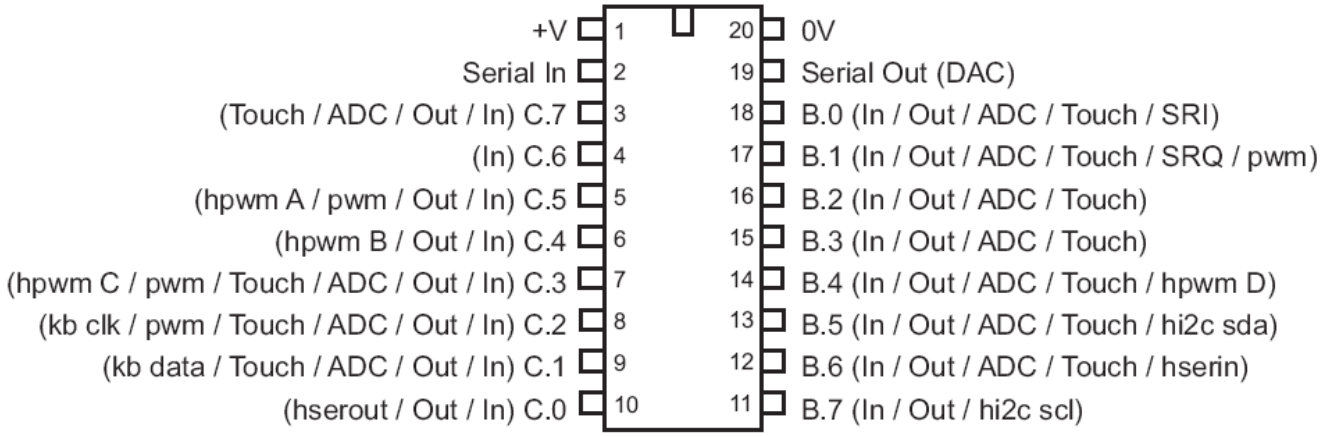
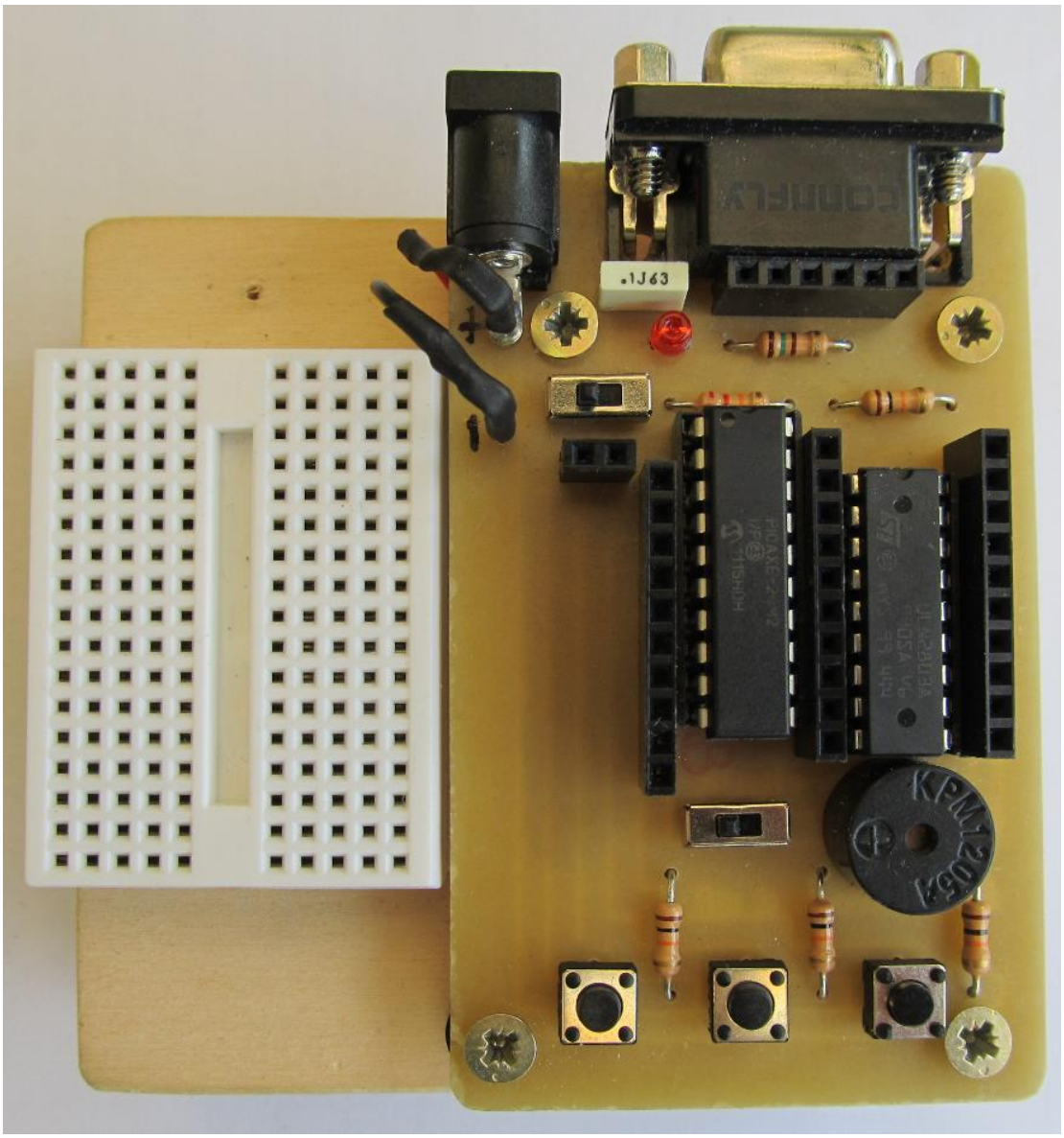
Das GSR-Picaxe-20M2-Basicboard v8

PICAXE-20M2





Inhaltsverzeichnis

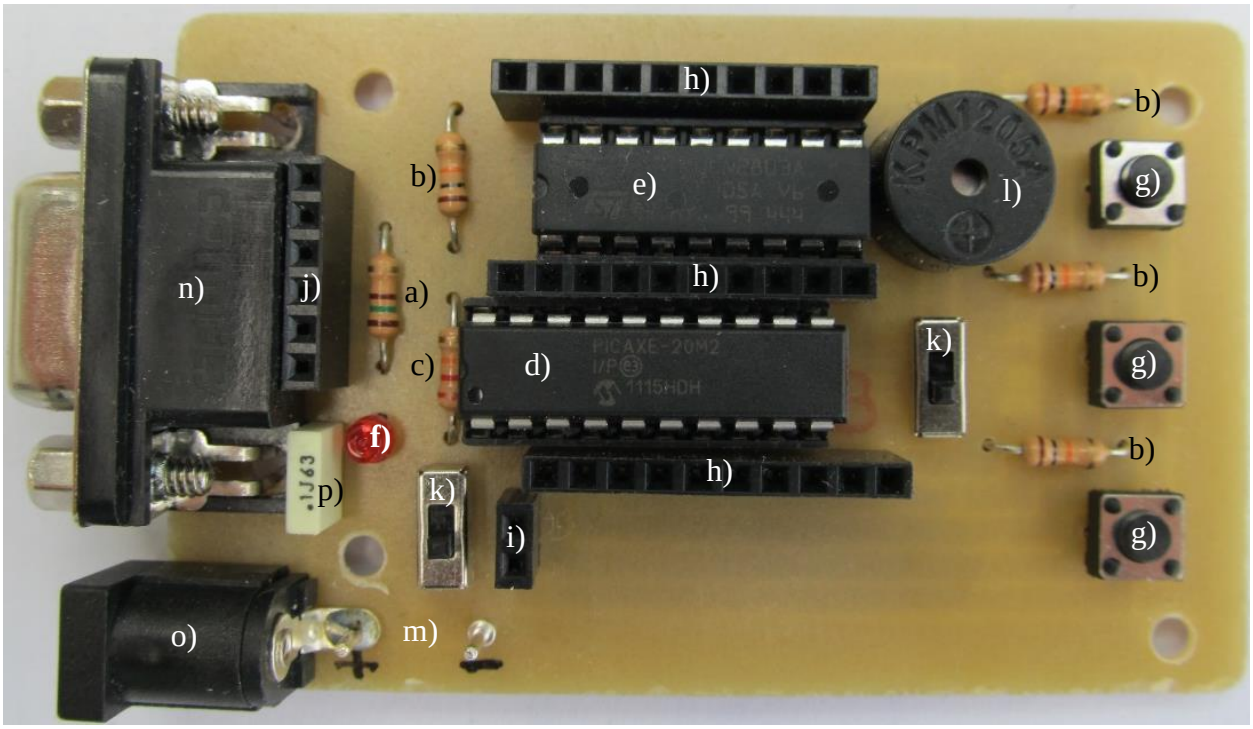
1. Die Herstellung des GSR-Picaxe-20M2-Basicboards Version 8

1) Die vier Befestigungslöcher an den Ecken mit einem 3 mm Bohrer aufbohren.
2) Die zwei Löcher für den Sub-D-9-Pol-Buchsenstecker mit einem 3 mm Bohrer aufbohren.
3) Das Loch für den Minus-Pol der Hohlbuchse mit einem 3 mm Bohrer aufbohren.
4) Die Ecken mit einer Eisenfeile abrunden.
5) Mit Stahlwolle die Kupferseite der Platine abreiben und die Grate entfernen.

Bauteilliste:

Teil	Menge	Bezeichnung	Hinweis
a)	1	Kohleschichtwiderstand 150 Ω	braun, grün, braun
b)	4	Kohleschichtwiderstand 10 kΩ	braun, schwarz, orange
c)	1	Kohleschichtwiderstand 22 kΩ	rot, rot, orange
d)	1	IC-Sockel 20 Pin	Kerbe zeigt nach links zum Stecker
e)	1	IC-Sockel 18 Pin	Kerbe zeigt nach links zum Stecker
f)	1	farbige LED 3 mm	low current / Pluspol unten
g)	3	Mikrotaster	2 Pins
h)	3	Buchsenleisten 10 Pins	
i)	1	Buchsenleiste 2 Pins	
j)	1	Buchsenleiste 6 Pins	
k)	2	Schiebeschalter	
l)	1	Summer	Pluspol unten (zeigt zum Schiebeschalter)
m)	2	Lötstifte 1 mm	
n)	1	Sub-D-9-Pol-Buchsenstecker	
o)	1	Hohlbuchse 5,5 / 2,1 mm	Pluspol an Lötstift, Minuspol Unterseite
p)	1	Kondensator 100nF	ungepolt

Erst wenn alle Bauteile angelötet und kontrolliert wurden, dürfen die beiden ICs (Picaxe-20M2 und ULN2803A) auf die IC-Sockel aufgesteckt werden.



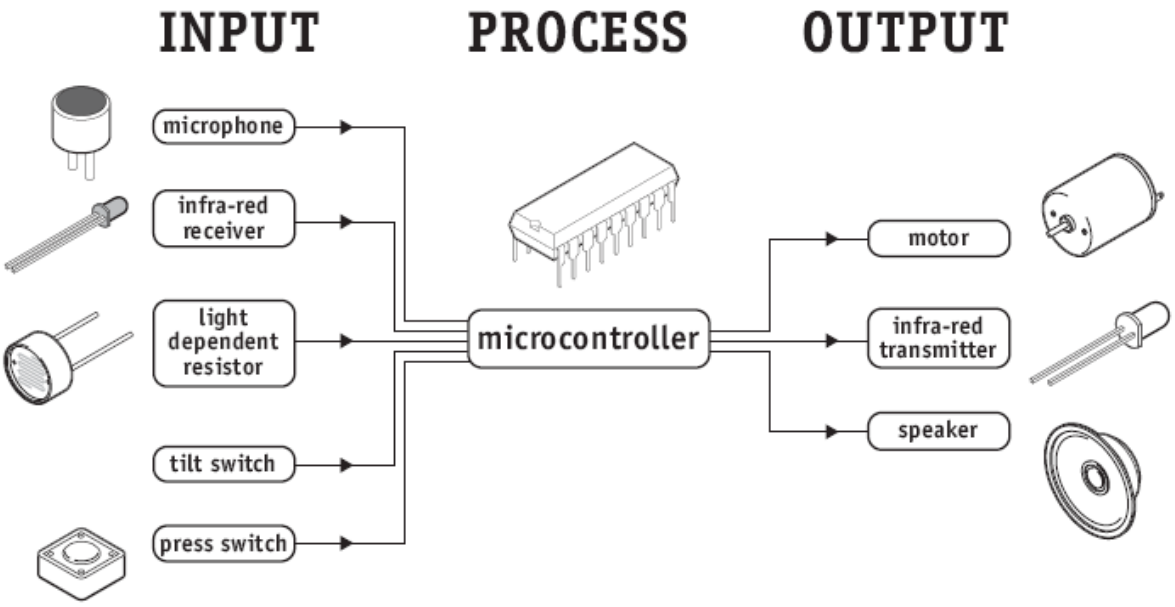
2. Woher bekommt man die Picaxe Mikrocontroller?

Die Mikrocontroller und andere Bauteile kann man sehr günstig direkt über den Picaxe-Store (http://www.picaxestore.com) in Großbritannien bestellen und per Kreditkarte bezahlen. Bei Bestellungen unter 12£ zahlt man nur 3£ Porto (Airmail (EU)). Die Kosy-Datei zum Fräsen der Platine unseres GSR-Picaxe-20M2-Basicboards kann man kostenlos unter www.gsr-technik.de herunterladen.

In Deutschland gibt es den Picaxe-Shop (http://www.picaxe-shop.de) oder roboter-teile.de (http://www.roboter-teile.de).

3. Was man mit einem Mikrocontroller so machen kann…

Der Picaxe-20M2 hat 16 Pins, die als Ein- oder Ausgänge geschaltet werden können. Die untere Grafik zeigt Möglichkeiten auf, den Picaxe-20M2 zu benutzen. Man kann z.B. über einen Input-Pin abfragen, ob ein Schalter offen (low) oder geschlossen (high) ist um z.B. eine LED ein- oder auszuschalten. Mit einer Universalfernbedienung kann über eine Infrarotdiode den Mikrocontroller angesteuert werden, der dann über einen Motortreiber ein Fahrzeug steuert usw..



(Quelle: Picaxe Manual 1, Revolution Education Ltd., www.picaxe.com)

4. Die Software

Der Picaxe Programming Editor kann kostenlos unter www.picaxe.com heruntergeladen werden.

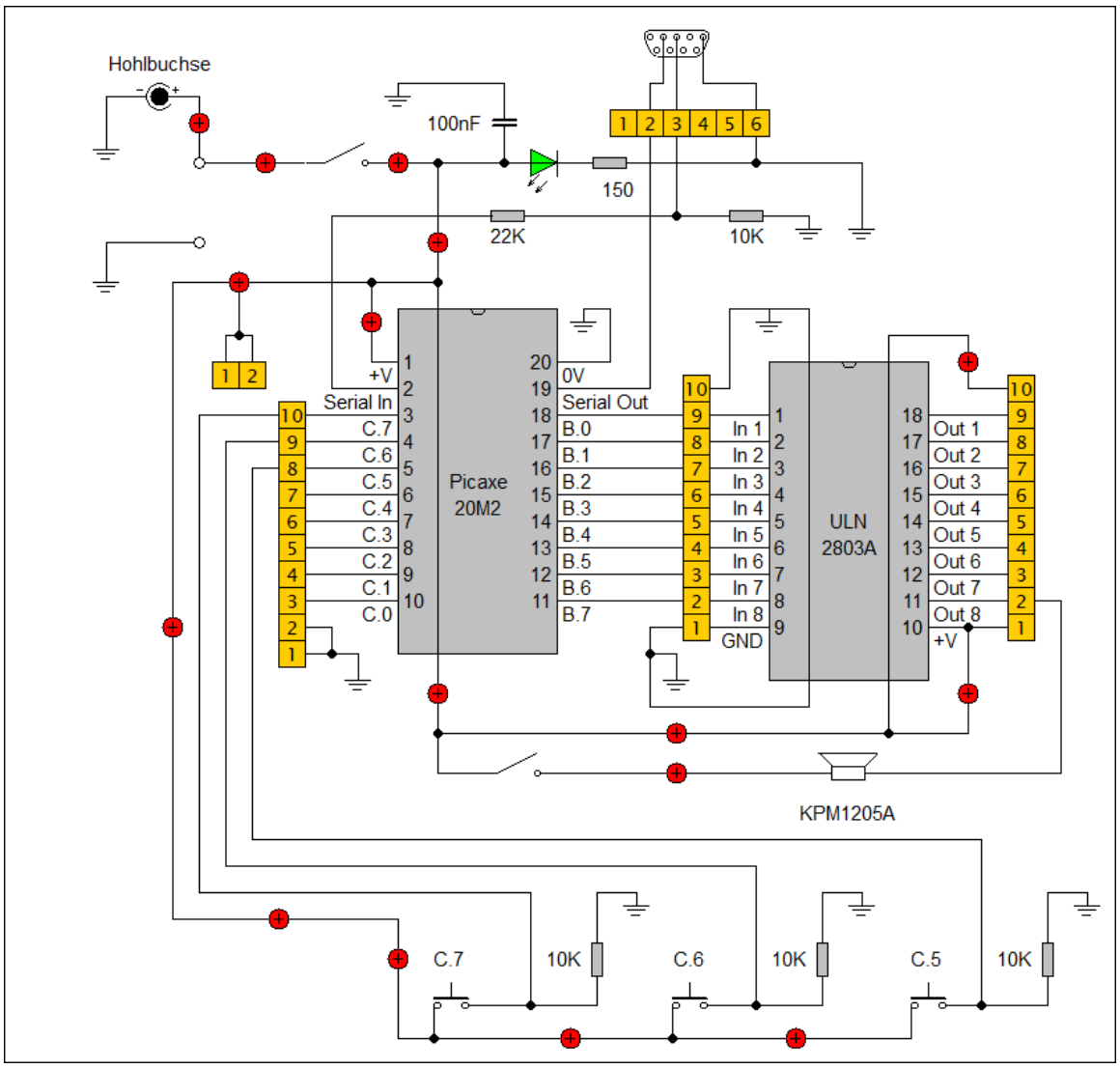
5. Spannungsversorgung und Ausgangsleistung

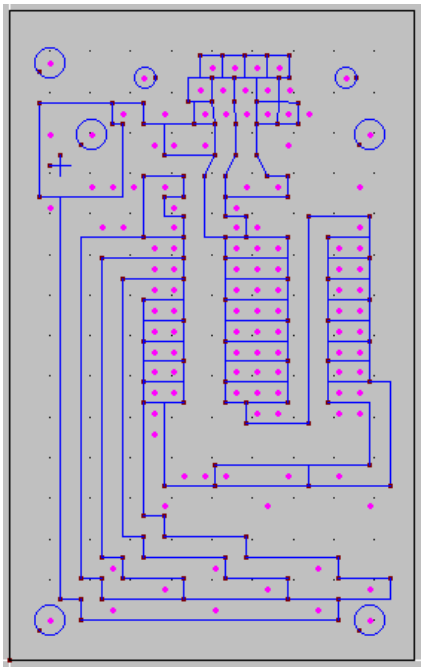
Das GSR-Picaxe-20M2-Basicboard sollte mit einer Spannung von 4,5 V bis 5 V versorgt werden. Eine höhere Spannung als 5,5V kann den Chip zerstören. Jeder Input oder Output Pin darf mit maximal 20 mA Stromstärke belastet werden. Der gesamte Picaxe-20M2-Chip mit maximal 90 mA.

Drei Vorschläge zur Spannungsversorgung:

1. Drei AA-Batterien in einem Batteriehalter und zwei Steckhülsen (3 x 1,5V = 4,5V)
2. Vier AA-Akkus in einem Batteriehalter und zwei Steckhülsen (4 x 1,2V = 4,8V)
3. Steckernetzteil mit 5V (1A) mit Hohlstecker (5,5/2,1 mm) (Pollin Art. 350 830 für 3,95€)

6. Der Schaltplan des GSR-Picaxe-20M2-Basicboards





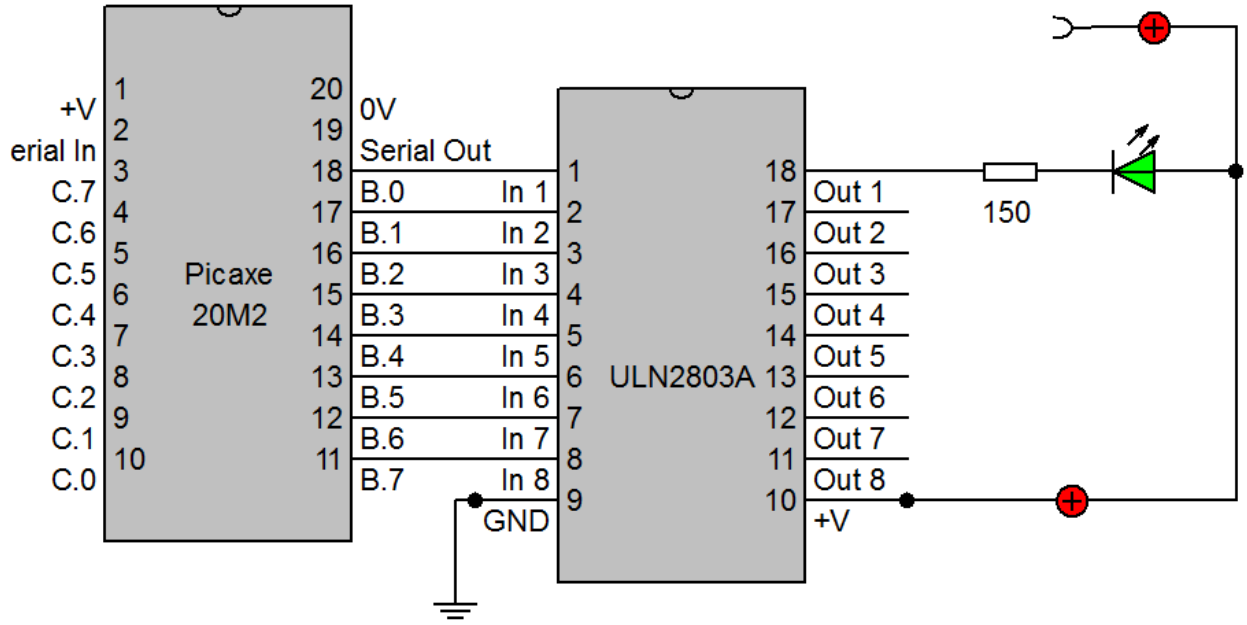
7. Die Ansicht der nccad-Datei

Gezeigt wird die nccad-Datei (Platinenansicht) mit ihren Fräsbahnen und Bohrlöchern. Vor dem Fräsen wurde das Layout natürlich gespiegelt (entspricht also nicht der Unterseite der Platine). Bohrlöcher und Leiterbahnen wurden im Rastermaß (RM) 1,27 erstellt. Nur der Sub-D-9-Pol-Buchsenstecker hat ein RM von 1,38.

8. Der ULN2803A Verstärkerchip

Der ULN2803A ist ein 8-Kanal Darlington-Treiber (DIP18 Gehäuse) und dient als Verstärker für den Picaxe-20M2-Chip (Pins B.0 bis B.7). Er sollte mit einer Spannung von 4,5V bis 50 V versorgt werden (Vs) und kann pro Kanal bis zu 500 mA schalten. Wenn wir bei unserem GSR-Picaxe-20M2-Basicboard den Ausgang B.0 auf high setzen, dann wird bei dem ULN2803A der Out 1 als Masse (GND) frei geschaltet. Der ULN 2803A schaltet also die Masseseite der Verbraucher frei und nicht die Versorgungsspannung (Pluspol, Potential).

Das nachfolgende Schaltbild soll die Funktion des ULN2803A verdeutlichen. Wird der Pin B.0 (Pin-Nummer 18) des Picaxe auf High gesetzt, dann bekommt der ULN2803A auf der linken Seite an Pin In 1 ein Signal und schaltet auf der rechten Seite den Pin Out 1 als Masse frei. Somit kann die angeschlossene LED leuchten.

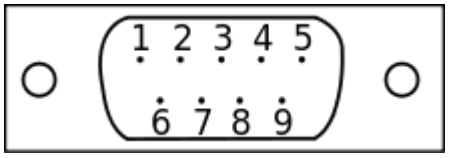


9. Anschluss des GSR-Picaxe-20M2-Basicboards an den Computer

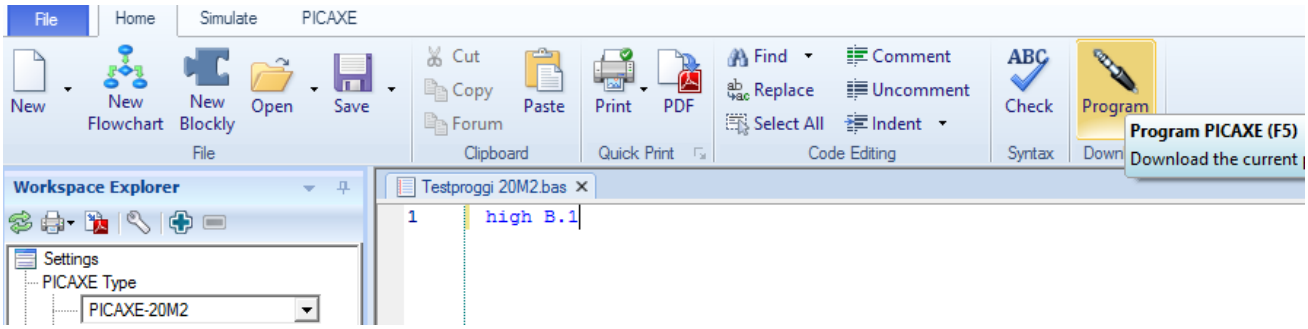
Das Basicboard kann entweder über ein 9-poliges Sub-D-Verlängerungskabel (Kupplung/Stecker) oder ein USB-Adapterkabel auf Seriell (9-polig Sub-D) angeschlossen werden. Bei den USB-Adapterkabeln kommt es auf den eingebauten Chip an. Picaxe empfiehlt Kabel mit FTDI-Chip, aber bietet noch ein USB-Adapterkabel auf Seriell mit dem günstigeren Prolific-Chip an (USB010).

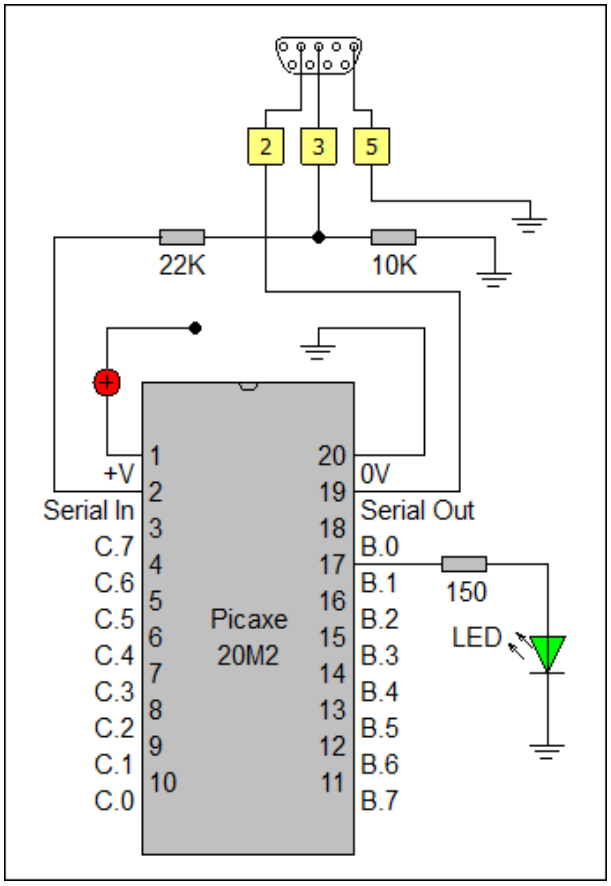
Alternativ habe ich bei Pollin zwei USB-Adapterkabel auf Seriell getestet. Bis jetzt fehlerfrei. Die günstige Version mit Prolific-Chip hat die Artikelnummer 721034 und kostet 6,95€. Die Variante mit FTDI-Chip ist von LogiLink (AU0034) und hat die Artikelnummer 722784 (9,95€). Sehr gut funktioniert auch das günstige USB RS-232 Adapterkabel (CH340-Chip) von CSL für 4,95€.

10. Direkter Anschluss eines Picaxe-Mikrocontrollers an einem seriellen Stecker



Damit man einen Picaxe-Mikrocontroller über ein serielles Kabel (oder USB2Serial) direkt programmieren kann, muss man die Pin-Belegung des Steckers kennen. Nun kann man dem Schaltplan entsprechend die Pins und Bauteile mit Kabeln auf einem Breadboard verbinden. Die Spannung an Pin 1 des Picaxe sollte zwischen 4,5V und 5V betragen.



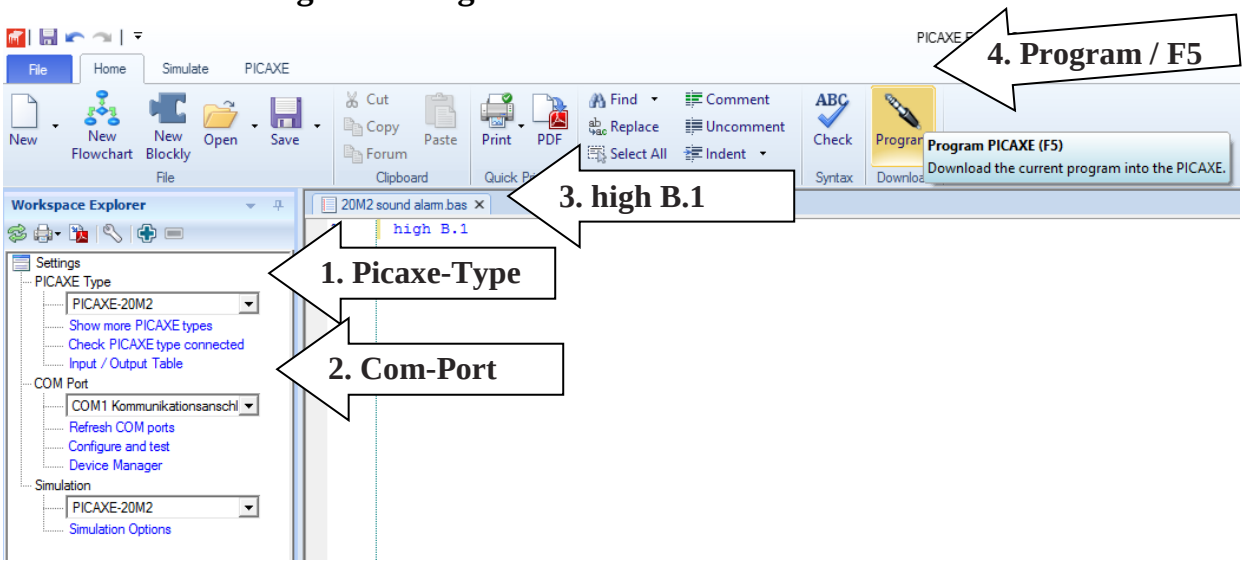


Mit dem einfachen Befehl **high B.1** kann man den Pin 17 (B.1) des Picaxe auf 5V setzen. Somit kann die angeschlossene LED leuchten. Mit **F5 (Program)** kann man das Programm auf den Picaxe übertragen.

Möchte man die LED blinken lassen, dann kann man das nachfolgende Programm benutzen. Der Text nach dem Semikolon (;) ist kein Befehl, sondern nur eine Erklärung und muss nicht mitgetippt werden.

```
start:  ;Schleifenanfang
  high B.1      ;Pin B.1 auf 5V setzen
  pause 200     ;200 Millisekunden warten
  low B.1       ;Pin B.1 auf 0V setzen
  pause 200     ;200 Millisekunden warten
goto start      ;Sprung zum Schleifenanfang
```

11. Das erste Programm eingeben



Starten Sie den **Picaxe Program-Editor**:

1. Wählen Sie links im Fenster unter **Picaxe-Type** den Picaxe-20M2 Chip aus.
2. Jetzt muss noch der passende **COM-Port** (je nach Kabel) ausgewählt werden.
3. Geben Sie das folgende Programm ein: **high B.1**
4. Mit der Schaltfläche **Program** oder **F5** wird das Programm auf das Interface übertragen.
5. Jetzt sollte die LED (B.1) dauerhaft leuchten.

12. Eine LED blinken lassen

Das folgende Programm schaltet Pin B.1 jede Sekunde an und aus. Das Programm zeigt die Benutzung der **high, low**, **pause** und **goto** Befehle. **Der Text nach dem Semikolon (;) erklärt die Befehle und muss nicht mitgetippt werden.**

```
main:          ; erzeuge eine Marke (Sprungadresse) mit dem Namen ‘main’
  high B.1     ; schalte Output B.1 an
  pause 1000   ; warte 1000ms (= 1 second)
  low B.1      ; schalte Output B.1 aus
  pause 1000   ; warte 1000ms (= 1 second)
goto main      ; springe zurück zum Start (main)
```

Die erste Zeile erzeugt eine **Marke (Label)** mit dem Namen ‘main’. Eine Marke wird benutzt als Positionsmerker (**Sprungadresse**) im Programm. In diesem Programm benutzt die letzte Zeile die Marke 'main’ um mit 'goto main’ zurück an die erste Zeile zu springen. Das erzeugt eine **Endlosschleife**, die immer wieder durchlaufen wird.
Beachten Sie, dass der ‘high B.1’ Befehl automatisch den Pin B.1 als Output Pin konfiguriert.
Eine Marke kann ein beliebiges Wort sein (Schlüsselwörter wie ‘high’ ausgenommen), müssen aber mit einem Buchstaben beginnen. Bei der Definition muss die Marke mit einem Doppelpunkt (:) abgeschlossen werden. Der Doppelpunkt ‘sagt’ dem Computer, dass das Wort als eine neue Marke zu interpretieren ist.
Normalerweise setzt man ein paar Leerzeichen (oder Tabs) an den Zeilenanfang jeder folgenden Zeile, beginnend ab der Marke. Das erhöht die Lesbarkeit und Verständlichkeit der Programme.

Kommentare können hinter einem Apostroph (‘) oder einem Semikolon (;) hinzugefügt werden, ebenfalls um die Lesbarkeit zu erhöhen.

Die Befehle **wait** und **pause** erzeugen Zeitverzögerungen. Dabei ist der Befehl **wait** für größere Verzögerungen (Auflösung in Sekunden) gedacht, **pause** kann auch kürzere Verzögerungen erzeugen (gemessen in Millisekunden).
(Quelle: Revolution Education Ltd. Web: www.rev-ed.co.uk Version 0.9, Deutsche Übersetzung: www.roboter-teile.de, Download: www.roboter-teile.de/datasheets/AXE003M_D.pdf)

13. Zuordnung der Pins und LEDs

Die LED-Platine kann auf das GSR-Picaxe-20M2-Basicboard aufgesteckt werden. Dann sind die acht LEDs an den Pins B.0 bis B.7 wie folgt angeschlossen:

B.0	LED 1
B.1	LED 2
B.2	LED 3
B.3	LED 4
B.4	LED 5
B.5	LED 6
B.6	LED 7
B.7	LED 8

Aufgabe 1: LEDs sollen leuchten
Die LEDs 1 bis 8 (an den Pins B.0 bis B.7) sollen nacheinander für je 1 Sekunde leuchten. Das soll sich endlos wiederholen.

Aufgabe 2: LEDs nacheinander vor- und rückwärts
Es sollen nacheinander die LEDs 1 bis 8 (an den Pins B.0 bis B.7) für 500 Millisekunden leuchten und dann die LEDs 7 bis 1 (also rückwärts). Dieses Programm soll auch endlos laufen.

14. Let dirs und let pins (mehrere LEDs gleichzeitig leuchten lassen)

Möchte man mehrer Pins gleichzeitig auf high oder low setzen, dann kann man dazu den Befehl **let pinsB = %11111111** benutzen. Die acht Zahlen nach dem %-Zeichen stehen für die acht Pins (B.0 bis B.7). Die nachfolgende Tabelle soll die Zuordnung verdeutlichen:

let pinsB = %	1	1	1	1	1	1	1	1
	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0

Mit dem Befehl **let pinsB = %10101010** werden also die Pins B.1, B.3, B.5 und B.7 auf high gesetzt. Die übrigen Pins bleiben low.

let pinsB = %	1	0	1	0	1	0	1	0
	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0

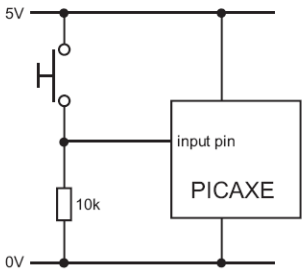
Damit man den Befehl let pinsB benutzen kann, muss man vorher festlegen, dass die B-Pins alle Ausgabe-Pins sind. Dies geschieht am Anfang des Programms mit dem Befehl **let dirsB = %11111111**. Eine 1 nach dem %-Zeichen bedeutet, dass es sich um einen Ausgabe-Pin handelt. Eine 0 bedeutet einen Eingabe-Pin.

Mit dem nachfolgenden Programm leuchten die LEDs 1 und 2 für je 500 Millisekunden.

```
let dirsB = %11111111          ; alle B-Pins als Ausgabepins setzen
main:                          ; erzeugt eine „Marke“ mit dem Namen main
  let pinsB = %00000001        ; der Pin  B.0 wird auf high gesetzt, alle anderen auf low
  pause 500                    ; Pause für 500 Millisekunden
  let pinsB = %00000010        ; der Pin B.1 wird auf high gesetzt, alle anderen auf low
  pause 500                    ; Pause für 500 Millisekunden
goto main                      ; Sprung zurück zur Marke main
```

Aufgabe 3: LEDs verschieben
Es sollen die LEDs 1, 3, 5, 7 und dann die LEDs 2, 4, 6, 8 für je 200 Millisekunden leuchten.

15. Nutzung der Taster und des Befehls If…Then



Die drei Mikrotaster C.7, C.6 und C.5. sind jeweils am Pluspol und an den Eingabe-Pins des Picaxe angeschlossen. Außerdem liegen sie über einen 10k-Pulldown-Widerstand an der Masse. Der Zustand der Pins kann mit dem Befehl If…Then (Wenn…Dann) abgefragt werden. Will man einen gedrückten Taster abfragen, dann benutzt man die 1 (If pinC.7 = 1 Then …). Bei einem offenen Taster die 0.

If pinC.6 = 1 Then flash_B0
Ist der Taster an dem PinC.6 gedrückt (1), dann springe in das Unterprogramm flash_B0.

Nun ein kleines Programm, das die Benutzung der Taster verdeutlichen soll:

```
main:                       ; Marke main
  If pinC.6 = 1 Then flash_B0   ; Abfrage des Tasters C.6; Wenn Taster gedrückt, dann
                              ; spring in das Unterprogramm flash_B0
  If pinC.5 = 1 Then flash_B7   ; Abfrage des Tasters C.5; Wenn Taster gedrückt, dann
                              ; spring in das Unterprogramm flash_B7
  low B.0                   ; Pin B.0 wird auf 0V gesetzt
  low B.7                   ; Pin B.7 wird auf 0V gesetzt
goto main                   ; Sprung zur Marke main

flash_B0:                   ; Unterprogramm flash_B0
  high B.0                  ; Die LED 1 an Pin B.0 leuchtet
goto main                   ; Sprung zur Marke main

flash_B7:                   ; Unterprogramm flash_B7
  high B.7                  ; Die LED 8 an Pin B.7 leuchtet
goto main                   ; Sprung zur Marke main
```

Es können auch mehrer Taster gleichzeitig abgefragt werden:
If pinC.6 = 0 and pinC.5 = 0 then LED_flash

Wenn PinC.6 und PinC.5 offen sind (kein Taster gedrückt), dann gehe in das Unterprogramm LED_flash.

Aufgabe 4: LEDs und Taster
Wird der Taster C.6 gedrückt, dann sollen die LEDs 1 bis 4 leuchten. Wird der Taster C.5 gedrückt, dann sollen die LEDs 5 bis 8 leuchten. Ohne Tastendruck sollen alle LEDs leuchten.

16. Umbenennung von Pins mit dem Befehl symbol

Manchmal ist es hilfreich, Pins umzubenennen. So könnte man in dem obigen Beispiel den PinC.7 in Taster1 umbenennen und im weiteren Programm dann **Taster1** benutzen.

```
symbol Taster1 = pinC.7
symbol Taster2 = pinC.6
main:
If Taster1 = 1 Then flash_B0
If Taster2 = 1 Then flash_B7
```

17. Nutzung von Variablen

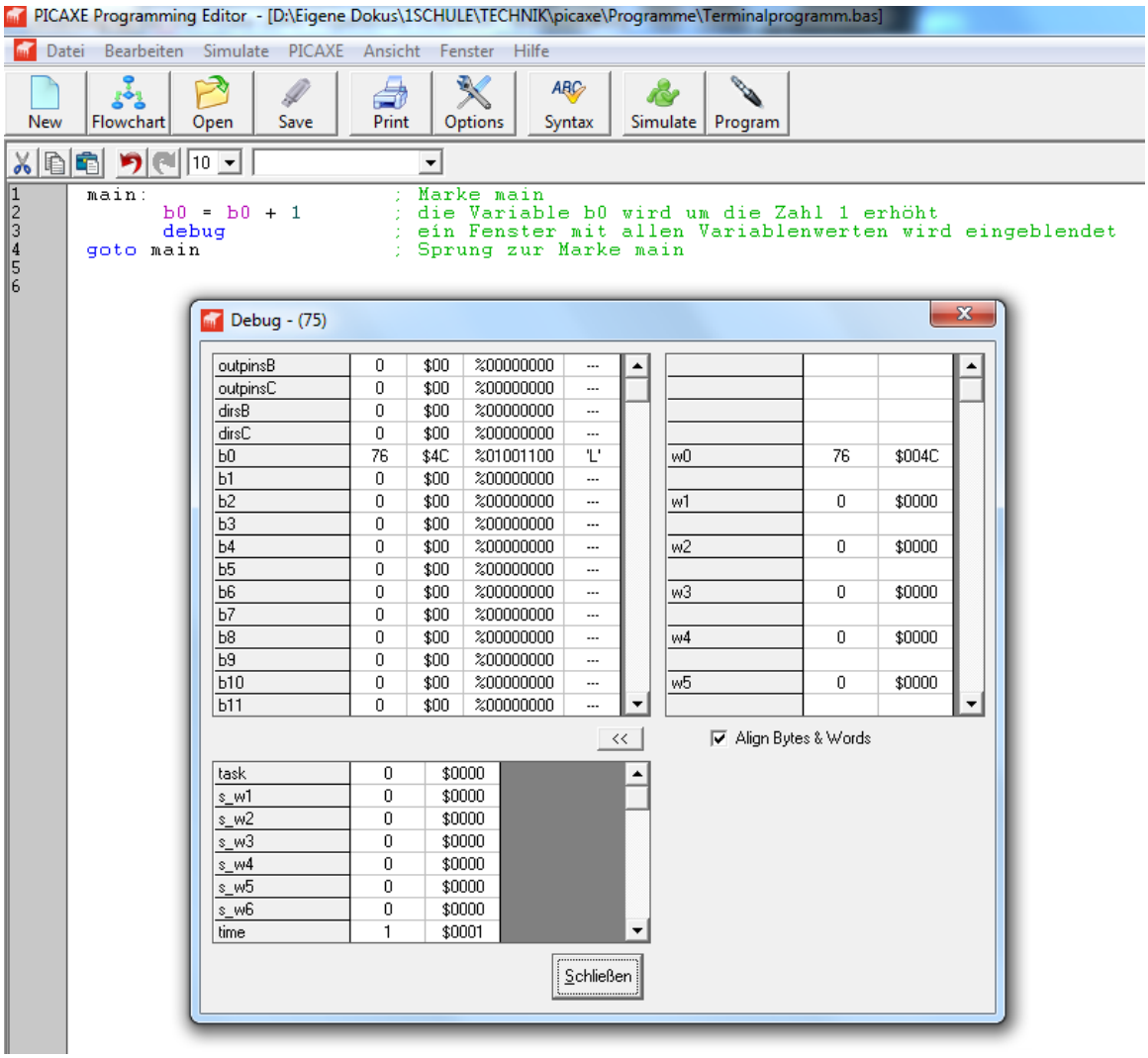
Variablen werden in Programmen genutzt, um Zahlen oder Buchstaben zu speichern. In Basic gibt es 14 Byte-Variablen (b0 bis b13), die jeweils Zahlen von 0 bis 255 speichern können. Wird der Zahlenbereich bis 255 überschritten, wird wieder ab der 0 weitergezählt (Beispiel: 255 + 2 = 1). Möchte man größere Zahlen speichern (0 bis 65535), dann benutzt man die Variablen w0 bis w6. Die Variablen b0 und b1 bilden zusammen die Variable w0.

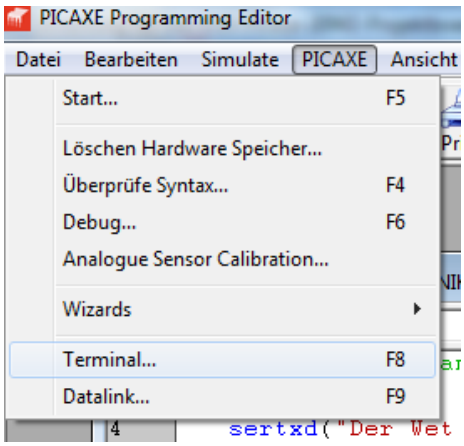
18. Der debug-Befehl

Will man den Wert einer Variablen auf dem Bildschirm ausgeben, dann kann man den Befehl **debug** verwenden. Der Wert der Variablen wird als Dezimalzahl, Hexadezimalzahl, Binärzahl und ASCII-Wert ausgegeben.

Aufgabe 5: debug
Gib das nachfolgende Programm ein:

```
main:
        b0 = b0 +1
        debug
goto main
```

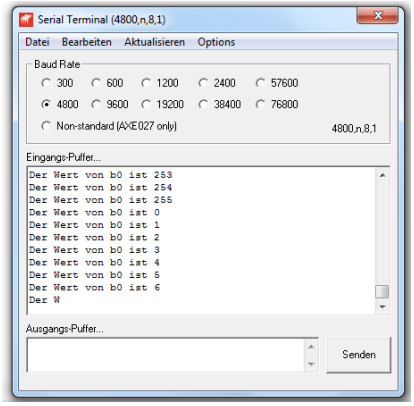




19. Der Befehl sertxd und das „Terminalfenster“

Möchte man neben dem Wert einer Variablen noch einen Text ausgeben, dann sollte man den sertxd-Befehl und das Terminalfenster nutzen. Nach dem Programmdownload muss man das „Bestätigungsfenster“ mit OK wegklicken und kann dann mit der Taste F8 das Terminalfenster öffnen (oder über das Menü PICAXE / Terminal…). Mit dem Befehl ***#terminal 4800*** öffnet sich das Terminalfenster automatisch nach dem Programmstart mit einer Baudrate von 4800 (4800 Symbole pro Sekunde). Die Befehle cr und lf werden benötigt, damit der Text und jeder Wert in einer neuen Zeile dargestellt werden.

Aufgabe 6: Terminalfenster
Gib das nachfolgende Programm ein!




```
#terminal 4800
main:
        b0 = b0 +1
        sertxd ("Der Wert von b0 ist ",#b0, cr, lf)
        pause 10
goto main
```

20. Die for…next-Schleife

Möchte man einen Teil des Programms mehrfach wiederholen, dann kann man die for..next-Schleife verwenden. In diesem Beispiel werden die Anweisungen in der Schleife wiederholt, solange die Variable b0 <= 10 ist. Hat sie den Wert 11, dann bricht die for…next-Schleife ab und es folgt die Anweisung **end** (Programmende).

In diesem Beispielprogramm blinkt die LED 8 (an Pin B.7) zehn Mal.

```
main:                   ‘ erzeugt eine Marke ‘main’
  for b0 = 1 to 10      ‘ Anfang der for…next Schleife; der Variablen b0 wird der Wert 1 zugewiesen
    high B.7            ‘ LED 8 (an Pin B.7) an
    pause 500           ‘ warte 0,5 Sekunden
    low B.7             ‘ LED 8 (an Pin B.7) aus
    pause 500           ‘ warte 0,5 Sekunden
  next b0               ‘ die Variable b0 wird um 1 erhöht, Sprung zum Schleifenanfang
end                     ‘ Programmende
```

Soll die Variable in der for…next-Schleife um einen anderen Wert als 1 erhöht werden, dann muss man die „Schrittweite“ (step) eingeben. Zum Beispiel so:

```
#terminal 4800
For b0 = 10 to 200 step 10
  sertxd ("Der Wert von b0 ist ",#b0, cr, lf)
  pause 1000
next b0
sertxd ("Ende", cr, lf)
```

Aufgabe 7: For…next
Alle LEDs sollen 20 Mal für 250 Millisekunden blinken (mit einer Pause von jeweils 500 Millisekunden).

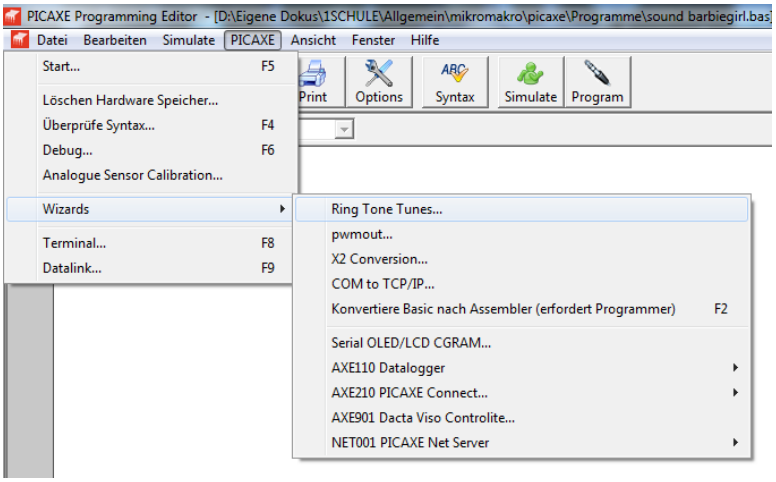
21. Klingeltöne über einen Miniatursummer abspielen

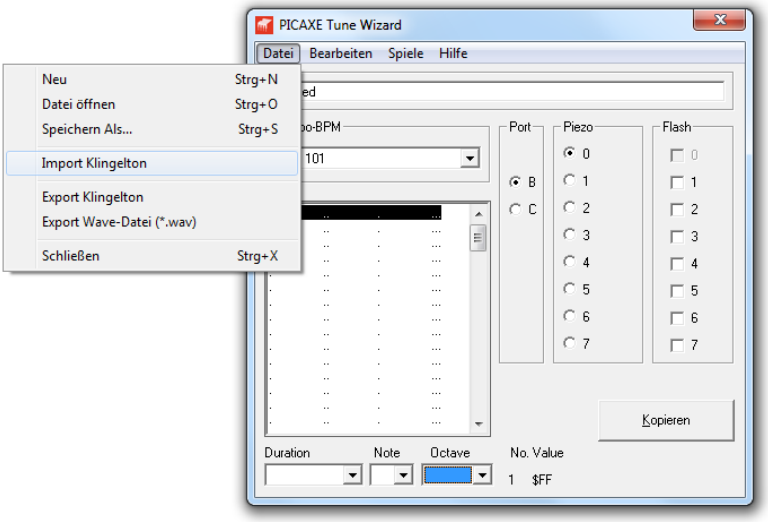
Klingeltöne kostenlos über den nachfolgenden Link herunterladen:

http://www.picaxe.com/RTTTL-Ringtones-for-Tune-Command/

Liegen die Klingeltöne nicht in einer bas-Datei vor (das sind alle Klingeltöne in den Ordnern rtttl, rtttl2 und rtttl3), dann müssen sie noch importiert werden!

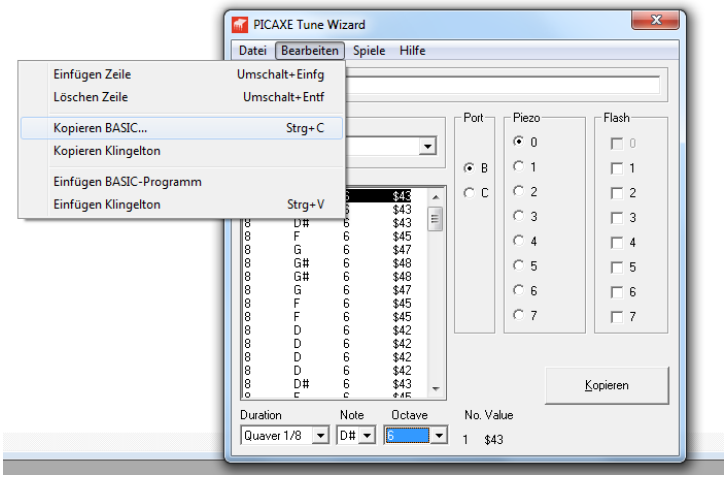
Im Picaxe Programming Editor über das Menü **Datei / Wizards / Ring Tone Tune…** anklicken.





Dann im geöffneten Tune Wizard auf **Datei / Import Klingelton** klicken.

Den importierten Klingelton über das Menü **Bearbeiten / Kopieren BASIC …** in den Editor einfügen. Die anschließende Frage: ***Einfügen in Hauptdokument?*** Mit Ja beantworten.



22. Klingeltöne mit den tune-Befehl abspielen

Es gibt fertige Klingeltöne (ringtones), die man über die folgende Seite herunterladen kann:

http://www.picaxe.com/RTTTL-Ringtones-for-Tune-Command/

Die zip-Dateien ***TV Theme Tunes*** und ***Christmas Tunes*** enthalten schon fertige Basic-Dateien. Sie müssen nur geöffnet werden und können dann direkt abgespielt werden (mit Program). **Ring Tone Text Transfer Language (RTTTL)** oder **Nokring** ist ein verbreitetes Format für Klingeltöne für Mobiltelefone. Über Google kann man mit dem Suchbegriff „RTTTL + Liedname“ viele kostenfreie Klingeltöne herunterladen.

23. Der Soundbefehl

Mit dem Sound-Befehl lassen sich Töne über den eingebauten Summer (KPM1205A) erzeugen. Dazu muss der Schiebeschalter an dem Summer geschlossen sein. Hier ein Beispiel für einen Soundbefehl:

SOUND B.7 (80,50)

Der Soundbefehl ist wie folgt aufgebaut:

SOUND *Ausgabe-Pin (Tonfrequenz, Dauer)*

Unser Soundgeber ist an dem Ausgabe-Pin B.7 angeschlossen. Um „brauchbare“ Töne zu erzeugen, sollte die **Tonfrequenz** zwischen 20 und 120 liegen. Die **Dauer** darf Werte zwischen 0 und 255 enthalten.

Das nachfolgende Programm erzeugt über den Ausgabe-Pin B.7 endlos einen „Piep-Ton“ mit Pause (Frequenz 80 und der Dauer 50).

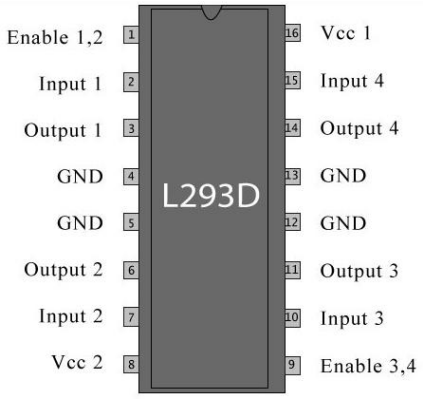
```
main:
  sound B.7, (80, 50)
  pause 500
goto main
```

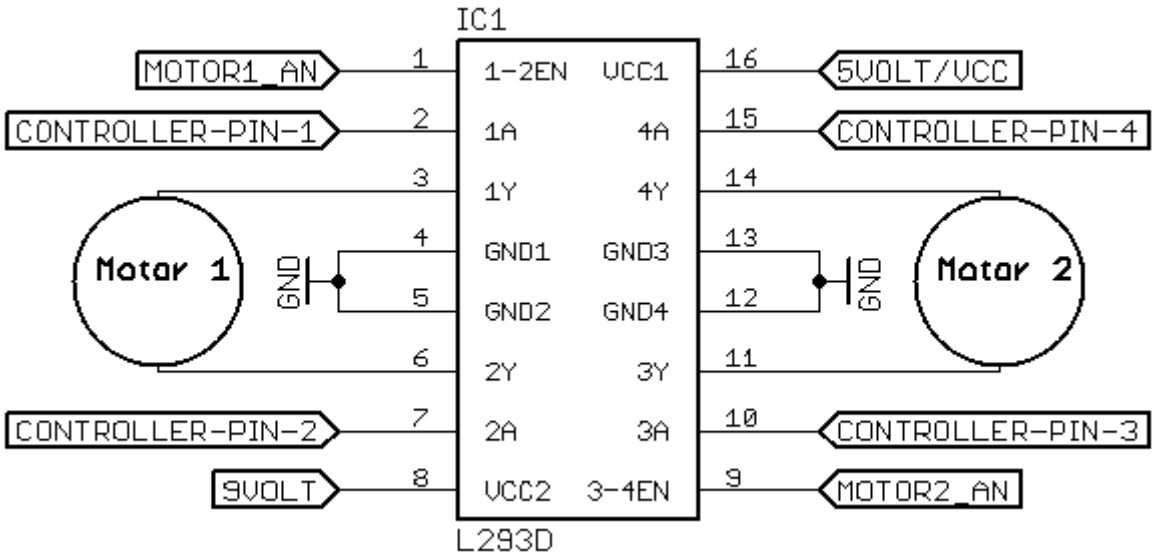
Aufgabe 8: Tonfreqenzen

Mit der for…next-Schleife sollen die Töne mit der Frequenz 20, 30 40 … bis 120 (also Schrittweite 10) für eine Dauer von jeweils 50 ausgegeben werden. Das Programm soll sich endlos wiederholen.

24. Steuern von Gleichstrommotoren mit dem Motortreiber-IC L293DNE

Mit dem Motortreiber IC L293DNE lassen sich zwei kleine Motoren (max. 600mA pro Motor) gleichzeitig steuern. Der L293DNE hat 16 Pins. An die Pins 1, 9 und 16 muss eine Versorgungsspannung von 4,5V bis 7V für den IC angelegt werden. An Pin 8 (Vcc 2) kommt die Spannungsversorgung für den Motor (4,5V bis 36V). Die Input-Pins 1 bis 4 sind am Mikrokontroller angeschlossen und steuern den IC. Die Output-Pins gehen zu den Motoren. Die Pins 4, 5, 12 und 13 sind an die Masse / Ground angeschlossen. Das folgende Schaubild soll die Anschlüsse nochmals verdeutlichen:

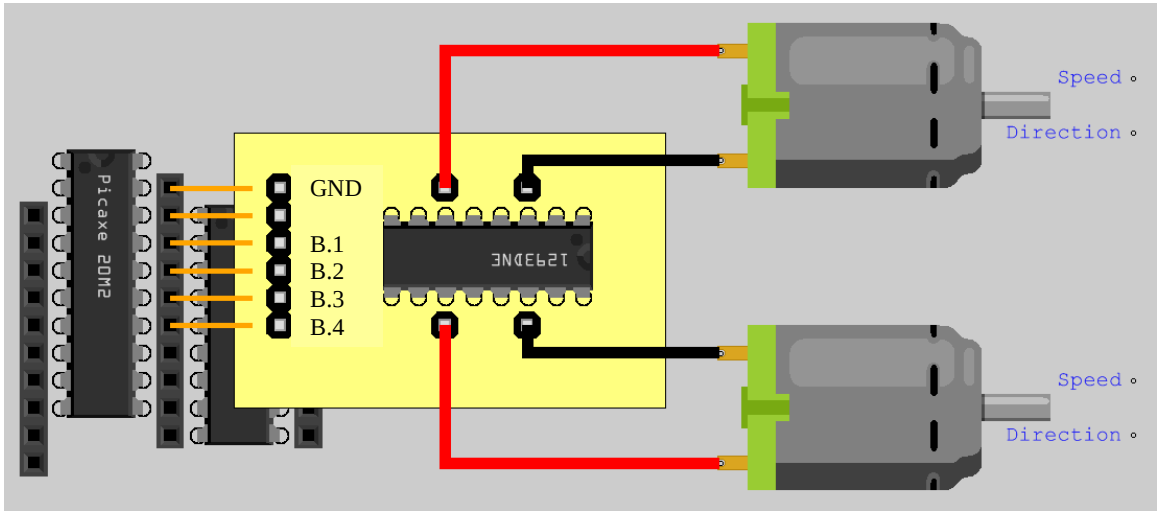




Quelle: http://www.wirmachenroboter.com/node/35987

25. Unser GSR-L293DNE-Aufsatz für Motoren

Unser GSR-L293DNE-Aufsatz lässt sich auf unser GSR-Picaxe-20M2-Basicboard in die Buchsenleiste zwischen dem Picaxe-20M2 und dem ULN2803A Verstärkerchip stecken. Die Stiftleiste des Aufsatzes sollte in die Buchsenleiste eingesteckt werden, sodass der obere (äußere) Masse-Pin und die Pins B.0 bis B.4 verbunden sind (siehe Schaubild).



Zum Steuern der Motoren des GSR-Kettenfahrzeugs brauchen wir die Pins B.1 bis B.4, die jeweils eine Verbindung zu den IC-Pins 1A, 2A, 3A und 4A haben. Schaltet man die Pins B.1 und B.4 auf High und die Pins B.2 und B.3 auf low, dann fährt das Fahrzeug vorwärts.

high B.1
low B.2
low B.3
high B.4

Das geht natürlich mit dem Befehl *let PinsB* schneller!

Let dirsB = %11111111 ; alle B-Pins werden als Ausgabe-Pins definiert
Let pinsB = %00010010 ; die Pins B.1 und B.4 sind high, alle anderen low

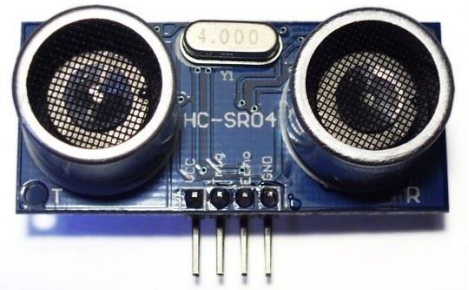
Hier eine „Wahrheitstafel“ für die Steuerung der beiden Motoren:

	B.7	B.6	B.5	**B.4**	**B.3**	**B.2**	**B.1**	B.0	**Fahrtrichtung:**
%	0	0	0	**1**	**0**	**0**	**1**	0	vorwärts
%	0	0	0	**0**	**1**	**1**	**0**	0	rückwärts
%	0	0	0	**1**	**0**	**1**	**0**	0	Drehung rechts
%	0	0	0	**0**	**1**	**0**	**1**	0	Drehung links
%	0	0	0	**0**	**0**	**0**	**0**	0	Motoren stoppen
%	0	0	0	**1**	**1**	**1**	**1**	0	Motoren stoppen

Aufgabe 9: geradeaus fahren
Das Kettenfahrzeug soll fünf Sekunden geradeaus fahren, 180° drehen und wieder fünf Sekunden geradeaus fahren.

Aufgabe 10: Quadrat abfahren
Das Kettenfahrzeug soll ein Quadrat abfahren.

26. Der Ultraschallsensor HC-SR04



Der Ultraschall Distanz Sensor HC-SR04 ist sehr günstig. Er benötigt eine Betriebsspannung von 5 Volt, hat eine Stromaufnahme von 15 mA und einen Öffnungswinkel <= 15 Grad. Er misst Entfernungen von 2 bis 450 cm.

Um den HC-SR04 benutzen zu können, muss die interne Geschwindigkeit des Picaxe-20M2-Mikrokontrollers mit dem Befehl ***setfreq M8*** von 4 Mhz auf 8 Mhz erhöht werden. Somit muss auch die Baudrate des Terminalfensters verdoppelt werden (von 4800 auf 9600).

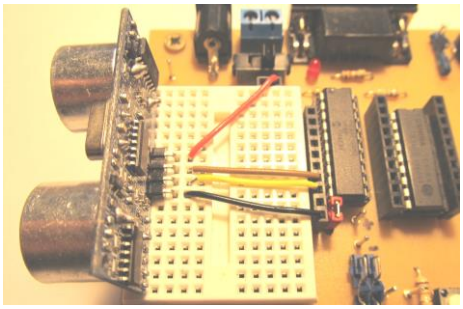
Mit dem Befehl **PULSOUT** sendet der HC-SR04 (bei 8Mhz) einen 5 µs langen Ultraschall-Impuls und misst die Zeit im µs, bis das Signal wieder zurückkommt. Die Schallgeschwindigkeit beträgt etwa 29µs/cm und der Ultraschall-Impuls muss den doppelten Weg laufen (hin und zurück). Um die Dauer des Echosignals von µs in cm umzurechnen, brauchen wir folgende Formel: **Distanz in cm = Dauer des Echosignals * 5 / 29 / 2**

Das nachfolgende Programm misst die Entfernung zum Ziel und gibt den Abstand in cm über das Terminalfenster aus.

```
setfreq M8                          ; Die Geschwindigkeit des 20M2 wird auf 8 Mhz erhöht
#terminal 9600                      ; öffnet das Terminalfenster mit 9600 Baud

symbol triggerpin = C.1             ; Pin C.1 wird in triggerpin umbenannt
symbol echopin    = C.0             ; Pin C.0 wird in echopin umbenannt
symbol echosignal = w0              ; Die Variable w0 wird in echosignal umbenannt
symbol distance = w1                ; Die Variable w1 wird in distance umbenannt

main:
 PULSOUT triggerpin, 1              ; Ein 5 µs langer Ultraschall-Impuls wird gesendet
 PULSIN echopin, 1 , echosignal     ; Das Signal am Echo-Pin wird in echosignal gespeichert
 let distance = echosignal * 5/29/2 ; Umwandlung von µs in cm
 sertxd ("Der Abstand beträgt ", #distance," cm", cr, lf)
 pause 500
goto main
```

27. Anschluss des HC-SR04 am GSR-Picaxe-20M2-Basicboard

Wie der HC-SR04 an das GSR-Picaxe-20M2-Basicboard angeschlossen wird, zeigt die folgende Tabelle:



HC-SR04	GSR-Picaxe-20M2-Basicboard
VCC	Pluspol (2-pol-Buchsenleiste)
Trig	Pin C.1
Echo	Pin C.0
GND	Minuspol

Aufgabe 11: Abstandswarner

Entwickle das Programm für einen akustischen Abstandswarner. Je kleiner der Abstand, desto höher soll der Ton am Soundgeber sein. Nur Abstände, die kleiner als 100 cm sind, sollen berücksichtigt werden. Gleichzeitig soll der Abstand in cm über das Terminalfenster angegeben werden.

28. Ansteuern des LCD-Displays AXE133

Mit dem LCD-Display AXE133 kann man zwei Zeilen mit je 16 Zeichen darstellen. Man benötigt dazu den Befehl **Serout**. Das Verbindungskabel des Displays muss an einen B-Pin des Picaxe-20M2 Mikrokontrollers angeschlossen werden. Auf der Rückseite des AXE133 kann man mit dem Poti den Kontrast einstellen. Es gibt verschiedene Befehle, die man zum Steuern des Displays verwenden kann:

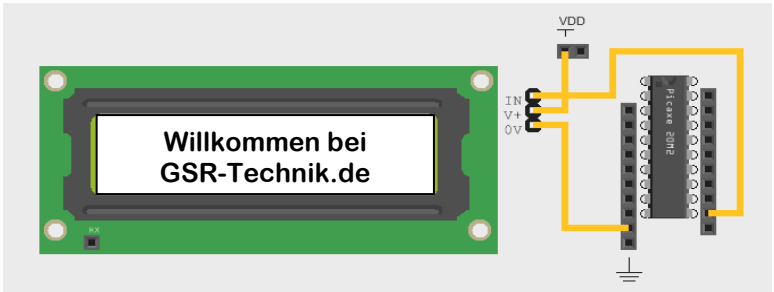
```
serout B.7, N2400, (254,1)      ;löscht das Display (benötigt dann: pause 30)
serout B.7, N2400, (254,128)    ;setzt den Cursor in die erste Zeile, Position 1
serout B.7,N2400,(254,192)      ;setzt den Cursor in die zweite Zeile, Position 1
```

Das nachfolgende Programm gibt den Text: **Hallo Welt** auf dem Display aus.

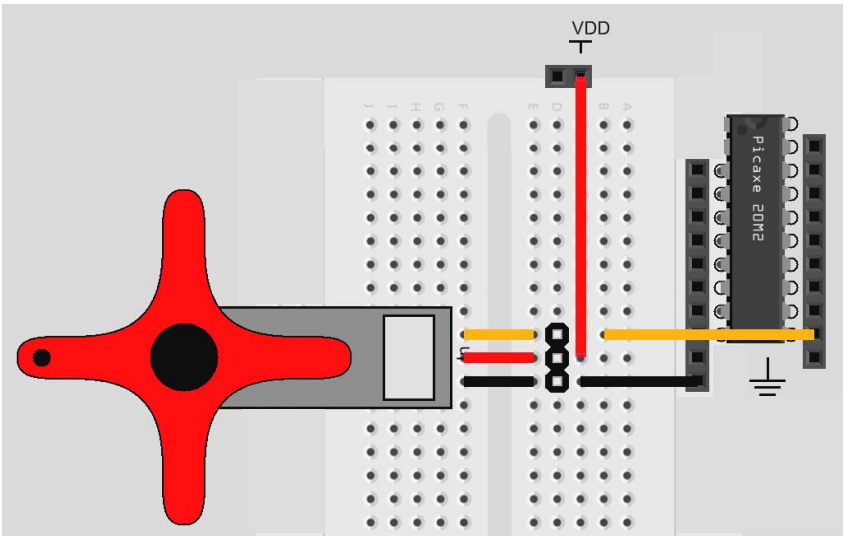
```
init: pause 500                 ;initialisiert das Display
main:                           ;Sprungmarke main
  serout B.7, N2400, (254,1)    ;löscht das Display
  pause 30                      ;benötigte Pause für den Löschbefehl
  serout B.7, N2400, (254,128)  ;setzt den Cursor in die erste Zeile, Position 1
  serout B.7, N2400, (" Hallo Welt")  ;gibt den Text Hallo Welt aus
```

Aufgabe 12: LCD-Display

In der ersten Zeile soll der Text: „Willkommen bei“ und in Zeile zwei der Text: „GSR-Technik.de“ mittenzentriert ausgegeben werden.



29. Steuern von Servos



Um Servos steuern zu können, muss man sie richtig an das GSR-Picaxe-20M2-Basicboard anschließen. Das orangene Kabel ist die Signalleitung, die an einen B-Pin angeschlossen werden muss. Rot ist die Spannungsversorgung (5V) und schwarz die Masse (0V).

Mit dem Befehl servo [Signalpin, Position (Zahl) zwischen 75 und 225] wird der Servo initialisiert und in eine bestimmte Position gebracht. Bewegungen des Servos sollten mit dem Befehl servopos [Pin, Variable zwischen 75 und 225] umgesetzt werden.

Hier ein Beispielprogramm. Der Servo startet in der Position 75 (Maximalstellung im Uhrzeigersinn). Dann wird der Servo in kleinen Schritten gegen den Uhrzeigersinn bewegt. Ist die Maximalposition (220) im Uhrzeigersinn erreicht, bewegt sich der Servo in die Anfangsposition zurück und das Programm wiederholt sich endlos.

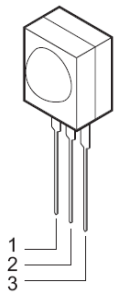
```
; Kabelanschluss: orange=Signal an Pin B.7 (muss ein B-Pin sein), rot=5V, braun=Masse
init:   servo B.7, 75          ; servo ist das Initialisierungskommando und setzt die
                               Signallänge der Servoimpulse auf 20ms, B.7 ist der Ausgangspin
                               und 75 die Startposition des Servos
b0=75                          ; die Variable b0 wird auf 75 gesetzt

main:   servopos B.7, b0       ; bewegt den Servo in kleinen Schritten gegen den Uhrzeiger
        pause 100
        b0=b0+5
        if b0 > 220 then gosub anfangswert
        goto main              ; zurück zur Sprungmarke main

        anfangswert:
        b0 = 75
return
```

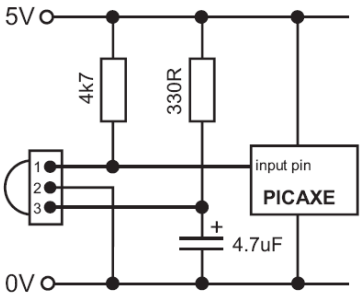
Aufgabe 13: Servo per Taster bewegen
Mit den beiden Tastern soll der Servo nach links und rechts bewegt werden können.

30. Steuerung des GSR-Picaxe-20M2-Basicboards per Infrarot

Unser Projektboard lässt sich mit dem Infrarotsensor LED020 von Picaxe steuern. Dazu wird eine (Universal)-Fernbedienung oder ein Smartphone (inkl. IR-Diode mit passender App) benötigt, die Sony-Fernseher steuern kann. Unsere Fernbedienung e+p FB13 muss zuerst eingestellt werden.

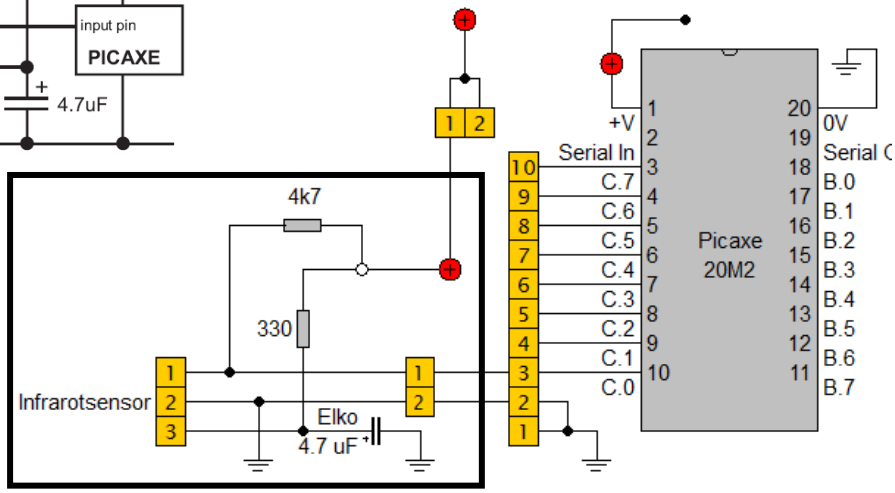
1. Taste SET und TV gleichzeitig drücken.
2. Code 030 oder 044 eingeben.





Das linke Schaltbild zeigt, wie der Infrarotsensor LED020 an unser Interface angeschlossen werden muss.

Das nebenstehende Schaltbild zeigt unsere IR-Platine inkl. Anschluss an unser GSR-Picaxe-20M2-Basicboard.



Mit dem Befehl **irin** können die Signale der Fernbedienung über den Infrarotsensor empfangen werden. In dem kurzen Programmbeispiel ist der Pin 1 des Infrarotsensors am Pin C.0 des Picaxe-20M2 angeschlossen. Der empfangene „Wert“ wird in der Variablen b1 gespeichert und über **das Terminalfenster** kann man den Wert angezeigt bekommen.

```
#terminal 4800
main:
      irin [50], C.0,b0
      sertxd ("Der Wert von b0 ist ",#b0, cr, lf)
      pause 10
goto main
```

Taste	Wert
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
0	9

Die Tabelle rechts zeigt, bei welchem Tastendruck der Fernbedienung welche Werte über den irin-Befehl „erzeugt“ werden. Im erscheinenden Terminalfenster muss eventuell noch der passende COM-Port eingestellt werden.

Beim Drücken der Taste 1 auf der Fernbedienung wird die LED 1 eingeschaltet.

```
main:
  irin C.0, b1          ; wartet auf ein Signal an Pin C.0; der Wert wird dann in b1 gespeichert
  if b1 = 0 then led1   ; wird die Taste 1 gedrückt (b1 = 0) dann gehe ins Unterprogramm led1
goto main               ; Sprung zum Label main

led1:                   ; Anfang des Unterprogramms led1
  high B.0              ; der Pin B.0 wird high
goto main               ; Sprung zum Label main
```

Der Befehl **irin** kann noch erweitert werden: **IRIN [timeout, address], pin, variable**
timeout = eine Variable oder ein Wert, der eine Pause in Millisekunden erzeugt.
adress = wird keine Taste gedrückt, springt das Programm in das Unterprogramm **ledoff**.

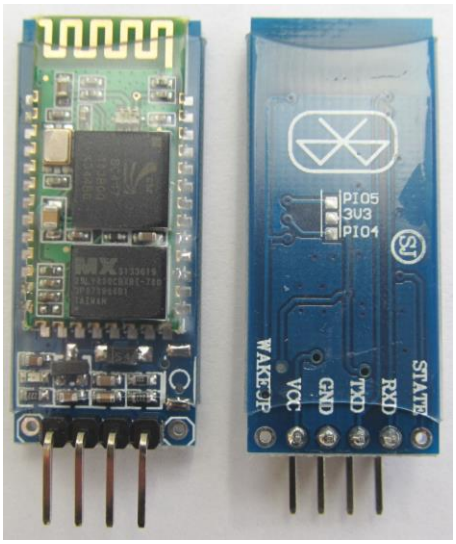
```
main:
  irin [100, ledoff], C.0, b1
  if b1 = 0 then led1
 goto main

led1:
  high B.0
goto main

ledoff:
  low B.0
goto main
```

Beim linken Programmbeispiel leuchtet die LED 1, wenn die Taste 1 gedrückt wird. Wird keine Taste gedrückt, dann folgt ein Sprung in das Unterprogramm (Label) **ledoff**.

Achtung!!! Die Schaltung mit dem Infrarotsender muss an den Picaxe-20M2 angeschlossen sein, wenn ein neues Programm aufgespielt werden soll. Sonst kann die Fehlermeldung: **Hardware nicht gefunden …** kommen. Bei Fehlern das Programmierkabel austauschen.

Aufgabe 14: LEDs mit einer Fernbedienung einschalten
Mit den Tasten 1 bis 8 auf der Fernbedienung sollen die LEDs 1 bis 8 auf den GSR-Picaxe-20M2-Basicboard eingeschaltet werden können. Wird keine Taste gedrückt, dann soll auch keine LED leuchten.

31. Steuern mit dem HC-06 Bluetooth-Controller und der App: GSR Drive



Den HC-06 kann man mit backplane (wichtig) über ebay, banggood, aliexpress oder andere Portale erwerben. Das eigentliche HC-06 Bluetooth Modul ist mit einer blauen Adapterplatte (backplane) verbunden und kann wegen eines Spannungsreglers mit 3,3V bis 6V Spannung versorgt werden. Der HC-05 kann als Master (senden und empfangen) und der HC-06 nur als Slave (empfangen) arbeiten.

Der serielle Bluetooth-Controller HC-06 hat mehrere Anschlüsse, die mit dem **GSR-Picaxe-20M2-Basicboard** verbunden werden müssen. VCC an den Pluspol (2-Pin-Buchsenleiste), GND an Minuspol (Buchse 1 der 6-Pol-Buchsenleiste), TXD an Pin C.O, RXD (frei). Das Blinken der LED bedeutet, dass der HC-06 betriebsbereit ist aber noch nicht mit einem anderen Gerät (Handy, PC) verbunden ist.

Dann muss der HC-06 mit dem Handy gekoppelt werden. Einfach nach neuen Bluetooth Geräten suchen. Der serielle Bluetooth Controller wird als HC-06 gefunden. Um sich koppeln zu können, muss noch die PIN (Standard ist 1234) eingegeben werden.

Zum Steuern des **GSR-Picaxe-20M2-Basicboards** kann man die kostenlose App: **GSR Drive** im Google Play Store herunterladen.

Hat man die App gestartet, muss man den Button „Mit BT-Geräten verbinden“ drücken. Anschließend den HC-06 auswählen und los geht’s…

Hier das einfache Steuerprogramm für den Picaxe-20M2:

; Dateiname: 20M2 GSR Drive HC-06 Kettenfahrzeug.BAS

```
let dirsB=%11111111      ; alle B-Pins werden Ausgabepins
let pinsB=%00000000      ; alle B-Pins werden auf 0V gesetzt (low)
setfreq M8               ; Die Geschwindigkeit des 20M2 wird auf 8 Mhz erhöht

main:                    ; Start des Hauptprogramms
 serin C.0,N9600_8,b1    ; C.0 empfängt über HC-06 Signal und speichert es in b1
 If b1 = 127  then vw    ; wenn Vorwärtsbutton dann gehe ins Unterprogramm vw
 If b1 = 63   then re    ; wenn Rechtsbutton dann gehe ins Unterprogramm re
 If b1 = 126  then rw    ; wenn Rückwärtsbutton dann gehe ins Unterprogramm rw
 If b1 = 31   then li    ; wenn Linksbutton dann gehe ins Unterprogramm li
 If b1 = 125  then stehen ; wenn Stopbutton dann gehe ins Unterprogramm stehen
goto main

vw:
let pinsB=%00010010      ; B1 und B4 high => L293D Pin 1 und Pin 4 high => Vorwärts
goto main

re:
let pinsB=%00010100      ; B2 und B4 high =>L293D Pin 2 und Pin 4 high => Rechtsdrehung
goto main

rw:
let pinsB=%00001100      ; B2 und B3 high => L293D Pin 2 und Pin 3 high => Rückwärts
goto main

li:
let pinsB=%00001010      ; B1 und B3 high =>L293D Pin 1 und Pin 3 high => Linksdrehung
goto main

stehen:
let pinsB=%00000000      ; alle Pins low =>L293D werden alle Pins low => Stop
goto main
```

32. Mehrere Programme auf dem Interface ausführen

Möchte man mehrere Programme auf den Picaxe aufspielen und diese über die Taster (C.7 und C.5) ansteuern, dann braucht man ein Hauptprogramm und mehrere Unterprogramme.

Man kann die Funktion des Programmes gut mit einem Fahrstuhl und den verschiedenen Stockwerken vergleichen. Die Variable b0 wird benötigt, um zu den anderen Unterprogrammen (Stockwerken) zu navigieren. Da wir mit dem Prog1 starten wollen, setzen wir zuerst die Variable b0 auf den Wert 1. Im Hauptprogramm wird immer überprüft, ob einer der Taster (C.7 nach oben, C.5 nach unten) gedrückt wurde. Wird C.7 gedrückt, dann wird über das Unterprogramm countplus der Wert von b0 um 1 erhöht und somit gelangt man wegen des Befehls `if b0 = 2 then Prog2` in das Unterprogramm Prog2.

```
let dirsB=%11111111
let b0 = 1
main:
      If pinC.7 = 1 then countplus
      If pinC.5 = 1 then countminus
      if b0 = 1 then Prog1
      if b0 = 2 then Prog2
      if b0 = 3 then Prog3
goto main

countplus:
      b0=b0+1
      pause 500
      if b0>3 then b0_ist_3
goto main

countminus:
      b0=b0-1
      pause 500
      if b0<1 then b0_ist_1
goto main

b0_ist_1:
      b0=1
goto main

b0_ist_3:
      b0=3
goto main

Prog1:
      let pinsB=%00000001
goto main

Prog2:
      let pinsB=%00000010
goto main

Prog3:
      let pinsB=%00000100
      If pinC.6 = 1 then sound_Ghostbusters
goto main

sound_Ghostbusters:
      tune 7, 7,($48,$48,$53,$43,$6C,$51,$4A,$EC,$6C,$4A,$4A,$4A,$4A,$4A)
goto main
```