NAVODILA ZA UPORABO
Učni paket Conrad Components Arduino™ razumeti in uporabljati 10174 od 14. leta starosti naprej

Kataloška št.: 138 41 45

Kazalo

1. Zgoščenka za učni paket

Učni paket ima priloženo zgoščenko, ki vsebuje različne programe in primere. Zgoščenka vam olajša delo s to knjigo. Primere, ki so natisnjeni tukaj, lahko najdete tudi na zgoščenki.

1. 1 Vsebina zgoščenke

- Arduino™ razvojno okolje (IDE)
- Primer programske kode za učni paket

1. 2 GPL (Splošna javna licenca)

Svoje programe lahko izmenjujete z ljudmi preko spleta. Programski primeri so predmet odprtokodne licence GPL (Splošna javna licenca). Zato lahko spreminjate, objavljate ali ponudite programe drugim uporabnikom pod pogoji GPL, med drugim da vsebino programov na voljo tudi GPL.

1. 3 Sistemske zahteve

Za Windows 7/8/8.1, 32 in 64 bit ali višje, Linux 32 in 64 bit, Mac OS X, pogon za zgoščenke, Java.

Opomba
Učni paket vsebuje VB. NET programe, ki delujejo samo v okolju Windows. Osnovni Arduino™ programi za te poskuse delujejo tudi v drugih operacijskih sistemih. Samo .NET-PC programi potrebujejo operacijski sistem Windows z .NET ogrodjem za poskuse.

1. 4 Posodobitve in podpora

Arduino™ je predmet nenehnega razvoja. Posodobitve si lahko brezplačno naložite s spletne strani http://arduino.cc.

2. Vsebina učnega paketa

Učni paket vsebuje vse dele, ki jih potrebujete za poskuse. Prosimo preverite, če so deli poškodovani, preden začnete s poskusi.

Opomba
Arduino™-UNO-micro krmilnik PCD ni vključen v dostavo. Ker je to učni paket namenjen naprednim Arduino™ programerjem, morate na delovnem mestu že imeti Arduino™ UNO ali MEGA. Plošče so na voljo po nizki ceni pri Conrad Electronics, v franšiznih prodajalnah ali drugih spletnih trgovinah.

Seznam delov:
1x maketna plošča, 1x LCD 16 x 2 moder, 1 x pin trak 16 pinov za spajkanje, 1 x tipka, 1 x NTC 72, 1 x foto tranzistor, 1 x 10 kΩ, 1 x 330 Ω, 14 x skakalci.

2. 1 Informacije o varnosti

Arduino™ PCB in zaslon sta večinoma zaščitena pred napakami, tako da je skoraj nemogoče poškodovati računalnik. Priključki USB vtičnice niso izolirani na spodnji strani PCB. Če namestite PCD na kovinski prevodnik lahko pride do višjega el. toka, kar lahko poškoduje računalnik in PCB.

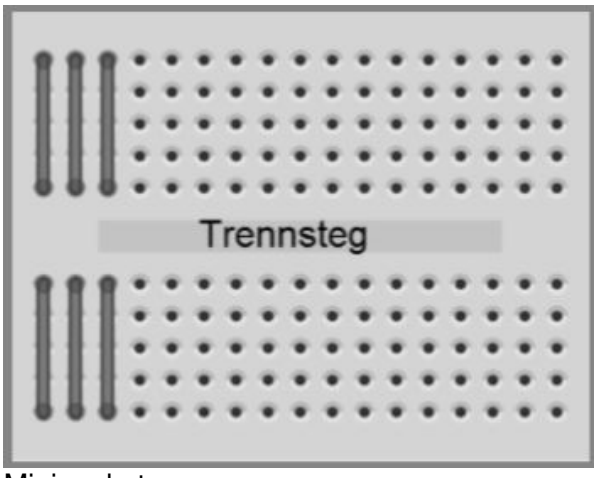
Upoštevajte naslednja varnostna pravila!
- Izogibajte se kovinskim predmetom pod PCB ali izolaciji celotnega spodnjega dela z neprevodno zaščitno ploščo ali izolirnim trakom.
- Držite glavne enote, kot so viri prenapetosti ali živi prevodniki z več kot 5 volti (V), stran od eksperimentalnega PCD.

3. Sestavni deli in njihove funkcije

Sestavni deli učnega paketa so na kratko predstavljeni tukaj. Naslednji poskusi bodo ponudili praktične izkušnje s tehnologijo vezja v elektroniki.

3. 1 Maketna plošča

Na maketni plošči lahko nastavite vezje brez da bi bilo potrebno spajkanje. Naša maketna plošča je sestavljena iz 15 kolon in 5 vrst. Kolone, vsaka s 5 kontakti, so povezane med seboj v seriji (od zgoraj navzdol, glejte sliko). Ločilni most v sredini maketne plošče označuje da ni povezave z drugimi polji 17 kolon in 5 vrst. Kot koristno se je izkazalo odvitje povezovalnih žic najprej po diagonali, da se ustvari nekakšen klin na koncih žic. To omogoča lažjo priključitev delov v maketno ploščo. Težko je priključiti dele, najbolje je uporabiti majhne visoko natančne mehanske klešče za potiskanje delov v maketno ploščo in pri tem uporabiti več pritiska.



Mini maketa

3. 2 Skakalci

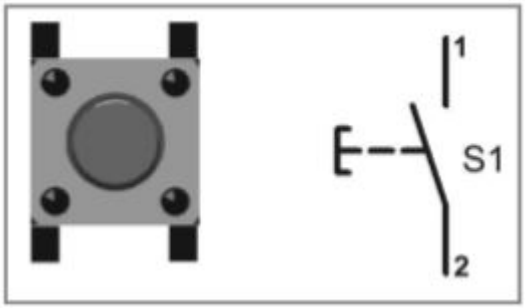
Učni paket vsebuje več posebej prilagojenih skakalcev. Uporabljajo se za povezavo med maketno ploščo in Arduino™-PCB. Skakalci imajo majhne pine na obeh koncih, ki jih lahko enostavno potisnete v Arduino™-PCB. Vendar bodite kljub temu previdni, da po nesreči ne odlomite ali zvijete pina!

Skakalci.

3. 3 Tipke

Tipka ima podobno funkcijo kot stikalo. Stikala že poznate iz vaših domov, kjer z njim vklapljate ali izklapljate luči. Ko potisnete stikalo navzdol, se luč vklopi. Ko potisnete stikalo navzgor, se luč ponovno ugasne. Stikalo ostane v svojem položaju. V tem se razlikuje od tipk. Ko pritisnemo tipko, je vezje sklenjeno. Sklenjeno bo ostalo dokler ponovno ne pritisnemo tipke. Ko tipko spustimo, se bo vezje ponovno razklenilo in luč se bo ugasnila. Tipka se bo samodejno vrnila v svoje prvotno stanje ali stanje počivanja, s pomočjo mehanizma ki ga vsebuje.

Obstajajo tipke ki sklenejo vezje ko so aktivirane in takšne ki vezje razklenejo. Tipke, ki vezje sklenejo so pogosto imenovani »N.O.« (normalno odprte) in tisti ki vezje razklenejo »N.C.« (normalno zaprte). Primer prikazuje tipko, ki je priložena učnemu paketu. Ob pritisku tipka sklene vezje in tok lahko teče skozi iz stika 1 na 2. Druga dva kontakta sta medsebojno povezana.



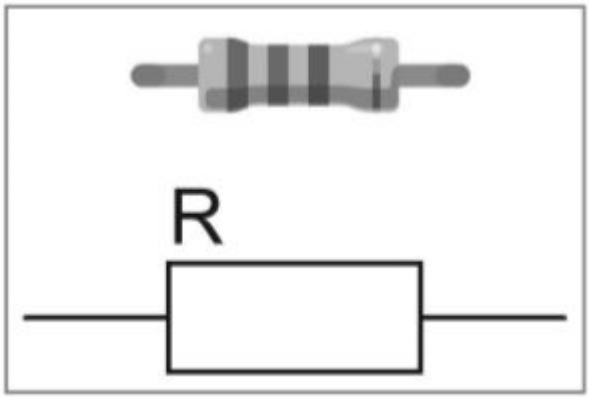
Tipka.

3. 4 Uporniki

Uporniki so potrebni za omejitev toka in da nastavijo točke delovanja ali kot delilci napetosti v električnih vezjih. Enota za električno upornost je Ohm (Ω). Predpona kilo (k, tisoč) ali mega (M, milijon) omogočata okrajšavo velikih upornosti.

1 kΩ = 1000 Ω
10 kΩ = 10.000 Ω
100 kΩ = 100.000 Ω
1 MΩ = 1.000.000 Ω
10 MΩ = 10.000.000 Ω

V diagramih vezja je simbol Ω po navadi izpuščen in 1 kΩ je ponavadi okrajšan na 1 k. Vrednost upornika je po navadi označena z barvo. Običajno so trije barvni obroči in dodaten četrti obroč, ki označuje natančnost upornika. Kovinski uporniki imajo po navadi toleranco 1 %. To je označeno z rjavim tolerančnim obročem, ki je širši od ostalih štirih obročev. To zmanjšuje možnost napake za normalne vrednosti obročev s pomenom »1«.

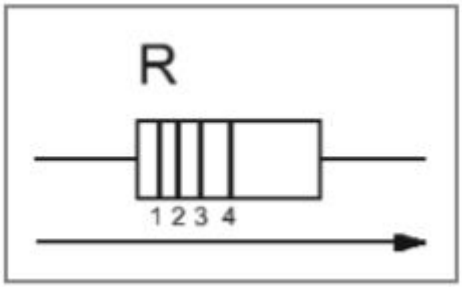


Upornik z označbami.

Uporniki s toleranco +/-5 % so na voljo v vrednosti E24 različice, kjer vsaka dekada vsebuje 24 vrednosti na približno enaki razdalji do sosednje vrednosti.

Uporniki E24 standardne različice so sledeči:
1,0 / 1,1 / 1,2 / 1,3 / 1,5 / 1,6 / 1,8 / 2,0 / 2,2 / 2,4 / 2,7 / 3,0 / 3,3 / 3,6 / 3,9 / 4,3 / 4,7 / 5,1 / 5,6 / 6,2 / 6,8 / 7,5 / 8,2 / 9,1

Barvna oznaka se začne brati z obroča ki je bližje robu upornika. Prva dva obroča predstavljata dve številki, tretji obroč je množitelj vrednosti upora v Ohm. Četrti predstavlja toleranco.



Smer branja vrednosti na uporniku.

Upornik z barvami rumena, vijolična, rjava in zlata, ima vrednost 470 Ω s toleranco 5 %. Poskusite sedaj prepoznati upornike priloženemu učnemu paketu.

Barva	Obroč 1	Obroč 2	Obroč 3 (faktor)	Obroč 4 (toleranca)
Srebrna	-	-	$1x10^{-2}$=0,01 Ω	+/-10 %
Zlata	-	-	$1x10^{-1}$=0,1 Ω	+/-5 %
Črna	0	0	$1x10^{0}$=1 Ω	-
Rjava	1	1	$1x10^{1}$=10 Ω	+/-1 %
Rdeča	2	2	$1x10^{2}$=100 Ω	+/-2 %
Oranžna	3	3	$1x10^{3}$=1 kΩ	-
Rumena	4	4	$1x10^{4}$=10 kΩ	-
Zelena	5	5	$1x10^{5}$=100 kΩ	+/-0,5 %
Modra	6	6	$1x10^{6}$=1 MΩ	+/-0,25 %
Vijolična	7	7	$1x10^{7}$=10 MΩ	+/-0,1 %
Siva	8	8	$1x10^{8}$=100 MΩ	-
Bela	9	9	$1x10^{9}$=1000 MΩ	

Tabela za upore s štirimi barvnimi obroči.

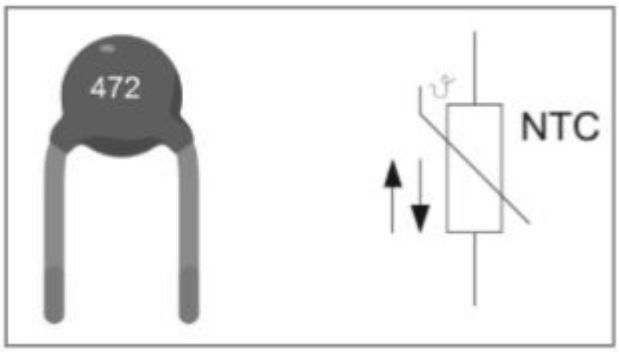
Nasvet
Vstavljanje iskalnega izraza »Resistance code calculator« (računanje vrednosti upornikov) na spletu, vam bo našlo računala za računanje, na primer:
http://www.ab-tools.com/de/software/resistancesrechner/
ali
http://www.dieelektronikerseite.de/Tools/resistancesrec hner.htm.

V primeru spodaj je prikazana tudi starejša različica. Ta merilnik upora ali vitrometer dovoljuje hitro določitev upora, brez uporabe računalnika, samo z vrtenjem koles. Na ta način si boste hitreje zapomnili vrednosti barvnih oznak v računalniški različici.

Merilni upora (vitrometer).

3. 5 Senzor temperature

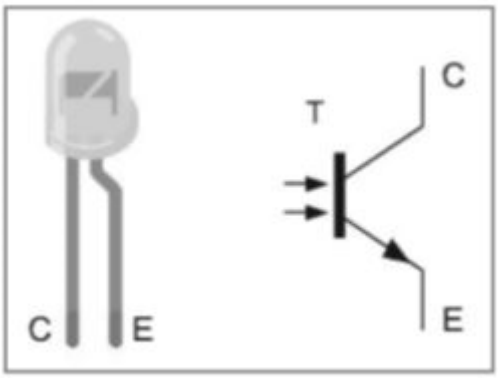
Za spremljanje temperature je priložen senzor temperature. Oznaka NTC pomeni (negativni temperaturni koeficient) in pomeni, da bo upornost padla z naraščanjem temperature. To je vroč prevodnik. NTC učnem paketu ima vrednost upora 4,7 kΩ pri 25 °C / 298,15 K (Kelvin).



NTC senzor temperature in skica kako se priključi.

3. 6 Fototranzistor

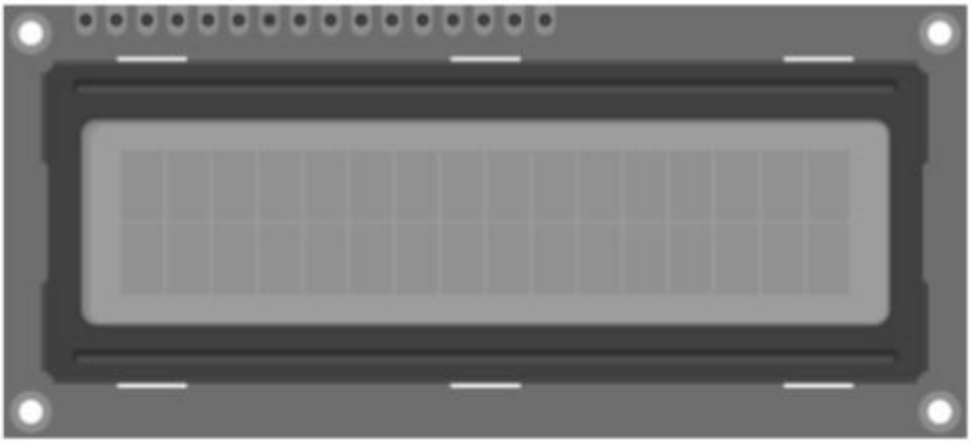
Za določitev svetlosti uporablja sodobna elektronika pogosto fototranzistorje. Učni paket vsebuje dele, ki izgledajo zelo podobno kot dioda ki oddaja belo barvo, vendar je foto tranzistor. Izgleda drugače kot normalni bipolarni tranzistor, vendar nima tudi nobene povezave. Podstavek, to je vhod za normalni trazistor, ki je odgovoren za nadzor toka med prevodnikom in oddajnikom, kjer svetloba pada v ohišje na foto tranzistor. Svetloba tam pade na silikon in ustvari manjši ali večji tok med prevodnikom in oddajnikom glede na moč svetlobe.



Fototranzistor in njegova priključitev.

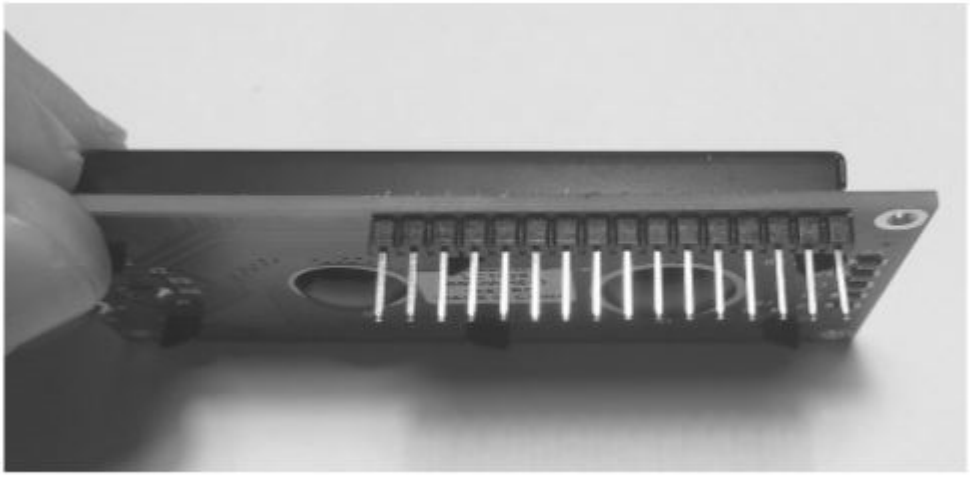
3. 7 LCD zaslon

Glavni del tega učnega paketa je modro beli LCD. Učni paket vsebuje zaslon z dvema vrstama 16 kolon z 5 x 8 pikami vsak. Ta zaslona lahko sedaj kupite tudi ločeno v katerikoli trgovini z elektroniko ali spletni trgovini, za samo nekaj eurov. Na voljo so v zeleni, modri, rumenorjavi, rumeni in tudi v posebnih barvah, ki pa so po navadi dražje. V našem primeru je naložen moder LCD. Naložen LCD krmilnik je KS0066/HD44780, ki ga izdeluje več proizvajalcev – več o tem kasneje.



LCD zaslon z dvema vrstama 16 kolon.

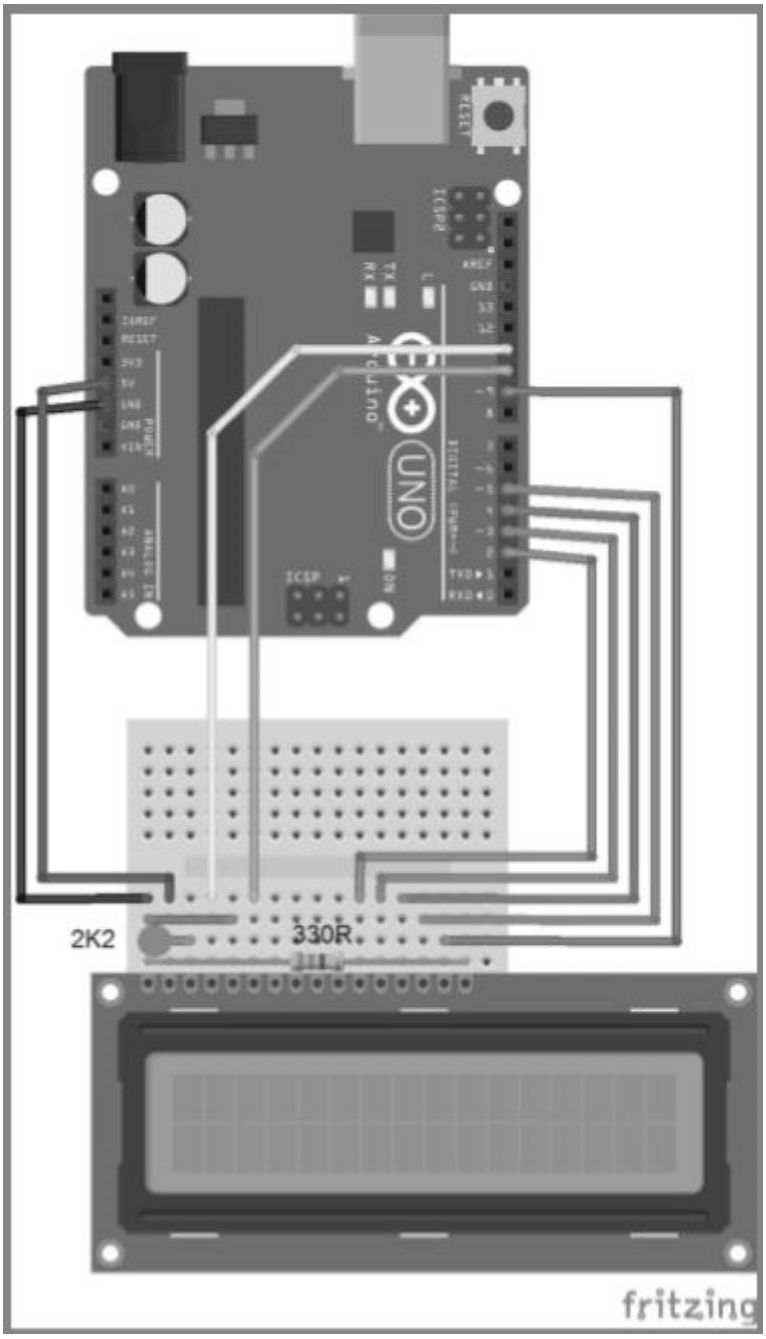
Preden lahko uporabite LCD za vaše poskuse morate spajkati priložen 16 pinski trak v stike na LCD. Zato vtaknite pinski trak s kratkimi stiki v LCD od zadaj in spajkajte najprej samo en stik. Na ta način lahko poravnate pinski trak pod kotom 90°. Ko je trak poravnan, lahko spajkate tudi ostale pine. Če še nimate pištole za spajkanje, si dobite stroškovno učinkovito ročno pištolo za spajkanje z izhodom med 20 in 30 W in električni spajkalnik. Ta naložba se vam bo izplačala, ko se boste ukvarjali z Arduino™ in drugo elektroniko.



Pinski trak po koncu spajkanja.

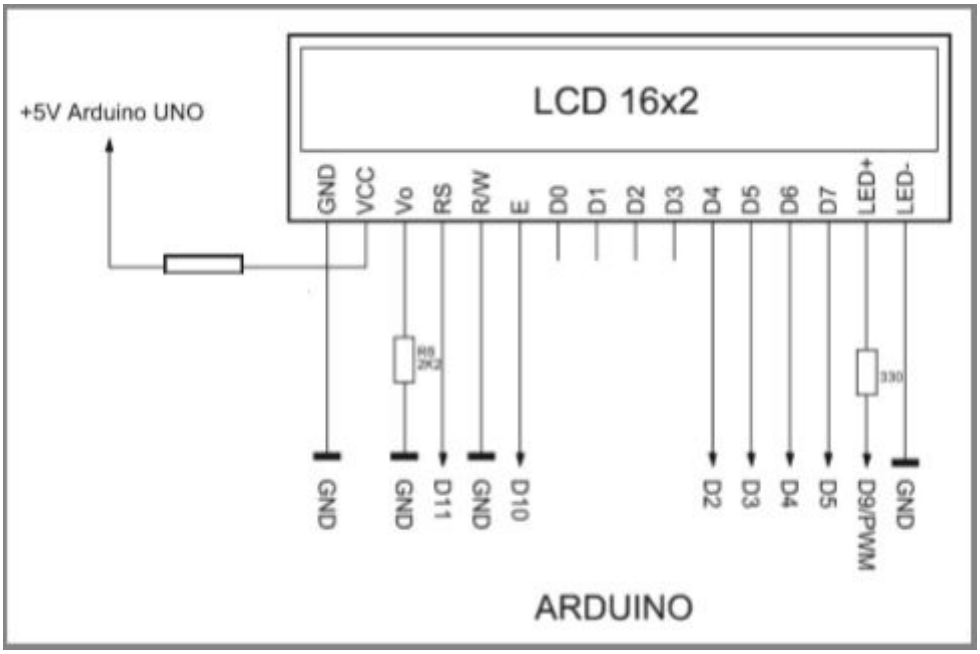
4. Test prvih funkcij

Povežite poskus kot je prikazano na sliki spodaj. Bodite pazljivi da ne zvijete ali zlomite pinov skakalcev.



Eksperimentalna povezava z LCD.

Na koncu natančno preverite vezje, da se izognete morebitnim poškodbam delov.



Priključni načrt za povezavo z LCD.

Informacije
Če prvič delate z Arduino™ morate najprej naložiti Arduino™ razvojno okolje. Najdete ga na uradni Arduino™ spletni strani http://www.arduino.cc.
Tukaj lahko izberete vaš operacijski sistem in določite ali želite uporabiti program za nameščanje ali ZIP različico. V programski različici za nameščanje, namestite Arduino™ kot normalen program. V ZIP različici ne potrebujete nameščanja. Razširite ZIP datoteko in jo shranite na želeno mesto na vašem računalniku. To je koristno, če želite na primer shraniti Arduino™ na USB ključek in ga odnesti s seboj.

Pozor!
Arduino shranite samo tam kjer imate vse pravice za branje, pisanje itd.!

Za prvi test funkcij, naložite sledeč program na Arduino™ ploščo. Programske primere lahko najdete na priloženi zgoščenki v mapi Examples.

Program bo ustvaril tekst in prikazal se bo nekakšen števec na LCD, kar je zelo primerno za prvi test funkcij, saj preveri če vse deluje pravilno, ker je zelo majhen in dobro strukturiran.

Primer kode: LCD

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 void setup()
009 {
010     // LED-Hintergrundbeleuchtung
011     analogWrite(9, 150);
012
013     // LCD Ausgabe
014     lcd.begin(16, 2);
015     lcd.setCursor(0, 0);
016     lcd.print("**ARDUINO LCD**");
017     lcd.setCursor(0, 1);
018     lcd.print("CNT:");
019 }
020
021 void loop()
022 {
023     lcd.setCursor(5, 1);
024     lcd.print(millis() / 1000);
025 }
```

V prvi vrstici programa lahko vidite, da je za delovanje LCD potrebna integracija Arduino™ knjižnice z imenom LiquidCrystal.h. Vsebuje bolj kompleksno kodo, ki je potrebna za nadzor zaslona. Lahko pogledate v mapo Arduino™, ki se nahaja v Arduino\libraries\LiquidCrystal, in pregledate LiquidCrystal.h in LiquidCrystal.cpp datoteki, da dobite idejo funkcij knjižnice. Priporočamo da za odprtje teh datotek uporabite program Notepad++. Lahko si ga brezplačno naložite s spletne strani http://www.notepad-plus-plus.org.
Videli boste da bo ta knjižnica opravila veliko dela namesto vas. V našem Arduino™ programu, smo integrirali samo naslovno datoteko LiquidCrystal.h. Arduino™ bo sedaj samodejno poznal vse LCD funkcije.
V naslednji vrstici seznanimo Arduino™ kateri pini na LCD so priključeni na Arduino™-PCD.

```
001 LiquidCrystal lcd(11, 10, 2, 3, 4, 5)
```

Naslednji ukaz določi svetlost osvetlitve zaslona. LED LCD je povezana z Arduino™ digitalnim/PWM-vrati P9. Lahko so uporabljena kot preprosta digitalna vrata ali PWM (vrata z modularno širino pulza). V našem testu je uporabljen kot PWM-vrata. Tako lahko postopoma nastavljamo svetlost osvetlitve zaslona. Vrednost 150 naredi LED dovolj svetel. Če je izbrana nižja PWM vrednost, bo LED temnejša. Poskusite spreminjati vrednost in opazujte kaj se dogaja.

```
001 analogWrite(9, 150)
```

Namestitev je skoraj končana. Sedaj morate nastaviti koliko kolon in vrstic ima LCD: 16 kolon/posamezni znaki in 2 vrstici.

```
001 lcd.begin(16, 2)
```

Osnovno nameščanje je sedaj končano. Sedaj lahko uporabimo lcd.setCursor, za določitev položaja kazalke in tako teksta za izpis.

```
001 lcd.setCursor(0, 0)
```

Prvi parameter določa položaj znotraj kolone, to je 0 do 15 v tem primeru. Drugi parameter določa številko vrstice, to je od 0 do 1.
Sedaj lahko izpišemo tekst na tej določeni lokaciji na LCD z ukazom commandlcd.print.

```
001 lcd.print("**ARDUINO LCD**")
```

Lahko vidite da je vedno potrebno napisati ».lcd« preden zapišete pravo funkcijo LCD izpisa. To določi da uporabimo razred lcd, ki smo ga integrirali z #include <LiquidCrystal.h>. Sedaj je Arduino™ seznanjen od kod prihaja klic in kateri razred je odgovoren za »prevajanje«, ali kot to imenujejo strokovnjaki »zbiranje« (compiling).
Če ste se kdaj prej ukvarjali s programskim jezikom C++, boste prepoznali, s končnico *.cpp, da so to C** razredi. Arduino™ v osnovi temelji v C++. To je dober način za programiranje lastnih razredov ali knjižnic in jih ponuditi drugim Arduino™ uporabnikom.
Po tej kratki C++ ekskurziji, se vrnimo nazaj na naš primer. Do sedaj smo bili vedno znotraj funkcije Setup(), ki se izvede vsaj enkrat ko zaženete program in se večinoma uporablja za začetne nastavitve. Znotraj nje lahko nastavimo spremenljivke v naprej, preden se zažene dejanski program in nastavimo v naprej tudi strojno opremo.
Sledeče funkcija Loop() je neskončna zanka, ki se ne konča. To je hkrati glavna Arduino™ zanka za naš program. Tukaj kličemo izvajanje v milisekundah v vsakem izvajanje s funkcijo millis(). Delitev s 1000 vodi v izpis v sekundah. Na LCD zaslonu bomo prikazali izvajanje v sekundah.

```
001 lcd.setCursor(5, 1)
002 lcd.print(millis() / 1000)
```

Ker je funkcija millis() zelo zanimiva, bomo poskusili še en eksperiment, preden se ukvarjamo z LCD zaslonom bolj podrobno, ker lahko funkcijo millis() uporabimo za izmero časa izvajanja programa, kot je prikazano v sledečem primeru.

Primer kode: TIME_DIFF

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 long time_diff, diff;
009
010 void setup()
011 {
012     // LED-Hintergrundbeleuchtung
013     analogWrite(9, 150);
014
015     // LCD Ausgabe
016     lcd.begin(16, 2);
017     lcd.setCursor(0, 0);
018     lcd.print("**ARDUINO LCD**");
019     lcd.setCursor(0, 1);
020     lcd.print("TIME DIFF: ");
021 }
022
023 void loop()
024 {
025     diff = millis();
026
027     // Mein Programm Start
028
029     lcd.setCursor(11, 1);
030     lcd.print(diff - time_diff);
031     delay(100);
032
033     // Mein Programm Ende
034     time_diff = diff;
035 }
```

Ta primer kode prikazuje kako določiti čas izvajanja programa. Zato beremo trenutno stanje števca funkcije millis() v vsakem novem izvajanju programa in odštejemo zadnjo števko, ki je bila shranjena, na koncu programa. Ukaz Delay(100) na programu simulira daljši čas izvajanja. Vaša programska koda mora biti med »// My program start and // My program end«, da se določi dovoljen čas izvajanja vaše kode.
Prvi poskus in s tem tudi test funkcij je s tem končan. Pustite vezje povezano tako kot je. Potrebovali in razširili ga boste v naslednjih primerih. V naslednjem poglavju se boste naučili več o LCD zaslonu in njegovih lastnostih.

5. Nastavitev in delovanje LCD zaslona

LCD se v glavnem uporabljajo v mnogih elektronskih napravah, kot so zabavna elektronika, merilnih napravah, mobilnih telefonih, digitalnih urah in žepnih računalnikih. Head-up-Displays in video projektorji tudi uporabljajo to tehnologijo. Sledeči primeri prikazujejo LCD ki je del učnega paketa. To je standarden 5 x 8 matrični zaslon z 2 vrstama s 16 znaki v vsaki.

16 x 2 LCD zaslon v uporabi.

LCD je na splošno sestavljen iz 2 individualnih steklenih panelov in posebne tekočine med njima. Posebna lastnost te tekočine je, da obrne polarizacijski nivo svetlobe. Na ta učinek vpliva uporaba električnega polja. Na te dve plošči je tako nanesena tanka plast kovine. Da dobite polarizirano svetlobo, polariziran film obtiči na zgornji stekleni ploščici. Ta se imenuje polarizator. Še en tak film mora biti uporabljen na spodnji stekleni ploščici, vendar je tam polarizacija zasukana za 90°. To je analizator.
Ko ne deluje, tekočina zasuka polarizacijski nivo vpadne svetlobe za 90°, tako da lahko preide skozi analizator neovirano. LCD je tako prosojen. Uporaba specifične napetosti na uplinjeni kovinski plasti bo povzročila spremembo kristalov v tekočino. To bo spremenilo polarizacijski nivo svetlobe, na primer za dodatnih 90°: Analizator blokira svetlobo, LCD postane neprosojen.

5. 1 Polarizacija zaslonov

Polarizacija v LCD zaslonu ne pomeni polarnosti v električni napetosti, ampak strukturo plina, tekočine in filtra zaslona. Večina LCD so TCN zasloni (zasukan nematski zaslon). Vsebujejo tekočino, ki zasuče polarizacijski nivo za 90°. STN (super zasukana nemacija) zasuče polarizacijski nivo svetlobe za vsaj 180°. To izboljša kontrast zaslona. Vendar ta tehnika vodi do določene obarvanosti zaslona. Najpogostejša obarvanost se imenujejo rumeno-zelen in moder način. Siv način se pojavi bolj moder kot siv v praksi. Za nadomestitev neželenega barvnega učinka, FSTN tehnologija uporablja še eno zunanjo folijo. Rezultat izgubljene svetlobe naredi to tehnologijo smiselno samo za zaslone z osvetljenim ozadjem. Različne barve se pojavijo v neosvetljenih ali belo osveteljnih zaslonih. Ko je enkrat osvetlitev obarvana (na primer LED osvetlitev rumeno-zeleno), se ustrezna barva zaslona pomakne v ozadje. Vsi modri načini LCD z rumeno zeleno LED osvetlitvijo, bodo vedno izgledali rumeno zeleno.

5. 2 Statični nadzor, multipleksno delovanje

Majhni zasloni z nizkim dometom delovanja so po navadi nadzorovani statično. Statični zasloni imajo najboljši kontrast in maksimalni mogoč kot. TN tehnologija tukaj doseže svoj polni namen (črno beli zasloni, stroškovno učinkoviti). Vendar ko se zasloni večajo, več in

več vrstic potrebuje statično delovanje (na primer 128 x 64 grafično = 8192 segementov = 8192 vrstic). Ker se tako veliko število vrstic ne bi prilegalo na zaslon, niti na IC krmilnika, je izbrano Multipleks delovanje. Zaslon je strukturiran v vrsticah in kolonah in segment se nahaja na vsakem križišču (128 + 64 = 192 vrstic). Tukaj je skenirana vrstica za vrstico (64x, to je multipleks stopnja 1 : 64). Ker je naenkrat aktivna samo 1 vrstica, vendar se kontrast in kot gledanja slabšata z povečevanjem števila multipleks stopnje.

5. 3 Kot gledanja 6:00 ali 12:00

Vsak LCD zaslon ima priporočen kot gledanja. Gledanje iz te smeri pomeni da ima zaslon najboljši kontrast. Večina zaslonov je narejenih za gledanje iz smeri 6:00 (tudi: gledanje od spodaj, BV). Ta kot se sklada s tistim od žepnega računalnika, ki leži ravno na mizi. Zasloni 12:00 (pogled od zgoraj, TV) so najbolje integrirani iz sprednje strani mize. Iz vseh zaslonov lahko gledamo navpično od spredaj.

5. 4 Odsev, transprosojni, prepustni

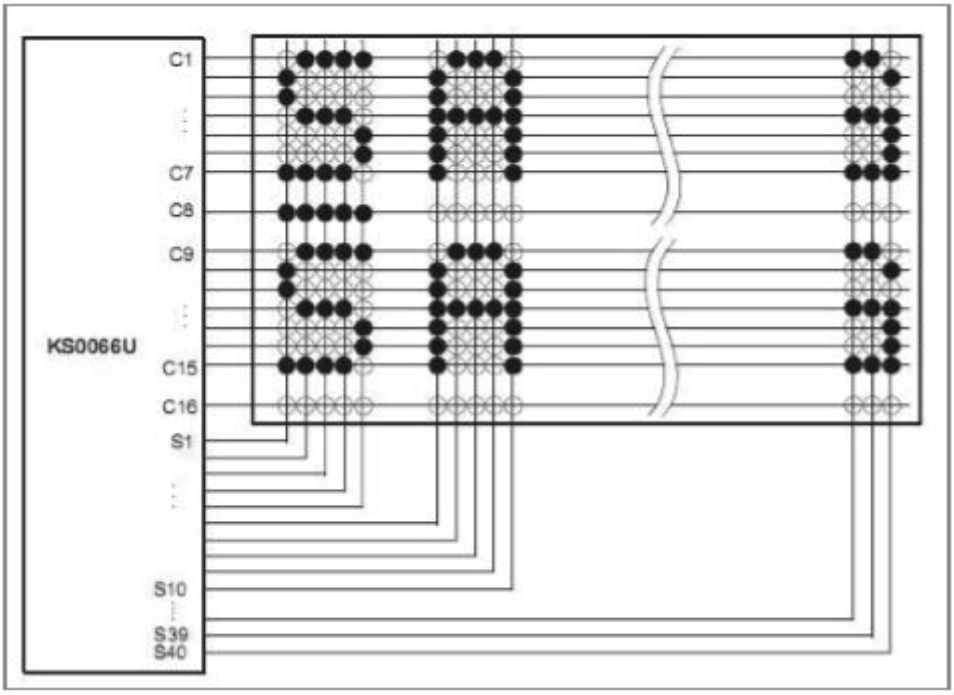
Odsevni (neosvetlejeni) zasloni imajo 100 % odsevnost ozadja. Osvetlitev od zadaj ni mogoča. Transprosojni zasloni imajo delno preprustno odsevnost ozadja. Lahko se jih bere brez osvetlitve. To jih naredi neosvetljene, vendar malo zamegljene kot odsevno različico. Kljub temu je najboljši kompromis za osvetljen LCD. Prepustni zasloni nimajo nobene odsevnosti. Iz njih se lahko bere samo s pomočjo osvetlitve, vendar so zelo svetli. LCD priložen učnemu paketu je transprosojni LCD.

5. 5 Krmilnik LCD zaslona

Matrični zasloni so narejeni s strani različnih proizvajalcev na svetu (še posebno v Tajvanu). Zraven velikih proizvajalcev, kot je Data-Vision, obstajajo tudi manjši, neznani, proizvajalci zaslonov. Na srečo je delovanje in povezava zaslonov vedno enaka. V tem učnem paketu, se bomo ukvarjali z zasloni, ki uporabljajo tip krmilnika HD44780 (ali združljivo različico), na primer KS0066. Da se vsi zasloni dosledno obnašajo je to zaradi čipa krmilnika ki je postal vzpostavljen kot standard med vsemi proizvajalci. To je HD44780 Hitachi.

5. 6 Tako je zaslon nadzorovan s pomočjo krmilnika zaslona

Naslednji primeri kažejo kako je krmilnik zaslona (KS0066) priključen na zaslon. Ta vezja ne potrebujete ustvariti sami. So že prisotna v LCD modulih.

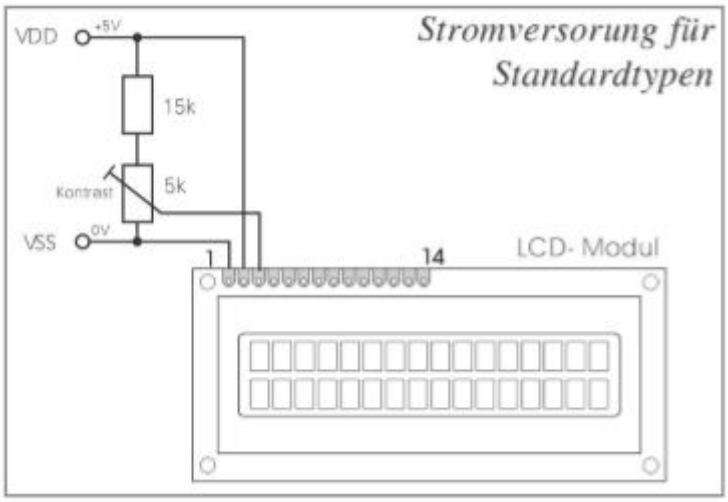


Dvovrstični 5 x 8 matrični zaslon (Vir: podatkovni list Samsung).

Krmilniki so delno različno povezani na zaslon in so lahko vklopljeni drugače, glede na proizvajalca. Tako je mogoče da je enovrstični 16 znakovni zaslon narejen iz 2 x 8 znakov. Za te informacije morate preveriti liste s podatki. Večji zasloni velikokrat uporabljajo dva krmilnika, ki imajo Chipselect (CS) ali dve Enable vrstici. Razlog zato je, da ima krmilnik znakovni spomin 80 znakov. S priključitvijo večjega števila krmilnikov zaslona, se bo znakovni spomin povečal za dodatnih 80 znakov. Zasloni imajo tudi druge priključke in jih je možno dokaj enostavno ločiti od standardnih LCD modulov. Lahko predvidevate, da ima zaslon brez osvetlitve 14 pinov in tisti z osvetlitvijo 16 pinov.

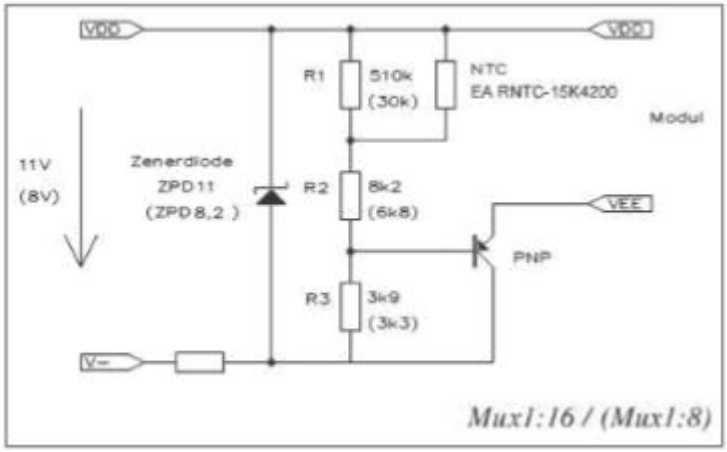
5. 7 Nastavitev kontrasta na zaslonu

Kot na drugih zaslonih, lahko tudi na LCD modulih nastavite kontrast. To naredite tako da 10 kΩ potenciometer priključite kot spremenljivi distributer napetosti. Ker imajo LCD zelo nizko razpršitev električnih lastnosti, je bila tehnologija uporabljena v učnem paketu z enim stalnim upornikom. Za to smo uporabili 2,2 kΩ upornik, ki je nameščen ozemljitvijo in kontrastno Vee povezavo LCD, tako da je kontrast dobro nastavljen.
Ko uporabljate potenciometer brez dodatnega balasta je območje prilagoditve, ki vpliva na kontrast, zelo nizka. Za boljšo razpršitev območja kontrasta, je priporočeno povezati ustrezni balast med Vcc (+5V) in enim koncem potenciometra.



Enostavna nastavitev kontrasta s potenciometrom (Vir: podatkovni list Electronic Assembly).

Napetost na pinu Vee bi morala biti prilagodljiva med 0 in 1,5 V. To vezje je primerno za uporabo na sobni temperaturi 0 do 40 °C. Če prilagoditveno območje ni optimalno (nekateri LCD od tega odstopajo) morate zamenjati balast. Praktične vrednosti so na območju med 10 do 22 kΩ.



Temperaturno nadzorovana nastavitev kontrasta (Vir: podatkovni list Electronic Assembly).

Če uporabljate zaslon izven normalnega temperaturnega območja (0 do 40 °C), je priporočeno izvesti ožičenje kot je prikazano na diagramu vezja, zgoraj. To vezje prilagodi kontrast sobnim pogojem. Temperatura bo izmerjena s temperaturnim senzorjem NTC (negativni temperaturni koeficient sonde), ki bo pomaknila kontrastno napetost proti PNP tranzistorju. LCD moduli ne morajo biti brani pravilno pri temperaturah nižjih od 0 °C. Kontrast je temperaturno odvisen.

5. 8 Nabor znakov

Zasloni imajo nabor znakov, ki je trdno integriran v krmilnik zaslona. Z zaporedjem zgornjih in spodnjih 4 bitov, bo ustvarjen podatkovni bajt za ustrezni ASCII znak. Na primer za ASCII znak A: 01000001.

Nabor znakov na LCD (Vir: podatkovni list Samsung).

Naredimo primer z LCD naborom znakov. Sledeča programska koda prikazuje kako lahko zapišete znake iz tabele znakov na zaslon. Posebni znaki, kot je znak za stopinje ali Ohm simbol, niso mogoči s pomočjo močnega izpisa. Ker je to razširjen nabor znakov LCD, moramo uporabiti znakovno tabelo.

```
001 lcd.write(B11110100)
```

Tukaj bomo zapisali binarno vrednost v krmilnik zaslona, za izpis znaka Omega. B, na začetku zaporedja primera, označuje da je to zaporedje številk v binarnem zapisu.
Zgornji (upper) in spodnji (lower) 4 biti za znak Omega so narejeni sledeče:

```
001 upper       =       1111
002 lower       =       0100
```

Prav tako poskusite izpisati druge znake, s preverjanjem tabele znakov.

Nalaganje
Poskus potrebuje osnovno ožičenje LCD, ki ste ga nastavili že v testu funkcij.

Primer kode: Nabor znakov

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 void setup()
009 {
010     // LED-Hintergrundbeleuchtung
011     analogWrite(9, 150);
012
013     // LCD-Ausgabe
014     lcd.begin(16, 2);
015     lcd.setCursor(0, 0);
016     lcd.write(B11110100);
017
018 }
019
020 void loop()
021 {
022     // Nix zu tun ...
023 }
```

5. 9 Določitev pinov pogostih LCD

Večina zaslonov brez osvetlitve ima pine dodeljene kot je zapisano v naslednji tabeli. Če kasneje uporabite drug LCD, kot priloženega v učnem paketu, priporočamo da si najprej pogledate liste s podatki, da se izognete poškodbam LCD.

Pinbelegung			
Pin	**Symbol**	**Pegel**	**Beschreibung**
1	VSS	L	Versorgung 0V, GND
2	VDD	H	Versorgung +5V
3	VEE	-	Displayspg. 0..1,5V Kontrasteinstellung
4	RS	H/L	Register Select
5	R/W	H/L	H: Read / L: Write
6	E	H	Enable
7	D0	H/L	Datenleitung 0 (LSB)
8	D1	H/L	Datenleitung 1
9	D2	H/L	Datenleitung 2
10	D3	H/L	Datenleitung 3
11	D4	H/L	Datenleitung 4
12	D5	H/L	Datenleitung 5
13	D6	H/L	Datenleitung 6
14	D7	H/L	Datenleitung 7 (MSB)

Določitev pinov na LCD brez osvetlitve (Vir: podatkovni list Electronic Assebmly).

LCD moduli z osvetlitvijo vedno potrebujejo malo več nege. Nekateri proizvajalci ne dajo stike LED osvetlitve na pina 15 in 16, ampak na pina 1 in 2. Še enkrat, preverite podatkovni list proizvajalca preden priključite LCD.

Informacija
LCD v učnem paketu ima LED povezavo na pina 15 (+=anoda) in 16 (-=katoda).

Če nimate pri roki podatkovnega lista LCD, na primer če ste zaslon kupili na elektronskem boljšem trgu, morate vseeno slediti prevodnikom in najti povezave za osvetlitev. Po navadi so nekoliko debelejše od ostalih prevodnikov. Ko se prepričate katere povezave so odgovorne za osvetlitev, lahko uporabite multimeter za določitev polarnosti. Vrnjena napetost mora znašati 2 do 4 V. Druga možnost za prepoznavo LED pinov in polarnosti, ko LED zasveti z glavno enoto ali baterijo približno 5 V v zelo visokem balastu (približno 1–4,7 kΩ). Visok balast predstavlja majhno nevarnost za uničenje LCD.

Pinbelegung			
Pin	**Symbol**	**Pegel**	**Beschreibung**
1	VSS	L	Versorgung 0V, GND
2	VDD	H	Versorgung +5V
3	VEE	-	Displayspannung 0..0,5V
4	RS	H / L	Register Select
5	R/W	H / L	H: Read / L: Write
6	E	H	Enable
7	D0	H / L	Datenleitung 0 (LSB)
8	D1	H / L	Datenleitung 1
9	D2	H / L	Datenleitung 2
10	D3	H / L	Datenleitung 3
11	D4	H / L	Datenleitung 4
12	D5	H / L	Datenleitung 5
13	D6	H / L	Datenleitung 6
14	D7	H / L	Datenleitung 7 (MSB)
15	LED +	-	LED-Versorgung Plus /Vorwiderstand!
16	LED -	-	LED-Versorgung Minus

Določitev pinov na LCD z osvetlitvijo (Vir: podatkovni list Electronic Assembly).

6. ARDUINO™ LIQUITCRYSTAL knjižnica

Kot ste se že naučili v testu funkcije, ima Arduino™ LiquidCrystal knjižnica številne funkcije, ki so določene za izpis na LCD. Sedaj se boste naučili več o LCD funkcijah.

6. 1 LiquidCrystal

LiquidCrystal določi kateri Arduino™ pin je povezan z LCD. LCD je lahko nastavljen v 4- ali 8-bitnem načinu. Za uporabo 8-bitnega načina morate pokazati osem namesto štiri podatkovne pine (D0 do D7) in jih priključiti na Arduino™-PCB.

Arduino™ sintaksa

```
001 LiquidCrystal lcd(rs, enable, d4, d5, d6, d7)
002 LiquidCrystal lcd(rs, rw, enable, d4, d5, d6, d7)
003 LiquidCrystal lcd(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
004 LiquidCrystal lcd(rs, rw, enable, d0, d1, d2, d3, d4,
                                                    d5, d6, d7)
```

Naš učni paket uporablja naslednjo konfiguracijo:

```
001 // RS, E, D4, D5, D6, D7
002 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
```

RS = Arduino™-Pin D11
E = Arduino™-Pin D10
D4= Arduino™-Pin D2
D3= Arduino™-Pin D3
D2= Arduino™-Pin D4
D1 = Arduino™-Pin D5

6. 2 .begin()

.begin() sproži LCD z navedbo vrstic in kolon. Naš LCD ima 2 vrstici in 16 kolon. Nastavljanje mora zato biti sledeče:

Arduino™ sintaksa

```
001 lcd.begin(16, 2)
```

6. 3 .clear()

.clear() izbriše izpisane znake in postavi kazalec v levi zgornji kot.

Arduino™ sintaksa

```
001 lcd.clear()
```

6. 4 .home()

.home() postavi kazalec v zgornji levi kot. Noben znak se ne izbriše.

Arduino™ sintaksa

```
001 lcd.home()
```

6. 5 .setCursor()

.setCursor postavi kazalec na določen položaj. Kot pogosto v informatiki se to štetje začne z nič. Zgornji levi kot, to je prvi znak v vrstici 1, je sledeč:

Arduino™ sintaksa

```
001 lcd.setCursor(0, 0)
```

Prvi parameter je položaj znaka, drugi parameter je vrstica.

6. 6 .write()

.write() zapiše en znak na LCD. To lahko uporabite za izpis posebnih znakov iz tabele znakov ali prikaz ASCII kode za znak.

Arduino™ sintaksa

```
001 lcd.write(64)
```

Znak @ ima digitalno število 64 v ASCII kodi.
Posamezni ASCII znaki so označeni z opuščajem. Lahko ga zapišete tudi sledeče:

Arduino™ sintaksa

```
001 lcd.write('@')
```

6. 7 .print()

S .print() lahko izpišete celotno zaporedje znakov imenovanih nizi (Strings). Na ta način je možno izpisati tudi spremenljivke. Za to obstaja število parametrov za oblikovanje (BASE), ki so navedeni kot drugi parametri.

Arduino™ sintaksa

```
001 lcd.print(data, BASE)
002
003 lcd.print("Arduino")  // es wird nur ein Text ausgeben
004
005 int variable1 = 100
006 lcd.print(variable1) // es wird der Wert der "Variablen1"
                                                    ausgeben
007
008 lcd.print(40 + 2)    // es wird die Summe aus 40 + 2
                                                   ausgegeben
009
010 lcd.print(3.1415, 2) // es wird nur 3.14 ausgeben
011
012 lcd.print(42, BIN)   // es wird die 42 binär ausgeben
```

6. 8 .cursor()

.cursor() vklopi kazalec. Kazalec je bil izklopljen, sedaj je spet viden.

Arduino™ sintaksa

```
001 lcd.cursor()
```

6. 9 .noCursor()

.noCursor izklopi kazalec (neviden).

Arduino™ sintaksa

```
001 lcd.noCursor()
```

6. 10 .blink()

.blink() vklopi kazalec in kazalec utripa.

Arduino™ sintaksa

```
001 lcd.blink()
```

6. 11 .noBlink()

.noBlink() izklopi kazalec in ta preneha utripati.

Arduino™ sintaksa

```
001 lcd.noBlink()
```

6. 12 .noDisplay()

.noDisplay izklopi zaslon. Položaj znaka in kazalca je shranjen.

Arduino™ sintaksa

```
001 lcd.noDisplay()
```

6. 13 .Display()

.display ponovno vklopi LCD po .noDisplay(). Zadnje vrednosti so obnovljene.

Arduino™ sintaksa

```
001 lcd.display()
```

6. 14 .scrollDisplayLeft()

.scrollDisplayLeft() pomakne vsebino zaslona na levo za en znak vsakič ko je klicana.

Arduino™ sintaksa

```
001 lcd.scrollDisplayLeft()
```

6. 15 .scrollDisplayRight()

.scrollDisplayRight() pomakne vsebino zaslona na desno za en znak vsakič ko je klicana.

Arduino™ sintaksa

```
001 lcd.scrollDisplayRight()
```

6. 16 .autoscroll ()

.autoscroll() pomakne vsebino zaslona iz desne proti levi. Ko je dosežen konec niza znakov, se smer pomikanje zamenja. Pomaknjena je 1x na vsak klic.

Arduino™ sintaksa

```
001 lcd.autoscroll()
```

6. 17 .noAutoscroll ()

.noAutoscroll() prekine .autoscroll() funkcijo.

Arduino™ sintaksa

```
lcd.noAutoscroll()
```

6. 18 .leftToRight()

.leftToRight() določi smer izpisa znakov. Zapisani so od leve proti desni.

Arduino™ sintaksa

```
lcd.leftToRight()
```

6. 19 .rightToLeft()

.rightToLeft() določi smer izpisa znakov. Zapisani so od desne proti levi.

Arduino™ sintaksa

```
lcd.rightToLeft()
```

6. 20 .createChar()

.createChar() ustvari namenski znak. Za to moramo ustvariti Array z osmimi podatkovnimi polji, z določitvijo naših znakov. lcd.createChar določi znakom serijsko številko s prvim parametrom. Lahko je ustvarjenih do osem lastnih znakov, ki so klicani z 0 do 7.

Arduino™ sintaksa

```
byte myChar[8] = {
  B00000,
  B10001,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
}

void setup()
{
  lcd.createChar(0, myChar)
  lcd.begin(16, 2)
  lcd.write(byte(0));
}
```

7. LCD funkcije

Sledeč primer povzame LCD funkcije, ki so pojasnjene v zgornjem daljšem primeru. Poglejte programsko kodo in zamenjajte nekaj opisanih parametrov, da boste popolnoma razumeli funkcije.

Nalaganje

Za poskus potrebujete osnovni zapis LCD, ki ste ga že nastavili v testu funkcij.

Primer kode: Funkcije

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define Backlight   9
009
010 int i;
011
012 void setup()
013 {
014     analogWrite(Backlight, 200);
015
016     lcd.begin(16, 2);
017     lcd.setCursor(0, 0);
018     lcd.print("ARDUINO LCD");
019     delay(1000);
020     lcd.clear();
021 }
022
023 void loop()
024 {
025     // Cursor blinken und Position wechseln
026     lcd.clear();
027     lcd.setCursor(0, 0);
028     lcd.print("blink/setCursor");
029     delay(1000);
030     lcd.clear();
031
```

```
032     lcd.setCursor(0, 0);
033     lcd.blink();
034     delay(1500);
035
036     lcd.setCursor(15, 0);
037     delay(1500);
038
039     lcd.setCursor(0, 1);
040     delay(1500);
041
042     lcd.setCursor(15, 1);
043     delay(1500);
044
045
046     // Cursor on/off
047     lcd.noBlink();
048     lcd.clear();
049     lcd.setCursor(0, 0);
050     lcd.print("cursor on/off");
051     delay(1000);
052     lcd.clear();
053     lcd.home();
054     lcd.cursor();
055
056     char txt[6] = {"HALLO"};
057      for(i = 0; i < 5; i++)
058     {
059         lcd.print(txt[i]);
060         delay(500);
061     }
062
063     lcd.noCursor();
064     delay(2000);
065
066
067     // Scroll LCD
068     lcd.clear();
069     lcd.noBlink();
070     lcd.setCursor(0, 0);
071     lcd.print("scroll LCD");
072     delay(1000);
073     lcd.setCursor(0, 0);
074
075     for(i = 0; i < 16; i++)
076     {
077         lcd.scrollDisplayLeft();
```

```
078         lcd.setCursor(0, 0);
079         lcd.print("FRANZIS ARDUINO IST MEGA SPITZE!");
080         delay(350);
081     }
082
083     delay(1500);
084
085     for(i = 0; i < 16; i++)
086     {
087         lcd.scrollDisplayRight();
088         lcd.setCursor(0, 0);
089         lcd.print("FRANZIS ARDUINO IST MEGA SPITZE!");
090         delay(350);
091     }
092
093     delay(1500);
094 }
```

Novo je to da navedemo pin za osvetlitev z #define backlight. To je predprocesorski ukaz, ki bo zamenjal vsa imena, ki se pojavijo v izvorni kodi z oznako Backlight vrednosti 9. Na ta način lahko naredite spremembe na parametrih zelo hitro, brez da bi morali preiskati celotno izvorno kodo.

Nasvet
Za več informacije na temo predprocesorjev si poglejte spletno stran: http://www.mikrocontroller.net/articles/C-Pr%C3%A4prozessor

Sledeča programska točka ponuja možnost izpisa posameznih znakov kot v starem pisalnem stroju:

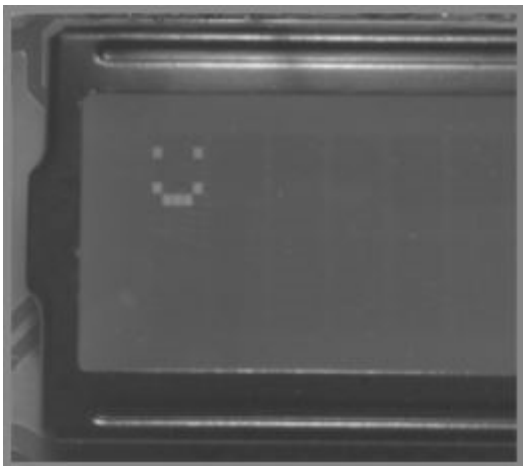
```
001     char txt[6] = {"HALLO"};
002      for(i = 0; i < 5; i++)
003     {
004         lcd.print(txt[i]);
005         delay(500);
006     }
```

Tukaj je Array z 6 znaki nastavljen in že v naprej določen z nizom (string) »HELLO«. Vedno morate nastaviti Array večji kot 1, ker je nevidna prekinitev niza (\0) dodana samodejno.
For() zanka izpiše vsak posamezen znak iz Arraya s kratkim premorom. Za vso operacijo je kazalnik vklopljen, da zgleda bolj kot pisalni stroj.

8. Ustvarjanje lastnih znakov

Ustvarjanje lastnih znakov, kot je že bilo opisano z uporabo .createChar(), je pogosto potrebno ko uporabljate matrični LCD, ker veliko znakov potrebnih v praksi ni vključenih v tabelo znakov LCD. Obstaja možnost ustvarjanja lastnih znakov piko za piko in jih lahko

izpišete. Če potrebujete na primer smeška, ga lahko definirate preko Array in ga pošljete na LCD.
Obstaja možnost zapolnitve do osmih znakov v RAM (spomin) na LCD. Array naših posebnih znakov mora biti velik 8 bajtov in je najbolje zapisan kot je prikazano v poskusni kodi. Znaki so lahko na primer ustvarjeni na preverjeni risbi pad. Vidite lahko, da je sestavljena iz 8 vrstic in vsaka iz 5 vrednosti, kar odraža 5 x 8 pik v LCD. Kjer nastavimo 1 v binarni kodi, se kasneje pojavi bela pika. Z lcd.write(byte(0)) zapišemo znak na LCD.
Primer naredi celotno stvar jasnejšo. Poskusite ustvariti simbol za baterijo ali termometer.


Samodejno narisan smeško.

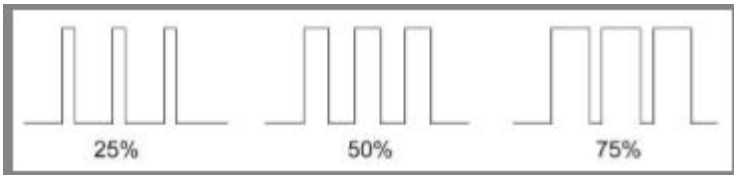
Nalaganje
Za poskus potrebujete osnovni zapis LCD, ki ste ga že nastavili v testu funkcij.

Primer kode: lastni znaki

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 byte myChar[8] = {
009   B00000,
010   B10001,
011   B00000,
012   B00000,
013   B10001,
014   B01110,
015   B00000,
016 };
017
018 void setup()
019 {
020     // LED-Hintergrundbeleuchtung
021     analogWrite(9, 150);
022
023     lcd.createChar(0, myChar);
024     lcd.begin(16, 2);
025     lcd.write(byte(0));
026 }
02/
028 void loop()
029 {
030
031     // Nix zu tun ...
032
033 }
```

9. Zatemnitev ozadja

Sledeč primer prikazuje kako lahko samodejno nastavite LCD osvetlitev na svetlejšo ali temnejšo. S spremembo PWM vrednosti na pinu D9 se svetlost osvetlitve prilagaja postopoma. Če izberete višjo PWM vrednost bo LED svetlejši. Z nižjo vrednosti PWM zatemnite osvetlitev. S spremembo PWM vrednosti boste spremenili razmerje plus-pause med trajanjem aktivacije in deaktivacije 5 V signala na D9. To prikazuje sledeč primer.



Majhna PWM vrednost pomeni krajši čas vklopa in s tem temnejšo osvetlitev.

Prenesite program in opazujte LCD. Zgleda tako kot da bi zaslon oživel.

Nalaganje
Za poskus potrebujete osnovni zapis LCD, ki ste ga že nastavili v testu funkcij.

Primer kode: LCD_LED

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define Backlight   9
009
010 byte i = 0;
011 byte flag = 0;
012 unsigned long previousMillis = 0;
013 const long interval = 10;
014
015 void setup()
016 {
017     analogWrite(Backlight, 0);
018
019     lcd.begin(16, 2);
020     lcd.setCursor(0, 0);
021     lcd.print("**ARDUINO LCD**");
022 }
023
024 void loop()
025 {
026     unsigned long currentMillis = millis();
027     if(currentMillis - previousMillis >= interval)
028     {
029         if(flag==0)i++;
030         if(flag==1)i--;
031
032         if(i==255)flag=1;
033         else if(i==0)flag=0;
034
035         analogWrite(Backlight, i);
036
037         previousMillis = currentMillis;
038     }
039 }
```

Primer prikazuje kako deluje samodejni števec gor/dol v praksi. Pomembno je da spremenljivka imenovana flag sprejme določeno začetno vrednost 0. Ko zaženete program se spremenljivka i šteje do 255. Če želite začeti s polno svetlostjo morate določiti spremenljivko flag z 1 in spremenljivko i z 255.

Ko doseže vrednost števca i 255 ali 0, bo spremenljivka flag vedno nastavljena od 0 do 1 in smer števca se bo spremenila (i++ povečuje stanje števca za 1, i— zmanjšuje stanje števca za 1). V Loop() funkciji tokrat ne rabimo uporabiti break, ampak določiti čas s pomočjo funkcije millis(). Šele ko bo razlika, ki smo jo določili v spremenljivki previousMillis, potekla se bo osvetlitev spremenila na eno stopnjo. Na ta način bo program nadaljeval s polnim delovanjem izven If() funkcije. Samo če je PWM vrednost zamenjana se bo čas delovanja malo zamenjal, ker bodo klicane nekatere funkcije, ki potrebujejo časovno spremembo. Tukaj lahko poskusite določiti različne čase delovanje s funkcijo Millis(), ki ste se jo že naučili uporabljati.

10. Ura matričnega LCD

V veliko aplikacijah je ura potrebna za nadzor programa – kot preprost časovnik, nadzor nad točno določenim urnikom ali kot delujoči števec ur. Aplikacije ki potrebujejo uro so različne.
Poskus prikazuje kako lahko sami programirate zelo preprosto uro. Program se izvaja v loop() funkciji, je končen in šteje v ciklih dolgih 10 ms. Če števec prebere cnt=100 se izpiše čas. To se zgodi vsako sekundo. LED L bo zasvetil vsako sekundo. Spremljajte delovanje programa da zagotovite, da se še vedno izvaja in če je slučajno prišlo do napake. Vendar lahko opazite da ura ni tako natančna kot dejanska quartz ura, ker je cikel in odstopanje mikrokrmilnika quartz pri 16 MHz veliko višje, kot pri večji quartz uri z kilohertz dosegom (quartz ura =32,768 kHz). Odstopanje za več kot eno minuto na dan ni nič nenavadnega. Natančnost je odvisna tudi od sobne temperature. Če močno niha s časom, bo ura imela veliko časovno napako. Časovno odstopanje lahko popravimo z ukazom delay(). Za še boljšo popravilo napake lahko uporabite delayMicroseconds(). Za to nastavite digitalne pine kot izhod in preklopite ob vsakem izvajanju programa, zamenjajte pogoje enkrat na izvajanje. Za signal je lahko natančno naravnan na izhodni čas 10 ms s pomočjo osciloskopa. Lahko določite odstopanje čez razširjeno časovno periodo, s primerjavo časa z različno, bolj natančno uro, na primer DCF uro (1 ali dva dni), računate razliko in jo nato popravite odstopanje z delayMicroseconds().

Ura v uporabi.

Spremenljivke so nato uporabljene za nastavitev ure:

```
001 Sekunde = 12
002 Minute = 0
003 Stunde = 0
```

Nasvet
Za več informacij o quartz uri si oglejte spletno stran:
http://de.wikipedia.org/wiki/Uhrenquarz

Nalaganje
Za poskus potrebujete osnovni zapis LCD, ki ste ga že nastavili v testu funkcij.

Primer kode: RTC

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define Backlight   9
009
010 int cnt, Sekunde, Minute, Stunde=0;
011 int LED=13;
012
013 void setup()
014 {
015       pinMode(LED, OUTPUT);
016       analogWrite(Backlight, 200);
017       lcd.begin(16, 2);
018
019       // Zeitvorgabe
020       Sekunde = 12;
021       Minute = 0;
022       Stunde = 0;
023 }
024
025 void loop()
026 {
027
028   cnt++;
```

```
029   if(cnt == 50)digitalWrite(LED, LOW);
030
031   if(cnt == 100)
032   {
033       digitalWrite(LED, HIGH);
034
035       lcd.setCursor(3, 0);
036
037     if(Stunde < 10) lcd.print("0");
038     lcd.print(Stunde);
039     lcd.print(":");
040
041     if(Minute < 10) lcd.print("0");
042     lcd.print(Minute);
043     lcd.print(":");
044
045     if(Sekunde < 10) lcd.print("0");
046     lcd.print(Sekunde);
047
048     Sekunde++;
049     if(Sekunde == 60)
050     {
051         Sekunde = 0;
052         Minute++;
053         if(Minute == 60)
054         {
055             Minute = 0;
056             Stunde++;
057             if(Stunde == 24)
058             {
059                 Stunde = 0;
060             }
061         }
062     }
063     cnt = 0;
064   }
065
066   delay(10);
067 }
```

Če bi želeli izpisati števčeve izpise enega za drugim na zaslon, bi izpisi števca pod 10 izgledali nenavadno, ker vodilna ničla ne bi bila izpisana. Da ima ura znano obliko zapisa »00:00:00«, morate preveriti vrednost, ki mora bit nižja od 10 pred izpisom.

```
001 if(Sekunde < 10) lcd.print("0")
```

Če je vrednost pred izpisom nižja od 10, bo preprost izpis prikazal, na primer »12:1:8«. Vendar preverite, če je vrednost nižja in dodajte »0« ročno, da zapolnite desetice.

11. Merilnik zmogljivosti

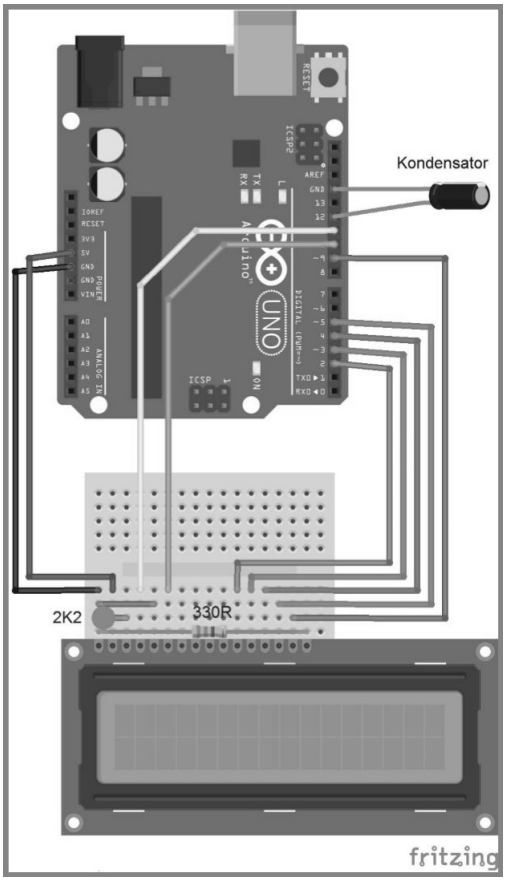
Izgradnja lastnih merilnikov s preprostimi stvarmi je vedno zanimiva in zabavna. Arduino™ dovoljuje programiranje merilnika zmogljivosti za manjše kondenzatorje v območju 1 nF do približno 100 µF za vaše laboratorije po zelo nizki ceni in z malo truda. Tukaj lahko vidite kako deluje kondenzator s samodejnim območjem:
Pričetek meritve, spremenljivka c_time je določena z nič. Vrata D12 so nastavljena kot izhod in nato nemudoma preklopljena na nizko (GND), da se izprazni kondenzator (testni del) pred dejansko meritvijo.
Po kratkem premoru 1 sekunde ko se praznjenje konča, bodo vrata 12 konfigurirana kot vhod in zvišanje notranjega upora bo aktivirano. Notranji upor bo sedaj polnil kondenzator, ki bo testiran, dokler vrata D12 ne bodo prepoznala visoko. Meja nad katero digitalna vrata prepoznajo visoko je približno 3,5 V pri delovni napetosti 5 V. Ta nivo je zato odvisen od delovne napetosti, kot je napisano na podatkovnem listu mikrokrmilnika, Vcc x 0,7.

```
001 HIGH = 5V x 0,7
```

Preden je še prepoznana visoka vrednost na digitalnem vhodu, mine nekaj časa. Merimo znotraj Do-while zanke z uporabo spremenljivke c_time. c_time se približno ujema s kapaciteto kondenzatorja, to pomeni ko je c_time zelo velik, je tudi izmerjena kapaciteta zelo velika.
Da dobite pravilno meritev, morate pretvoriti v spremenljivko (c_time x factor). Vrednost (factor) mora biti določena s pomočjo eksperimenta z nekaj že umerjenimi kondenzatorji, ker bo prepoznava visokega nivoja odstopala med različnimi krmilniki, kljub informaciji Vcc x 0,7 v podatkovnem listu, in frekvenca oscilatorja 16 MHz ni 100 % enaka na vsaki plošči (quartz toleranca). Končno je izmerjena vrednost deljena v Nanofarad (nF) in Mikrofarad (µF) s preprostim if() stavkom in izpisom na LCD pred novo meritvijo.

11. 1 Sestavljanje merilnika zmogljivost

Kondenzator je priključen na pin D12 in GND. Poskrbite da dodatno izpraznite kondenzator, ko ga priključite na Arduino™-PCB s stikom dveh žic kondenzatorja. Tudi če program izprazni kondenzator pred začetkom meritve, je mogoče da je kondenzator, ki ste ga prej izbrali, deloval na višji napetosti od 5 V. To lahko poškoduje vašo Arduino™ ploščo.



Sestavljanje merilnika.

11. 2 Umerjanje vašega merilnika zmogljivosti

Najdite različne kondenzatorje – vedno uporabite kondenzatorje katerih kapaciteto natančno poznate. Lahko pustite da vam jo izmeri strokovnjak za elektroniko. Novi kondenzatorji imajo po navadi odtisnjeno vrednosti +/-20 %, odvisno od tipa.
V tej meritvi vstavite faktor 1 (c_time=1.0). Priključite enega od kondenzatorjev v merilnik in odčitajte vrednost, ko se ta pojavi za zaslonu. Delite vrednost priključenega kondenzatorja z izmerjeno vrednostjo na LCD in vstavite rezultat kot faktor, na primer 1 µF/ 19,55 µF = 0,0511.

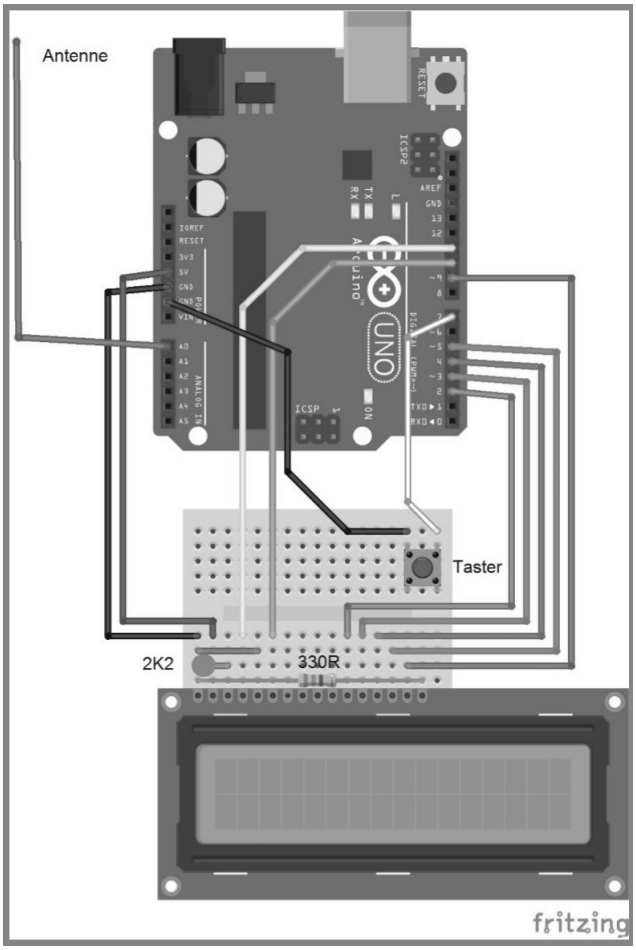
Primer kode: CAPA

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define Backlight   9
009
010 int messPort=12;
011 float c_time=0.0;
012 float kapazitaet=0.0;
013
014 void setup()
015 {
016     analogWrite(Backlight,200);
017
018     lcd.begin(16, 2);
019     lcd.setCursor(0,0);
020     lcd.print("C-MESSGERAET");
021 }
022
023 void loop()
024 {
025       // Entladen
026       pinMode(messPort,OUTPUT);
027       digitalWrite(messPort,LOW);
028       c_time=0.0;
029       delay(1000);
030
031       // Laden
032       pinMode(messPort,INPUT);
033       digitalWrite(messPort,HIGH);
034
035       // Messen
036       do
037       {
038         c_time++;
039       }while(!digitalRead(messPort));
040
041       // Umrechnen
042       kapazitaet=(c_time*0.06162)*10.0;
043
044       // Bereich
```

```
045         if(kapazitaet<999)
046         {
047             lcd.setCursor(0,1);
048             lcd.print(kapazitaet);
049             lcd.print("nF    ");
050         }
051         else
052         {
053             lcd.setCursor(0,1);
054             lcd.print(kapazitaet/1000);
055             lcd.print("uF    ");
056         }
057
058         delay(1000);
059 }
```

12. Naključna števila - generator loterijskih števil

Med zapisovanjem, merjenjem, nadzorom, regulacijo in izvajanjem programa je pogosto koristno ustvarjanje naključnih števil, na primer ko se luči vklapljajo in izklapljajo v hiši ob različnih časih. Za ta namen lahko uporabite funkcijo Arduino™-random(). To dovoljuje ustvarjanje preprostega generatorja loterijskih števil. Ne boste rabili več razmišljati katerega števila izbrati, ko boste izpolnjevali vaš loterijski listek.
Za nastavitev generatorja loterijskih števil potrebujete tipko in anteno. Tipka se znotraj programa odbija. Tipka in stikalo ne zapreta stika 100 % naenkrat, ampak sta namenjena sprožitvi večkrat zapored po pritisku. To lahko primerjate z metom žoge ob tla. Nekajkrat se bo odbila, preden bo obležala na tleh. To se zgodi veliko hitreje s tipko, vendar je Arduino™ mikrokrmilnik tako hiter, da bo zaznal te milisekundne skoke. Da se temu izognete, se tipka odbije, s tem da je prebrana dvakrat zapored v presledku 50 ms, kar je dovolj za odbojno rutino v praksi. Edino če je druga ocena še vedno prepoznala nizko stanje izhoda D7, se bo navodilo med oklepaji izvršilo.
Na začetku programa preklopimo vrata D7 na izhod in aktiviramo notranji dvig upornosti s pomočjo digitalWrite() z zapisom 1 za visoko stanje vhoda. Sedaj napetost približno 5 V čaka na vhodu v stanju mirovanja. Sedaj lahko usmerite vhod proti GND (ozemljitvi) s tipko. Ta dvig upornosti je integriran v mikrokrmilniku in ima vrednost približno 20 do 50 kΩ. Kar se tiče funkcije, je enako kot da bi priključili zunanji upornik iz vhoda D7 na +5 V. V stanju mirovanja bo program vedno prepoznal visoko vrednost na vratih D7 in nizko stanje ko bo pritisnjena tipka. Zato je stavek tipke zapisan v programu z vprašajem. To se imenuje NOT-operator v programskem jeziku C. Ker je znano da If() stavek preverja za TRUE, bo vse drugo interpretirano kot FALSE. Če je bil If() stavek izveden brez tega operatorja, bo pogoj vedno TRUE. Izvedel se bo še pred pritiskom tipke. NOT-operator obrne stanje tipke. 1 obrne v 0 in 0 obrne v 1 in If() stavek je sedaj samo TRU, če je tipka bila dejansko pritisnjena.



Sestavljanje generatorja loterijskih števil. Kot anteno lahko uporabite kos žice. Pritisnite tipko, da dobite izpisane zmagovalna števila.

Primer kode: Loterija

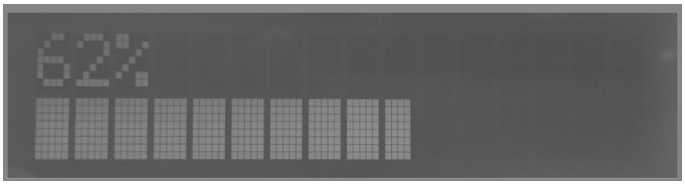
```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define Backlight   9
009 #define TASTER      7
010
011 int i, zahl=0;
012 int Anz = 6;
013
014 void setup()
015 {
016     analogWrite(Backlight, 200);
017
018     pinMode(TASTER, INPUT);
019     digitalWrite(TASTER, HIGH);
020
021     lcd.begin(16, 2);
022     lcd.setCursor(0, 0);
023     lcd.print("LOTTO 6 aus 49");
024     delay(1000);
025     lcd.clear();
026
027     randomSeed(analogRead(0)*5);
028
029 }
030
031 void loop()
032 {
033     lcd.clear();
034     lcd.setCursor(0, 0);
035     lcd.print("TASTER DRUECKEN");
036
037     while(digitalRead(TASTER));
038     {
```

```
039         if(!digitalRead(TASTER))
040         {
041             delay(50);
042             if(!digitalRead(TASTER));
043             {
044                 lcd.clear();
045                 lcd.setCursor(0, 0);
046                 lcd.print("IHRE LOTTOZAHLEN");
047
048                 lcd.setCursor(0, 1);
049                 for(i=0;i<Anz;i++)
050                 {
051                    zahl=random(49);
052                    zahl++;
053                    lcd.print(zahl);
054                    lcd.print(" ");
055                    delay(500);
056                 }
057
058                 delay(5000);
059             }
060         }
061     }
062 }
```

Ko zaženete program, randomSeed() ustvarite izhodiščno vrednost za funkcijo Random(). Ko spreminjate vrednost randomSeed(), bodo ustvarjena različna naključna števila. Če je vrednost randomSeed() vedno enaka, bodo ustvarjena vedno ista naključna števila, kar ne bo pomagalo pri igranju lota.
Tukaj uporabimo našo anteno. Za tvorbo različnih vrednosti s funkcijo randomSeed() uporabite ADC vhod za priključenega skakalca in drugi del skakalca odprt. To deluje kot antena in proizvaja višji šum na analognem vhodu in tako različne vrednosti randomSeed(). To deluje najbolje če daste vašo roko zraven antene ali če se antena nahaja zraven električne naprave. Rezultat z različnimi vrednostmi števil lahko vidite samo ob pritisku tipke za ponovni zagon na Arduino™-PC in nato izpišete števila. Namesto analogRead() lahko enkrat vpišete določeno vrednost, da vidite isto vrednost izpisa po vsakem ponovnem zagonu.

13. Zaslon za izpis stolpčnega diagrama

Zasloni za izpis stolpčnega diagrama se v merilni tehnologiji pogosto uporabljajo. Imenujejo se tudi stolpčni zasloni. Prikazujejo trend izmerjenih vrednosti. Ko nastavljate elektronska vezja, vam stolpčni prikaz vrednosti olajša delo, ker lahko lažje razberete trende proti maks. ali min., kot pa iz digitalnih vrednosti. Ta stolpčni izpis poznamo tudi iz računalniških programov, ko nameščamo programe. Tukaj stolpec prikazuje koliko programa je že bilo naloženega. Na splošno je stolpčni diagram, prikazan v tem poskusu, analogni zaslon na digitalni osnovi. Elektromehanski stolpčni diagrami so bili uporabljeni že v obdobju zgodnje elektronike. Naši stolpčni diagrami uporabljajo sodoben LCD in mikrokrmilnik.
V preprostem primeru boste lahko prikazali celoten znak za vsak korak zaslona (5 x 8 pik). Lahko implementiramo zelo splošen zaslon od 0 do 16. Bilo bi lepo če bi lahko uporabili dodatnih 5 stolpcev vsakega znaka. Ker lahko ustvarite 8 lastnih znakov, je lažje sprogramirati stolpčni diagram z 5 x 16 = 89 znaki/pogoj.



Stolpčni diagram med izvajanjem.

Nalaganje
Za poskus potrebujete osnovni zapis LCD, ki ste ga že nastavili v testu funkcij.

Primer kode: BARGRAPH

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
```

```
007
008 #define LCD_LENGHT 16.0
009
010 int value;
011 byte flag = 0;
012
013 byte MyChar0[8] = {
014 B00000,
015 B00000,
016 B00000,
017 B00000,
018 B00000,
019 B00000,
020 B00000,
021 B00000,
022 };
023
024 byte MyChar1[8] = {
025 B10000,
026 B10000,
027 B10000,
028 B10000,
029 B10000,
030 B10000,
031 B10000,
032 B10000,
033 };
034
035 byte MyChar2[8] = {
036 B11000,
037 B11000,
038 B11000,
039 B11000,
040 B11000,
041 B11000,
042 B11000,
043 B11000,
044 };
045
046 byte MyChar3[8] = {
047 B11100,
048 B11100,
049 B11100,
050 B11100,
051 B11100,
052 B11100,
```

```
053 B11100,
054 B11100,
055 };
056
057 byte MyChar4[8] = {
058 B11110,
059 B11110,
060 B11110,
061 B11110,
062 B11110,
063 B11110,
064 B11110,
065 B11110,
066 };
067
068 byte MyChar5[8] = {
069 B11111,
070 B11111,
071 B11111,
072 B11111,
073 B11111,
074 B11111,
075 B11111,
076 B11111,
077 };
078
079 void draw_bargraph(byte percent)
080 {
081   byte i, c1, c2;
082
083   lcd.setCursor(0, 0);
084   lcd.print(percent);
085   lcd.print("%  ");
086
087   lcd.setCursor(0, 1);
088
089   percent = map(percent, 0, 100, 0, 80);
090
091   c1 = percent / 5;
092   c2 = percent % 5;
093
094   for(i = 0; i < c1; ++i)
095   {
096     lcd.write(byte(5));
097     lcd.write(c2);
098   }
```

```
099
100   for(i = 0; i < 16 - (c1 + (c2 ? 1 : 0)); ++i)
101   {
102     lcd.write(byte(0));
103   }
104 }
105
106 void setup()
107 {
108     analogWrite(9,200);
109
110     lcd.createChar(0, MyChar0);
111     lcd.createChar(1, MyChar1);
112     lcd.createChar(2, MyChar2);
113     lcd.createChar(3, MyChar3);
114     lcd.createChar(4, MyChar4);
115     lcd.createChar(5, MyChar5);
116
117     lcd.begin(16, 2);
118 }
119
120 void loop()
121 {
122     double percent;
123
124     if(flag == 0)value++;
125     if(flag == 1)value--;
126     if(value > 1024)flag=1;
127     else if(value == 0)flag=0;
128
129     percent = value / 1024.0 * 100.0;
130     draw_bargraph(percent);
131     delay(10);
132 }
```

Posamezni deli zaslona stolpčnega diagrama so določeni z Array MyChar0 in MyChar1. Programska koda že prikazuje kako se deli zapolnijo z Array za Array. V funkciji setup(), so znaki ustvarjeni z lcd.createChar().
V funkciji Loop(), je števec gor/dol, ki ga poznate iz prejšnjega primera, tukaj znova uporabljen. Vendar tokrat šteje od 0 do 1024. Na ta način je lahko števec zamenjan z Arduino™ digitalnim vhodom, ki pokriva vrednosti od 0 do 1023. Vrednost števca je pretvorjena v procente in predana funkciji draw_bargraph().
draw_bargraph() funkcija je zanimiva. Tukaj je stolpčni diagram zložen skupaj in prikazan. Vsakič ko je funkcija klicana bo kazalka postavljena v položaj (0,0). To je začetni položaj za ponovno pisanje na zaslon. V prvi vrstici izpišemo procente iz spremenljivke percent in za njo izpišemo znak za procente. Za znakom za procente bosta dve prosti mesti. Izpisa digitalnih vrednosti v procentih je tako končan.
Nato uporabimo setCursor(0, 1) za namestitev kazalnika v spodnji, to je drugo vrstico LCD: Za delitev vrednosti procentov, ki gre od 0 do 100 % v 80 posameznih območij (pikslov),

uporabimo Map() funkcijo. To spremeni lestvico vhodnih vrednosti od 0 do 100 na izhodno vrednosti 0 do 80, glede na vhodno vrednost percent.
Sedaj določimo vrednost percent z delitvijo števila polj, ki jih moramo zapolniti s 5 in zapišemo rezultat v spremenljivko c1. Operator % določi ostale delne vrednosti. Spremenljivko zapišemo v spremenljivko c2. Sedaj vemo koliko polj moramo čisto zapolniti in katere samo delno.
Sledeča For() zanka šteje dokler polja niso zapolnjena, in izpiše celotno polje na LCD, ob vsakem končanju zanke. Ker vsak naknadni izpis na LCD samodejno premakne znak za ena, bomo imeli stolpec s polnimi polji na koncu zanke. Ta del programa prikazuje da For() zanka ne uporablja oklepajev. Prevajalnik v For() zanki naknadno vzame sledečo vrstico v zanko. V tem primeru bo klicana lcd.write(byte(5)). Po prevodu prve zanke, bo škatla z delno zapolnitvijo prikazana na LCD: To vodi do stolpiča ko je sestavljen piksel za pikslom na LCD:

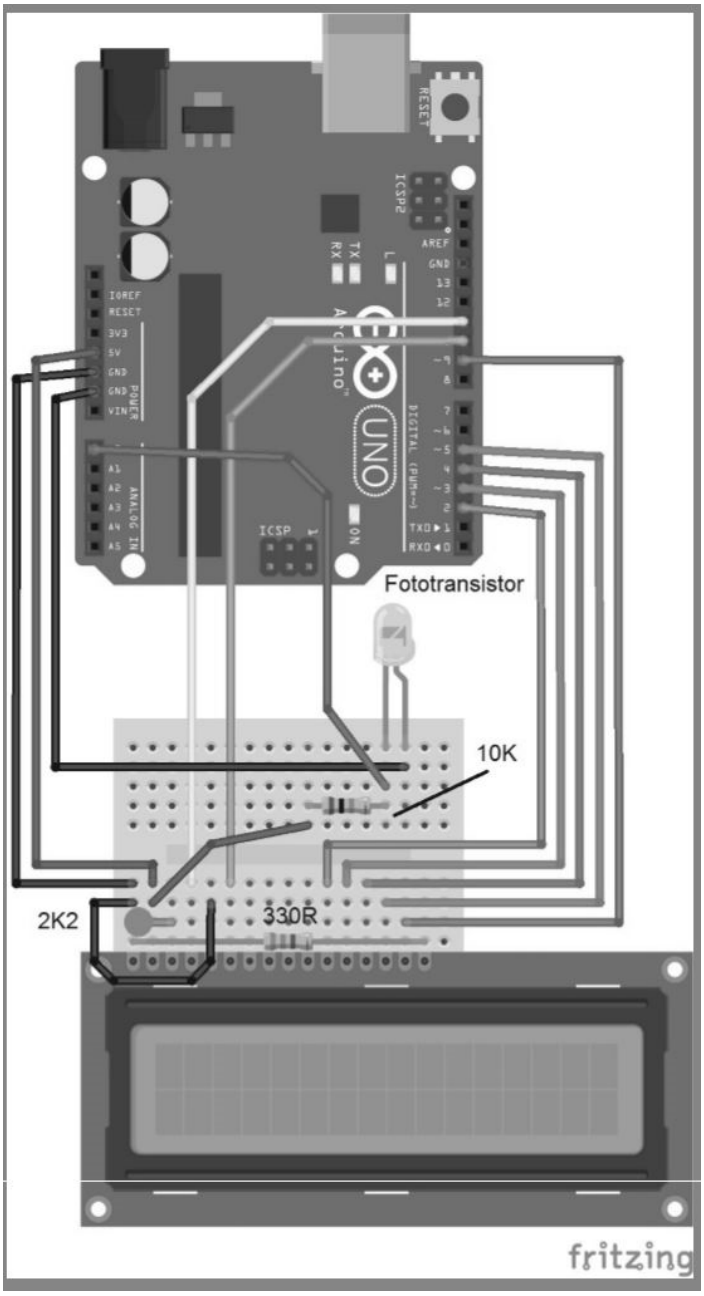
Primer
Prikazana bo vrednost 43. Delite vrednost 43 s 5, kar znaša 4,6. Ker je spremenljivka c1 razglašena kot bajt, bo samo 8 shranjena v njej. Če ostanek 43 delimo s 5 je 3, kar pomeni 3 delne udarce.
Če se vrednost ponovno zmanjša, moramo izbrisati odvečni znak iz LCD. Nova operacija, ki se imenuje pogoji izraz ali ternarni izborni operator, je dodana.
Na splošno lahko na celotno stvar gledate kot If-Else navodila, vendar je samo okrajšana C izpeljava. Sintaksa za pogojna navodila bi bila: Condition ? Expression1 : Expression2
Se vam to zdi znano? Ni nobene razlike od:

```
001 if(Bedingung)
002 {
003       // Ausdruck1
004 }
005 else
006 {
007       // Ausdruck2
008 }
009
010 for(i = 0; i < 16 - (c1 + (c2 ? 1 : 0)); ++i)
011 lcd.write(byte(0))
```

Zanka izbriše odvečen znak iz LCD z določitvijo kako dolgo je območje ki ni uporabljeno, ter ga prepiše z znaki.
Potrebnega je nekaj znanja za izpis stolpčnega diagrama, vendar ko enkrat razumete, pride zelo prav v številnih aplikacijah.

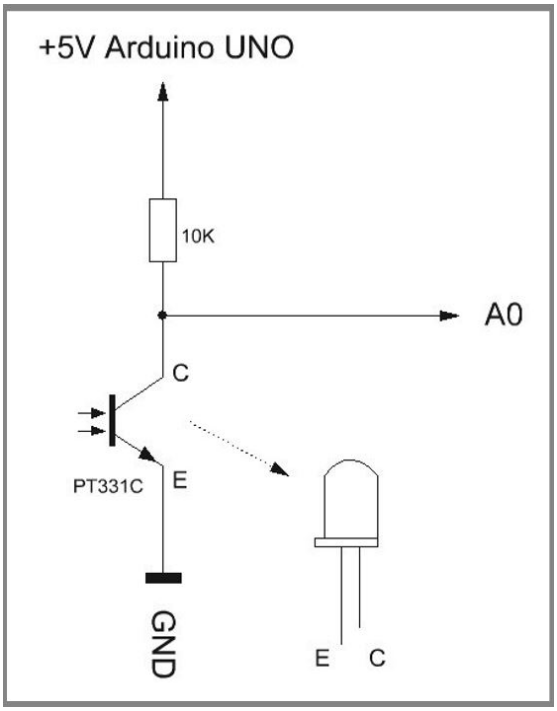
14. Merilec svetlobe - fotometer

Fotometer je merilnik ki določi gostoto svetlobe ali moč svetlobe. Uporabljajo ga na primer fotografi kot merilnik osvetlitve ali v astronomiji za določitev svetlosti zvezd.



Sestavljanje fotometra. LCD povezava je vzpostavljena.

V kemiji se uporablja za določitev koncentracije. V zadnje primeru smo programirali stolpčni diagram, tokrat bomo namesto gor/dol števca, predali v uporabo pravo fizikalno vrednost in programirali preprost fotometer.



Načrt priključitve fototrazistorja tipa PT331C na Arduino™ plošči.

V diagramskem vezju lahko vidite da je fototranzistor priključen na analogni vhod (ADC) na Arduino™ plošči. Fototranzistor ni lahko ločiti od LED regulatorja. Ima čisto, transparentno ohišje, vendar je to res tudi za nekatere LED. Če niste prepričani ali imate LED ali fototranzistor, lahko preverite s pomočjo multimetra, tako da ga nastavite na določeno upornost. Za to priključite zbiralnik (C- ravni del ohišja) na pozitivni del in oddajnik (E) na negativni kabel multimetra. Če potemnite fototranzisotr, se bo upornost močno spremenila. Na LED bo učinek veliko manjši. Multimetri s samodejno funkcijo range so tukaj idealni, saj je lahko meritev nekje med kilo Ohm in nekaj mega Ohm.

Primer kode: FOTOMETER

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define LCD_LENGHT 16.0
009
010 const int numReadings = 10;
011 unsigned int readings[numReadings];
012 int index = 0;
013 unsigned int total = 0;
014 int average = 0;
015
016 byte MyChar0[8] = {
017 B00000,
018 B00000,
019 B00000,
020 B00000,
021 B00000,
022 B00000,
023 B00000,
024 B00000,
025 };
026
027 byte MyChar1[8] = {
028 B10000,
029 B10000,
030 B10000,
031 B10000,
032 B10000,
033 B10000,
034 B10000,
035 B10000,
036 };
037
038 byte MyChar2[8] = {
039 B11000,
040 B11000,
041 B11000,
042 B11000,
043 B11000,
044 B11000,
045 B11000,
046 B11000,
```

```
047 };
048
049 byte MyChar3[8] = {
050 B11100,
051 B11100,
052 B11100,
053 B11100,
054 B11100,
055 B11100,
056 B11100,
057 B11100,
058 };
059
060 byte MyChar4[8] = {
061 B11110,
062 B11110,
063 B11110,
064 B11110,
065 B11110,
066 B11110,
067 B11110,
068 B11110,
069 };
070
071 byte MyChar5[8] = {
072 B11111,
073 B11111,
074 B11111,
075 B11111,
076 B11111,
077 B11111,
078 B11111,
079 B11111,
080 };
081
082 int adc_AVG(byte channel)
083 {
084     total= total - readings[index];
085     readings[index] = analogRead(channel);
086     total= total + readings[index];
087     index = index + 1;
088
089     if (index >= numReadings)index = 0;
090
091     average = total / numReadings;
092     return average / 1024.0 * 100.0;
093 }
```

```
094
095 void draw_bargraph(byte percent)
096 {
097     byte i, c1, c2;
098
099     lcd.setCursor(0, 0);
100     lcd.print("Brightness: ");
101     lcd.print(percent);
102     lcd.print("%  ");
103
104     lcd.setCursor(0, 1);
105
106     percent = map(percent, 0, 100, 0, 80);
107
108     c1 = percent / 5;
109     c2 = percent % 5;
110
111     for(i = 0; i < c1; ++i)
112     lcd.write(byte(5));
113
114     lcd.write(c2);
115
116     for(i = 0; i < 16 - (c1 + (c2 ? 1 : 0)); ++i)
117     lcd.write(byte(0));
118 }
119
120 void setup()
121 {
122     analogWrite(9,200);
123
124     lcd.createChar(0, MyChar0);
125     lcd.createChar(1, MyChar1);
126     lcd.createChar(2, MyChar2);
127     lcd.createChar(3, MyChar3);
128     lcd.createChar(4, MyChar4);
129     lcd.createChar(5, MyChar5);
130
131     lcd.begin(16, 2);
132 }
133
134 void loop()
135 {
136     int raw_adc = adc_AVG(0);
137     draw_bargraph(100 - raw_adc);
138     delay(20);
139 }
```

Ko ste nastavili vezje in prenesli kodo na Arduino™ ploščo, se bo izpisala vrednost blizu 100 % na LCD v svetlem okolju, in stolpec na stolpčnem diagramu bo skoraj povsem zapolnil spodnjo vrstico LCD.
Če sedaj skoraj popolnoma zatemnite fototranzistor, se bo vrednost zmanjšala na blizu 0. Natančneje si poglejte vezje. Fototranzisotr je priključen na 10 kΩ upornik na sprejemniku, ki je priključen na +5 V in na GND (ozemljitev) na oddajniku. Analogni vhod 0 Arduino™-PCB je priključen na presek med sprejemnikom in uporom. Če je sedaj fototranzistor izpostavljen svetlobi, bo postal prevoden. Padec napetosti med sprejemnikom in oddajnikom se bo zmanjšal. Ko je fototrazistor zatemnjen bo teklo zelo malo toka, fototrazistor se bo zaklenil in sprejemnik-oddajnik napetost se bo povečala. Sedaj merimo na območju skoraj celotnih 5 V.
Med dvema ekstremoma bo fototranzistor zelo dinamičen, odzval se bo tudi na majhne spremembe v svetlobi. Ker bo zaslon deloval obratno, zelo svetlo bi bila nizka vrednost in temno zelo visoka vrednost, moramo prilagoditi izmerjeno vrednost. Za to odštejemo izmerjeno vrednost od 100 %, da dobimo želen rezultat. Obrnemo analogno izmerjeno vrednost.
Če nočete da se stolpčni diagram neprenehoma spreminja, zaradi majhnih sprememb v svetlobi, ima funkcija stolpčni zaslon dodano povprečno obliko. Zgladi analogno izmerjeno vrednost in izračuna drseče povprečje, imenovano AVG za povprečje.
Trenutna izmerjena vrednost je zato dodana v Array v vsakem izvajanju glede na to, kako visoko vrednost ima števec. Neprekinjeno bo vračal povprečno vrednost. Število meritvenih serij za določitev povprečja je določeno s spremenljivko numReadings. Višja kot je vrednost vrednosti ki bodo povprečne, večja bo natančnost, vendar bo tudi dlje časa zaslon brez izpisa. Tukaj se lahko igrate z vrednostmi. Za najboljše so se izkazale vrednosti med 8 in 64.
Na koncu AVG funkcije, je izračunana vhodna vrednost (0 do 1023) za vrednost v procentih za funkcijo izpisa stolpčnega diagrama.
Takšen fotometer je lahko programiran tudi za vklapljanje ali izklapljanje luči ali, kot prikazuje naslednji primer, za uporabo alarma.

15. Alarmni sistem

Fotometer lahko uporabite tudi kot alarmni sistem, ki se odzove že na majhne spremembe svetlobe. Na začetku programa za alarm je določena trenutna moč osvetlitve, ki bo uporabljena na analognem vhodu A0. Če se vrednost napetosti v neprekinjenih meritvah, poveča ali zmanjša, zaradi sprememb v svetlobi (na primer ker se mimo sprehodi oseba) in nato preseže ali pride pod določeno mejo, se sproži alarm.
Ker se bo svetloba spreminjala čez cel dan, mora biti nova referenčna vrednost (trenutna napetost fotometra) določena samodejno vsakih 10 sekund, da je uporabljena kot nova referenčna točka za neprekinjene meritve.
Zato primerjamo točno določene vrednosti svetlobe, ki so merjene na novo vsakih 10 sekund, z vrednostjo svetlobe v neprekinjenih meritvah. Samo določeno odstopanje bo povzročilo sprožitev alarma, kar je +/- neka vrednost odstopanja od trenutne osvetlitve.

```
001 Schwelle = 25
002 if(analogRead(PHOTOTRANSITOR) > (value + Schwelle) ||
          analogRead(PHOTOTRANSITOR) < (value - Schwelle))
```

Če je vrednost nad 2000, bo prebrana nova vrednost iz analognega vhoda A0. Vstavljen premor, ki vpliva na hitrost izvajanja programa, bo zvišala spremenljivko cnt za 1 vsakih 5 ms. To vodi do vrednosti 5 ms x 2000 = 10000 ms = 10 sekund.

```
001 Schwelle = 25
002 if(analogRead(PHOTOTRANSITOR) > (value + Schwelle) ||
          analogRead(PHOTOTRANSITOR) < (value - Schwelle))
```

Tukaj je nastavljena vrednost spremenljivke imenovane Treshold. Odgovorna je za občutljivost sprožilca in ne sme biti nastavljena previsoko ali prenizko, ker bo alarmni sistem reagiral zelo slabo ali pa bo sprožil preveč lažnih alarmov. Ocena neprekinjenih meritev, ki je posneta vsakih 5 ms se bo izvedla v programski vrstici.
Da to preizkusite potegnite roko čez fototranzistor na razdalji približno 50 cm v normalno osvetljeni sobi in alarm se bo sprožil. To lahko storite tudi zelo hitro. Detektor vas bo zaznal če zatemnitev preseže 5 ms.
Alarmni sistem lahko namestite v hladilnik in lahko dodate vrednost števca v alarmno sporočilo, da izmerite cikle odprtja hladilnika. V poskusu s hladilnikom boste videli da LCD spremeni svoj kontrast in bo tudi bolj neaktiven. Poskusite.

Nalaganje
Uporabite enako nastavitev kot pri merilniku svetlobe!

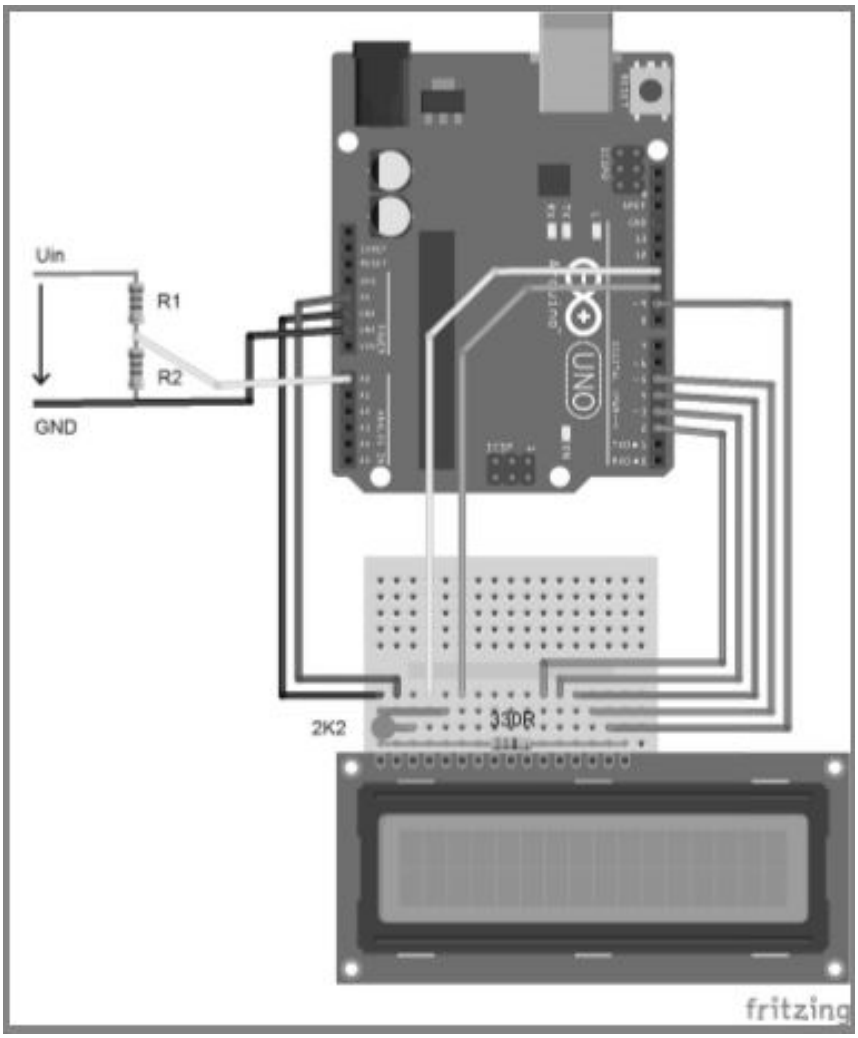
Primer kode: ALARM

```
001  // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define Backlight   9
009
010 int PHOTOTRANSITOR = 0;
011 int cnt = 0;
012 int value, Schwelle;
013
014 void setup()
015 {
016     analogWrite(Backlight, 200);
017     lcd.begin(16, 2);
018     lcd.setCursor(1, 0);
019     lcd.print("ALARMANLAGE!!!");
020     value = analogRead(PHOTOTRANSITOR);
021 }
022
023 void loop()
024 {
025
026   Schwelle = 25;
```

16. Digitalni voltmeter z zaslonom za stolpčni diagram in USB vmesnikom

S tem kar ste se naučili do sedaj lahko programirate digitalni voltmeter z analognim zaslonom za stolpčni diagram. Zaslon za stolpčni diagram je zelo uporaben za nastavitve. Natančneje lahko vidite kje se nahajata maksimum in minimum na analognem zaslonu, kot na digitalnem zaslonu s čistim numeričnim izpisom. Kot posebni dodatek, zagotovimo programu serijski izhod, ki pošlje vaše izmerjene podatke na računalnik preko USB

vmesnika. Tukaj uporabimo USB vmesnik,ki se že nahaja na Arduino™-UNO plošči in ga že uporabljate za programiranje.
Upornika R1 in R2 nista potrebna v tem poskusu in nista priložena učnemu paketu! Vendar bosta v nadaljevanju kljub temu razložena. Če ju potrebujete ju lahko dokupite v trgovini z elektroniko.



Sestavljanje digitalnega voltmetra.

Za zelo natančno izmero napetosti, med 0 in 5 V, lahko uporabite vezje in analogni izhod A0 brez uporabe upornikov R1/R2. Vendar zagotovite da ne priključite višje napetosti na povezavo, saj lahko s tem poškodujete Arduino™ ploščo. Zelo natančno lahko izmerite napetost ene izmed dveh Mignon celic (AA) ali mikro celic (AAA) z uporabo vezja. Primer je zelo podoben tistemu z fotometerom z nekaterimi manjšimi razlikami. Tokrat uporabimo tudi serijski vmesnik (UART = univerzalni asinhroni sprejemnik oddajnik) na Arduino™ mikrokrmilniku. Izmerjeni podatki so poslani preko serijskega vmesnika na mikrokrmilnik (UART) na pretvornik UART v USB na Arduino™-PCD, ki pošlje podatke na računalnik. Serijski povezavi D0/RX in D1/TX sta trdno povezani na pretvornik in ni potrebe po dodatnem ožičenju. Na strani računalnika se naredi navidezni vhod, ko namestite Arduino™-PCD. To se že uporablja za programiranje. Sedaj ga lahko uporabimo za prenos podatkov na računalnik. Zato moramo sprožiti UART vmesnik v programu. Ta je nastavljen z Serial.begin(). Parameter 19200 med oklepaji predstavlja hitrost prenosa. Sprožitev mora biti izvedena samo enkrat na začetku programa v funkciji Setup().

```
001 Serial.begin(19200)
```

Baud je enota za hitrost prenosa v telekomunikacijski tehnologiji. 19200 Baud pomeni, da bo prenesenih 192000 simbolov na sekundo. Znak za hitrost lahko vsebuje različno število bitov glede na kodiranje in mora biti nastavljeno enako na oddajniku in sprejemniku da je prenos dovoljen.
Sledeče vrstice so sedaj uporabljene za pošiljanje merjenih vrednosti ADC (0 do 1013) neposredno na računalnik brez predhodne pretvorbe. Pretvorba v Volte se zgodi v računalniškem programu, ker moramo poslati samo dva posamezna bajta na računalnik, ki jih program veliko lažje oceni kot pa niz (ASCII znakovni niz).
Sedaj bo spomin UART vmesnika izpraznjen z ukazom flush.

```
001 Serial.flush()
```

Tako bomo razstavili analogno izmerjene vrednosti, ki imajo razpon med 0 in 102, v visoke in nizke bajte. Visok bajt dobite tako, da delite izmerjeno vrednost z 256.

```
001 highbyte = adc_raw / 256
```

Nizki bajt dobimo z modulo (iskanjem ostanka) operacijo 256.

```
001 lowbyte = adc_raw % 256
```

Nato pošljemo na računalnik najprej visoke in nato nizke bajte.

```
001 Serial.write(highbyte)
002 Serial.write(lowbyte)
```

Za preverjanje če so bile poslane prave vrednosti, lahko preverimo vsoto na koncu prenosa, ki bi naj bila točna številka in XOR formacija tega, kot tudi dva bajta.

```
001 crc = 170^highbyte^lowbyte
002 Serial.write(crc)
```

Informacije
Program deluje tudi brez računalniškega programa in je lahko uporabljen kot samostojen voltmeter.

Primer kode: VOLTMETER

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define LCD_LENGHT  16.0
009 #define ADC_CHANNEL 0
010
011 const int numReadings = 20;
012 unsigned int readings[numReadings];
013 int index = 0;
014 unsigned int total = 0;
015
016 byte MyChar0[8] = {
017 B00000,
018 B00000,
019 B00000,
020 B00000,
021 B00000,
022 B00000,
023 B00000,
024 B00000,
025 };
026
027 byte MyChar1[8] = {
028 B10000,
029 B10000,
030 B10000,
031 B10000,
032 B10000,
033 B10000,
034 B10000,
035 B10000,
036 };
037
038 byte MyChar2[8] = {
039 B11000,
040 B11000,
041 B11000,
042 B11000,
043 B11000,
044 B11000,
```

```
045 B11000,
046 B11000,
047 };
048
049 byte MyChar3[8] = {
050 B11100,
051 B11100,
052 B11100,
053 B11100,
054 B11100,
055 B11100,
056 B11100,
057 B11100,
058 };
059
060 byte MyChar4[8] = {
061 B11110,
062 B11110,
063 B11110,
064 B11110,
065 B11110,
066 B11110,
067 B11110,
068 B11110,
069 };
070
071 byte MyChar5[8] = {
072 B11111,
073 B11111,
074 B11111,
075 B11111,
076 B11111,
077 B11111,
078 B11111,
079 B11111,
080 };
081
082 int adc_AVG(byte channel)
083 {
084     total= total - readings[index];
085     readings[index] = analogRead(channel);
086     total= total + readings[index];
087     index = index + 1;
088     if (index >= numReadings)index = 0;
089     return total / numReadings;
090 }
```

```
091
092 void draw_bargraph(byte percent)
093 {
094   byte i, c1, c2;
095
096   lcd.setCursor(0, 1);
097
098   percent = map(percent, 0, 100, 0, 80);
099
100   c1 = percent / 5;
101   c2 = percent % 5;
102
103   for(i = 0; i < c1; ++i)
104   lcd.write(byte(5));
105
106   lcd.write(c2);
107
108   for(i = 0; i < 16 - (c1 + (c2 ? 1 : 0)); ++i)
109   lcd.write(byte(0));
110 }
111
112 void setup()
113 {
114     analogWrite(9, 200);
115
116     lcd.createChar(0, MyChar0);
117     lcd.createChar(1, MyChar1);
118     lcd.createChar(2, MyChar2);
119     lcd.createChar(3, MyChar3);
120     lcd.createChar(4, MyChar4);
121     lcd.createChar(5, MyChar5);
122     lcd.begin(16, 2);
123
124     Serial.begin(19200);
125 }
126
127 void loop()
128 {
129     double percent;
130     float voltage;
131     byte highbyte, lowbyte, crc;
132     int adc_raw = adc_AVG(ADC_CHANNEL);
133
134     voltage = (5.0 / 1024.0) * adc_raw;
135
136     lcd.setCursor(0, 0);
```

```
137     lcd.print(voltage, 2);
138     lcd.print(" V   ");
139
140     percent = voltage / 5.0 * 100.0;
141     draw_bargraph(percent);
142
143     Serial.flush();
144     highbyte=adc_raw/256;
145     lowbyte=adc_raw%256;
146     Serial.write(highbyte);
147     Serial.write(lowbyte);
148     crc=170^highbyte^lowbyte;
149     Serial.write(crc);
150
151     delay(20);
152 }
```

VB.NET program v delovanju.

Da zaženete računalniški program, morate zagnati EXE datoteko v mapi ...\VOLTMETER\vb.net\bin\Release. Nato izberete Comport (vhod), ki je enak tistemu, ki ste ga že nastavili v Arduino™-IDE za prenos programa. Če kliknete Connect, bo prikazana napetost v računalniškem programu.

VB.NET program je priložen kot izvorna koda in je lahko uporabljen kot osnova za vaše poskuse. Zato si morate naložiti brezplačen Visual-Basic-Express-Version podjetja Microsoft. Najdete ga na https://www.visualstudio.com/downloads/download-visual-studio-vs. Ko odprete program z Visual Basic, boste videli izvorno kodo in oblikovalca. Oblikovalec izpiše vizualne nadzorne elemente kot so tipke, tekstovna polja itd. Sedaj si poglejte izvorno kodo programa Voltmeter s preklopom na pogled za izvorno kodo.

```
001 Imports System.IO.Ports.SerialPort
002 Imports System.Text.Encoding
```

Najprej bomo uvozili funkcije potrebne za serijski vmesnik in kodiranje. Brez uvoza Encoding knjižnice ne bo mogoča ocena vrednosti ki presegajo 128, ker ne moremo preklopiti vmesnika na UTF-8 format. Izmerjeni rezultati bodo napačni.

```
001 Dim input_data(10) As Byte
```

input_data(10) je uporabljena za nastavitev Arraya ki lahko vzame do 10 bajtov. Prejeti bajti iz serijskega vmesnika bodo vpisani kasneje.

```
001 Private Sub Form1_Load(ByVal sender As System.Object,
        ByVal e As System.EventArgs) Handles MyBase.Load
```

V funkciji Form1_load() bodo poiskani trenutni računalniški vhodi (Comport). Zapisani so v posebnem seznamu. Funkcija Form1_Load() bo samodejno klicana ob zagonu program, podobno kot Setup() funkcija v Arduino™ programu.

```
001 Private Sub Button_Connect_Click(ByVal sender As System.
            Object, ByVal e As System.EventArgs) Handles
            Button_Connect.Click
```

V funkciji Button_Connect_Click() lahko nastavite in odprete serijske vmesnike istočasno. Ta funkcija je klicana ko pritisnete tipko Connect.

```
001 SerialPort1.PortName = ComboBox_Comport.Text
002 SerialPort1.BaudRate = 19200
003 SerialPort1.Encoding = System.Text.Encoding.UTF8
004 SerialPort1.Open()
```

Tukaj določimo Comport, Baud hitrost in Encoding (kodiranje) ter odpremo vmesnik z SerialPort1.Open(), ki bo pripravljen za prenos in prejemanje.

```
001 Private Sub Button_Disconnect_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs) Handles
        Button_Disconnect.Click
```

Funkcija Button_Disconnect_Click() zapre vmesnik. Sedaj je na voljo tudi drugim programom, kot je Arduino™-IDE. Če želite prenesti spremembe v programu ali druge programe na Arduino™ ploščo, morate prekiniti program ali klikniti Disconnect, da znova sprostite vmesnik.

```
001 Private Sub SerialPort1_DataReceived(sender As Object
        , e As System.IO.Ports.SerialDataReceivedEventArgs)
        Handles SerialPort1.DataReceived
002
003         Dim cnt As Byte
004         Dim in_bytes As Byte
005         Dim HighByte As Byte
006         Dim LowByte As Byte
007         Dim crc As Byte
008         Dim crc_ok As Byte
009         Dim data_Word As Integer
010         Dim voltage As Single
011
012         Try
013
014             ' Hier werden die Daten empfangen
015             If SerialPort1.IsOpen Then
016
017                 Control.CheckForIllegalCrossThreadCalls
                                                    = False
018
019                 ' wie viele Bytes sind im Puffer
020                 in_bytes = SerialPort1.BytesToRead
021
022                 ' Alle Bytes holen
023                 For cnt = 1 To (in_bytes)
024                   input_data(cnt) = SerialPort1.ReadByte
025                 Next
026
027                 ' Puffer leeren
028                 SerialPort1.DiscardInBuffer()
029
030                 HighByte = input_data(1)
031                 LowByte = input_data(2)
032                 crc = input_data(3)
033
034                 ' Checksumme
035                 crc_ok = 170 Xor input_data(1) Xor input_
```

```
                                                    data(2)
036
037                 If crc = crc_ok Then
038
039                     ' High und Low Byte wieder
                                                zusammensetzen
040                 data_Word = ((HighByte * 256) + LowByte)
041                    voltage = data_Word * (5.0 / 1024.0)
042
043                     ' RAW Wert umrechen und als Spannung
                                                    Anzeigen
044                  Label1.Text = Format(voltage, "0.00 V")
045
046                 End If
047
048             End If
049
050         Catch ex As Exception
051         End Try
052
053     End Sub
```

V funkciji SerialPort1_DataReceived() bo sedaj sledil najbolj zanimiv del programa VB.NET. Ta funkcija bo klicana vsakič ko bodo prejeti podatki serijskega vmesnika. Tukaj preberemo bajte ki jih pošlje program Arduino™ in jih takoj procesira. Pred branjem in procesiranjem vedno preverimo ali je povezava najprej odprta in nato preverimo koliko bajtov je na voljo v sprejemnem spominu. Nato preberemo bajte v input_data() in jim določimo prejete vrednosti spremenljivkam HighByte, LowByte in crc. Na koncu preverimo, če se izračunana in prejeta vsota ujemata, da ni prišlo do napake med prenosom. Sedaj izračunamo vrednost izmerjenih meritev. Da dobimo določeno obliko uporabimo funkcijo Format() v VB.NET in izpišemo urejeno vrednost v Label1.

16.1 Razširitev merilnega območja

Če želite meriti višje napetosti, potrebujete prenapetostni delivec, sestavljen iz upornika R1 in R2. Lahko ju uporabite za razširitev območja vhodne napetosti po želji. Vendar se bo ločljivost zmanjšala z večjim merilnim območjem.
V našem primeru, ki je namenjen za vhodno napetost 5 V, je ločljivost 0,00488 V ali 4,88 mV na pretvorbeni korak. Digitalna vrednost analognega vhoda lahko razreši 1024 korakov,ker ima digitalno ločljivost 10 bitov.

Koraki pretvorbe (koraki): $1024 = 2^{10}$
Ločljivost na število = U_{ADC}/koraki
5 V / 1024 = 0,00488 V = 4,88 mV

Če bi dvignili vhodno napetost razpona na 30 V, bi se ločljivost zmanjšala za petkrat ((30 V – 5 V) / 5 V = 5). Delali bi z ločljivostjo 0,0244 V = 24,4 mV.
Sedaj lahko določimo delivec napetosti za območje delovanja do 30 V. Kot že vemo hočemo razširiti vhodno območje iz 5 V na 30 V, kar ustreza faktorju 5. Vhodni signal za merjenje napetosti ne sme imeti prenizkih Ohmov in mora biti vsaj 100 kΩ. Sodobni voltmetri imajo, za primerjavo, vhodno upornost 10 MΩ, da dajo kar najmanj obremenitve na vir napetosti in da se izognejo napačnim rezultatom meritev.

Da bo vhodna upornost približno 100 kΩ določimo upornik R1 pri 100 kΩ. V serijskem vezju je razmerje napetosti enako razmerju upornikov med seboj. Upornik R2 mora biti le 1/5 tako veliko kot je upornik R1. Vzamemo 100 kΩ/5 in dobimo vrednost 20 kΩ za R2.
Sedaj določimo tok ki teče v vezju. V serijskem vezju je tok skozi upornik enak in napetosti bodo deljene. Tok skozi upornika je izračun sledeče:

$I = U / R_{Total}$
30 V / 120 kΩ = 0,25 mA

Sledeč padec napetosti bo na R2:
U = R2 x I 20 kΩ x 0,25 mA = 5 V

Pri uporabljeni napetosti 15 V bo vrednost ADC:
15 V / 120 kΩ = 0,125 mA
20 kΩ x 0,125 mA = 2,5 V

Izračunajmo izgubo moči skozi upornik pri maksimalni napetosti:
P = U x I
30 V x 0,25 mA = 7.5 mW

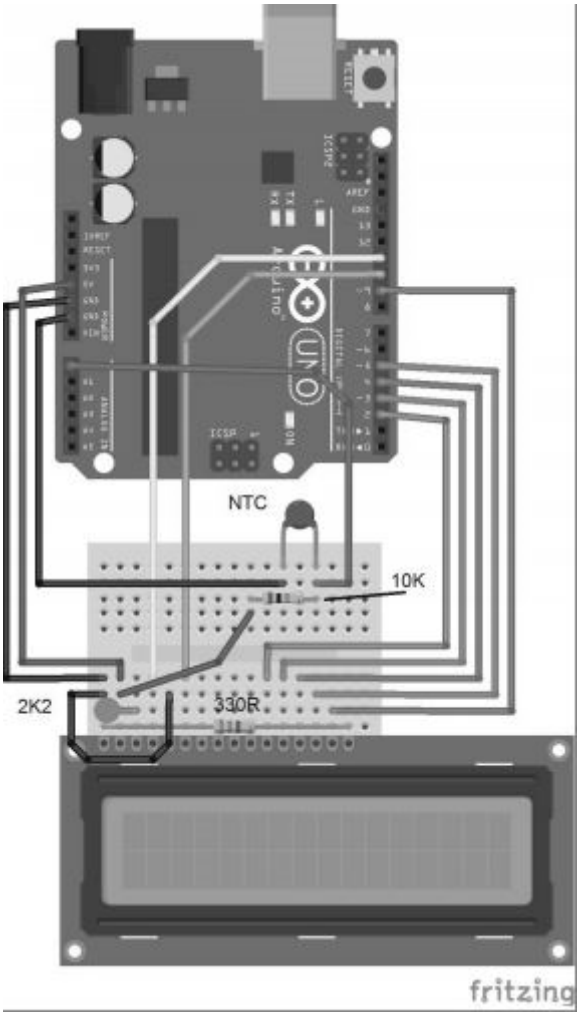
Sedaj lahko izračunate vaš delivec napetosti in prilagodite merjeno vrednost v programu z pomnožitvijo z vašim faktorjem delivca napetosti. Meritvena napaka je lahko določena s poskušanjem z natančnim voltmetrom in je lahko nato vključena v izračun faktorja za pomnožitev. Opazujte vrednost upornika E-serije in ne poskusite vključiti matematično določenih vrednosti v vezju z uporabo E-serije ali zamenjave upornikov v seriji ali vzporedno, da bi dobili natančno izmerjeno vrednost.
Ta izračun ne upošteva vhodne upornosti ADC. V Arduino™-UNO ATmega328 znaša približno 100 kΩ.
Zato ne smete iti višje kot 100 kΩ z delivcem vhodne napetosti da dobite uporabne merilne rezultate. Za bolj natančne rezultate lahko naredite izračun za obremenjen delivec napetosti in vključite upor ADC in R2.

Upoštevajte tudi da se toleranca sešteva v serijskih vezjih. Če uporabljate upornik s 5 % toleranco za prikazano vezje, je skupna toleranca že 10 %.
Vrednosti E12-serije so se izkazale kot najboljše. Na voljo so v vsaki elektronski trgovini in so del standardne serije. Ko izdelujete merilnike morate paziti da je toleranca komponent čim nižja za natančnejšo meritev.

Nasvet
Sledeča povezava vas bo usmerila na spletno orodje za izračun delivcev napetosti:
http://www.peacesoftware.de/einigewerte/spannungsteiler.html

17. Temperaturni zaslon v stopinjah Celzija in Fahrenheit

Ta poskus prikazuje kako lahko uporabite stroškovno učinkovit temperaturni upor kot NTC (negativni temperaturni koeficient termistor) ki je tukaj uporabljen za programiranje preprostega LCD termometra.



Sestavljanje NTC-LCD termometra. Povezava je podobna kot pri fotometru z razliko, da je namesto fotranzistorja uporabljen NTC.

NTC je upor ki spreminja svoj upor glede na temperaturo. NTC se imenuje vroč prevodnik. To pomeni, da se njegova upornost zmanjša, ko se temperatura poveča.

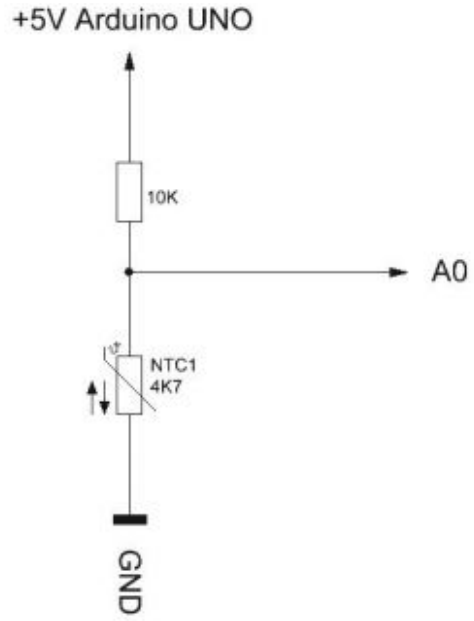


Diagram povezave NTC termometra.

Diagram vezja prikazuje to nastavitev podrobneje. To je še ena spremenljivka delilnika napetosti, ki sestoji iz 10 kΩ fiksnega upora in spremenljivega NTC-upora. Ko temperatura pade, se upornost NTC povečuje in tako tudi napetost na analognem vhodu A0.

Primer kode: LCD THERMO

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define Backlight   9
009 #define ADC_NTC     0
010
011 float temp_celsius, temp_fahrenheit;
012 int ADC_raw;
013
014 float Grad_to_Fahrenheit(float grad)
015 {
016     return (9.0 / 5.0) * grad + 32;
017 }
018
019 void setup()
020 {
021     analogWrite(Backlight, 200);
022     lcd.begin(16, 2);
023     lcd.setCursor(0, 0);
024     lcd.print("THERMO - ARDUINO");
025     Serial.begin(9600);
026     delay(2000);
027     lcd.clear();
028 }
029
030 void loop()
```

```
031 {
032     ADC_raw = analogRead(ADC_NTC);
033     temp_celsius = (580.0 - ADC_raw) / 10;
034     temp_fahrenheit = Grad_to_Fahrenheit(temp_celsius);
035 
036     lcd.setCursor(0, 0);
037     lcd.print(temp_celsius, 1);
038     lcd.write(223);
039     lcd.print("C  ");
040 
041     lcd.setCursor(0, 1);
042     lcd.print(temp_fahrenheit, 1);
043     lcd.write(223);
044     lcd.print("F  ");
045 
046     Serial.print("Temperatur = ");
047     Serial.print(temp_celsius);
048     Serial.print(" °C");
049 
050     Serial.print(" | ");
051     Serial.print(temp_fahrenheit);
052     Serial.println(" °F");
053 
054     delay(1000);
055 }
```

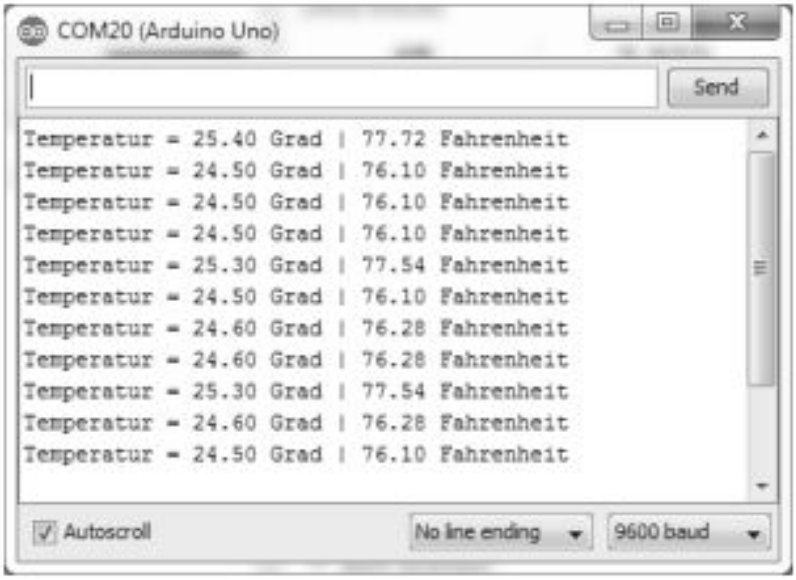
Krivulja upornosti NTC ni ravno linearna in zato jo je potrebno prilagoditi z izračunom.

```
001 temp_celsius = (580.0 - ADC_raw) / 10
```

Če želite prejeti prikaz, ne le v stopinjah Celzija, se bo vrednost pretvorila v stopinje Fahrenheita in prikazala na zaslonu.

```
001 float Grad_to_Fahrenheit(float grad)
002 {
003     return (9.0 / 5.0) * grad + 32;
004 }
```

Kot lahko vidite, lahko zapišemo izračune naenkrat po ukazu return in ni potrebno, da najprej predate spremenljivko. Ta funkcija izračuna vrednost v stopinjah Fahrenheit, kot se uporablja v ameriškem sistemu, ki pa temelji na stopinjah Celzija.



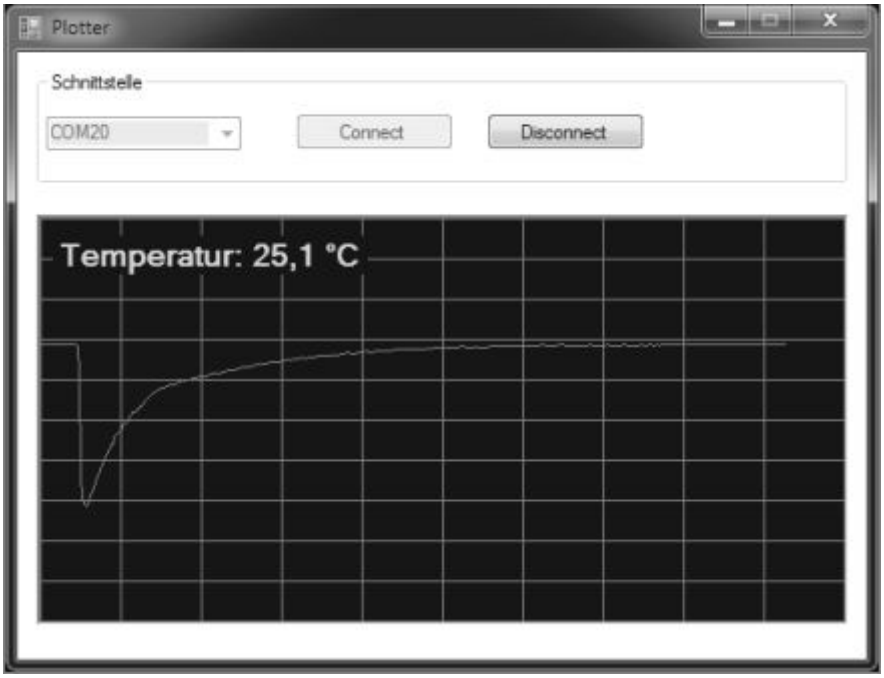
V tem programu je tudi uporabljen serijski vmesnik in enaka vrednost je prikazana na LCD in dodatno preko ASCII niza preko UART vmesnika.
Pri odpiranju notranjega terminalskega programa Arduino ™ in nastavitvi lastnega vmesnika za 9600 Baud, boste prejeli izmerjeno vrednosti v golo besedilo v terminalski program.

18. Ploter za temperaturo z USB vmesnikom

Razširimo termometer z VB.NET-programom in spremenimo programsko kodo, tako da se bo golo besedilo prikazano kot predhodnik programa nadomeščen le s pošiljanjem temperaturne vrednosti na računalnik. Vezje ostaja enako, vendar se program spremeni, kot sledi:

```
001 Serial.flush()
002 highbyte=ADC_raw/256
003 lowbyte=ADC_raw%256
004 Serial.write(highbyte)
005 Serial.write(lowbyte)
006 crc=170^highbyte^lowbyte
007 Serial.write(crc)
```

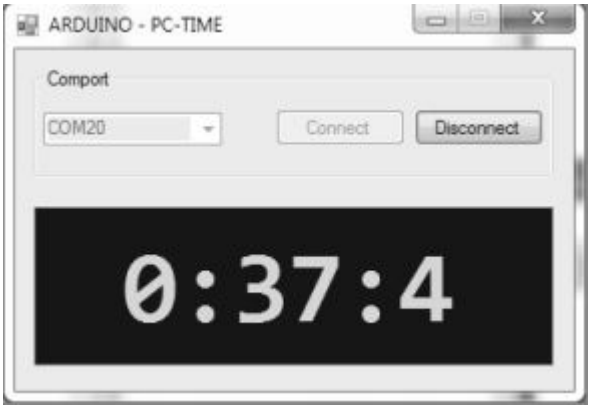
Kot lahko vidite, smo prenesli vrednost temperature, kot so izmerjene vrednosti napetosti v digitalnem USB voltmetru. Tu lahko najdete celoten program na priloženem CD-ju. To se imenuje »TEMP_PLOT«. Programske kode ni potrebno našteti preko tega, saj ustreza predhodnemu programu.
Vendar so se nekatere stvari spremenile v VB.NET programu. Ta se je razširil z grafično močjo. Za to je bila programirana kontrola elementa z imenom »AutoRedraw«, ki riše neprekinjeno črto v X in Y smeri, odvisno od vhodne spremenljivke in služi kot temperaturni ploter. Ta program je priložen kot izvorna koda in se lahko uporablja za lastne poizkuse. Podroben opis funkcij za risanje pa bi presegel obseg tega učnega paketa. Uporabite lahko sorodne spletne strani, kot na primer www.vb-paradise.de in VB.NET priročnike, če želite izvedeti več o VB.NET programiranju.



Temperaturni ploter v uporabi.

19. Ura Websynchronous

V tem učnem paketu smo že programirali uro. Ker to ni zelo natančna ura in ima zelo veliko odstopanje v okviru obratovalnega časa, bomo sedaj programirali uro Websynchronous, s poznavanjem serijskega prenosa podatkov med računalnikom in Arduino ™. To je Websynchronous, ker je Windows čas kot privzeto samodejno usklajen s spletnim časovnim strežnikom v ozadju. V tem VB.NET programu bomo sedaj poslali čas računalnika na Arduino ™ in ga oddajali na LCD. VB.NET program je priložen kot izvršljiva EXE datoteka in kot izvorna koda.



Računalnik pošlje čas na Arduino™ ploščo.

Arduino™ z natančnejšim prikazom časa.

Nalaganje
Poskus zahteva osnovno pisanje LCD, ki ste ga nastavili v testu funkcije.

Primer kode: PC čas

```
001 // LCD-Library einbinden
002 #include <LiquidCrystal.h>
003
004 // LCD-Pins festlegen
005 // RS, E, D4, D5, D6, D7
006 LiquidCrystal lcd(11, 10, 2, 3, 4, 5);
007
008 #define Backlight   9
009
010 byte Stunde, Minute, Sekunde;
011
012 void setup()
013 {
014     Serial.begin(9600);
015
016     analogWrite(Backlight, 200);
017
018     lcd.begin(16, 2);
019     lcd.clear();
020     lcd.setCursor(0, 0);
021     lcd.print("ARDUINO PC-UHR");
022
023     Stunde = 0;
024     Minute = 0;
025     Sekunde = 0;
026 }
027
028 void loop()
029 {
030
031     if(Serial.available()>3)
032     {
033         Stunde = Serial.read();
034         Minute = Serial.read();
035         Sekunde = Serial.read();
036
037         lcd.setCursor(0, 1);
038         lcd.print("NOW: ");
039
040         if(Stunde < 24)
041         {
042             if(Stunde < 10) lcd.print("0");
043             lcd.print(Stunde);
044             lcd.print(":");
045         }
046         if(Minute < 60)
047         {
048             if(Minute < 10) lcd.print("0");
049             lcd.print(Minute);
050             lcd.print(":");
051         }
```

```
052         if(Sekunde < 60)
053         {
054             if(Sekunde < 10) lcd.print("0");
055             lcd.print(Sekunde);
056         }
057     }
058
059     Serial.flush();
060     delay(100);
061 }
```

Osnovna struktura programa ustreza točnemu času (RTC - Real Time Clock), ki smo se ga že naučili na začetku učnega paketa.

Tokrat nismo prejeli nobenih podatkov v VB.NET programu, ampak poslane podatke iz računalnika na Arduino ™. Podatki ustrezajo uram, minutam in sekundam v obliki bajtov. Beremo jih z Arduino ™, ko je Serial.available () večji od tri. To je značilno da so trije bajti v teku v sprejemnem vmesnem pomnilniku. Uporabljamo Serial.read () za sistematično branje v treh bajtih ki prihajajo in jih dodelijo Arduinu ™ spremenljivke za ure, minute in sekunde. Tako je na strani Arduino ™. Poglejmo si funkcijo prenosa v programu VB.NET. Za to smo uporabili časovnik nastavljen na 500 ms.

```
001 Private Sub Timer1_Tick(sender As System.Object, e As
                    System.EventArgs) Handles Timer1.Tick
002
003         If SerialPort1.IsOpen Then
004
005             SerialPort1.Write(Chr(Now.Hour) & Chr(Now.
                                Minute) & Chr(Now.Second))
006             Label1.Text = Now.Hour & ":" & Now.Minute &
                                        ":" & Now.Second
007
008         End If
009 End Sub
```

Časovnik lahko nastavite tudi na 1000 ms. Možno je tudi, da bo drugi izhod poskočil na LCD ker se podatki prekrivajo. Bolje je, da nastavite čas posodabljanja malo hitreje ali pa samo do polovice, da bi prikaz bil videti bolj tekoče.
V funkciji VB.NET-timer se preverjanje opravi pred dejanskim prikazom za varnostne namene, da se ugotovi ali se serijska povezava odpre. Če se to zgodi bo uporabljen SerialPort1.Write () za prenos časovnih podatkov. Za to smo uporabili funkcijo Now ().Vsebuje čas in datum in se lahko naroči s parametri uro, minuto in sekundo, za prikaz samo želenih mest. Za pretvorbo vrednosti za prenos bajtov bo vsaka vrednost, ki se pretvori z Char(), en bajt.
Naša PC ura sedaj kaže točen čas na LCD.

20. Dodatek

20. 1 Merske enote za elektriko

Razlikujemo med napetostjo, tokom, uporom in enotami, v katerih so izmerjene vrednosti (npr. Volt ali Amper). Vsaka merska enota ima kratico, ki jo uporabljamo v enačbah. Kratice omogočajo kratek in dobro strukturiran zapis.
Namesto toka, ki je 1 Amper pišemo samo I = 1 A.

Te kratice so uporabljene v vseh enačbah v tem priročniku.

Vrednost	**Kratica**	**Enota**	**Kratica**
Napetost	U	Volt	V
Tok	I	Amper	A
Upor	R	Ohm	Ω
Moč	p	Watt	W
Frekvenca	f	Hertz	Hz
Čas	t	sekunda	s
Valovna dolžina	Λ (Lambda)	meter	m
Induktivnost	L	Henry	H
Kapaciteta	C	Farad	F
Ploščina	A	kvadratni meter	m^2

20. 2 ASCII tabela

Znak	**Decimalka**	**Heksadecimalka**	**Binarno**	**Opis**
NUL	000	000	00000000	Ničti znak
SOH	001	001	00000001	Začetek v glavi
STX	002	002	00000010	Začetek teksta
ETX	003	003	00000011	Konec teksta
EOT	004	004	00000100	Konec prenosa
ENQ	005	005	00000101	Poizvedba
ACK	006	006	00000110	Odgovor
BEL	007	007	00000111	Zvonec
BS	008	008	00001000	Vračalka
HAT	009	009	00001001	Vodoraven TAB
LF	010	00A	00001010	Dovod
VT	011	00B	00001011	Navpični TAB
FF	012	00C	00001100	Vnos oblike
CR	013	00D	00001101	Vračalka
SO	014	00E	00001110	Pomik ven
SI	015	00F	00001111	Pomik notri
DLE	016	010	00010000	Povezava za podatkovni izhod
DC1	017	011	00010001	Nadzor nad napravo 1
DC2	018	012	00010010	Nadzor nad napravo 2
DC 3	019	013	00010011	Nadzor nad napravo 3
DC4	020	014	00010100	Nadzor nad napravo 4
NAK	021	015	00010101	Negativni odgovor
SYN	022	016	00010110	Sinhron prosti tek
ETB	023	017	00010111	Konec prenosnega bloka
CAN	024	018	00011000	Prekiniti
EM	025	019	00011001	Konec medija
SUB	026	01A	00011010	Zamenjava
ESC	027	01B	00011011	Izhod
FS	028	01C	00011100	Izločevalec datotek
GS	029	01D	00011101	Izločevalec skupin
RS	030	01E	00011110	Prošnja za pošiljanje, izločevalec snemanja
US	031	01F	00011111	Izločevalec enot
SP	032	020	00100000	Presledek
!	033	021	00100001	Klicaj
«	034	022	00100010	Dvojni narekovaj

#	035	023	00100011	Znak za števila
$	036	024	00100100	Dolar znak
%	037	025	00100101	Procent
&	038	026	00100110	Ampersand
‘	039	027	00100111	Enojni narekovaj
(	040	028	00101000	Levi odprti oklepaj
)	041	029	00101001	Desno odprti oklepaj
*	042	02A	00101010	Zvezdica
+	043	02B	00101011	Plus
,	044	02C	00101100	Vejica
-	045	02D	00101101	Minus ali pomišljaj
.	046	O2E	00101110	Pika
/	047	O2F	00101111	Poševnica
0	048	030	00110000	
1	049	031	00110001	
2	050	032	00110010	
3	051	033	00110011	
4	052	034	00110100	
5	053	035	00110101	
6	054	036	00110110	
7	055	037	00110111	
8	056	038	00111000	
9	057	039	00111001	
:	058	O3A	00111010	Dvopičje
;	059	O3B	00111011	Podpičje
<	060	O3C	00111100	Manj kot
=	061	O3D	00111101	Enako
>	062	O3E	00111110	Več kot
?	063	O3F	00111111	Vprašaj
@	064	040	01000000	@ znak
A	065	041	01000001	
B	066	042	01000010	
C	067	043	01000011	
D	068	044	01000100	
E	069	045	01000101	
F	070	046	01000110	
G	071	047	01000111	
H	072	048	01001000	
I	073	049	01001001	
K	075	04B	01001011	
L	076	04C	01001100	
M	077	04D	01001101	
N	078	04E	01001110	
O	079	04F	01001111	
P	080	050	01010000	
Q	081	051	01010001	
R	082	052	01010010	
S	083	053	01010011	
T	084	054	01010100	
U	085	055	01010101	
V	086	056	01010110	
W	087	057	01010111	
X	088	058	01011000	

Y	089	059	01011001	
Z	090	05A	01011010	
[	091	05B	01011011	Levi zaprti oklepaj
\	092	05C	01011100	Leva poševnica
]	093	05D	01011101	Levi zaprti oklepaj
^	094	05E	01011110	Strešica
_	095	05F	01011111	Podčrtaj
`	096	060	01100000	
a	097	061	01100001	
b	098	062	01100010	
c	099	063	01100011	
d	100	064	01100100	
e	101	065	01100101	
f	102	066	01100110	
g	103	067	01100111	
h	104	068	01101000	
i	105	069	01101001	
j	106	06A	01101010	
k	107	06B	01101011	
l	108	06C	01101100	
m	109	06D	01101101	
n	110	06E	01101110	
o	111	06F	01101111	
p	112	070	01110000	
q	113	071	01110001	
r	114	072	01110010	
s	115	073	01110011	
t	116	074	01110100	
u	117	075	01110101	
v	118	076	01110110	
w	119	077	01110111	
x	120	078	01111000	
y	121	079	01111001	
z	122	07A	01111010	
{	123	07B	01111011	Levi odprti oklepaj
|	124	07C	01111100	Ravna črta
}	125	07D	01111101	Desni odprti oklepaj
~	126	07E	01111110	Tilda
DEL	127	07F	01111111	Brisanje

21. Vir naročanja

Conrad Electronic SE Klaus-Conrad-Straße 1
92240 Hirschau
www.conrad.de
Electronic Assembly GmbH Zeppelinstraße 19
82205 Gilching bei München
www.lcd-module.de

GARANCIJSKI LIST

Conrad Electronic d.o.o. k.d.
Ljubljanska c. 66, 1290 Grosuplje
Fax: 01/78 11 250, Tel: 01/78 11 248
www.conrad.si, info@conrad.si

Izdelek: Učni paket Conrad Components Arduino™ razumeti in uporabljati 10174 od 14. leta starosti naprej
Kat. št.: 138 41 45

Garancijska izjava:
Proizvajalec jamči za kakovost oziroma brezhibno delovanje v garancijskem roku, ki začne teči z izročitvijo blaga potrošniku. **Garancija velja na območju Republike Slovenije. Garancija za izdelek je 1 leto**.
Izdelek, ki bo poslan v reklamacijo, vam bomo najkasneje v skupnem roku 45 dni vrnili popravljenega ali ga zamenjali z enakim novim in brezhibnim izdelkom. Okvare zaradi neupoštevanja priloženih navodil, nepravilne uporabe, malomarnega ravnanja z izdelkom in mehanske poškodbe so izvzete iz garancijskih pogojev. **Garancija ne izključuje pravic potrošnika, ki izhajajo iz odgovornosti prodajalca za napake na blagu**.
Vzdrževanje, nadomestne dele in priklopne aparate proizvajalec zagotavlja še 3 leta po preteku garancije.
Servisiranje izvaja proizvajalec sam na sedežu firme CONRAD ELECTRONIC SE, Klaus-Conrad-Strasse 1, Nemčija.
Pokvarjen izdelek pošljete na naslov: Conrad Electronic d.o.o. k.d., Ljubljanska cesta 66, 1290 Grosuplje, skupaj z izpolnjenim garancijskim listom.

Prodajalec: _____

Datum izročitve blaga in žig prodajalca:
_____

Garancija velja od dneva izročitve izdelka, kar kupec dokaže s priloženim, pravilno izpolnjenim garancijskim listom.