu einstellt, wird die entfernte Grenze automatisch auf 4“ = 10 cm weiter weggestellt. Ist das in Ordnung, drückt man S1 einfach noch ein weiteres Mal. Um einen anderen Wert hierfür einzustellen, fährt man sein Auto etwas vor auf die gewünschte Stelle und drückt den Taster dann wie zuvor zweimal innerhalb einer Sekunde.

Die neuen Positionen werden sofort gespeichert und die zugehörigen Distanzen berechnet. Bei der Einstellung sollte man also nicht versehentlich im Bereich der Ultraschall-Sendekeule stehen. Nachdem die zweistufige Einstellung erledigt ist, schaltet das Programm automatisch in den Normalmodus.

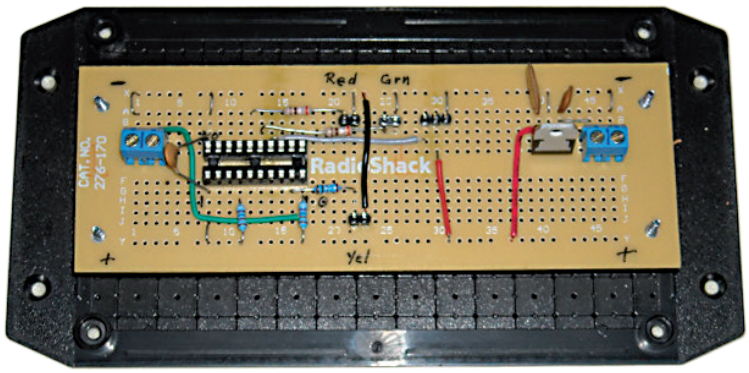


Bild 3.
Das Innenleben des Prototyps. Die Elektor-Version unterscheidet sich leicht davon.

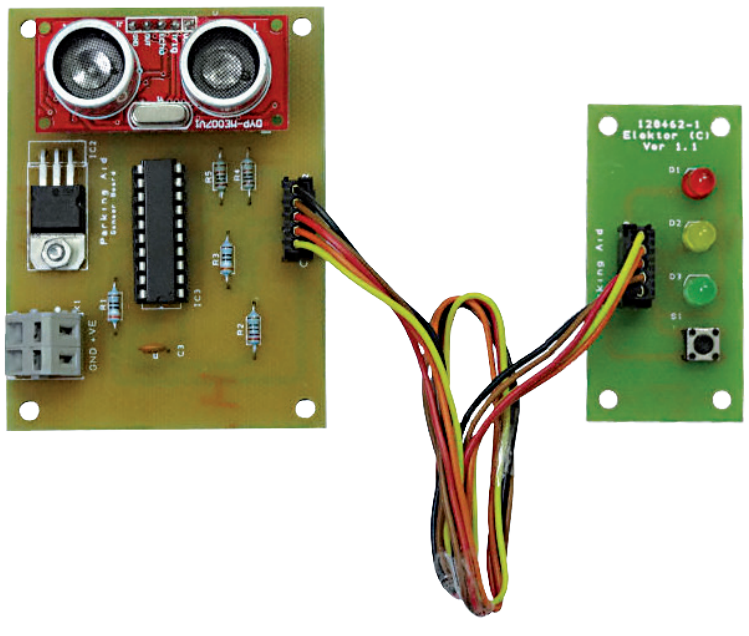


Bild 4.
Die im Elektor-Labor entwickelten Platinen. Die Trennung in zwei Platinen macht es einfacher, die LEDs an einer gut sichtbaren Stelle entfernt von der restlichen Elektronik und dem Ultraschall-Modul anzubringen.

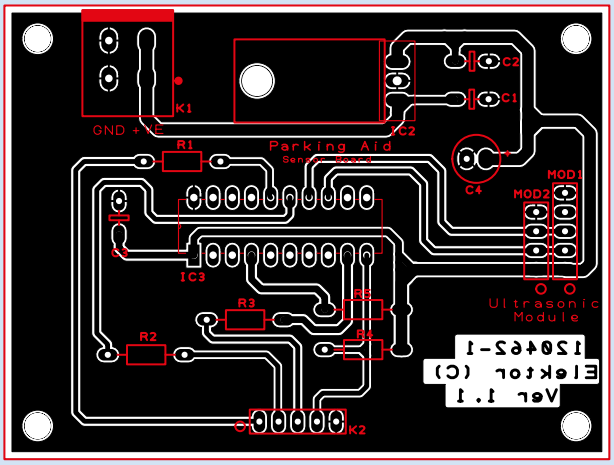
Stückliste

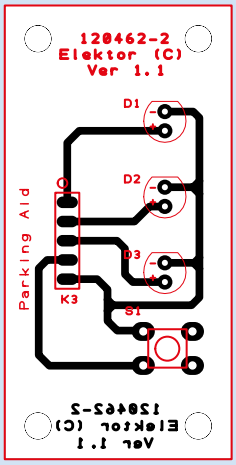
Widerstände:
(alle 0,25W, 5%)
R1,R3 = 390Ω
R2 = 270Ω
R4,R5 = 10K

Kondensatoren:
C1..C3 = 100n, keramisch
C4 = 10µ/25V, Elko, radial

Halbleiter:
D1 = LED, 5mm, rot
D2 = LED, 5mm, gelb
D3 = LED, 5mm, grün
IC2 = 7805, Spannungsregler
IC3 = PIC16F690 (Microchip)

Außerdem:
K1 = 2-pol. Schraubklemme für Platinenmontage, RM 5mm (Multicomp MC000046)
K2,K3 = 5-pol. SIL-Header, RM 1/10“
S1 = Taster, 1,2 x 1,2cm (Multicomp MCDTS2-1N)
MOD1 = 4-pol. SIL-Buchsenleiste, RM 1/10“
MOD2 = 5-pol. SIL-Buchsenleiste, RM 1/10“
Ultraschall-Modul DYP-ME007 oder HC-SR04 (siehe Text und [1])





Die Werte für nahe und entfernte Grenze der Zielposition werden im EEPROM abgelegt, sodass sie auch ohne Strom erhalten bleiben.

Was nun?

Wenn man mit so einem netten Ultraschall-Modul herumspielt, kommen einem dauernd Ideen für neue Anwendungen.
Eine denkbare Verbesserungsmöglichkeit bestände in der Anpassung der Elektronik an Batteriebetrieb. Die Schaltung braucht zwar nicht so viel Strom, aber dennoch ist hier Raum für Optimierungen, da sie ja dauernd eingeschaltet sein muss. Daher muss man Sleep-Modi einbauen, aus denen der Mikrocontroller periodisch aufwacht, um nachzuschauen, ob sich ein Objekt in Detektionsweite befindet.
Auf jeden Fall eignet sich die Schaltung auch jetzt schon prima zur Erleichterung des Einparkens. Wenn die Stoßstange dann einen neuen Kratzer aufweist, gibt es keine Ausrede mehr.

(120462)

Weblinks

[1] www.elektor-labs.com/120462
[2] www.microchip.com/wwwproducts/Devices.aspx?dDocName=en023112
[3] www.htsoft.com

Über den Autor

Terry Hinrich wurde kürzlich pensioniert und kann nun seinem Hobby Elektronik in vollen Zügen frönen. Er hat einen Abschluss als Mathematiker und blickt auf ein fast 50-jähriges Berufsleben zurück, in dem er sich mit vielen Aspekten von Software beschäftigt hat. Er hat Programme in vielen Sprachen für unterschiedlichste Computer geschrieben. Privat setzt er auf Delphi, eine Art Pascal. Die Firmware für ParkHilfe bzw. englisch ParkAid ist sein erstes Programm für so kleine Prozessoren wie einen Mikrocontroller. Dieses Projekt hatte also nicht nur einen praktischen Nutzen für ihn, sondern er hat dabei auch viel gelernt.