

U kraljevstvu Arduina

Igor S. RUŽIĆ, Svet kompjutera, 2016

Početak priče o poznatom malenom elektronskom uređaju nas vodi u 2005. godinu, na severozapad Italije, u regiju Pijemont, grad Ivrea i tamošnji fakultet interaktivnog



dizajna, gde su predavač Masimo Banzi i student Dejvid Melis krenuli na zadatak da stvore hardversku razvojnu platformu pristupačnu plitkim studentskim džepovima, zasnovanu na mikrokontroleru *ATmega128*. Važnom se pokazala zamisao da ta platforma radi sa programskim frejmworkom pod nazivom *Wiring*, čiji je autor kolumbijac Hernando Baragan. Sticajem okolnosti, Banzi je imao ulogu Baraganovog postdiplomskog supervizora, pa je dobio ideju da *Wiring* bude iskorišćen za programiranje mikrokontrolera u njihovom uređaju namenjenom studentima interaktivnog dizajna (dakle, više umetnicima nego programerima). *Wiring* je u suštini pojednostavljena verzija programskog jezika C/C++ optimizovana za obraćanje hardveru. Osim toga, vidljiv je uticaj programske platforme *Processing* koja ima za cilj da omogući programiranje onima koji se u njega mnogo ne razumeju. Pokazalo se da je to bila dobitna kombinacija, pošto nova platforma ne samo da je bila jeftina, već je prosto plenila svojom jednostavnošću implementacije hardverskih i softverskih rešenja. *Arduino* nikada nije bio sinonim za vrhunske računarske performanse. Naprotiv, radilo se o vrlo skromnom i jednostavnom hardveru koji u poređenju sa procesorskom snagom i memorijskim kapacitetima današnjih najjeftinijih kineskih tableta izgleda smešno. Sam uređaj po svojoj konstrukciji je toliko prost da bi ga bez problema mogao sastaviti i nešto talentovaniji početnik u svetu elektronike. Ali, da parafriziramo Njegoša i njegove reči o Vuku Mandušiću, umeće korišćenja nekog uređaja važnije je od njegove sirove snage. Uspeh *Arduina* leži u kombinaciji niske cene, brzine postizanja rezultata, prostote priključivanja periferala, jednostavnosti programiranja i mogućnosti proširivanju osnovnih mogućnosti programskog jezika sa dodatnim bibliotekama. Sve to je uticalo na promenu načina razmišljanja o elektronici koje je od pomalo dosadnog, standardnog

inženjerskog pristupa sada okrenuto prema privlačnijem i zanimljivijem principu „uradi sam“. I što je najvažnije, takav pristup donosi rezultate i elektronika je postala pristupačna kao nikada ranije.

Koncept uređaja se brzo dopao brojnim obrazovnim institucijama koje su naručivale serije u stotinama primeraka. Nije trebalo čekati dugo i *Arduino* je sišao u mase, gde su ga sa oduševljenjem dočekale desetine hiljada hobista i onih koji su želeli da nauče osnovne principe elektronike. Počele su da se izdaju knjige, internet sajtovi su krenuli sa objavljivanjem projekata, interesovanje za hardver je naglo poraslo i bili smo svedoci ponavljanja stvaralačke euforije karakteristične za vreme buma kućnih računara, kada su informatički časopisi iz broja u broj objavljivali hardverske projekte za samogradnju.

Još da razjasnimo odakle potiče naziv same hardverske platforme. Ekipa okupljena oko tog projekta je volela da se sastaje u kafiću koji nosi naziv po Arduinu od Ivree, italijanskom kralju u razdoblju od 1002. do 1014. godine.



Kada se radi o platformi *Arduino* vrlo je važno istaknuti činjenicu da je u pitanju projekat otvorenog koda. U prevodu, sve informacije o arhitekturi uređaja su dostupne svima i ne samo da tvorci na takav način ne sprečavaju pojavljivanje klonova, nego ga direktno i stimulišu. *Arduino* je sa pravne strane pokriven Creative Commons licencom koja trećim licima dozvoljava modifikacije osnovnog hardvera i prilagođavanje sopstvenim željama i potrebama. Za *Arduino* se često kaže da je najuspešniji hardverski projekat otvorenog koda.

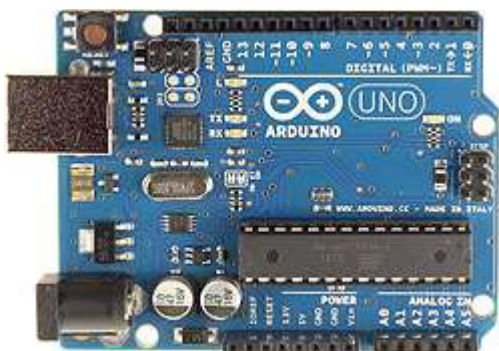
Dugi niz godina su se za osnovu ovih uređaja koristili mikrokontroleri sa AVR arhitekturom, koji su zbog svoje cene i jednostavnosti upotrebe čest izbor na mnogim fakultetima. Tek negde od prošle godine započinje znatna diversifikacija po pitanju izbora arhitekture mikrokontrolera koji predstavljaju pogonski motor uređaja *Arduino*. Svedoci smo pojavljivanja novih modela koji u sebi imaju ARM jezgro, a odnedavno je u igru ušao i veliki Intel sa svojim mikrokontrolerom baziranim na x86 arhitekturi.

Da bi komplikovana situacija oko mnoštva uređaja pod nazivom *Arduino* postala još komplikovanija, potrudili su se pojedinci iz samog projekta. Naime, jedan od ljudi iz originalnog tima *Arduino*, Đanluka Martino, 2008. godine mimo znanja ostalih kolega, tajno registruje ime *Arduino* na sebe. Posledica svega toga je da danas postoje dve firme *Arduino*. Čak su i njihovi sajtovi vizuelno vrlo slični, pa ih je dosta teško razlikovati.

Ekipa sa sajtom arduino.cc okuplja najveći deo prvobitne družine „poštenih Arduinovaca“, dok se iza sajta arduino.org nalaze „nečasni Arduinovci“, koji nisu ni malo popularni kod fanatičnih ljubitelja ove platforme. Mnogi su korisnici godinama davali prednost originalnim (i skupljim) modelima koji su bili prepoznatljivi po natpisu „Made in Italy“, misleći da na taj način finansijski pomažu razvoj platforme. Na kraju balade se saznalo da i nije baš bilo tako. Sada je glavni način raspoznavanja modela prisustvo URL adrese sajta o kojima smo prethodno govorili, kao i uvođenje marke *Genuino* koja se odnosi na modele za sva tržišta osim SAD. Po svemu sudeći, rat ova dva klana će u budućnosti umnogome iskomplikovati pitanja kompatibilnosti programskih alatki i one više neće biti univerzalne kao do sada.

Osim modela pod zvaničnim nazivom *Arduino/Genuino*, postoje stotine raznoraznih varijanti tih uređaja. Modeli sa originalnim dizajnom se uglavnom proizvode na zapadu (nazivaju se još i derivatima), dok se po pravilu u Kini prave jeftine kopije komercijalno najuspešnijih varijanti. Pri tome su cene modela kineske proizvodnje 4-5 puta niže, ali je kvalitet (po običaju) dosta relativan i varira od slučaja do slučaja.

Uno

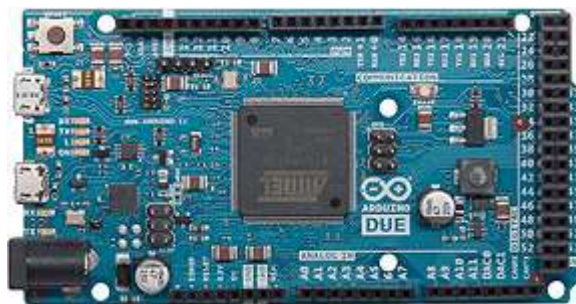


Bez ikakve sumnje, ubedljivo najuspešniji *Arduino* do sada (Slika 1). Reč je o modelu od kojega je ova platforma i postala planetarno poznat fenomen. Za popularnost je u početku bila važna njegova relativno niska cena i bolje mogućnosti u odnosu na prethodnike, pa su autori projekata računajući na što širi auditorijum

uzimali *Uno* za osnovu svojih projekata. Njegova glavna prednost nad snažnijim modelima je u tome što podržava najveći procenat postojećih šildova (naziv za hardverske dodatke koji se priključuju direktno na uređaj), tako da će početnici u najvećoj meri biti zaštićeni od naručivanja pogrešnih komponenti. Radi na brzini od 16 megaherca i ima dva kilobajta RAM i 32 kilobajta fleš memorije. U narednom broju ćemo upravo na ovom modelu (zato ovde ne pišemo opširnije) objasniti neke od najvažnijih stvari koje treba znati kada se govori o *Arduino* platformi. Trenutna cena originalnih *Uno* modela je oko 20 evra, dok kineske kopije koštaju tek nekih 3-4 evra.

Due

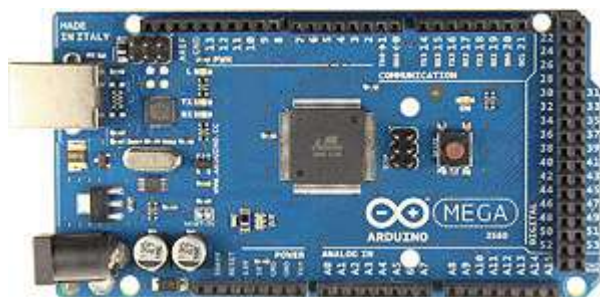
Iako smo rekli da su kreatori *Arduino* platforme tek negde tokom 2015. godine prešli na mikrokontrolere sa arhitekturama različitim od AVR, *Arduino Due* je izuzetak od tog pravila (Slika 2). Reč je o modelu koji se pojavio još 2012. godine i baziran je na *Atmelovom* ARM Cortex-M3 mikrokontroleru AT91SAM3X8E, koji radi na frekvenciji od solidnih 84 megaherca i razvija brzinu do 103 MIPS-a. Na raspolaganju su nam pedeset i četiri U/I pina, sa dvanaest analognih ulaza frekvencije smplovanja od 12 bita. Tu su još i četiri UART (serijska) porta i dva DAC izlaza, što je najviše od svih dosadašnjih modela. Sve u svemu, dobar potencijal.



Na ploči su prisutna dva mikro USB konektora od kojih je jedan po običaju namenjen programiranju uređaja i napajanju strujom, dok drugi ima mogućnost priključivanja standardnih USB periferija. S obzirom na to da *Arduino Due* raspolaže sa 96 kilobajta RAM (skoro 50 puta više od modela *Uno*) i 512 kilobajta fleš memorije za zapisivanje programa, moguće ga je koristiti u prilično složenim projektima.

Prilikom korišćenja ove ploče ne treba gubiti iz vida da ona radi na naponu od 3,3 volte, za razliku od ranijih modela koji su predviđeni za pet volti, jer u suprotnom može doći do pregorevanja uređaja. Prilikom nabavljanja dodatnih šildova, treba da birate one koji rade na 3,3 volta. Ovaj model je trenutno moguće kupiti preko sajta arduino.org po ceni od 40 evra, dok se kineske kopije mogu pazariti za nekih desetak evra.

MEGA 2560



Ovaj model je pre svega namenjen onima kojima je potrebna istovremena kontrola nad većim brojem ulaznih i izlaznih uređaja. *MEGA 2560* (Slika 3) je zasnovana na osmobitnom kontroleru *Atmega2560*, koji ima četiri

puta više radne i osam puta više fleš memorije od modela *Atmega328* i pri tome oba rade na frekvenciji od 16 megaherca. Osim toga, na raspolaganju nam stoje 54 U/I digitalne linije i celih 14 PWM izlaza (Pulse Width Modulation, način emulacije analognog signala uz pomoć digitalnog kvadratnog talasa) uz

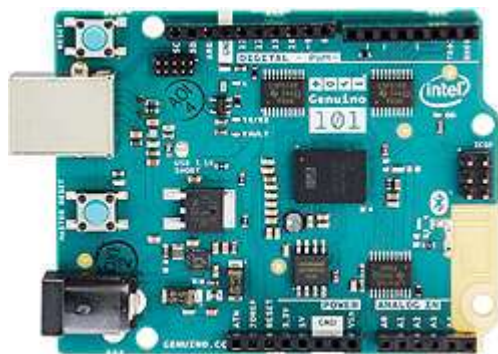
četiri hardverska UART-a (serijska porta). Veće mogućnosti su za posledicu imale i znatno veće gabarite uređaja, pa je po dužini *MEGA 2560* za nekih 3,3 centimetra veća od dimenzija koje ima *Uno*, dok je širina ostala jednaka. Generalno, *MEGA 2560* se zaista može smatrati kao „mega” varijanta *Uno* modela i kao takva je prvenstveno orijentisana prema naprednim korisnicima koji su već ispekli hardverski zanat. Za prvi put i za edukativne potrebe, *Uno* je više nego dovoljan. Ukoliko baš želite da od samog početka budete „mega”, onda ćete za originalnu verziju izdvojiti 35 evra, dok se ekvivalenti sa oznakom „Made in China” mogu nabaviti već za 6-7 evra.

101

Intel je davno shvatio da dolazi vreme kada će sićušni mikrokontroleri biti ugrađivani u praktično sve predmete kojima se svakodnevno koristimo. Posle pametnih telefona, naočara, televizora i satova, očekujte i pametne cipele, jakne, rukavice, torbe i ko zna šta još sve ne. Upravo za tu namenu su pre nekoliko godina razvili



familiju sićušnih mikroprocesora pod nazivom *Quark*, koja bi trebalo da obezbedi pristojne performanse uz vrlo malu potrošnju energije. Glavna ciljna grupa njihovog novog SoC procesora pod nazivom *Curie* (Slika 4) je tzv. *wearable* segment tržišta, pa su iz tog razloga u njega ugradili podršku za Bluetooth LE, žiroskop, RTC i termo senzor. Zanimljiva specifičnost ovog uređaja je da u sebi ima dva mikroprocesaorska jezgra od kojih je jedno x86, dok je drugo ARC (Argonaut RISC Core). I što je još zanimljivije, ovo drugo jezgro pogoni minijaturni operativni sistem pod nazivom *Zephyr*, a iza kojeg stoji Linux Fondacija. *Curie* ima 384 kilobajta fleš memorije ali je samo polovina dostupna korisniku (ostatak ide za pomenuti *Zephyr*). Slično tome, od 80 kilobajta statičke RAM, korisnik vidi tek 20 kilobajta. Što se tiče x86 seta mašinskih instrukcija koje podržava *Curie*, on je ekvivalentan sa onim kod starih *Pentium* mikroprocesora bez MMX instrukcija. Međutim, ne postoji način da (za sada) direktno pristupamo tim resursima. Benčmark testovi su pokazali da performanse u radu sa FP brojevima mizerne, što može da bude posledica loše optimizacije kompajlera. Treba imati u vidu da je cela platforma još uvek u fazi razvoja i da će u budućnosti mnoge stvari biti ispravljene.



Iako je učinjeno dosta na polju održanja kompatibilnosti sa modelom *Uno* (zadržan je isti raspored pinova), „sto keć” ima dva PWM izlaza manje, dakle, četiri. Koristi se napajanje od 3,3 volta, ali su pinovi konektora tolerantni i na napon od pet volti. Ipak, za očekivati je da će najveći deo šilдова kompatibilnih sa *Uno* bez problema raditi i na

ovom modelu. Cena originala se kereće u rasponu od 15-30 evra (zavisno od toga gde se kupuje), što uz njegove dodatne mogućnosti i performanse predstavlja dobar izbor.

Zero

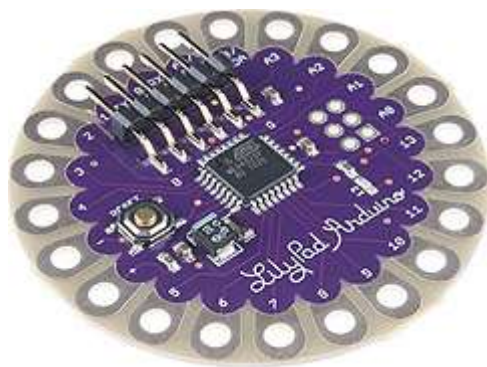
Naziv dolazi od arhitekture ugrađenog mikrokontrolera *ARM Cortex M0+* (kod ekipe oko arduino.org se ovaj model naziva *Arduino M0*). U ovom modelu je to *Atmel SAMD21 MCU* koji radi na 48 megaherca i razvija brzinu od 43 MIPS-a. U pitanju je arhitektura koja je prvenstveno okrenuta ka maksimalnoj ekonomiji energije, što je u ovakvim uređajima često od velikog značaja. Kada tome dodamo 256 kilobajta fleš memorije i 32 kilobajta RAM, jasno je da je u pitanju model koji spada u višu klasu *Arduino* porodice. I ovde na raspolaganju imamo dva mikro-USB konektora od kojih je jedan namenjen komunikaciji uređaja sa računarom i (eventualno) snabdevanju strujom, dok drugi možemo da koristimo kao USB hab za dodavanje raznoraznih periferala. Napomena koju smo dali kada smo opisivali *Arduino Due* važi i ovde, pa treba da budete oprezni i da na konektore ne dovodite napon od pet volti.



U pitanju je prvi *Arduino* model koji podržava funkciju EDBG (Embedded DeBuGger), koja omogućava direktno analiziranje programa bez bilo kakvog dodatnog hardvera. Što se tiče cene, varijanta arduino.cc je u vreme pisanja teksta na njihovom sajtu koštala 42 evra, dok je kod rivala iz „.org” bila čak za 20 evropskih novčanica niža.

LilyPad

Ako bismo pripadnicama lepšeg pola predložili da za svoje potrebe izaberu neki Arduino uređaj, više smo nego sigurni da bi izbor pao upravo na ovaj simpatični model (Slika 7). I zaista, na internetu se može primetiti da su žene dosta česti korisnici *LilyPada*, a i reč je o proizvodu koji je konstruisala dama, Lia Bikli, sa namenom da se koristi u kreacijama



odevnih predmeta. Prečnik pločice iznosi pet centimetara, što za pomenutu namenu i nije tako malo. Za povezivanje sa periferijama se koristi specijalni konac koji provodi električnu energiju (oveće klupko košta oko 40 USD). Postoji više različitih modula za napajanje koji koriste litijumske CR22, AAA ili polimer baterije i svi oni se ušivaju unutar odeće. Od ostalih stvari koje se mogu koristiti uz *LilyPad*, tu su: LED lampice, prekidački tasteri, mini vibratori, piezo elementi, Bluetooth i Wi-Fi moduli. A onda kada se na to dodaju svakojaki senzori za temperaturu, vazdušni pritisak, jačinu svetlosti, ultraljubičasto zračenje, žiroskopi i razna druga čudesa tehnike, eto prave hardverske infrastrukture u našoj odeći. S obzirom na to da se pojavio još pre nekih 7-8 godina, ne možemo se oteti utisku da je *LilyPad* kreacija ispred svog vremena. Neće biti iznenađenje ako se uskoro pojavi sličan uređaj baziran na Intelovom mikrokontroleru *Curie* o kome smo govorili prilikom opisivanja *Arduina 101*. Originali koštaju oko 15 evra dok se kineske kopije mogu nabaviti već po ceni od tri evra. Pored standardnog, postoje još i modeli *LilyPad Arduino Simple* i *SimpleSnap*. Oba imaju duplo više memorije i svega jedanaest pinova, dok model *SimpleSnap* umesto rupica koristi kontaktne tastere i ima ugrađenu Li-Po bateriju sa zadnje strane.

• • •

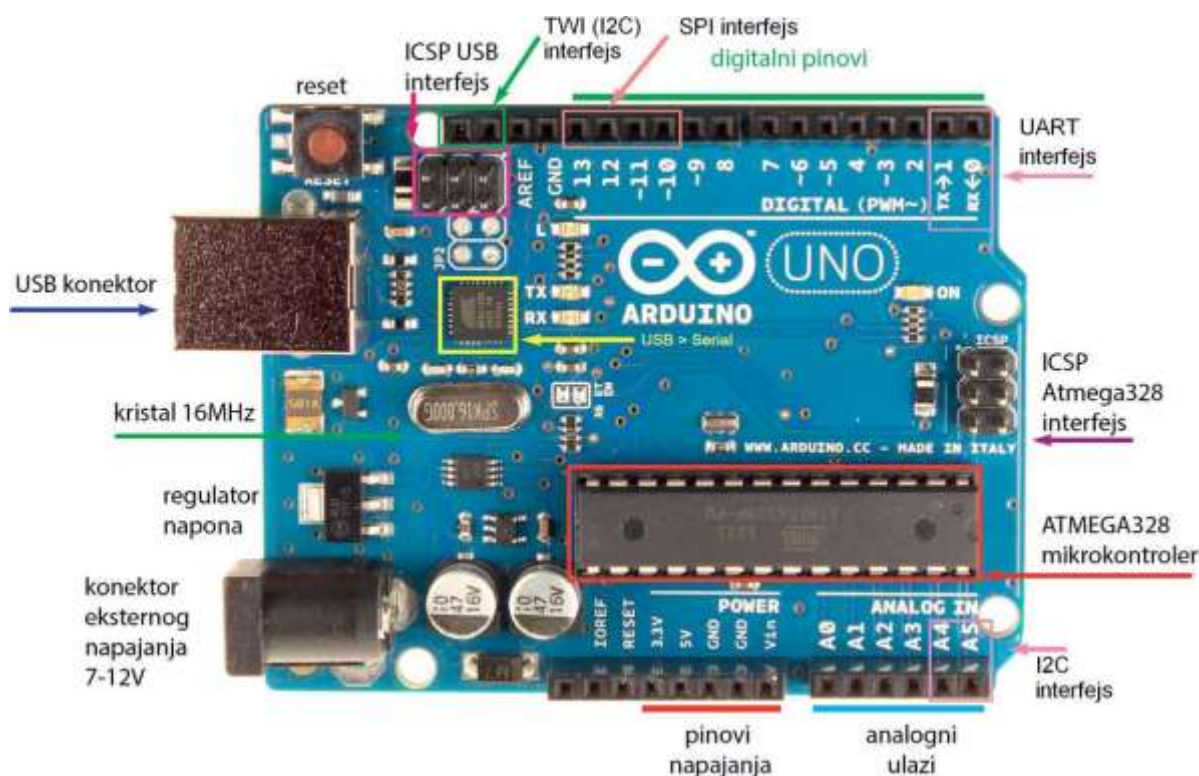
Glavna svrha ovog teksta je da čitaoce upozna sa najzanimljivijim predstavnicima *Arduino* porodice koji su trenutno dostupni na tržištu. Najsigurnija varijanta za početnike je kupovina modela *Uno*, dok zahtevnijim korisnicima više odgovara model *MEGA2560*. Model *Due* je najbrži *Arduino* do sada i nudi mnoštvo U/I kanala. *Zero* je svojevrsni *Uno* sa bržim ARM procesorom i sa ugrađenom podrškom za analizu programskog koda. Još nije najjasnija situacija oko svežeg modela *101* koji uvodi Intelove tehnologije u *Arduino* svet, ali je njegov potencijal svakako izvestan. *LilyPad* je kreiran za specifične namene pa nije najpodesniji za početnike. Zbog ograničenog prostora nismo prikazali još neke zanimljive modele. Jedan od njih je sićušni ($1,8 \times 3,3$ centimetara) *Arduino Pro Mini* koji ima karakteristike modela *Uno*, sa cenom originala od osam evra i kopija koje koštaju manje od 1,2 evra!

U sledećim nastavcima ćemo obraditi filosofiju korišćenja *Arduina* na primeru modela *Uno*, osnovne pojmove vezane za programiranje, razvojna okruženja i slično. Posle toga, na red dolaze hardverski projekti koji će vas uveriti kako je lako učiti elektroniku uz pomoć *Arduina*.

Arduino modeli	Uno	Due	101	MEGA2560	Zero	LilyPad
Mikrokontroler	ATmega328P	AT91SAM3X8E	Intel Curie	Mega2560	ATSAMD21G18	ATmega328V
Arhitektura	AVR	ARM	x86	AVR	ARM	AVR
Frekvencija	16 MHz	84 MHz	32 MHz	16 MHz	48 MHz	8 MHz
SRAM	2 KB	96 KB (64 + 32 KB)	24 KB (80)	8 KB	32 KB	1 KB
Flash memorija	32 KB	512 KB	196 KB (384)	256 KB	256 KB	16 KB
Radni napon	5 V	3,3 V	3,3 V (5 V tol. I/O)	5 V	3,3 V	2,7-5,5 V
Ulazni napon (min./maks.)	6-20 V	6-20 V	7-20 V	7-20 V	5-15 V	2.7-5.5 V
Ulazni napon (preporučen)	7-12 V	7-12 V	7-12 V	7-12 V	7-12 V	2.7-5.5 V
Struja po U/I pinu	40 mA	130 mA (za sve)	20 mA	40 mA	7 mA	40 mA
UART	1	4	-	4	2	-
Digitalnih U/I linija	14 (6 PWM)	54 (6 PWM)	14 (4 PWM)	54 (14 PWM)	20	14 (6 PWM)
Analognih ulaza	6 (10-bit ADC)	12 (12-bit ADC)	6	16 (10-bit ADC)	6 (12-bit ADC)	6 (10-bit ADC)
Analognih izlaza	0	2 (12-bit DAC)	0	0	1 (10-bit DAC)	0
Brzina [MIPS]	16	103	47	16	43	8

Arduino Uno

Igor S. RUŽIĆ, Svet kompjutera, 2016



Prvi član kraljevske porodice

Kada nadobudni ljubitelji računara vide tehničke karakteristike modela *Arduino Uno* obično odmahnu rukom smatrajući da se radi o sporom uređaju sa smešno malom količinom memorije. Međutim, kada biste se malo udubili u tematiku, shvatili biste da se u 32 kilobajta može smestiti i te kako mnogo korisnog koda. Arhitektura AVR nudi brzinu izvršavanja od milion instrukcija u sekundi, što kada se pomnoži sa 16 megaherca daje više nego dovoljno procesorske snage kako za obradu informacija pristiglih sa spoljašnjih izvora, tako i za upravljanje priključenim uređajima.

Arduino Uno dolazi u vidu malene pločice dimenzija $6,9 \times 5,3$ centimetara koja se programira putem USB interfejsa. Za pisanje programa je moguće koristiti

kako računare koji rade pod Windowsom tako i one sa OS x i Linuksom. Na pločici se nalazi USB-B konektor koji služi za programiranje uređaja, kao i njegovo napajanje dok je priključen na računar. Ukoliko je potrebno da uređaj funkcioniše nezavisno od računara, na raspolaganju nam je konektor (2,1x5,5 milimetara) preko koga priključujemo eksterno napajanje sa naponom u rangu 5-12 volti. Često se u takvim slučajevima koristi standardna „kockasta” PP3 baterija od devet volti za koju je potrebno nabaviti odgovarajući adapter. Ako uređaj napajamo preko USB porta, maksimalna jačina struje na koju možemo računati je 500 miliampera, dok u slučaju konektora za napajanje po specifikaciji može da ide do jednog ampera (realno oko 800 miliampera).



Najuočljiviji element na pločici je sam mikrokontroler *Atmega328P* (P je oznaka za „*PicoPower*” tehnologiju uštede energije), dok manji kvadratni čip u sebi sadrži USB/Serial interfejs. Dva bloka od po šest iglica predstavljaju ICSP (*In-Circuit Serial Programming*) interfejse. Reč je o specijalizovanoj varijanti ISP (*In-System Programming*) interfejsa čija je namena da dozvoli menjanje sistemskog programa (*bootloadera*) u čipovima na pločici, čime se izbegava odlemljivanje komponenti. Jedan blok je namenjen programiranju mikrokontrolera, a drugi za programiranje USB/Serial kontrolera. Od ostalih uočljivijih elemenata na pločici možemo izdvojiti taster za resetovanje uređaja, regulator napona koji se nalazi pored ulaza za napajanje i kristal od 16 megaherca u prepoznatljivom metalnom kućištu. Tu su i četiri minijturna LED svetla od kojih dva prikazuju slanje i prijem podataka preko serijskog porta (Rx,Tx), jedno prikazuje da je uređaj pod naponom, dok je četvrto fizički povezano sa pinom 13 i služi kao indikator njegove aktivnosti.

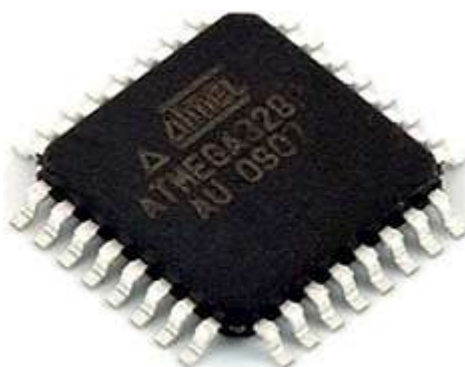
Pored edukacije, glavna namena *Arduina* je da olakša posao projektovanja uređaja koji u sebi imaju mikrokontroler i uz njegovu pomoć proveravamo da li naša zamisao funkcioniše u praksi. Obično se uz *Arduino* koristi prototipska ploča sa konektorima (*breadboard*) kojima nije potrebno lemljenje i na koju se postavljaju komponente koje se međusobno povezuju sa provodnim žicama. Ako se radi o edukaciji, nakon što učenik završi sa postavljanjem komponenti i uveri se (uz obavezno divljenje svojoj inteligenciji) da sve radi kako treba, obično sledi uklanjanje svih delova da bi se oslobodio prostor za nove eksperimente. Ako se pak *Arduino* koristi za konstrukciju nekog uređaja koji će biti korišćen u praksi, situacija ja malo drugačija. Kada se dokaže ispravnost

projekta, *Arduino* nam suštinski više nije potreban i umesto njega na štampanu pločicu možemo postaviti mikrokontroler koji će obavljati ulogu koju je prethodno imao mikrokontroler na našoj prototipskoj platformi. U narednim brojevima ćemo prikazati neke od alatki koje maksimalno pojednostavljaju izradu štampanih pločica koristeći za osnovu prototip koji smo kreirali na *Arduino*.

Uno, due, tre...

Igor S. RUŽIĆ, Svet kompjutera, 2016

Priča o *Arduino Uno* započinje u Nju Jorku početkom jeseni 2010. godine kada je najavljen u okviru sajma posvećenog ljubiteljima tehnike „uradi sam”. Inače, reč je o sajmu koji organizuje poznati američki časopis „Make” i koji se održava u velikom broju gradova širom sveta. Nije nikakva tajna da *Arduino* platforma tom časopisu duguje dobar deo svoje popularnosti.



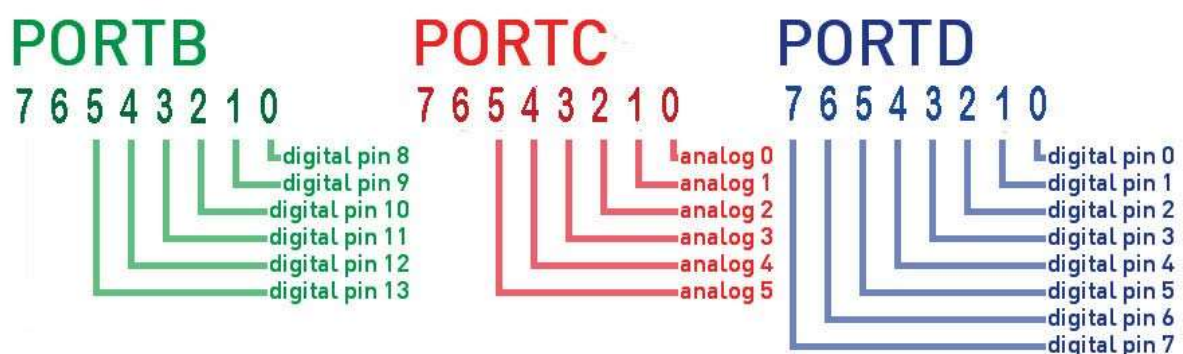
Uno svojom pojavom u suštini nije doneo ništa spektakularno novo. Radi se o modelu koji blago evoluirao iz svojih prethodnika. Kao prvi popularni predak *Una* se može smatrati model *Arduino NG* u koji je ugrađen mikrokontroler *ATmega8* sa svojih osam kilobajta fleš memorije. Nakon njega, 2007. godine sledi model *Arduino Diecimila* koji donosi *ATmega168* i 16 kilobajta memorije za programe. Dve godine posle dolazi red na model *Duemilanove* koji sa „mozgom” baziranim na *ATmega328* već dobro počinje da liči na *Uno*. Glavna razlika se odnosi na promenu čipa koji konvertuje podatke sa USB porta u serijski oblik. U prethodnim verzijama se radilo o čipu *FT232RL*, dok *Uno* dolazi sa modelom *Atmega8u2* u prve dve revizije i *Atmega16u2* za *R3*. Za prosečnog korisnika između tih modela nema nikakve razlike. Do pojave treće revizije *Unaimali* su i identičan broj i raspored pinova. Postavlja se pitanje, zašto je *Uno* najpopularniji model *Arduina*, a ne neki drugi? Iz prostog razloga što se tajming velike popularnosti cele platforme poklopio sa njegovim pojavljivanjem.

Često se uz naziv *Uno* modela može videti oznaka R3 koja predstavlja treću reviziju pločice uređaja. Oznake R1 i R2 je danas vrlo teško negde susresti, pošto su te varijante davno prestale da se proizvode. Razlike između ta tri modela su minimalne, a glavnu smo opisali u prethodnom pasusu. Postoje dve varijante *Arduino Uno* R3 uređaja koje su funkcionalno jednake i razlikuju se samo u tipu kućišta mikrokontrolera. Naime, prvobitna verzija dolazi sa duguljastim DIP kućištem (Slika 2) koje ima 28 nožica, ali je proizvođač zbog nedostatka tih čipova na tržištu bio primoran da koristi SMD varijantu (Slika 3) *Atmega328* u kvadratnom kućištu sa 32 pina. Informacije radi, SMD verzija mikrokontrolera sadrži dva neiskorišćena ADC ulaza (A6 i A7) koji nemaju izvode na kućištu uređaja. Korisnicima generalno ova razlika ne znači mnogo, ali je velika prednost DIP varijante to što je u slučaju pregorevanja mikrokontrolera moguće na jednostavan način zameniti неисправan deo, što je u SMD varijanti mnogo složeniji posao. Modeli sa SMD čipom donose oznaku „*SMD Edition*” uz inverziju osnovnih boja na leđnoj strani pločice.

Veza sa svetom

Igor S. RUŽIĆ, Svet kompjutera, 2016

Arduino je po svojoj funkciji U/I ploča (*I/O board*), što znači da preko svojih izvoda omogućuje rad sa ulaznim i izlaznim uređajima. Za priključivanje se koriste plastični konektori koji se tradicionalno nazivaju pinovima, iako su ovde u pitanju konektori ženskog tipa. Izvodi su grupisani u četiri gnezda i to po dva sa dve bočne strane pločice.



Gnezdo sa izvodima vezanim za napajanje uređaja u R3 varijanti ima osam kontaktnih mesta od kojih krajnje levo nema nikakvu funkciju. Tu su dva pina za uzemljenje (Gnd), izlazni pinovi za 3,3 i pet volti, pin za ulazni napon (Vin), pin za dovođenje RESET signala i (samo na R3) pin IOREF, čija je funkcija da uskladi napon priključenog šilda (dodatni uređaj) sa naponom samog uređaja.

Sledeće gnezdo ima šest pinova nazvanih A0-A5, čija je namena da prihvate informacije u analognom obliku koje su predstavljene različitim naponima. Te informacije se dalje prosleđuju u tzv. ADC (*Analog to Digital Converter*), gde se pristigli napon pretvara u neku vrednost izraženu 10-bitnim brojem. Znači, vrednost u opsegu 0-1024 (2^{10}). Pinovi A4 i A5 su interno povezani sa pinovima SDA i SCL, što nam omogućuje uspostavljanje komunikacije sa drugim uređajima putem I2C magistrale. I2C je jedan od najjednostavnijih protokola za serijski prenos podataka i za to su mu potrebne svega dve žice. Brzine prenosa ove magistrale se kreću od 1-5 Mb/s.

Sa gornje strane pločice se nalaze dva konektora (8 i 10 pinova) koji su najvećim svojim delom namenjeni radu sa digitalnim signalima. Pinovi sa oznakama 0-13 po potrebi mogu da imaju funkciju kako ulaznih, tako i izlaznih konektora. Pored pinova 3,5,6,9,10 i 11 se nalazi znak „~” koji napominje da se oni mogu koristiti kao pseudoanalogni izlazni portovi. Šta to znači?

Pošto *Atmega328* nema DAC (*Digital-to-Analog Converter*) logiku, koristi se tehnika PWM (*Pulse Width Modulation*) za generisanje analognih signala putem modulacije kvadratnog talasa. Šest nabrojanih pinova je podeljeno u tri para koji mogu raditi na različitim frekvencijama. Pinovi 5 i 6 se kontrolišu preko tajmerskog registra TCCR0B, 9 i 10 preko TCCR1B, a 3 i 11 preko TCCR2B. Kombinovanjem kvadratnog talasa sa vrednostima navedenih registara stvara promenjive nivoe napona u opsegu 0-5 volti. Funkcija `analogWrite()` svake sekunde može da bude izvršena nešto manje od 500 puta.

Na digitalnim pinovima D0 i D1 se nalaze oznake Rx i Tx koje sugerišu da se radi o kontaktima koji mogu biti korišćeni kao USART (*Universal Synchronous/Asynchronous Receiver/Transmitter*), odnosno, serijski interfejs preko koga možemo komunicirati sa drugim računarima i uređajima, a u slučaju potrebe ga možemo koristiti za programiranje samog *Arduina*.

Pinovi D10-D13 obavljaju i ulogu SPI (*Serial Peripheral Interface*) porta. Reč je o serijskoj magistrali koja za razliku od ostalih serijskih protokola koje smo obradili omogućuje rad u dupleks režimu (istovremeno slanje i primanje podataka). Interfejs koristi četiri pina za svoj rad i to: D10 – SS (*Slave Select*) pin koji koristi *master* za (de)aktivaciju uređaja; D11 – MOSI (*Master Out Slave In*) *master* linija za slanje podataka na *slave*; D12 – MISO (*Master In Slave Out*) *slave* linija za slanje podataka *masteru* i D13 – SCK (*Serial Clock*) klok za sinhronizaciju prenosa.

Pin 15 pod nazivom AREF (*Analog REference*) služi za određivanje gornje vrednosti ulaznog opsega za analogno-digitalni konverter. Podrazumevana

Atmega328

Srce i duša *Arduina Uno* je mikrokontroler koga proizvodi kompanija Atmel, jer pinovi koje smo prethodno pominjali su zapravo izvodi nožica ovog mališana. Procesorsko jezgro Atmelovih kontrolera na bazi AVR tehnologije je izgrađeno po RISC dizajnu koji najčešće jednu mašinsku instrukciju obrađuje u toku jednog sistemskog takta. Mikrokontroler *Atmega328* radi na maksimalnom taktu od 20 megaherca, ali je zbog kompatibilnosti sa prethodnim modelima *Arduina* radni takt ograničen na 16 megaherca. Osim toga, ovaj čip podržava različite brzine u zavisnosti od ulaznog napona. U slučaju da je napon na ulazu 1,8 volti, procesor će raditi na maksimalnoj brzini od četiri megaherca. Ako je pak u pitanju napon od 2,7 volti brzina će biti 10 megaherca. Za maksimalnu brzinu od 20 megaherca je potreban napon od minimalno 4,5 volti.

U slučaju kada ne postoji dovoljan broj pinova na nekom uređaju, pribegava se interfejsu koji se naziva GPIO (*General Purpose Input/Output*) i koji po potrebi može menjati funkciju nožica MCU čipa. Na ilustraciji pod brojem 1 je šematski prikazan mikrokontroler *Atmega328P* sa oznakama svih funkcija dodeljenih pojedinim njegovim nožicama.

Iz priloženog vidimo da, recimo, nožica označena brojem 5 može imati pet različitih funkcija. Ako malo pažljivije pogledamo ilustraciju, primetićemo da su pinovima mikrokontrolera 23-28 dodeljene funkcije ADC konvertera i analognih ulaza (A0-A5). Međutim, odmah pored vidimo da isti pinovi obavljaju i funkciju digitalnih portova sa oznakama D14-D19. Iako to nije obeleženo na samom uređaju (niti se o tome govori u specifikaciji), po potrebi je moguće raditi sa 20 digitalnih U/I kanala.

Kao što se vidi na ilustraciji, sve nožice mikrokontrolera osim onih čija je funkcija vezana za napajanje, imaju konektore sa početnim slovima PB, PC i PD iza kojih sledi brojna oznaka. Ta tri porta se u *Arduino* žargonu nazivaju PORTB, PORTC i PORTD. PORTB se odnosi na digitalne pinove D13-D8, PORTC na analogne pinove A5-A1, dok se PORTD odnosi na digitalne pinove D7-D0. Svaki od portova se kontroliše preko tri posebna registra (r ima vrednost B,C ili D):

DDRr – ukazuje na to da li je pin INPUT ili OUTPUT
PORTr – ukazuje na to da li je pin HIGH ili LOW
PINr – sadrži vrednost INPUT, ukoliko je pin u režimu čitanja ulaza.

Upravljanje ovim registrima spada u naprednije tehnike programiranja za *Arduino* i početnicima nije neophodno njihovo detaljno poznavanje. Ipak, treba istaći to da se njihovim korišćenjem prilikom rada sa portovima može

uveliko smanjiti obim koda, kao i povećati brzina njegovog izvršavanja za čak deset do petnaest puta.

Što se tiče organizacije memorije, *Atmega328* kao pripadnik harvardske škole deli memoriju na programsku i onu za podatke. Programska je fleš tipa, dok je memorija za podatke na bazi brzih SRAM memorijskih ćelija. Veličina programske memorije iznosi 32 kilobajta, od kojih treba oduzeti 512 bajtova namenjenih smeštanju tzv. *bootloader* programa koji omogućuje programiranje mikrokontrolera sa računara. Memorija za podatke sadrži 32 osmobitna registra opšte namene, 64 U/I registra, 160 registara za proširenu U/I memoriju i dva kilobajta namenjena smeštanju podataka potrebnih za izvršavanje programa iz programske memorije. Šest registara (R26-R31) ima posebnu namenu i mogu se kombinovati u indeksne pokazivače šesnaestobitnog adresnog opsega.

Unutar procesora su nam na raspolaganju tri registra koji po potrebi mogu vršiti funkciju tajmera ili brojača. Oni se nazivaju *Timer/Counter* 0, 1 i 2. Prvi i treći rade sa osam bita i međusobno su dosta slični, dok drugi ima šesnaestobitni registar. O njima smo govorili kada smo pominjali konverziju digitalnog u PWM analogni signal, gde igraju važnu ulogu.

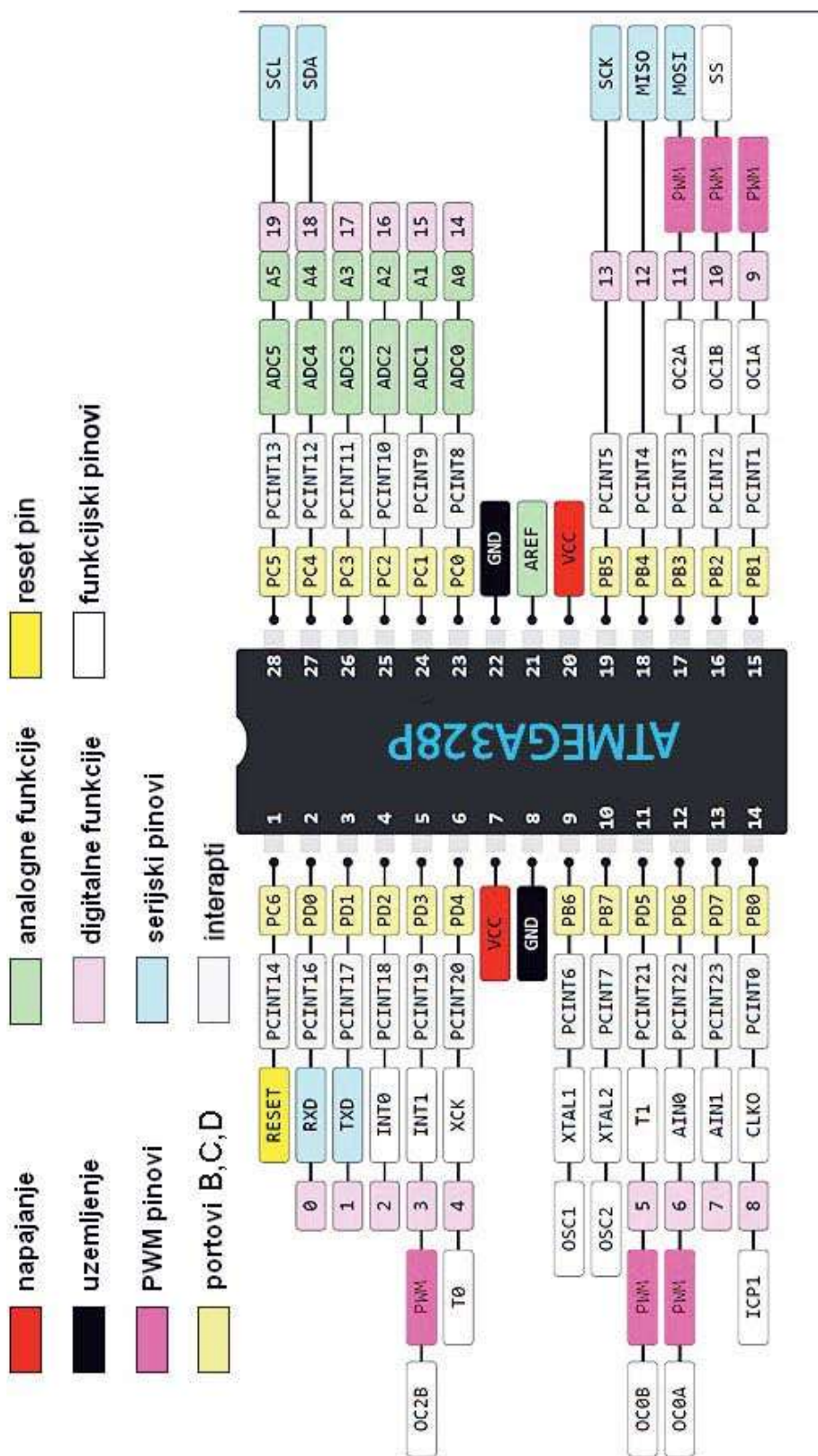
Na čipu se našlo mesta i za EEPROM (*Electrically Erasable Programmable Read-Only Memory*) veličine jednog kilobajta koji može sadržati podatke koji ostaju i nakon isključivanja struje, nešto kao minijturni hard-disk. Tu se uglavnom upisuju podaci koji predstavljaju podešavanja koja se ne menjaju tako često. Ovoj memoriji nije moguće pristupiti direktno, već se to obavlja preko posebnih registara. Na sreću, postoje programske biblioteke koje krajnje pojednostavljaju tu proceduru.

Specifikacija mikrokontrolera kaže da je zagarantovano 10000 ciklusa zapisivanja podataka u programsku memoriju i 100000 u EEPROM. U praksi je to najčešće nekoliko puta više od deklariranih vrednosti.

Mere opreza

Kao i svi uređaji i ovaj pod određenim uslovima može da se pokvari. Jedan od klasičnih načina da nešto zabrljamo je da napravimo kratak spoj postavljajući uređaj na materijal koji funkcioniše kao provodnik. Da bi se ovo izbeglo, preporučuje se korišćenje posebnih podnožja ili kućišta koja će obavljati funkciju izolatora. Moguće je izazvati kvar tako što ćemo utaknuti žicu u pogrešan konektor. Jedan od primera je povezivanje Vin i GND pinova. Potreban je oprez prilikom proračuna potrošnje struje, jer

ukoliko *Atmega328P* daje struju veću od 200 miliampera (ili 40 miliampera sa pojedinačnog pina), dolazi do pregorevanja. Najčešći kvarovi su posledica dovođenja elektriciteta koji premašuje propisane vrednosti za više od 10%. Recimo, dovođenje šest volti na U/I pinove. Postoje i načini da kvar izazovemo programskim putem. Na primer, povežemo dva U/I pina sa jednom žicom i postavimo im funkciju izlaza, pa jednom od njih dodelimo vrednost *High*, a drugom *Low*, što će izazvati pregorevanje na oba konektora.



Arduino Nano

Šibicar na italijanski način
Dejan PETROVIĆ

Ako je Arduino UNO prevelik za projekte veličine kutije šibica, Arduino Nano nije. Među najmanjim članovima porodice Arduino, Nano ima impresivno male dimenzije, 18 × 45 milimetara, i težak je nestvarnih sedam grama. Hardverska ploča tako malih dimenzija definitivno staje u kutiju šibica, zajedno sa po nekim senzorom, a trenutno je aktuelna verzija 3.0. Mikrokontroler koji zauzima centralno mesto ove malene PCB ploče jeste Atmega328P na 16 megaherca, da kažemo standardnom kloku za Arduino. Mikrokontroler radi sa 32 kilobajta fleš memorije, od kojih je dva kilobajta rezervisano za bootloader. Tu su još i dva kilobajta SRAM i jedan kilobajt EEPROM.

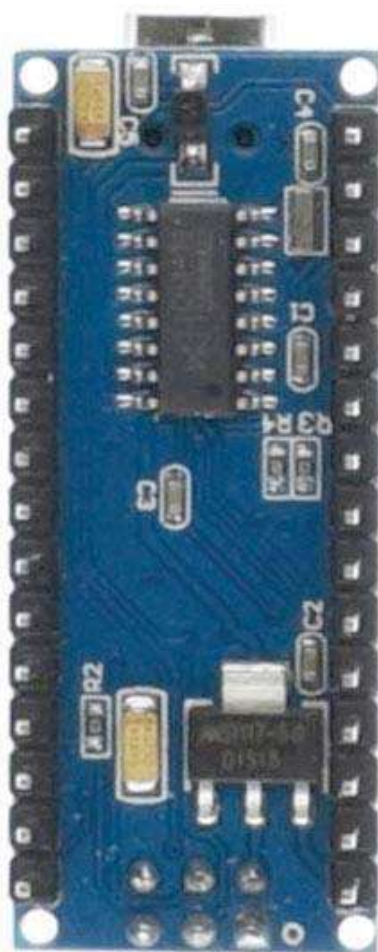


Ovako mali Arduino nema puno mesta za portove i konektore, ipak, svi su tu. Za napajanje i vezu sa računarom Nano koristi mini-B USB port. Za napajanje preko njega potreban je strujni adapter na pet volti. Za takozvano „eksterno” napajanje Nano može da radi na naponima od šest do dvanaest volti (maksimalno do dvadeset) preko pina 30 (Vin) sa napajanjima bez naponskog kontrolera ili preko pina 27 (5V) sa napajanjima sa kontrolerom. Naponski kontroler zadužen za peglanje napona nalazi se sa donje strane, između pinova. Na sredini je taster za reset, a odmah pored su četiri LED (TX, RX, ON i L). L (onboard LED) je povezana sa pinom 13. Na samom kraju nalazi se šest ICSP pinova i njihova uloga je ista kao i na Unu, o čemu je bilo reči. Pinovi su poređani sa donje strane u dva reda po petnaest i u razmaku od 0,1 inča, tako da su podesni za prototipsku ploču. Pinout je drugačiji od Una, ali je način njihovog povezivanja i upotrebe ostao isti. Svih četrnaest digitalnih pinova se mogu koristiti za input ili output i svi rade na pet volti. Šest od njih imaju osmokitnu PWM funkciju (`analogWrite()`). Nano ima osam analognih inputa, od kojih se dva (6 i 7) ne mogu koristiti kao digitalni pinovi. S obzirom na veličinu

same ploče, označavanje nije tako pregledno kao na Unu, već su u pitanju sitne oznake pored samih pinova i zgodno je ispratiti povezivanje preko pinout šeme. Nano je zadržao dobar deo funkcionalnosti velike braće uz kompaktniju formu. Downside je nemogućnost upotrebe velikog broja „šildova“, iako postoje, ali u daleko manjem broju.

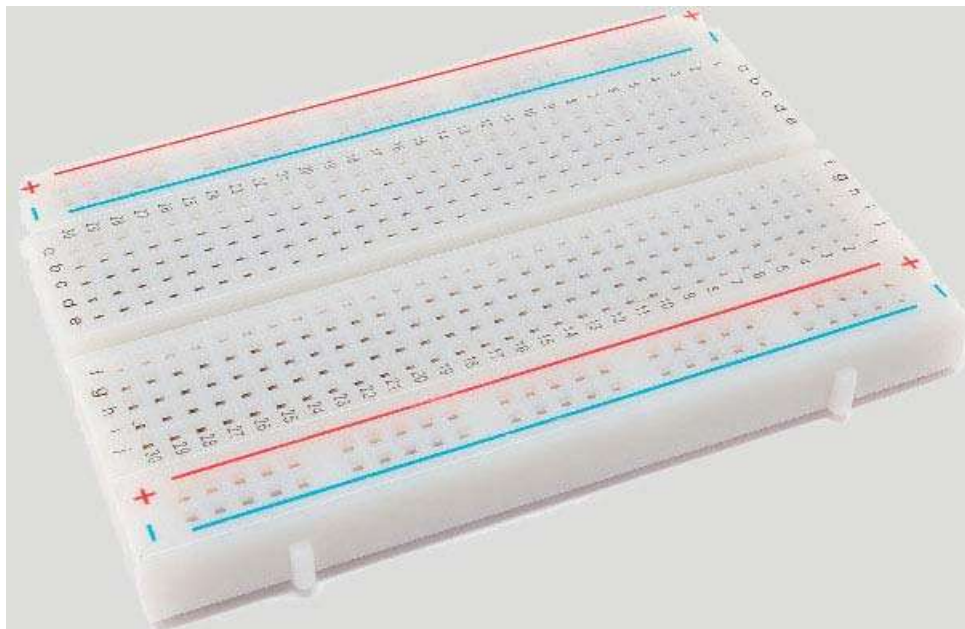
(Klik za uvećanje: JPEG, 22 KB)

Već smo pominjali da postoji veliki broj klonova. Najveći broj njih za komunikaciju između mikrokontrolera i računara koristi jeftin CH340G čip, za koji su neophodni drajveri. Originalni Nano koristi FTDI FT232RL, gde se neophodni drajveri dobijaju uz Arduino softver. Napomenućemo i to da mnogi klonovi dolaze bez bootloadera, što početnicima može da zada glavobolju, ali se do (relativno jednostavnog) rešenja dolazi guglanjem ključnih reči „arduino bootloader install“. Ipak, koristili original ili kopiju, to neće praviti razliku da biste, recimo, sve to povezali sa nekim „meteo-modulom“, spakovali u kutiju šibica i napravili mini-meteorološku stanicu. Klonovi vrlo često dolaze bez zalemljenih pinova, što, ako će se Nano namenski upotrebiti za određeni projekat, može da bude i dobro, jer je bez pinova još kompaktniji, a žice se mogu zalemiti direktno na PCB.



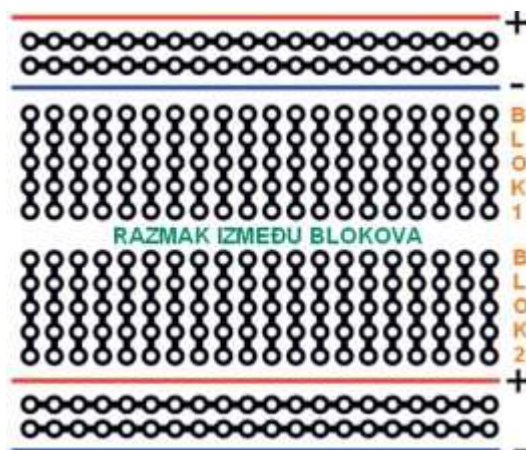
Prototipske pločice

Igor S. RUŽIĆ, Svet kompjutera, 2016



Prototipske ploče koje ne zahtevaju lemljenje izum su koji je za red veličine pojednostavio dizajniranje i testiranje jednostavnijih hardverskih uređaja. Sa njihovom pojavom je prestala potreba za prepravljanjem prototipskih štampanih ploča i lemljenjem komponenti u fazi konstrukcije. Naziv breadboard potiče od kuhinjske daske za sečenje hleba koja je često korišćena kao „matična ploča” za radio amaterske uređaje pre mnogo decenija.

Logički se svi kontakti na ploči mogu podeliti u dve grupe. Prva obuhvata konektore koji se nalaze po bokovima (na modelima sa 400 i 830 rupica su odvojene obojenim linijama i praćene oznakama + i –) i namena im je da služe za napajanje strujom komponentata na centralnom delu ploče. Konektori u ta dva reda (modeli sa 700 rupica imaju samo po jedan) su povezani i jedan red je namenjen razvođenju napona (+), a drugi uzemljenju (-). U neku od (obično krajnjih) rupica ovog bloka žicama dovodimo struju sa izvora napajanja, a zatim je razvodimo iz ostalih konektora tog reda. Ispod plastike su skrivene minijaturene štipaljke koje pridržavaju utaknutu žicu ili nožicu komponente i stvaraju električni kontakt. Prosečan vek štipaljki na kvalitetnim pločama se računa na oko 10000 korišćenja.



Centralni deo je namenjen smeštanju elektronskih komponenata i sastoji se iz dva bloka konektora koji su razdvojeni kanalom. Svaki blok je sačinjen iz većeg broja kolona od po pet rupica. Svaka od tih kolona konektora je ispod plastike povezana paralelnom vezom i sve žice/komponente koje imaju dodir sa tom kolonom ostvaruju međusobni električni kontakt. Razmak između blokova služi za galvansko razdvajanje i to je jedino mesto na koje se mogu postaviti integrisana kola koja su jednim redom nožica priključena na prvi, a drugim redom na drugi blok. Rastojanje između rupica iznosi 2,5 milimetra (1/10 inča), što je ujedno i standardni razmer za razmak između kontakata elektronskih komponenata.

Osim standardnih ploča koje smo opisali, postoje i one bez redova za napajanje i sa kolonama sa manjim brojem otvora od pet. Često je na pločama moguće videti brojne i slovne oznake koje mogu biti od pomoći kada se sklapaju projekti koji navode tačne pozicije komponenata. Postoje i modeli koji kombinuju dve ploče u jednu.

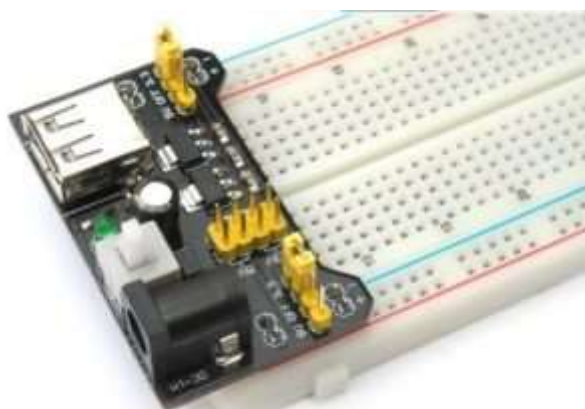


Za ostvarivanje veza se uglavnom koriste obične izolovane žice manjih profila. Te žice se često nazivaju džamperima. Za malu sumu novca je moguće nabaviti pakovanja od više desetina žica koje imaju igličaste konektore na obe strane, pa su izuzetno pogodne za povezivanje sa Arduinoom. Ovakve prototipske ploče u zavisnosti od veličine i kvaliteta koštaju 1,5-5 evra, što je zaista malo u odnosu na udobnost koju pružaju.

Napajanje Arduina

Igor S. RUŽIĆ, Svet kompjutera, 2016

Koristite li eksperimentalnu pločicu, tzv. *breadboard*, zasigurno vam je poznato kako napajanje ponekad nije jednostavno spojiti. Zato je tu ORIGINALNO napajanje za eksperimentalnu pločicu. Radi se o pločici koja je dizajnirana da točno odgovara širini eksperimentalne pločice i da pinovi za napajanje sjednu na linije za napajanje na pločici.



Ulazni napon 6.5V - 12V se dovodi na DC konektor dimenzija 5.5x2.1mm (poznat sa Arduino Uno pločice) ili preko USB konektora. Izlazni napon može biti 5V ili 3.3V i odabire se pomoću kratkospojnika (jumpera) na pločici. Izlazni naponi su dovedeni i na konektor u sredini pločice. Najveća dopuštena izlazna struja je 700mA, što je sasvim dovoljno za većinu mikrokontrolerskih aplikacija. Treba napomenuti da za napajanje preko USB-a treba koristiti 5V. Naime, kad se pločica napaja preko USB-a, ulazni napon ne ide na naponski regulator, već je spojen direktno na izlaz.



Napajanje dolazi zalemljeno i spremno za rad. Jedinu zamjerku upućujemo USB konektoru; da je micro USB bi bilo lakše priključiti punjač od mob. telefona koji je idealan za napajanje ove pločice.

ATX (računarsko) napajanje

Napajanje od računala, odnosno ATX napajanje je zasigurno jedina komponenta koju praktički bez ikakvih izmjena možemo koristiti u našoj radionici kao laboratorijski izvor napajanja. Iako su naponi fiksni i nepromjenjivi, vrlo često su nam dovoljni za naše projekte.



Zašto baš ATX napajanje?

ATX napajanje ima istosmjernu napone koji nam najčešće trebaju u projektima koje radimo, a to su +12V, +5V i +3.3V. Također ima i -12V, a starije verzije i -5V koji su nam potrebni za napajanje sklopova koji traže simetrično napajanje kao npr. operacijska ili audio pojačala. Samo napajanje ima ugrađenu zaštitu od preopterećenja i kratkog spoja, naponi su filtrirani, a na +5V i +12V može dati i nekoliko desetaka ampera struje, a nije zanemarivo niti da su takva napajanje jeftina, lako dostupna, imaju ugrađeno hlađenje i tvornički su ugrađena u kućište.

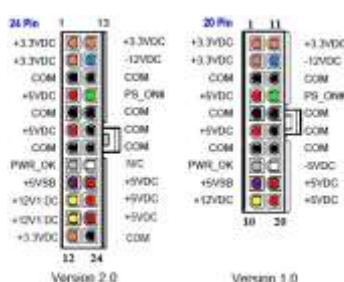


Opis pinova

OZNAKA	OPIS	BOJA PROVODNIKA
+3.3V	+3.3V napon	narančasta
+5V	+5V napon	crvena
+12V	+12V napon	žuta
GND	masa, zajednična za sve napone	crna
-12V	-12V napon	plava
-5V	-5V napon (na ATX 2.0 i novijim ga nema)	bijela
Power good	Ponekad se označava i sa PWR_OK. Napon na ovom pinu je +5V kad su +3.3V i +5V naponi u granicama tolerancije, odnosno 0V kad nisu ili napajanje ne radi	siva
+5V standby	+5V napon i kad je napajanje ugašeno	ljubičasta
Power on	Označava se i sa PS_ON. Kad je ovaj pin kratko spojen prema masi (GND), napajanje će se upaliti. Odspajanjem, napajanje se gasi.	zele

Na ATX power konektoru (na slici ispod) potrebno je kratko spojiti zelenu (PS_ON) i bilo koju crnu žicu (COM) i napajanje će se upaliti. Naravno, provjerite prije toga je li napajanje uključeno u struju i ako sa stražnje strane ima prekidač, upalite ga, odnosno prebacite na 1. Neka napajanja se neće upaliti bez opterećenja, no i ona koja se upale nije dobro dulje vrijeme koristiti bez opterećenja.

Ako je vašem napajanju potrebno opterećenje da bi se upalilo, možete spojiti neko malo trošilo, npr. 12V žaruljicu od auta, ventilator i sl. no preporučam da spojite otpornik vrijednosti 10 ohma, snage 5W ili 10W između crvene žice (5V) i crne (COM) koji će imati ulogu trošila. Važno je da otpornik ima snagu barem 5W jer će se na njemu disipirati 2.5W snage. Neka napajanja imaju vodiče drugačijih boja od uobičajenih. U tom slučaju kratko spojite pinove 15 i 16 ako se radi o 24-pinskom konektoru, odnosno 13 i 14 ako se radi o 20-pinskom konektoru. Pinovi se označavaju brojevima kao na donjoj slici, a konektor gledate tako da stranu sa kontaktima okrenete prema sebi, a kukica koja drži konektor za matičnu ploču neka vam bude s desne strane.



Proto Shield

Dejan PETROVIĆ, Svet kompjutera, 2017

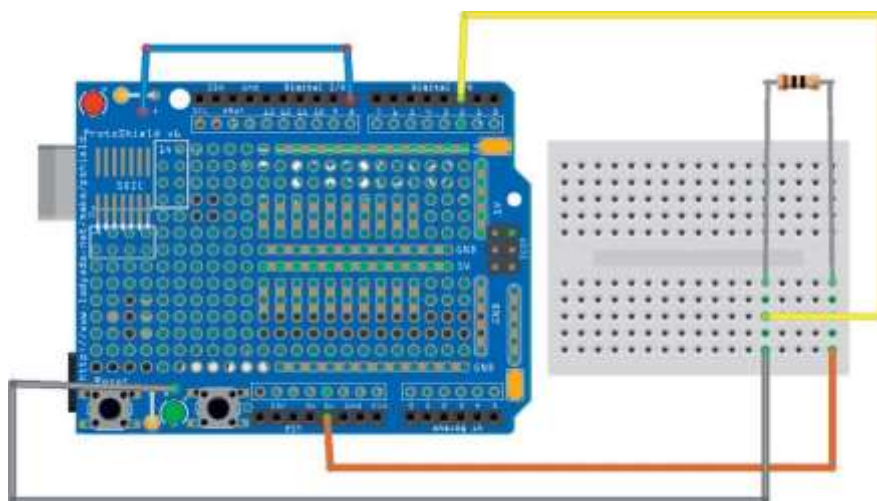
Kada govorimo o Arduinou, pod pojmom šild (shield) se podrazumevaju štampane ploč(ic)e (PCB, printed circuit board) koje se na Arduino naprosto „nabadaju”. Na taj način se, u zavisnosti od šilda, osnovne mogućnosti Arduina povećavaju, a mnogi projekti znatno olakšavaju.



Šildovi mogu imati takozvane prolazne pinove, pa je moguće dodavati više njih jedan na drugi. Neki nemaju takvu mogućnost iz tehničkih razloga, bilo da iskorišćavaju skoro sve pinove, pa je eventualni šild iznad bespredmetan, bilo da za prolazne pinove jednostavno nema mesta. Šildova ima mnogo, neki su specifični za određeni Arduino, neki su jednostavno varijacija na temu, a mi ćemo proći kroz najbitnije i najčešće. Seriju šildova otvaramo sa Proto(type) šildom.

Namena Proto šilda je da se određeni projekat prvo razradi, a potom da se na njega zaleme senzori, moduli i druge elektronske komponente. Na taj način se dobija „stand alone” projekat koji može da započne samostalan život. Proto šild koji ćemo vam predstaviti jeste kineska kopija Adafruit šilda verzije pet, a uz njega se dobija i mini breadboard ploča. Oblika i dimenzija je Arduino Una i najpodesniji je za rad sa njim, mada se može koristiti i za: Leonardo, Mega, NG, Duemilanove i druge. Najveći deo šilda zauzimaju sitni otvori obloženi sa obe strane ploče, tako da se komponente ili žice mogu lemiti sa obe strane. Iz praktičnih razloga, komponente se najčešće postavljaju sa gornje strane, dok se sa donje strane obično prave „mostovi” od kalaja. Otvori su $0,1 \times 0,1$ inč razmaka i odgovaraju onima na breadboard pločama, a samim tim i modulima. Primećujemo već pripremljene, definisane i obeležene vodove za GND i 5V u po dva reda. Ovi vodovi se protežu po sredini i krajevima površine namenjene

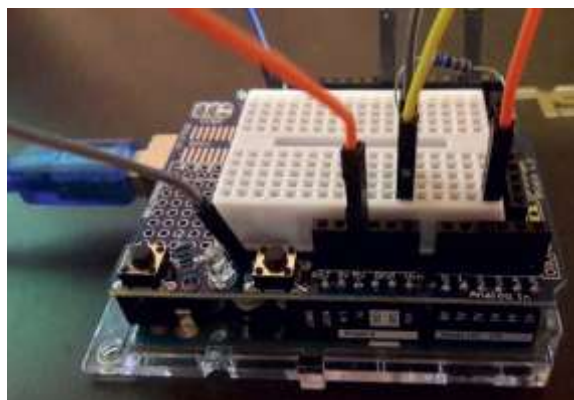
za rad. Tu je deo predviđen za postavljanje DIP (dual in line) integralnog kola do 20 pinova. U pitanju su čipovi sa nožicama koji prolaze kroz ploču. Nasuprot tome, ostavljeno je i mesto sa 14 pinova za SOIC (small outline IC), a u pitanju su čipovi koji se na ploču leme sa gornje strane. U samom uglu je predviđeno mesto za ICSP hedere, ako se za njima ukaže potreba.



Na šildu primećujemo dva „button” prekidača, od kojih je jedan reset samog Arduina, dok se drugi može programirati po potrebi. Ovaj drugi prekidač nema prateći otpornik (pull-up/pull-down), koji se prilikom izrade projekta mora dodati ili se uključiti onaj ugrađen u Arduinu (INPUT_PULLUP). Takođe, tu su i dve LED (obe crvene) sa pratećim otpornicima koje se takođe mogu programirati. Svi pinovi na Arduinu imaju svoje ženske pin hedere na šildu koji su uvučeni unutra, tako da je dalja nadogradnja šildova nemoguća. Nedostaje samo veza sa SCL i SDA pinovima. Dodat je još po jedan red ženskih hedera za GND i 5V i vidi se da je na šildu stavljen akcenat na 5V. Na kraju, tu su predviđena mesta za opcione kondenzatore od po 0,1 mikrofarađ za povećanje stabilnosti napona.

Napravili smo jedan skeč gde ćemo prekidačem na ploči upaliti jednu od LED dioda.

```
const int buttonPin = 2;
const int ledPin = 8;
int buttonState = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode (ledPin, INPUT);
  digitalWrite (ledPin, LOW);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW){
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin,LOW);
  }
}
```



U suštini, u pitanju je blago prepravljeni blink skeč, kome smo dodali mogućnost uključivanja putem prekidača. Skeč je postavljen klasično, dok se sve dešava u loop funkciji, gde čekamo stanje prekidača i, u zavisnosti od očitavanja, „palimo“ LED. Povezivanje ide prema fritzing šemi (slika gore), gde smo dodali jedan otpornik od 10K i povezali ga na 5V (pull-up) da bismo eliminisali „plivajući“ input i dozvolili ispravno očitavanje logičke nule i jedinice.

Proto šild je prilično jednostavan za ugradnju i pravljenje projekata, i sa njim neće imati problema ni oni manje iskusni. Mi nastavljamo seriju šildova u sledećem broju, a oni koje smo vam spremili u narednom periodu neće biti ovako lagani.

Arduino: Alatke za programiranje

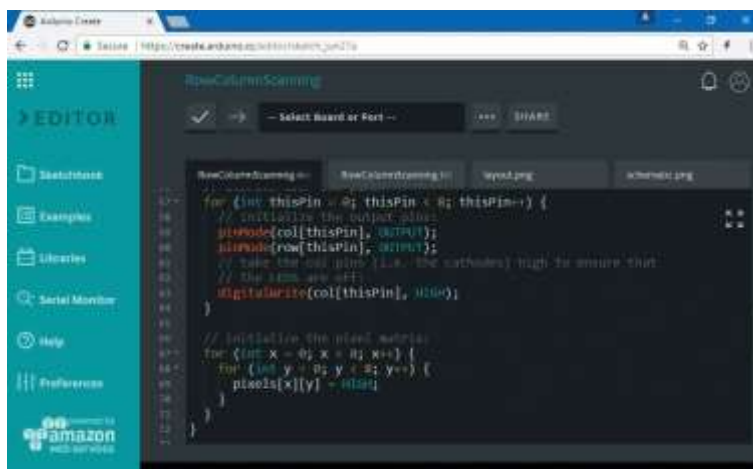
Da skečevi samo pršte

Igor S. RUŽIĆ, Svet kompjutera, 2017

Originalni Arduino IDE ima jednu fantastičnu karakteristiku – maksimalno je jednostavan za korišćenje. Napišemo kod, odredimo adresu COM porta i model uređaja, i za nekoliko trenutaka naš skeč biva prebačen u memoriju mikrokontrolera. Upravo zbog jednostavnosti veliki broj korisnika (posebno onih koji nemaju većeg iskustva u programiranju) i ne razmišljaju o nekoj alternativi. Ta jednostavnost korišćenja je, na žalost, plaćena vrlo ograničenim skupom funkcija za rad sa kodom, pa se svi oni koji su bili u prilici da osete udobnost rada u nekom od programskih editora osećaju kao riba na suvom. Jedine dve funkcije Arduino IDE koje bismo mogli da svrstamo u napredne mogućnosti savremenih editora odnose se na sintaksno bojenje koda i njegovo automatsko formatiranje tasterima 'Ctrl+T'. Sve ostalo je čista misaona imenica. Namera teksta je da čitaoc upozna sa alternativnim rešenjima kojih ima prilično mnogo.

Arduino Create

Pod ovim imenom se krije veb verzija standardnog desktop Arduinovog razvojnog okruženja, stoga nećemo imati napredne funkcije za uređivanje programskog teksta, pošto se i ovde poštuje filozofija minimalizma. Centralni deo je, kao i u slučaju desktop verzije, posvećen pisanju koda. Ispod njega je područje sa informacijama o procesu kompilacije, dok je na vrhu smešteno područje za izbor modela i ekranski tasteri za kompilaciju i prebacivanje koda.



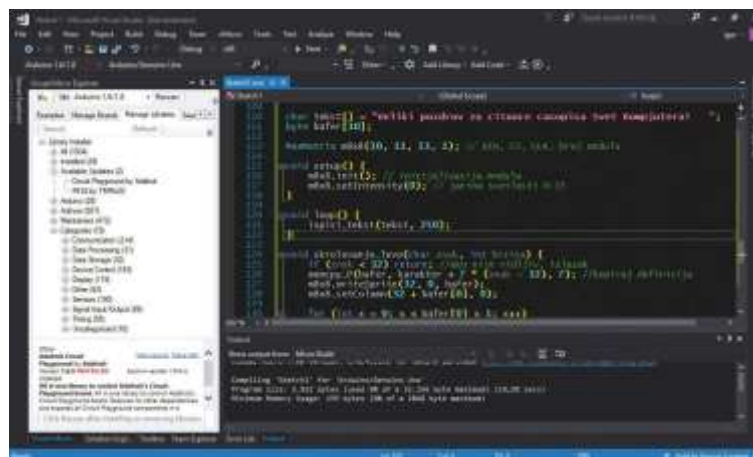
Meniji su ovde smešteni sa leve strane i sadrže programske biblioteke, demonstrativne primere, serijski monitor i opcije za podešavanje. Tamo je, između ostalog, moguće izabrati rad sa dve teme, od kojih je jedna u svetlim, a druga u tamnim bojama. Za sinhronizaciju sa uređajem se koristi poseban

program pod nazivom Arduino Create Agent, koji se nakon instalacije automatski pokreće podizanjem operativnog sistema i smešta se u područje systemske kasete.

Pozitivna strana korišćenja veb editora jeste mogućnost čuvanja projekata u oblaku, a za to je potrebno da otvorimo korisnički nalog na sajtu arduino.cc. Isto tako, ovaj editor možemo da koristimo na više različitih računara, bez potrebe za prenošenjem skečeva i za instalacijom desktop verzije. Doduše, potrebno je instalirati pomenuti Create Agent, ali on ne zauzima mnogo prostora na disku. Postoji i mogućnosti preuzimanja skečeva iz veb aplikacije na desktop računar.

Arduino IDE for Visual Studio

U prošlom broju „Sveta kompjutera“ (7/2017) smo predstavili izuzetno moćan programerski instrument Visual Studio 2017, koji se, zahvaljujući postojanju ekstenzije koju je kreirala kompanija Visual Micro (zbog toga se često naziva i Visual Micro for Visual Studio), pretvara u pravo profesionalno radno okruženje sa praktično svim funkcijama koje na raspolaganju imaju „pravi“ programeri. Čak bismo se usudili da kažemo da je u ovoj formi reč o „prejakom“ okruženju za jednu platformu skromnih zahteva kao što je Arduino.



Prilikom prvog pokretanja, nakon instaliranja ekstenzije pred nama se pojavljuje dijalog koji traži da odredimo folder sa izvornim Arduino IDE programom, te nam daje mogućnost da dodamo podršku za neke razvojne pločice koje ne spadaju direktno u klasu Arduino, ali mogu da se programiraju njegovim razvojnim alatima. Nakon instaliranja, u području ispod menija se pojavljuju nove linije sa alatima preko kojih možemo izabrati model Arduina, COM port i još neke detalje vezane za konfigurisanje programa.

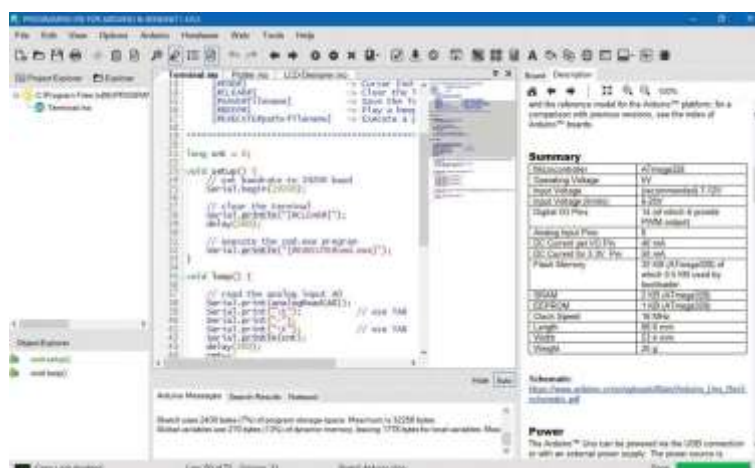
Pored standardnog Visual Studio taba pod nazivom Solution explorer, koji daje prikaz fajlova i programskih blokova u okviru projekta, preko menija pod nazivom vMicro možemo uključiti dodatni tab pod nazivom Visual Micro Explorer. Reč je o sjajnom modulu koji na jednom mestu objedinjuje prikaz

mnoštva stvari važnih za rad sa Arduinoom. Tu su na vrlo pregledan način prikazane informacije o instaliranoj podršci za različite modele Arduina, primeri koda, linkovi sa referentnim materijalom na internetu i tabovi preko kojih na vrlo jednostavan način možemo dodavati i udaljavati podršku za programske biblioteke i instalirane modele razvojnih pločica. Jedina primedba bi se odnosila na nemogućnost prikazivanja materijala sa tamnom pozadinom, pošto prilikom korišćenja tamne teme bela pozadina zna prilično da nervira.

Rad sa programom je veoma udoban, zahvaljujući mogućnostima konfigurisanja Visual Studia i podršci za tehnologiju Intellisense, koja nam pruža spisak potencijalnih naredbi već nakon tri otkucana karaktera.

Programino

U pitanju je razvojno okruženje namenski pisano za platformu Arduino. Nakon instalacije programa, pojavljuje se dijalog u kome je potrebno odrediti lokaciju izvršnog fajla standardnog Arduino IDE, kao i lokacije programskih biblioteka. Potrebno je obratiti pažnju na preporuku da verzije 1.6.10 i 1.6.12 ne rade ispravno, pa je potrebno instalirati neku noviju varijantu. Sa leve strane prozora smešteni su paneli sa Project Explorerom, File Explorerom i Object Explorerom.

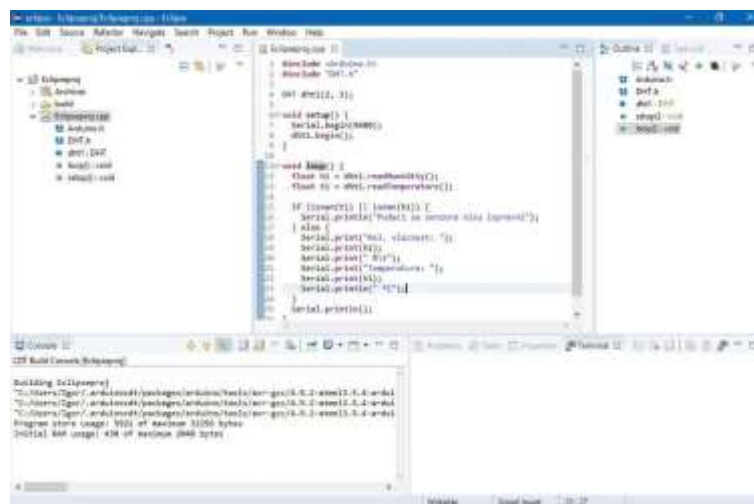


Centralni deo je posvećen editoru programskog koda, dok je na desnoj strani smešten panel sa šematskim prikazom izabrane Arduino platforme i osnovnim tehničkim informacijama o toj platformi. Editor programskog koda je jednostavan za upotrebu i nudi mogućnost sintaksnog bojenja, kao i sistem (polu)automatskog dopunjavanja koda. Sa desne strane editora nalazi se područje sa prikazom teksta programa u umanjenom obliku, što omogućava brzo lociranje željenog koda. Od ostalih stvari, mogli bismo da izdvojimo meni Tools, gde su smeštena dva serijska monitora, ploter podataka pristiglih sa analognih portova, komparator verzija programskog koda, dizajner 5 × 8

znakova za LCD displeje i još dosta drugih sitnica. Zbog jednostavnosti i udobnosti upotrebe, neko bi mogao da pomisli kako je reč o jednoj od najboljih zamena za standardni Arduino IDE, ali malu dozu razočaranja donosi činjenica da se radi o vlasničkom programu sa probnim periodom od 14 dana, nakon kojeg je potrebno kupiti odgovarajuću licencu.

Eclipse

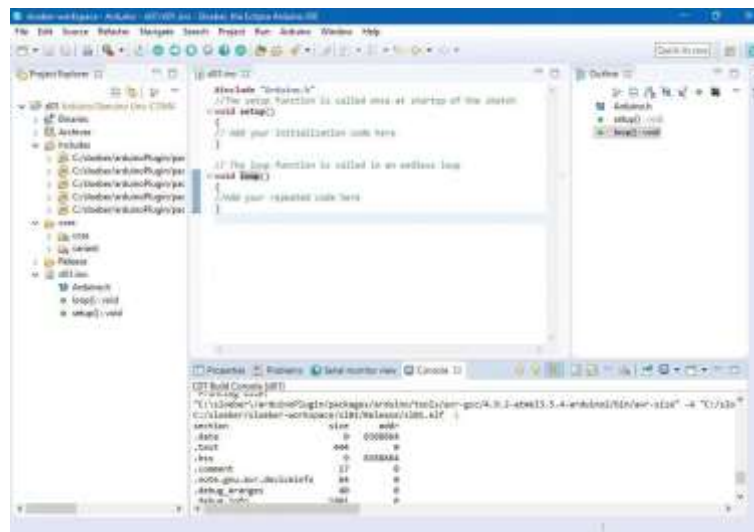
Eclipse je kvalitetno, besplatno, multiplatformsko razvojno okruženje poznato po svojoj univerzalnosti, pa je kao takvo čest izbor mnogih profesionalnih programera. Pored desetina drugih programskih platformi i jezika, našlo se mesta i za podršku popularnog Arduina. Da stvar bude još lepša, postoji nekoliko različitih varijanti pluginova za ovu platformu. Instalacija dodatka se u Eclipseu obavlja preko menija Help: Eclipse Marketplace. Prvo ćemo se pozabaviti pluginom pod nazivom Eclipse C++ IDE for Arduino.



Nakon instalacije se (ponovo) u okviru menija Help pojavljuje stavka pod nazivom Arduino Download Manager, preko koje preuzimamo podršku za željene modele Arduina, kao i programske biblioteke za podršku hardverskih uređaja. Dodavanje se obavlja klikom na ekranski taster Add, koji se nalazi sa desne strane. Sledeća stvar koju radimo je povezivanje sa uređajem preko COM porta. Za to koristimo opciju Window: Show View: Other: Connections. U okviru ovog panela postoji opcija New Connection, u kojoj biramo Arduino i zatim mu dodeljujemo odgovarajući COM port i model uređaja. Nakon što završimo sa pisanjem koda, on se kompajlira i prebacuje na uređaj. Celu proceduru možete videti ovde: goo.gl/n4WdnT.

Drugi plugin nosi naziv Arduino Eclipse IDE named Sloeber. Da bi se izbegle komplikacije, autor dodatka preporučuje da se preuzme kompletna portabilna varijanta Eclipsea, koja u sebi ima integrisan Sloeber verzije 4.1. To se može

učiniti sa lokacije goo.gl/Lpqcir. Skrećemo vam pažnju da je veoma važno da ne prihvatite podrazumevanu putanju do foldera sa radnim fajlovima, pošto ukoliko putanja sadrži više od 40 karaktera, dodatak neće raditi kako treba. Najbolje je da za to kreirate folder u korenu diska. Ako je sve odrađeno kako treba, program će sa interneta preuzeti potrebne datoteke, nakon čega možemo započeti sa radom. U meniju se pojavljuje nova opcija Arduino, iz koje biramo stavku New Sketch i popunjavamo potrebna polja (naziv, vrsta uređaja, COM port). Praktično sve funkcije za rad se nalaza u okviru menija Arduino i jednostavne su za korišćenje.



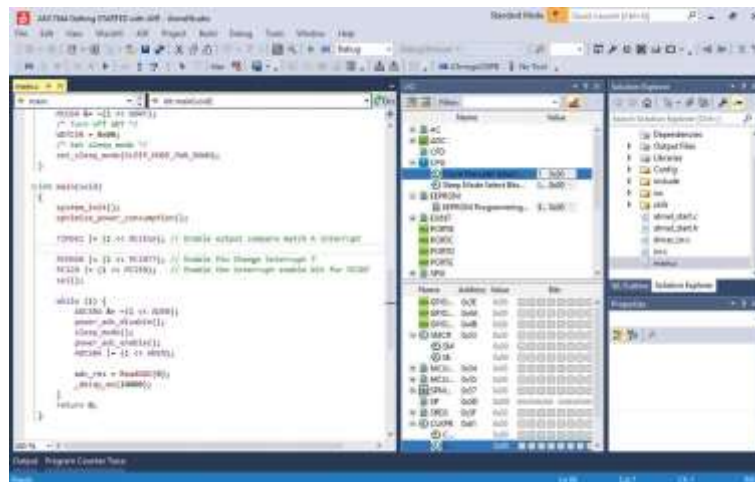
Treća ekstenzija radnog okruženja Eclipse za rad sa Arduinoom nosi naziv AVR Eclipse plugin. Reč je o nešto starijem alatu koji je namenjen programiranju Atmelovih AVR mikrokontrolera i prilično je komplikovan za instalaciju, pa ga zbog skućenog prostora nećemo detaljnije predstavljati.

Atmel Studio

Atmel je naziv kompanije konstruktora mikrokontrolera na kojima je zasnovano najviše modela Arduina. Oni već dugo besplatno nude razvojno okruženje namenjeno uređajima iz njihovog proizvodnog programa. Kada iskusan programer prvi put sedne da radi sa programom Atmel Studio, prva asocijacija mu je da ono veoma podseća na Visual Studio.

U stvari, to i jeste Microsoft Visual Studio (u slučaju verzije 7.0 VS2015), kome su dodati instrumenti za rad sa Atmelovim hardverom, dok su istovremeno izbačeni moduli za rad sa programskim jezicima kao što su VB.NET ili C#.

Podržan je jedino rad sa projektima u jeziku C (GCC kompajler), kao i pisanje asemblerskih programa. Osim toga, podržano je uvoženje koda iz Arduino skečeva. Slično kao kod Visual Studia, i ovde je moguće proširivanje osnovnih mogućnosti paketa preko dodatnih modula. Interesantno je da uz program besplatno dolazi poznati dodatak Visual Assist X.



Ovo razvojno okruženje je namenjeno ljudima koji se aktivno bave programiranjem Atmelovih mikrokontrolera, što znači da njegovo korišćenje generalno zahteva veći nivo stručnosti od onoga koji od korisnika traži platforma Arduino. Međutim, zahvaljujući Visual Micro ekstenziji, koja je veoma slična onoj za Visual Studio, pisanje Arduino skečeva je i ovde jednostavan posao. U okviru ovoga IDE bismo posebno izdvojili modul pod nazivom IO, koji vizuelno prikazuje stanje bitova svih podržanih hardverskih registara. Ovo je izuzetno funkcionalna i korisna stvar, posebno za one koji pišu kod koji se direktno obraća hardveru. Takođe je važno prisustvo simulatora podržanih mikrokontrolera, koji nam pružaju mogućnost debugovanja koda, što je u slučaju najvećeg broja Arduino modela složen posao koji zahteva dodatni hardver.

UECIDE

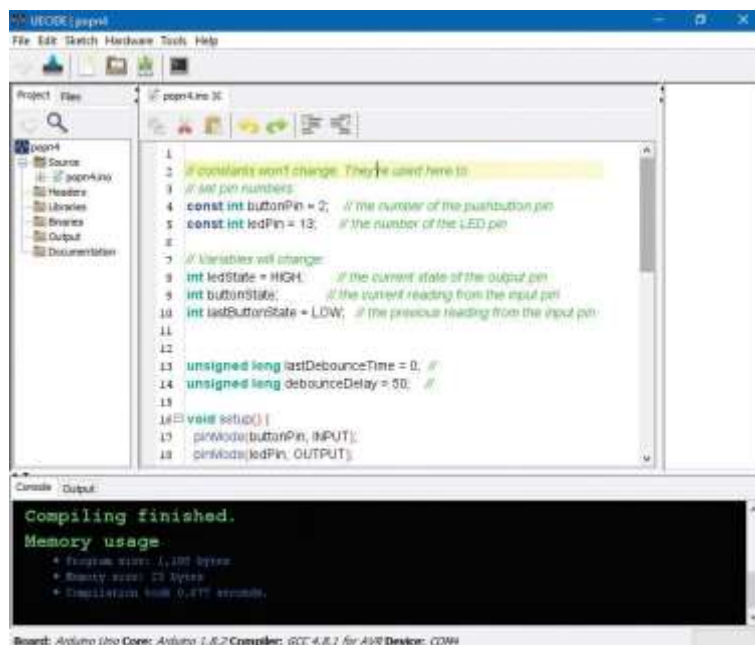
(Klik za uvećanje: JPEG, 68 KB)

Naziv je u stvari akronim za Universal Embedded Computing IDE, što znači da ovaj program nije namenjen radu samo sa Arduino i mikrokontrolerima iz porodice AVR, nego i sa mnogim drugim uređajima, uključujući mikrokontrolere PIC16/18/24/32, ESP8266 i ARM.

Prilikom prvog pokretanja pojavljuje se dijalog prozor (Plugin Manager) koji nam pomaže u izboru hardverske platforme za koju nameravamo da pišemo program. Ovo je ujedno i mesto odakle dodajemo biblioteke sa podrškom za priključeni hardver, kao i dosta drugih stvari.

Samo radno okruženje je jednostavno za korišćenje, ali je to praćeno nedostatkom naprednih mogućnosti. Editor koda jedino sadrži mogućnost sintaksnog bojenja ključnih reči, dok izostaje podrška bilo kakvoj pomoći prilikom kucanja teksta. Da biste uspešno zapisali program na Arduino,

potrebno je u meniju Hardware podesiti parametre koji odgovaraju našem modelu. Naročito je važno da u meniju Programmers bude precizirana opcija Avrdude for Arduino boards. Uz program dolazi serijski terminal sa mogućnošću definisanja teksta za devet ekranskih tastera.

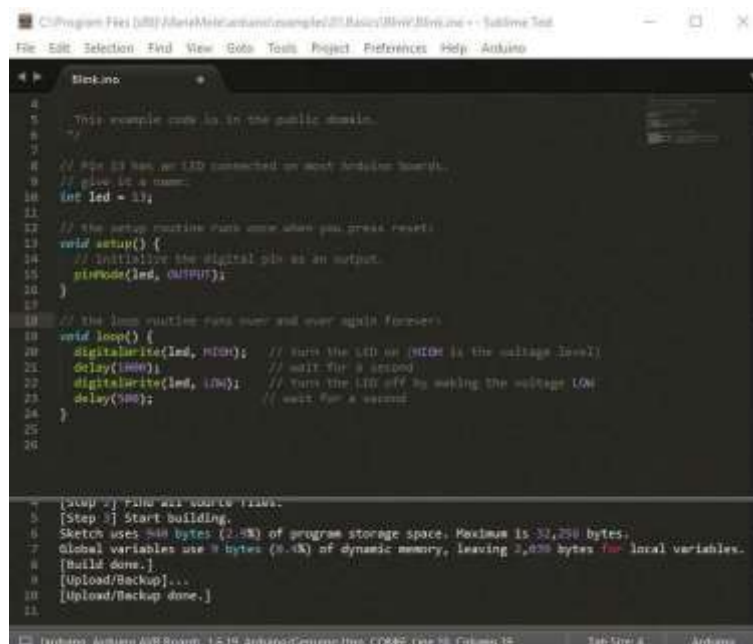


Sam autor priznaje da je zbog velikog posla oko multiplatformske podrške prilično zanemario editor i korisnički interfejs. Sa manjim doradama UECIDE bi mogao da postane zanimljiva alternativa, posebno za one koji istovremeno rade sa više hardverskih platformi.

Sublime Text

O programskom editoru Sublime Text smo pre nekog vremena pisali na stranicama našeg časopisa i tada smo za njega imali samo reči hvale. Lako se može desiti da svojom skromnošću korisničkog interfejsa odbije korisnike koji su navikli na razne grafičke đakonije i fancy izgled, ali zato odlično obavlja posao za koji je namenjen. Reč je o alatu veoma poštovanom među profesionalnim programerima.

Prvo moramo da instaliramo dodatak pod nazivom Package Control, što postižemo pomoću istoimene opcije u meniju Tools. Zahvaljujući njemu uveliko ćemo proširiti funkcionalnost ovog editora.



Kombinacijom tastera 'Shift+Ctrl+P' pozivamo dijalog za instalaciju nekog od mnoštva dodataka, a za to nam je potrebno da prvo izaberemo opciju Package Control: Install package. Kucamo reč Arduino i biramo modul Arduino-like IDE. Nakon uspešno završenog posla, na desnom kraju linije menija se pojavljuje opcija pod nazivom Arduino. Da bismo mogli da instaliramo Arduino skečeve iz Sublime Texta, potrebno je da odradimo nekoliko podešavanja. Prvo biramo opciju Install platform: arduino: Arduino AVR boards: x.x.x (verzija). Sledeća važna stvar je određivanje lokacije paketa Arduino IDE (najčešće C:\Arduino) preko opcije Install platform: Add Arduino IDE. Preostaje da dodelimo naziv modela (Arduino: Board: model) i adresu COM porta (Arduino: Serial Port). Ovoliki broj podešavanja može da izgleda komplikovano, ali verujemo da će svi oni koji imaju bar malo iskustva sa Arduinoom brzo shvatiti smisao predstavljenih opcija. Da biste shvatili punu snagu Sublime Texta, potrebno je da neko vreme provedete u radu sa njim i upoznate se sa njegovim dodacima koji značajno proširuju prvobitne mogućnosti.

• • •

Ovom prilikom smo predstavili alternativna radna okruženja za Arduino, za koja smatramo da su svojim kvalitetom i mogućnostima zaslužili da se nađu u pregledu. Tu nije kraj, postoji još mnoštvo različitih instrumenata koji se mogu upotrebiti za programiranje Arduina. Kada čovek jednom navikne na udobnost okruženja sa naprednim opcijama, rad u standardnom Arduino IDE mu izgleda kao povratak u kameno doba.

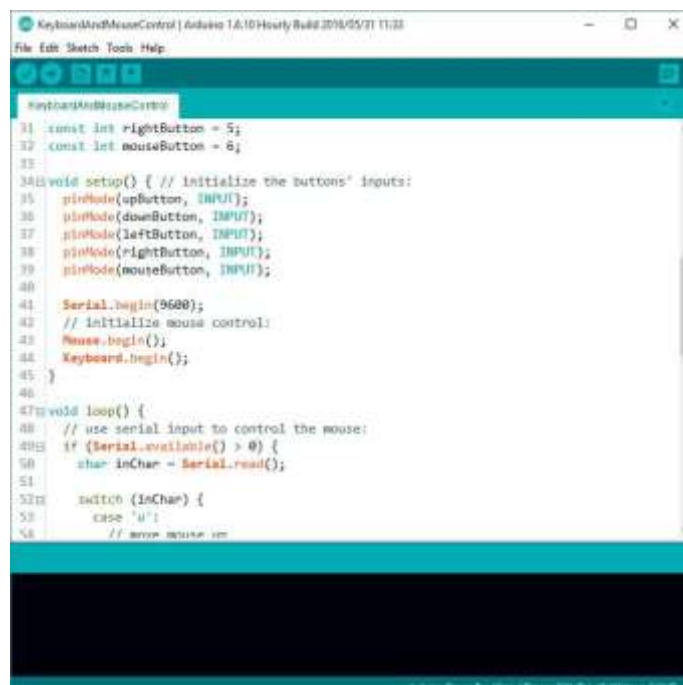
Programiranje na platformi Arduino

Skečevi druge vrste

Igor S. RUŽIĆ, Svet kompjutera, 2017

Tajna velike popularnosti Arduino platforme leži u činjenici da ona po svojoj koncepciji uveliko pojednostavljuje tradicionalni proces konstruisanja elektroničkih sklopova. Zahvaljujući mnoštvu gotovih modula koji se jednostavno dodaju u projekat, stvoreni su uslovi da i apsolutni početnici u svetu elektronike mogu u roku od nekoliko minuta formirati sklopove koji obavljaju dovoljno kompleksne zadatke. Naravno, nećemo daleko odmaći bez poznavanja osnovnih pojmova kao što je Omov zakon ili kakva je funkcija otpornika, kondenzatora i tranzistora, ali rad sa modulima omogućava početnicima da na najbezbolniji mogući način savladaju filozofiju funkcionisanja strujnih kola. Ali to nije sve. Uporedo sa jednostavnim korišćenjem hardvera, kreatori Arduino platforme su učinili možda i još važniju stvar sa uvođenjem mogućnosti jednostavnog programiranja naših hardverskih uređaja.

Arduino IDE



Istorija celog projekta Arduino vezana je za programski alat pod nazivom Processing od kojega je preuzeo mnoštvo stvari. Processing je zanimljiva platforma koja omogućuje pisanje aplikacija na pojednostavljenoj varijanti jezika Java. Reč je o jednom od najefektivnijih načina za programiranje na platformi Android (vlasnicima telefona toplo savetujemo da postavimo program APDE), a podržane su i sve popularne desktop platforme. Dodavanjem

biblioteke funkcija je moguće i programiranje Arduino uređaja direktno iz ove platforme. Arduino je od Processinga preuzeo razvojno okruženje koje je inače napisano na jeziku Java, čemu dugujemo njegovu portabilnost. Tu imamo dobru i lošu vest. Dobra je da je IDE izuzetno jednostavan za korišćenje i da će se u njemu bez problema snaći svaki početnik. Loša vest je ta da će iskusniji korisnici pronaći mnoštvo ograničenja za rad sa složenijim projektima. Od naprednijih opcija, tu su sintakso bojenje teksta (doduše, bez mogućnosti izbora sopstvenih boja), automatsko formatiranje koda (opcija u meniju Tools) i skrivanje koda u okviru funkcija. Nažalost, nisu podržane neke danas standardne mogućnosti, kao što je prikazivanja sintaksne greške prilikom editovanja ili autokompletiranje koda. U jednom od narednih brojeva osvrnućemo se na radna okruženja koja u sebi sadrže sve ono što Arduino IDE iz nekih razloga nije podržao, ali za sada nam je on više nego dovoljan. Kada završimo sa pisanjem skeča (engl. scatch, naziv programa pisanih za Arduino), potrebno je da proverimo da li sintaksa ima greške uz pomoć opcije Verify. Ukoliko nema grešaka, na dnu prozora će se pokazati podaci o zauzeću memorije za pohranjivanje programa i koliko je bajtova otišlo na podatke u RAM. U prošlom broju smo pisali da Arduino Uno ima 32 kilobajta programske memorije i dva kilobajta SRAM, tako da je potrebno ekonomično upravljanje sa ovim resursom. Opcija Upload kompajlira kod i prebacuje ga na mikrokontroler. Da bismo skeč pokrenuli na našem uređaju, potrebno je da prethodno iz menija Tools: Board izaberemo model koji trenutno koristimo. Osim toga je iz menija Tools: Port potrebno izabrati serijski port koji je dodeljen našem USB adapteru preko koga vršimo programiranje Arduina.

Već smo pominjali da je za ovu platformu napisan veliki broj programskih biblioteka koje uveliko olakšavaju korišćenje svakojakih hardverskih modula koje možemo uključiti u naše projekte. Umesto da se sami mučimo sa realizacijom neke funkcije (recimo, želimo da crtamo po displeju) jednostavno se poslužimo sa gotovom bibliotekom koju su neki dobri ljudi napisali pre nas. Ukoliko želimo da dodamo biblioteku koja nam je potrebna, to je najlakše obaviti preko menija Sketch: Import Library. U istom meniju ćemo videti sve biblioteke koje već postoje na našem sistemu. Izborom neke od njih u program će biti automatski dodata sva neophodna zaglavlja preko `#include <>` direktiva.

Iz tona C

Osnovni programski jezik koji se koristi na platformi Arduino je svojevrsna pojednostavljena varijanta jezika C. Tačnije, iz jezika su sklonjene neke kompleksnije stvari i uvedena je C/C++ biblioteka zvana Wiring, koja je preuzeta iz pomenute platforme Processing. Upravo iz činjenice da je

Processing pisan za jezik Java, a Wiring za C/C++ proizilaze i nevelike razlike u njihovoj sintaksi.

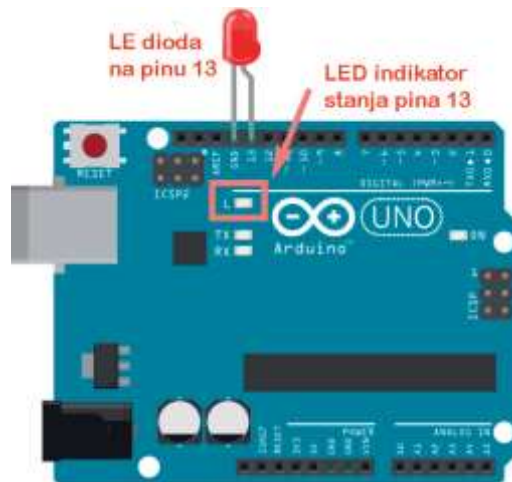
Iako C važi za jedan od težih jezika za savladavanje, ovde je to teško primetiti. Kostur svakog Arduino programa kreiranog uz pomoć razvojnog okruženja Arduino IDE izgleda ovako:

```
void setup() { // put your setup code here, to run once:
}
void loop() { // put your main code here, to run repeatedly:
}
```

Kao što vidimo, reč je o dve funkcije od kojih je prva (setup) zadužena za inicijalizaciju parametara programa i izvodi se samo jednom na početku, dok funkcija loop predstavlja beskonačnu petlju koja se ponavlja tokom rada programa. Pošto se radi o varijanti jezika C, mnogi će se upitati šta je sa nezaobilaznom funkcijom main()? Da bi pojednostavili rad, autori su rešili da je prosto sakriju, tako da ona i dalje obavlja svoj posao iza scene. U stvarnosti, osnovni skelet Arduino skeča izgleda ovako:

```
int main() {
init();           //inicijalizuj USB
setup();          //početne postavke
while(1){loop();} //beskonačna petlja
}
```

Funkcija init() reguliše neke parametre vezane za USB port. Sledi pozivanje funkcije setup() za koju smo pisali da je namenjena izvođenju operacija koje se obavljaju samo jednom na početku programa. Posle toga ulazimo u beskonačnu petlju koja će stalno pozivati funkciju loop() sve do prestanka rada, a u našem slučaju to znači do prekida dotoka električne energije ili pritiska na reset taster.



Zanimljivo je da skoro svi udžbenici na ovu temu za prvo upoznavanje sa kodom navode demonstracioni primer pod nazivom Blink. Razlog za to je svakako taj što svi modeli Arduina na sebi imaju malu LED lampicu koja je fizički povezana sa digitalnim izlazom 13 i pokazuje njegovo trenutno stanje. Ukoliko postavimo LED sa anodom utaknutom u pin 13 i katodom u pin GND, videćemo isti efekat. Dakle, moguće je demonstrirati rad uređaja bez ikakve dodatne opreme.

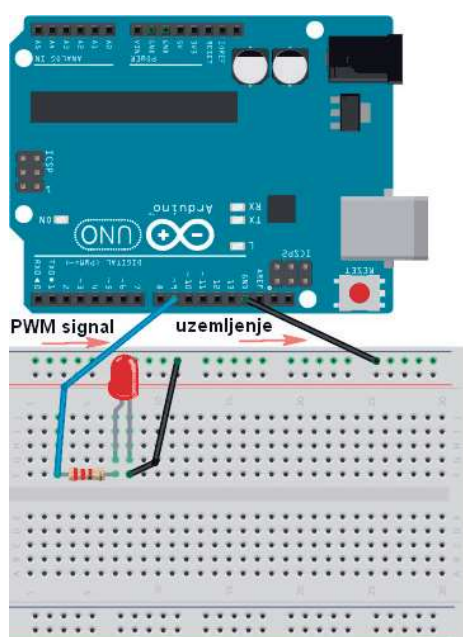
```
void setup() {
    pinMode(13, OUTPUT);
}
void loop() {
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
}
```

Unutar funkcije setup(), digitalnom U/I pinu broj 13 funkcijom pinMode() dodeljujemo funkciju izlaza. To je potrebno zbog toga što U/I pinovi mogu obavljati kako funkciju ulaza, tako i izlaza. Iako su svi U/I pinovi inicijalno postavljeni u režim rada INPUT, moguće ih je koristiti bez pozivanja funkcije pinMode() ukoliko preko njih očekujemo prijem podataka. Ipak, toplo se savetuje da se u cilju izbegavanja nesporazuma ta funkcija uvek eksplicitno koristi.

Unutar beskonačne petlje loop(), pomoću funkcije digitalWrite() na pinu 13 postavljamo stanje takozvane logičke jedinice uz pomoć konstante HIGH, što dovodi do uključivanja LED lampice. Logička jedinica i nula predstavljaju opsege gornjeg i donjeg napona sa kojim funkcioniše neka komponenta. Generalno, kod modela UNO i MEGA2560 ona predstavlja napon od dva do

pet volti, dok logička nula predstavlja napon u opsegu od nula do jedan volt. Kod modela koji rade na 3,3 volta sa HIGH predstavljamo naponsko stanje od dva do 3,3 volta, dok je LOW jednako naponu u opsegu od nula do 0,8 volta. U praksi su ovi opsezi nešto uži, recimo, u slučaju TTL kola gornji napon je u oblasti 2,7-5 volti, dok je donji 0-0,4 volti. Umesto konstanti HIGH i LOW mogli smo da koristimo brojeve 0 i 1 kako bismo označili u kakvom se logičkom stanju trenutno nalazi ukazani pin, ali je ipak elegantnije raditi sa njima.

Nakon što smo uključil LED lampicu, mikrokontroler izvršava funkciju `delay(1000)` koja pravi pauzu od 1000 milisekundi, odnosno jedne sekunde. Sledeći korak je izvođenje funkcije `digitalWrite()` sa kojom na digitalni izlaz 13 šaljemo stanje logičke nule, koje izaziva isključenje lampice i to u intervalu od 1000 milisekundi nakon čega nas beskonačna petlja ponovo vraća na početak. Rezultat rada programa je smenjivanje perioda kada dioda svetli i kada ne emituje svetlost.



Još jedan vrlo jednostavan primer koji demonstrira funkcionisanje PWM tehnike digitalno-analogne konverzije od nas zahteva da na uređaj priključimo LE diodu i otpornik od 220 oma.

Na anodu LE diode (poznaje se po dužoj nožici i maloj izbočini na plastici) dovodimo signal sa pina 10 preko otpornika od 220 oma (zavisi od vrste diode koja se koristi, u najvećem broju slučajeva će poslužiti otpornici od 160-600 oma) koji je štiti od pregorevanja. Na katodu (kraća nožica) dovodimo uzemljenje sa pina GND (na ilustraciji uzemljenje prvo dovodimo na liniju uzemljenja prototipske ploče pa odatle na katodu). Pin broj 10 Arduino Uno (kao i pinovi 3, 5, 6, 9 i 11) ima funkciju pretvaranja digitalnog signala u

analogni ekvivalent tako što emituje 256 različitih nivoa napona u opsegu nula do pet volti. Pa da vidimo kako se to ostvaruje programski:

```
int svetlost=0; // intenzitet LED svetla
void setup() {
  pinMode(10, OUTPUT); // pin 10 je PWM izlaz
}
void loop() {
  for (svetlost=0;svetlost<255;svetlost++)
  {
    analogWrite(10, svetlost);
    delay(10); // kratka pauza
  }
  for (svetlost=255;svetlost>0;svetlost--)
  {
    analogWrite(10, svetlost);
    delay(10); // kratka pauza
  }
  delay(2000); // duža pauza
}
```

Nivo osvetljenosti LE diode koji želimo da zadamo se nalazi u varijabli svetlost koju preko for-next petlje šaljemo na pin 10, povećavajući je do maksimalne vrednosti 255, da bismo zatim na isti način tu vrednost smanjivali do nule. Između svakog koraka pravimo pauzu od 20 milisekundi kako bi bio vidljiv efekat postepenog uključivanja i isključivanja lampice. Iza svakog ciklusa uključivanja i isključivanja imamo pauzu od dve sekunde. Primer je suštinski dosta sličan prethodnom sa jednom razlikom da ovde umesto funkcije digitalWrite koristimo analogWrite, pošto ova prva ima mogućnost prikazivanja samo dva stanja: uključeno i isključeno. Cela priča vezana za Arduino programiranje se uglavnom vrti oko čitanja i upisivanja analognih i digitalnih vrednosti, a ona je zahvaljujući konceptu platforme Wiring izvedena na jednostavan i razumljiv način.

Od ostalih stvari koje inicijalno dolaze u ovoj varijanti jezika C/C++, na raspolaganju su nam više-manje standardne funkcije za konverziju među različitim tipovima podataka (byte(), char(), int(), word(), long(), float()), matematičke funkcije (min(), max(), abs(), sqrt() i tako dalje), trigonometrijske (sin(), cos(), tan()), vremenske (delay(), millis(), micros()) za generisanje

slučajnih brojeva (`random()`, `randomseed()`)... Što se tiče manipulacije sa tekstom, izvorno je podržan rad sa baznim nizovima alfanumeričkih karaktera, ali nam je na raspolaganju i klasa `String` koja sa sobom donosi niz naprednih metoda za rad sa tekstom. Sa jedne strane, rad sa tipom podataka `String` i metodima istoimene klase pruža programeru moćno sredstvo za obradu ove vrste podataka, dok sa druge strane generiše dosta više koda, pa je potreban oprez u situacijama kada nemamo mnogo resursa na raspolaganju.

Ovaj pregled nije rađen sa namenom da bude udžbenik programiranja na platformi Arduino, već da posluži kao demonstracija jednostavnosti implementacije jednog dosta kompleksnog jezika kao što je C. Postoji mnogo udžbenika i tutorijala na temu savladavanja tog programskog jezika, pa vam, ukoliko ne poznajete njegovu sintaksu, toplo savetujemo da pročitate neki od njih. Svi oni koji poseduju veštinu rada sa nekim od programskih jezika izvedenih iz C (Java, JavaScript, C#...) sasvim sigurno će u vrlo kratkom roku biti u stanju da pišu skečeve za Arduino.

Lov na bube

Dobar debager para vredi, to zna svako ko je pisao iole složeniji program na računaru. Nažalost, mikrokontroleri ugrađeni u najveći broj Arduino modela nemaju hardversku podršku za analizu koda u realnom vremenu. Da bi se dobila mogućnost debugovanja na Atmel mikrokontrolerima, optimalno rešenje je hardverski uređaj pod nazivom AVR Dragon, čija se cena kreće oko 50 dolara. Mogu se nabaviti i neki jeftiniji (kao i skuplji) modeli koji omogućavaju debugovanje, ali treba voditi računa ukoliko je vaš Arduino baziran na mikrokontroleru Atmega328 da ti debageri podržavaju protokol `debugWIRE`, pošto je to jedini put za analizu koda na ovim uređajima. Postoje i softverska rešenja koja emuliraju funkciju analize koda od kojih je najpoznatije ono pod nazivom Visual Micro koji funkcioniše kao dodatak za MS Visual Studio, ali je nažalost reč o proizvodu čija je cena 25 dolara. Pošto je teško očekivati da neko ko je za Arduino platio nekoliko evra ulaže dosta više novca u stvari bez kojih se nekako može živeti, onda mu preostaje da koristi ono što omogućava kakvu-takvu funkcionalnost u praćenju izvršavanja koda.

Mogućnost te „kakve-takve” kontrole stanja promenljivih ostvariva je preko ugrađene klase pod nazivom `Serial`, koja je namenjena razmeni podataka putem serijskog interfejsa. Priključivanjem Arduina preko USB porta mi ostvarujemo serijsku komunikaciju između računara i uređaja. U okviru radnog okruženja u meniju Tools postoji opcija pod nazivom `Serial Monitor` koja omogućuje slanje i prikaz informacija preko serijske veze (prečica je ikonica u

gornjem desnom uglu prozora). Pošto Arduino u baznoj varijanti nema nikakav oblik displeja, ova opcija nam pruža mogućnost da prikazemo rezultate izvršenja programa na mikrokontroleru.

Uzmimo na primer kako bi izgledao kod koji računa vrednost hipotenuze preko Pitagorine teoreme.

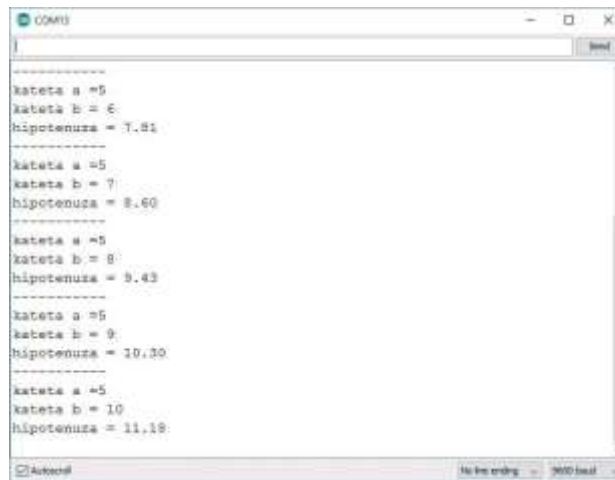
```
#include "math.h"
int kata=5;                //fiksna vrednost katete A
int katb;                  //promenljiva kateta B
float hipotenuza;
void setup(){
    Serial.begin(9600);      // brzina serijskog porta
    Serial.println("Pitagorina teorema");
    for (katb=3;katb<11;katb++) {
        Serial.println("-----");
        Serial.print("kateta a =");
        Serial.println(kata);
        Serial.print("kateta b = ");
        Serial.println(katb);
        hipotenuza = sqrt(kata*kata+katb*katb);
        Serial.print("hipotenuza = ");
        Serial.println(hipotenuza);
    }
}
void loop(){}

```

Kao što pratimo vrednosti promenljivih u navedenom primeru, na isti način možemo „loviti” vrednosti koje bi ukazivale na grešku. Recimo, moguće je unutar prethodnog koda zadati proveru uz pomoć If-Then konstrukcije koja bi se aktivirala u slučaju pojave neke specifične ili nedozvoljene vrednosti:

```
if (hipotenuza>10) {
    Serial.println("dužina hipotenuze veća od 10");
}

```



To bi bio neki osnovni demonstrativni šablon upotrebe klase Serial za praćenje stanja promenljivih i obaveštavanja korisnika o njemu. Koliko god da se radi o primitivnom načinu debugovanja, u manjim projektima je i ovo sasvim dovoljno da se otkrije kod koji izaziva grešku. Kao što to možemo videti na pratećoj ilustraciji, u prozoru Serial Monitora dobijamo podatke u tekstualnom obliku, na sličan način kao što ih dobijamo korišćenjem komandne linije operativnog sistema. Bez većih problema je moguće koristiti neki od brojnih programa za emulaciju terminala umesto onoga koji dolazi sa programom.

U gornjem delu ovog prozora se nalazi ekranski taster Send i polje za unos alfanumeričkih karaktera koji se šalju na uređaj i koji se mogu koristiti u okviru našeg programa. Jednostavan primer slanja podataka bi izgledao od prilike ovako:

```

char karakter="";
// varijabla za prihvatanje karaktera
void setup() {
  Serial.begin(9600); // brzina serijskog porta
}
void loop() {
  if (Serial.available()>0) { // tekst unesen?
    karakter=Serial.read(); // čitamo ga
    Serial.println(karakter); // ispis
  }
}

```

• • •

To bi bilo, u najkraćem, ono što bismo mogli da napišemo o programiranju za Arduino. Ukoliko posedujete određeno znanje iz programiranja vrlo brzo ćete se adaptirati na specifične aspekte ovog okruženja i početi da pišete upotrebljive skečeve. Ukoliko spadate u totalne početnike, lepota Arduino platforme je u tome da će vam kroz praksu upravljanja hardverskim uređajima omogućiti progresivno učenje programiranja koje ćete posle moći iskoristiti za savladavanje velikog broja programskih jezika zasnovanih na jeziku C. Programiranje za Arduino zaista nije teško i sa malo koda možemo postići vrlo zanimljive rezultate



Arduino: Korisnički unos podataka

Interakcija korisnika sa Arduino platformom
Igor S. RUŽIĆ, Svet kompjutera, 2017

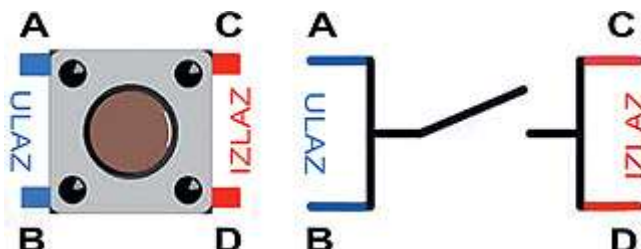
Do sada nismo poklanjali naročitu pažnju jednoj važnoj oblasti koja se odnosi na procesiranje informacija unesenih od strane korisnika, pa ćemo situaciju pokušati da popravimo ovim pregledom.

Tasteri i prekidači

Svi znamo da se razne vrste tastera veoma često koriste u sklopu elektronskih uređaja. Iako vizuelno mogu mnogo da se razlikuju, princip rada im je jednostavan i sličan.



Funkcija tastera je da na intervenciju korisnika o(ne)mogući protok struje iz jednog dela komponente u drugu, slično funkcionisanju visećeg mosta.

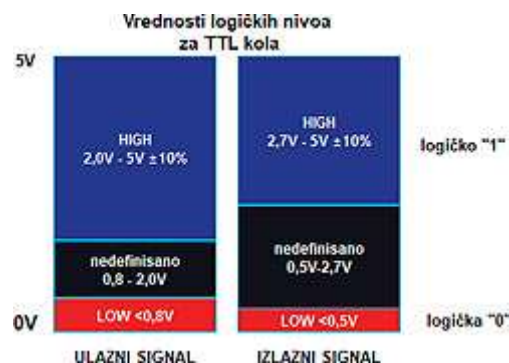


Nije moguć prelazak sa jedne na drugu obalu reke, sve do momenta dok most nije spušten. Treba pomenuti da postoji i poseban (ređe korišćen) tip tastera koji radi suprotno, odnosno, na izvršeni pritisak zaustavlja protok struje, dok u standardnom stanju struja kontinuirano teče. Takvi tasteri se označavaju kao NC (normally closed), dok su standardni označeni kao NO (normally open).

Osim tastera (push buttons), često se sreću i prekidači (switches), čija je funkcija da konstantno drže protok struje u uključenom ili isključenom stanju. Postoji i svojevrsna kombinacija klasičnog tastera push tipa sa prekidačem (na ilustraciji, sa plavom kapicom) koja konstantno propušta struju dok se komponenta nalazi u poziciji „uključeno”, odnosno, fiksirana u položaju prema dole. Tasteri se obično realizuju sa dva reda kontaktnih nožica. Razlog postojanja dva reda jeste u tome što veći broj nožica omogućuje bolju

stabilnost komponente na štampanoj ploči. U radu sa Arduinoom najčešće ćemo koristiti tastere push tipa sa po četiri pina koja smo na ilustraciji označili slovima AB (ulaz) i CD (izlaz). Dovoljno je dovesti napon na samo jedan od ta dva pina, pošto su oni međusobno povezani.

Šta se dešava kada direktno povežemo taster na Arduino? Iako bismo očekivali da, u zavisnosti od toga da li je taster pritisnut ili ne, na ulazu mikrokontrolera imamo stanje logičke nule ili jedinice, dešava se da u stvari imamo nedefinisano stanje visoke impedanse kod koga se vrednost stalno nepredvidivo menja usled elektromagnetnog šuma.

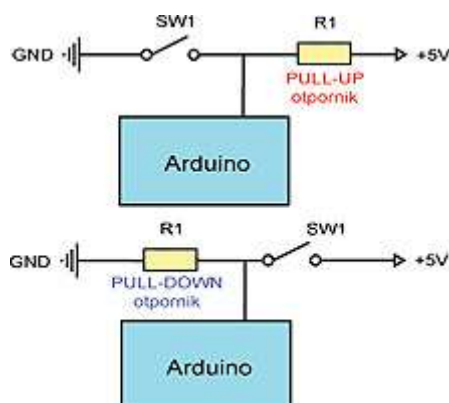


Tačnije, ovo stanje imamo samo kada taster nije pritisnut. Da bismo lakše shvatili o čemu se radi, pogledajmo sledeći primer:

```
void setup()
{
  pinMode(2, INPUT); //ovde priključujemo taster
  Serial.begin(9600);
}
void loop()
{
  Serial.println(digitalRead(2));
  delay(500);
}
```

Kôd čita ulaz na digitalnom pinu 2. Ukoliko pratimo pristigle podatke u prozoru terminala, videćemo da se oni sastoje od nasumičnih vrednosti nula i jedinica. Ako taster ili žicu koja vodi do pina mikrokontrolera, dodirnemo rukom ili ih izložimo nekom izvoru elektromagnetnog zračenja, videćemo da dolazi do promena u očitanim podacima.

Da bi se ovo izbeglo, potrebno je dodati otpornik na jedan od sledećih načina:



U pitanju je tehnika koja se vrlo često koristi u radu sa integrisanim kolima kada je potrebno da jasno postavimo pin u stanje inicijalne logičke nule ili jedinice. Gornji primer pokazuje vezu preko pull-up otpornika (povezuje se na Vcc liniju). Sve dok taster nije pritisnut, pin Arduina dobija napon koji odgovara vrednosti HIGH, dok nakon pritiska na taster dolazi do oticanja struje prema uzemljenju, a vrednost na posmatranom pinu pada na LOW. Dakle, kada imamo stanje LOW, to je znak da je taster pritisnut, dok HIGH znači da nije. Drugi primer je vrlo sličan, samo što sada pin dobija napon LOW dok taster nije pritisnut. Nakon pritiska dolazi do preusmeravanja struje od izvora 5V prema pinu Arduina i on dobija vrednost HIGH. Pull-down otpornici se uvek povezuju preko GND linije. Generalno, u praksi je moguće koristiti oba načina povezivanja, ali moramo obratiti pažnju na jedan važan detalj. Zbog uštede električne energije koristimo pull-up način ukoliko je taster retko u stanju kontakta (najčešći slučaj), dok se pull-down preferira tamo gde je taster/prekidač najveći deo vremena uključen.

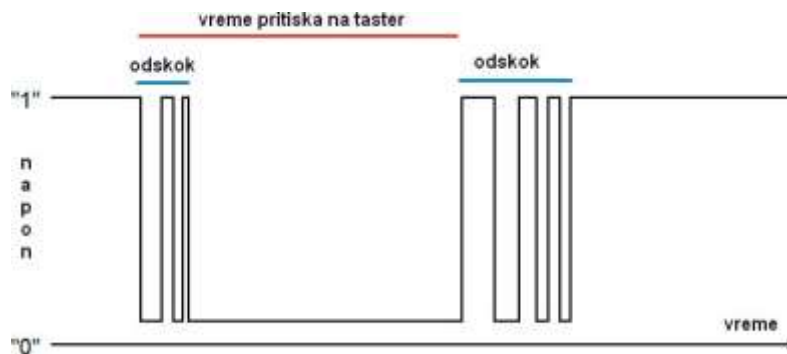
U radu sa naponom od 5V (Arduino Uno), najčešće vrednosti korišćenog otpornika su 4,7 i 10 kilooma. Moguće je izbeći korišćenje eksternih otpornika, aktiviranjem onih koji su ugrađeni u sam mikrokontroler. Naime, svi ulazni pinovi čipa Atmega328P u sebi sadrže sićušne otpornike veličine 20 kilooma, koje je moguće aktivirati na dva načina:

```
pinMode(pin, INPUT);  
digitalWrite(pin, HIGH);
```

ili

```
pinMode(pin, INPUT_PULLUP);
```

Treba imati na umu da zbog pada napona uzrokovano korišćenjem LE diode na pinu 13, ulazni napon iznosi 1,7 volti, umesto standardnih pet volti. U radu sa tasterima srećemo se sa još jednim problemom koji se naziva „odskok” (engl. bounce), a koji se manifestuje kao višestruka promena logičkih stanja u kratkom vremenskom periodu.



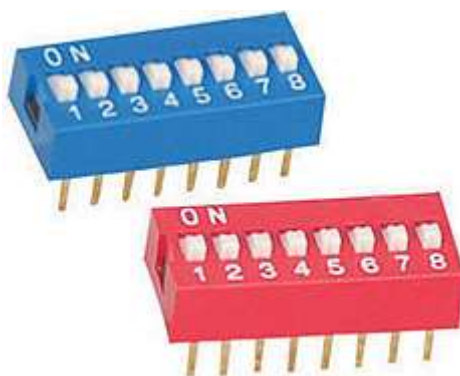
Pošto su u pitanju uređaji sa pokretnim delovima, prilikom pritiska na taster dolazi do sićušnih oscilacija koje se manifestuju gubljenjem kontakta i njegovim ponovnim uspostavljanjem, a čije vreme se meri mikrosekundama. Iako korisnik ne primećuje razliku, na osciloskopu je ta anomalija jasno vidljiva. Problemi mogu da nastanu kada, recimo, brojimo koliko je puta taster pritisnut, pa umesto jedne promene napona imamo više njih, što znači da je taster pritisnut više puta.

I ovde je moguće koristiti pristup sa korišćenjem biblioteke i bez nje. S obzirom na odlične praktično potvrđene rezultate rada, ovde pokazujemo primer pravilnog očitavanja tastera uz pomoć biblioteke Debounce2, koju preuzimamo sa goo.gl/7H99go, gde se nalazi i spisak podržanih funkcija.

```
#include <Bounce2.h>
    Bounce d_taster = Bounce(); //Instanciranje objekta
void setup() {
    pinMode(2,INPUT_PULLUP); //uključujemo ugradjeni otpornik
    pinMode(13,OUTPUT); //LED na pinu 13
    d_taster.attach(2); //pratimo taster na portu 2
    d_taster.interval(5); //interval (ms)
}
void loop() {
    d_taster.update(); //apdejtujemo stanje
    int stanje = d_taster.read(); //uzmi novu vrednost
    if ( stanje == LOW) { //taster pritisnut?
        digitalWrite(13, HIGH ); //upali LED
    }
    else {
        digitalWrite(13, LOW ); //LED ugase
    }
}
}
```


Postoji mogućnost da istovremeno koristimo više tastera preko jednog analognog ulaza na Arduino i to tako što svaki taster povežemo sa otpornikom iste vrednosti, a sve otpornike zatim međusobno povežemo serijski, dok sve izlaze šaljemo na analogni pin. U zavisnosti od toga koji taster je pritisnut, menja se i vrednost otpora koji će uzrokovati različit nivo napona na analognom pinu. Loša strana ovakvog vezivanja je to da istovremeno može biti očitano samo jedan taster. Ako želimo simultano očitavanje više njih, potrebno je da koristimo različite otpornike kojima je vrednost bar dvostruko veća od prethodnika u vezi. Zbog ograničenosti prostora, ovu temu ostavljamo za neki drugi put.

Da bismo zaokružili priču o tasterima i prekidačima, pomenućemo i interesantnu varijantu ulaznih uređaja koji se nazivaju DIP (dual in-line package) prekidačima. Reč je, najčešće, o plastičnim kutijicama koje sa donje strane imaju pinove kojima se fiksiraju na štampanu ploču i na sebi sadrže veći broj malih prekidača, uglavnom od dva do deset.

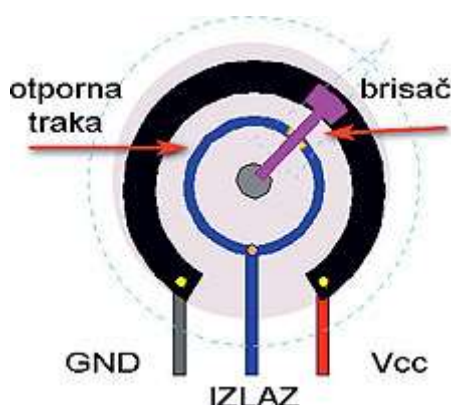


Njihova funkcija je jedino ograničena maštom kreatora, a najčešće se koriste kao džamperi za podešavanje uređaja. Sa cenom od 50 centi i malim dimenzijama, mogu odlično da posluže u mnogim projektima.

Potenciometri



Potenciometri su uređaji kojima se menja vrednost otpora pomeranjem pokretnog dela, koji je obično rotacionog tipa ili u obliku slajdera. Drugim rečima, u pitanju su promenjivi otpornici čija se vrednost definiše akcijom korisnika. Dva glavna tipa su: logaritamski (označeni slovom A ispred vrednosti) i linearni (slovo B).



Uzmimo, na primer, potenciometar od deset kilooma. Kada se nalazi na minimumu, generiše otpor od nula oma, dok na maksimumu postiže svoju nominalnu vrednost od deset kilooma. Za očitavanje je potrebno da se potenciometar priključi na neki od analognih ulaza Arduina, gde će se promene otpora manifestovati promenom napona u analogno-digitalnom konverteru mikrokontrolera. Sve se to dešava u skladu sa Omovim zakonom, pa ćemo najveći napon imati kada je potenciometar na minimumu, a najmanji kada ga postavimo u krajnju poziciju. Arduino prepoznaje 1024 (210) stanja na A/D ulazu, što u prevodu znači da se prepoznaju promene napona u koracima od 0,00488V (5 mV).

```

float alfa = 0.6; //koeficijent ublazavanja (0-1)
int EMA = 0; //inicijalizacija EMA
int stara = 0; //prethodna vrednost EMA
void setup(){
  Serial.begin(9600);
  EMA = analogRead(A3); //EMA pocetna vrednost
}
void loop(){
  int vrednost = analogRead(A3);
  if (vrednost != stara){ //doslo do promene?
    EMA = (alfa*vrednost) + ((1-alfa)*EMA);
    Serial.println(EMA);
    stara = vrednost; //pamti promenu
    delay(20);
  }
}

```

Kod za demonstraciju korišćenja je bio veoma jednostavan, pa smo odlučili da ga malo zakomplicujemo uvođenjem mehanizma za filtriranje podataka, koji je zasnovan na korišćenju funkcije eksponencijalne pokretne prosečne vrednosti (Exponential Moving Average). Iako zvuči prilično „matematički”, stvar je sasvim prosta i cela logika je sadržana u sledećoj liniji koda:

$$EMA = (alfa \cdot vrednost) + ((1 - alfa) \cdot EMA);$$

Što je ekvivalentno formuli: $EMA_t = a \cdot pt + (1 - a) \cdot EMA_{t-1}$

Alfa je konstanta koja može imati vrednost od 0 do 1 i predstavlja intenzitet filtera, pt predstavlja trenutnu vrednost A/D konvertera, dok je EMA_{t-1} vrednost EMA za prethodni period merenja.

Filtriranjem se umnogome otklanja šum koji se javlja u radu sa elektronskim komponentama. Postoji više načina za računanje pokretne srednje vrednosti, a EMA spada među one koji se najviše koriste. Karakteristika EMA je da nikada ne dobija vrednost nula i da daje prednost novijim rezultatima merenja u odnosu na one starije.

Senzorska tastatura

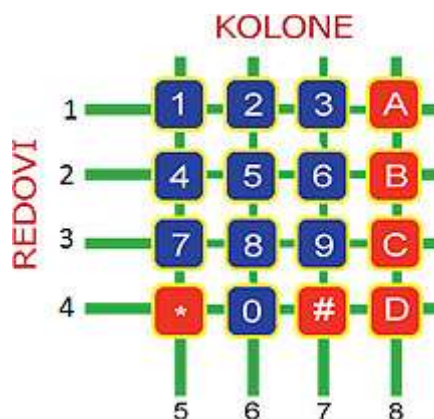
Stariji čitaoci se verovatno sećaju računara Sinclair ZX 80 i ZX 81, koji su početkom osamdesetih bili poznati po veoma niskoj ceni, crno-belom grafici, memoriji od samo jednog kilobajta i senzorskoj tastaturi koja je mnoge

korisnike izvodila iz takta, pošto se često dešavalo da računar pri unosu podataka ne detektuje pritisak tastera. Istina, senzorski tasteri i tastature ne spadaju u najudobnije ulazne uređaje, ali vrlo često mogu da budu veoma elegantno rešenje za unos manjeg obima podataka.



Senzorski paneli mogu da se pronađu u različitim oblicima i sa različitim brojem tastera. Najčešće sadrže brojeve i nekoliko pratećih funkcijskih tastera, ali je moguće kupiti i modele sa samo nekoliko tastera, kao i one na kojima je smeštena celokupna engleska azbuka. To su uređaji jednostavne konstrukcije, pa im je u skladu sa tim cena veoma povoljna. Mi smo za demonstraciju koristili panel od šesnaest tastera, čija se cena kreće u rangu od veoma prihvatljivih 50 centi.

Princip rada je veoma jednostavan. Svi tasteri se nalaze na preseku po jednog reda i kolone. Naš zadatak je da ustanovimo koji od njih je pritisnut. To radimo tako što svaki red postavljamo u stanje logičke nule i zatim proveravamo stanje na kolonama.



Pritisak tastera dovodi do prenosa logičke nule na kolonu sa pritisnutim tasterom, i kada očitamo da se na toj liniji pojavila logička nula, znamo da je taster pritisnut. Na primer, ukoliko je pritisnut taster 7, imaćemo na linijama 3 i 5 stanje logičke nule, što jasno identifikuje poziciju tastera.

```

const char taster[4][4]{
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

int red[4] {11, 10, 9, 8}; //linije 1,2,3,4 (izlaz)
int kol[4] {7, 6, 5, 4}; //linije 5,6,7,8 (ulaz)
void setup() {
  Serial.begin(9600);
  pinMode (11, OUTPUT); //red 1 (linija 1)
  pinMode (10, OUTPUT);
  pinMode (9, OUTPUT);
  pinMode (8, OUTPUT); //red 4 (linija 4)
  pinMode (7, INPUT); //kolona 1 (linija 5)
  pinMode (6, INPUT);
  pinMode (5, INPUT);
  pinMode (4, INPUT); //kolona 4 (linija 8)
  digitalWrite(7, HIGH); //input pinovi,
  digitalWrite(6, HIGH); //stanje logicko 1
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);
}

void loop() {
  for (int r = 0; r <= 3; r++) //4 reda
  {
    //saljemo LOW na red r
    digitalWrite(red[r], LOW);
    for (int k = 0; k <= 3; k++) //4 kolone
    {
      //proveri da li je ta kolona LOW?
      if (digitalRead(kol[k]) == LOW)
      {
        //jeste, ispisi pritisnuti taster
        Serial.print(taster[r][k]);
        Serial.print(" ");
      }
      //taster nije pritisnut
      //vracamo red na logicku jedinicu
      digitalWrite(red[r], HIGH);
    }
  }
}

```


Kao što to u Arduino svetu često biva, korisnicima na raspolaganju stoje biblioteke koje uveliko pojednostavljaju programiranje. U ovom slučaju to je biblioteka Keypad, koju je moguće učitati preko menadžera biblioteka Arduino IDE.

```
#include <Keypad.h>
Const char taster[4][4] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte red[4] = {11, 10, 9, 8}; //linije 1,2,3,4
byte kol[4] = {7, 6, 5, 4}; //linije 5,6,7,8
Keypad senz = Keypad(makeKeymap(taster), red, kol, 4, 4);
void setup(){
  Serial.begin(9600);
}
void loop(){
  char znak = senz.getKey();
  if (znak){
    Serial.print(znak); //salji na serijski port
    Serial.print(" ");
  }
}
```

Dvodimenzionalna promenjiva taster definiše koji znak će biti prikazan kada se identifikuje klik na određenom preseku redova i kolona. Pošto koristimo tastaturu sa 16 tastera, matrica će imati veličinu 4×4 znaka. U slučaju da imamo tastaturu bez tastera ABCD, tada bismo definisali matricu veličine 4×3 znaka. Za program je apsolutno nevažno koji znak će biti prikazan na kojem mestu, pa je umesto vrednosti koje odgovaraju natpisima na tastaturi moguće upotrebiti bilo koji znak, recimo, omogućiti prikaz heksadecimalnih brojeva.

Sledeće dve varijable red i kol definišu pinove Arduina koje ćemo koristiti za povezivanje sa redovima i kolonama tastature. U sledećem redu kreiramo konstruktor senz sa parametrima koji definišu pinove za redove i kolone, kao i dimenzije matrice. Kao parametar se prosleđuje i raspored tastera. Vrednost koja odgovara pritisnutom tasteru dobija se sa metodom getKey. U našem

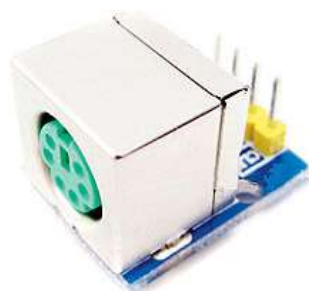
primeru samo ispisujemo taj znak, a moguće je uz pomoć petlje switch proveravati koji je taster pritisnut i na osnovu toga preduzimati željenu radnju.



Neko se može zapitati, zašto da pišemo komplikovani program bez biblioteke, kada je sa bibliotekom sve moguće odraditi brže i elegantnije. Odgovor glasi: varijanta sa bibliotekom zauzima 900 bajtova memorije više, što je važan parametar kada se radi sa mikrokontrolerima. Inače, biblioteka Keypad ima dosta zanimljiv skup funkcija koje omogućavaju pisanje vrlo fleksibilnog koda, kao i mogućnost korišćenja više tastatura odjednom.

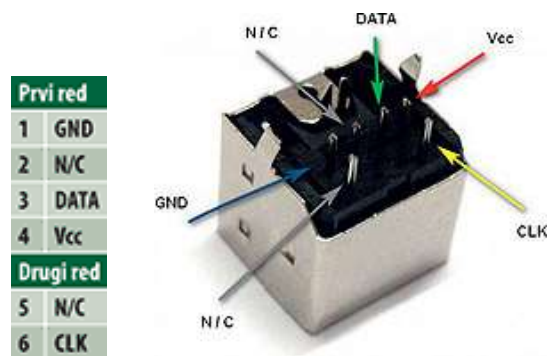
Desktop tastatura

Tastature sa PS/2 konektorom danas se retko koriste, ali se relativno lako mogu naći na starim računarima. Ono što je za nas veoma zanimljivo jeste činjenica da se takve tastature na vrlo jednostavan način mogu priključiti na Arduino i maksimalno olakšati unos podataka.

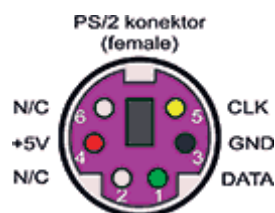


Šestopinske konektore ženskog tipa moguće je naručiti preko interneta (oko 50 centi) ili ih možemo skinuti sa neke stare neispravne matične ploče. Moguće je kupiti gotove module za priključivanje PS/2 tastature po ceni od jedan do jedan i po evro, što smatramo sasvim prihvatljivim.

Gledano sa prednje strane, konektori imaju sledeće kontakte za lemljenje na ploču:



Kao što vidimo, od šest kontakata koristi se četiri, dva za napajanje (Vcc i GND), jedan za sinhronizaciju prenosa (CLK) i jedan za prenos podataka (DATA). Postoji nekoliko biblioteka koje podržavaju rad sa PS/2 tastaturom, ali je najpoznatija ona pod nazivom PS2Keyboard (goo.gl/MjsjD8). Tu je i zanimljiva varijacija na temu pod nazivom PS2KeyAdvanced (goo.gl/ZUAU6X) koja, osim manjeg zauzeća memorije, omogućava i pojednostavljen ispis na priključenom LCD ekranu. Obe varijante je moguće preuzeti preko ugrađenog menadžera biblioteka.



U radu sa Arduinoom UNO, na raspolaganju imamo samo dva pina sa mogućnošću generisanja spoljašnjeg hardverskog prekida (interrupt) i oni su smešteni na pozicijama D2 i D3 (digitalni pinovi 2 i 3). Drugi Arduino modeli imaju veći izbor, pa tako MEGA 2560 omogućava priključivanje na pinove 2, 3, 18, 19, 20 i 21. Obavezno je priključiti CLK signal na neki od tih pinova, pošto u suprotnom tastatura neće funkcionisati. Pin za prenos podataka (DATA) možemo izabrati proizvoljno.

```

#include <PS2Keyboard.h>
PS2Keyboard kb;           //kb je instanca klase PS2Keyboard
void setup() {
  kb.begin(4, 3);          //DATA na pin 4, CLK na pin 3
  Serial.begin(9600);
}
void loop() {
  if (kb.available()) {    //otkucan novi znak?
    char znak = kb.read();  //da, citamo ga
                           //provera dela specijalnih karaktera

    if (znak == PS2_ENTER) {
      Serial.println();
    }
    else if (znak == PS2_DELETE) {
      Serial.print("[DEL]");
    }
    else if (znak == PS2_LEFTARROW) {
      Serial.print("[<--]");
    }
    else if (znak == PS2_RIGHTARROW) {
      Serial.print("[-->]");
    }
    else {                  //znak je slovo, ispiši ga
      Serial.print(znak);
    }
  }
}

```

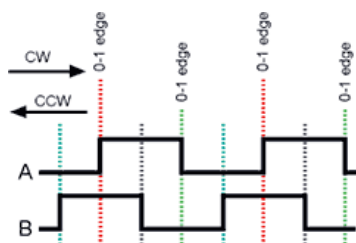
Kao što smo videli mnogo puta do sada, rad sa Arduinom je izuzetno jednostavan, što potvrđuje i ovaj skeč. Konstruktor klase u bloku `setup()` prima parametre koji ukazuju na pinove Arduina korišćene za povezivanje tastature. Sa programskom petljom `loop()` pratimo pojavljivanje novog karaktera u baferu tastature. Ukoliko on postoji, prvo proveravamo da li pripada grupi specijalnih karaktera. To su karakteri koji se na ekranu ne prikazuju kao tekst, već obavljaju neku funkciju. Recimo, Enter počinje sa ispisom u novom redu, Delete briše poslednji znak i tako dalje. U našem primeru ne postoji kod koji bi odgovarajuće reagovao na ponašanje tih kodova (osim za Enter), već jednostavno ispisuje tekst koji nam govori da su ti tasteri pritisnuti. Na završetku koda, ukoliko novi karakter ne spada u grupu specijalnih znakova, smatramo da je u pitanju alfanumerički karakter i ispisujemo ga.

Zaokret bez kraja

Rotary Encoder, Dejan PETROVIĆ Svet komputera 10/2018

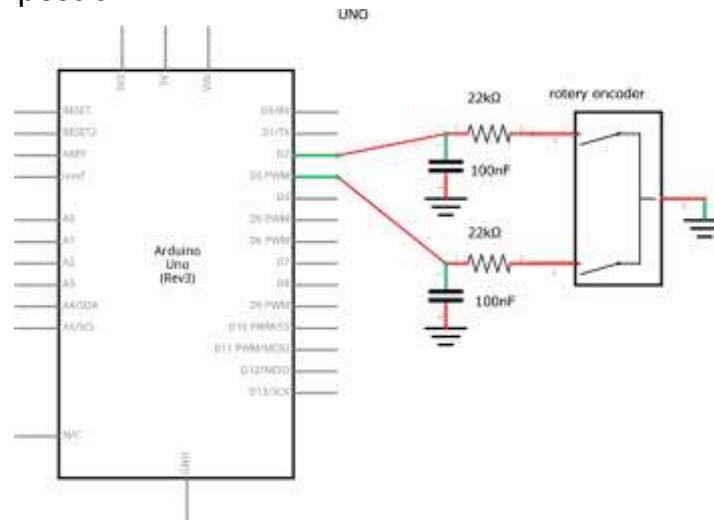


Rotary encoder je uređaj koji umnogome podseća na potencijometar, ali, osim fizičke sličnosti, sa njim nema nikakvih drugih dodirnih tačaka. Kao i u slučaju potencijometra i enkoder ima svoju primenu svuda oko nas. Jedna od čestih primena je na radio-uređajima u automobilima gde preko njega povećavamo i smanjujemo zvuk, a pritiskom na njega uključujemo ili isključujemo sam radio. Tipova ovakvih uređaja ima mnogo. Razlikujemo senzorske (magnetne, optičke i laserske) i apsolutne i postupene. Nas ovaj put interesuje postepeni rotacijski enkoder, kakav se najčešće sreće u Arduino svetu. Ovaj enkoder, osim čitanja zakretanja osovinice, ima i taster kada tu istu osovinicu pritisnemo na dole.



Sam enkoder radi tako što preko ploče generiše kvadratne talase. Ploča ima ravnomerno razmaknute zone preko kojih se ostvaruju kontakti, a koje imaju zajedničku dodirnu tačku (C), koja je zapravo GND i dve odvojene tačke (A i B) koje imaju svoje izlaze na pinovima CLK i DT. Prilikom zakretanja osovinice (shaft), a samim tim i ploče, krajevi ovih zona ostvaruju kontakte. Tom prilikom se stvaraju dva odvojena kvadratna talasa kao output. Koristeći bilo koji od ovih talasa, mi možemo tačno znati da li je došlo do zakretanja osovinice i koliko, ali da bismo znali smer, potrebna su nam oba signala. Ako se osovina zakreće u smeru kazaljke na satu A, izlaz će biti ispred B izlaza, dok će u suprotnom smeru to biti obrnuto. Na šemi vidimo da su oba signala uvek prisutna, ali su razmaknuta dovoljno da se kvadrati talasa, koji su pod 90 stepeni u odnosu na osnovu, ni u kojem slučaju ne poklapaju. Ako idemo u smeru kazaljke na satu, imaćemo pozitivna očitavanja na B pinu (DT), u suprotnom će biti negativna. Na A

pinu (CLK) ćemo brojati pulseve da bismo videli koliko se osovinica zakrenula, a praćenjem frekvencije možemo čak videti i kojom brzinom. Za razliku od pomenutog potencijometra, ovde zakretanju nema kraja u bilo kom smeru. Osovinica se može beskonačno zakretati u nekom smeru, a za potrebe našeg projekta, mi to možemo rešiti u okviru skeča i limitirati na vrednosti koje su nama potrebne. Naš enkoder ima 20 koraka za ceo krug, ali ovo zavisi od modela do modela. Ovo ujedno znači i minimum koraka koji se mogu postići.



Postepeni rotacijski enkoder može biti već ugrađen na PCB u obliku modula sa zalemljenim otpornicima ili u „goloj” varijanti. U zavisnosti od projekta, biramo odgovarajući. Mi ćemo se, naravno, prvo družiti sa ovom prvom varijantom koja je mnogo praktičnija za razradu projekta. Ovde imamo pet pinova gde CLK povezujemo sa digitalnim pinom 2, a DT sa pinom 3. SW pin tastera ide na D4. Pinove 5V i GND povezujemo, takođe, na odgovarajuće pinove Una.

```
//definisanje pinova i promenljivih
int encPinA = 2;
int encPinB = 3;
int switchC = 4;
int encoderPosition = 0;
int switchState = 0;
int switchOutState = 0;
int pos;
int lastPos;
void setup() {
  pinMode(encPinA, INPUT);
  pinMode(encPinB, INPUT);
  pinMode(switchC, INPUT_PULLUP);

  Serial.begin(9600);
  lastPos = digitalRead(encPinA);
}
void loop() {
```

```

pos = digitalRead(encPinA); //citamo poziciju po pinu A
  if(pos != lastPos){ //i ako nije poslednja pozicija
    if(digitalRead(encPinB) != pos){ //ako ni pozicija pina B nije isto sto i pozicija pina A
      encoderPosition --; //umanjujemo poziciju enkodera
    }

    else{
      encoderPosition ++; //u suprotnom je povecavamo
    }

    Serial.print("pozicija je ");
    Serial.println(encoderPosition);
  }
lastPos = pos; //dodeljujemo trenutnu poziciju poslednjoj poziciji
switchState=digitalRead(switchC); //citamo stanje prekidača
if(switchState == LOW){ //ako je pritisnut
  if(switchOutState == 0){ //ako je stanje prekidača 0
    switchOutState = 1; //menjamo ga u 1
    Serial.print("taster je ");
    Serial.println(switchOutState);
    delay(200);
  }

  else{
    switchOutState = 0; //u suprotnom, ako nije 0, menjamo ga u 0
    Serial.print("taster je ");
    Serial.println(switchOutState);
    delay(200);
  }
}
}
}

```

The screenshot shows the Arduino IDE interface. The top window is titled 'rotary_encoder | Arduino 1.8.5'. It displays the following code:

```

delay(200);
} else {
  switchOutState = 0; //u suprotnom ako nije 0 menjamo ga u 0
  Serial.print("taster je ");
  Serial.println(switchOutState);
  delay(200);
}
}
}

```

Below the code editor, the 'Tools' menu is open, showing 'Serial' selected. The 'Serial Monitor' window is open, displaying the following output:

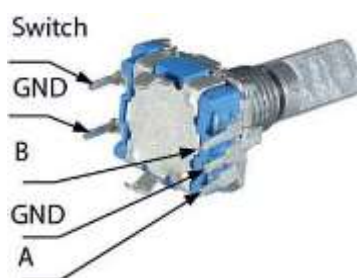
```

pozicija je 2
pozicija je 3
pozicija je 4
pozicija je 1
pozicija je 2
pozicija je 1
pozicija je 0
pozicija je -1
pozicija je -2
pozicija je -3
pozicija je -4
taster je 1
pozicija je -3
pozicija je -3
taster je 0

```

The bottom status bar shows 'No line ending', '9600 baud', and 'Clear output'.

Smatramo da do loop funkcije ne treba da dodatno pojašnjavamo. U loop funkciji čitamo poziciju encPinA (CLK) i, ako nije ista kao prethodna, upoređujemo je sa pozicijom na encPinB (DT). Ukoliko nije ni sa njom ista, umanjujemo encoderPosition, u suprotnom je povećavamo. Pri svakom narednom očitavanju ispisujemo encoderPosition kao trenutnu poziciju enkodera, a poslednju poziciju izjednačavamo sa pos vrednošću. Za taster smo napravili da nam se menja stanje logičke nule i jedinice pri svakom pritisku. Ovde ćemo napomenuti da je taster inicijalno „NO”.



Enkoder ima svoju primenu u dosta projekata. Vrlo često se koristi za stepper i servo motore. U principu, za bilo šta gde je potrebno povećavati vrednosti u koracima. Mikrokontroler će na osnovu toga izvršiti određene zadatke.

Ukoliko je u pitanju obična (non breakout) varijanta enkodera, pristup je malo drugačiji. S obzirom na to da ipak pričamo o mehaničkom uređaju, možemo imati problema sa šumom prilikom kontakata unutar enkodera. Govorimo o istom problemu kao kada smo pričali o tasterima i debounce proceduri. Za projekte gde ćemo enkoder koristiti za ne tako bitne stvari, šum i neće predstavljati prevelik problem. Ukoliko radimo složeniji projekat gde je svaki korak enkodera bitan, poželjno je ugraditi na A (CLK) i B (DT) pinove RC filter.

Filtriranje se može izvesti i kroz skeč, ali se na taj način dodatno komplikuje.

Uvek je bolje problem rešiti hardverski, pre nego posegnuti za softverskim rešenjem. Uz koji otpornik (22-27 kilooma) i keramički kondenzator (100 nanofarada) se ovaj problem može rešiti i ostaviti mikrokontroleru da radi svoj posao. Modul iznad već ima ugrađene otpornike ispod sebe, pa je na taj način šum sveden na minimum. Razlog više da se prilikom biranja dobro razmisli koji enkoder uzeti.

Arduino: Bežično vezivanje

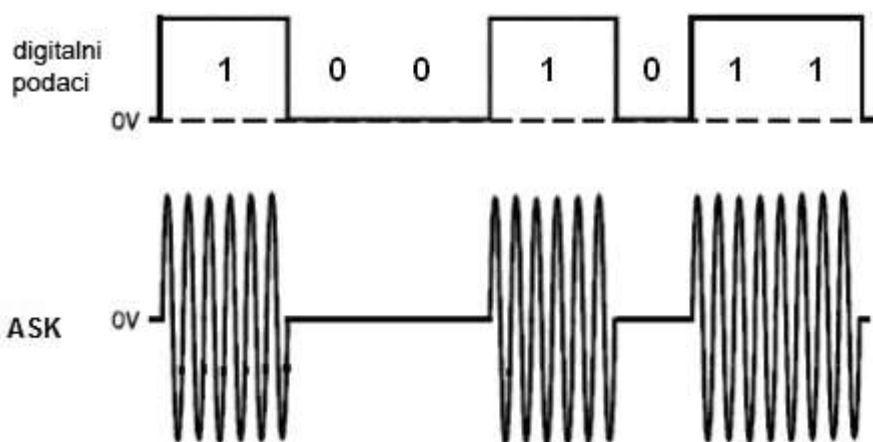
Igor S. RUŽIĆ, Svet kompjutera, 2016

Komunikacija preko radio signala

Avanturu po prostranstvima bežičnih komunikacija započinjemo od najjednostavnijeg (i najjeftinijeg) načina koji se u prenosu informacija oslanja na poznatu tehniku amplitudne modulacije (AM). Tačnije, koristi se srodna tehnika pod nazivom ASK (Amplitude Shift Keying) koja, za razliku od obične modulacije, omogućava manipulaciju digitalnim signalom.

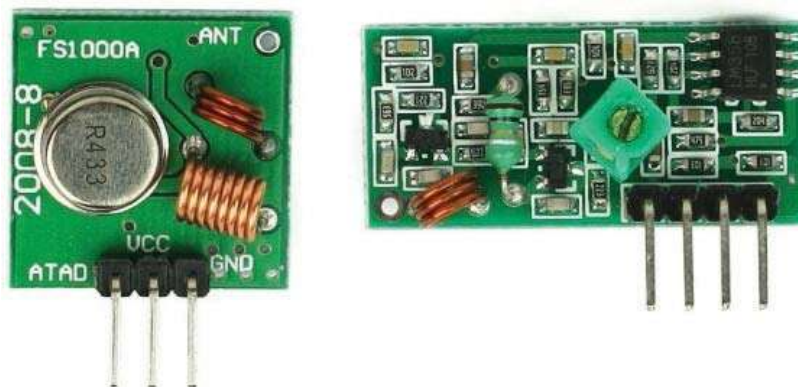
MX-05V/XD-RF-5V

Ovo su najčešći nazivi kompleta prijemnika i predajnika koji se mogu kupiti već za 50 centi. Razlog tako niskoj ceni je jednostavna konstrukcija tih uređaja.



Na tržištu je moguće naći dve vrste skoro jednakih modela, koji se razlikuju po frekvenciji na kojoj obavljaju prenos podataka. U pitanju su 315 i 433,92 megaherca. Dok je Amerikancima potpuno svejedno koji će od ova dva modela da izaberu, evropski zakoni o radio difuziji (CE R&TTE) striktno predviđaju korišćenje drugog.

Postoje još neke razlike u konstrukciji i nazivima radio modula, ali u suštini svi dolaze sa karakteristikama u tabeli dole.



Moduli ove vrste najčešće imaju nejasne oznake na štampanoj pločici.

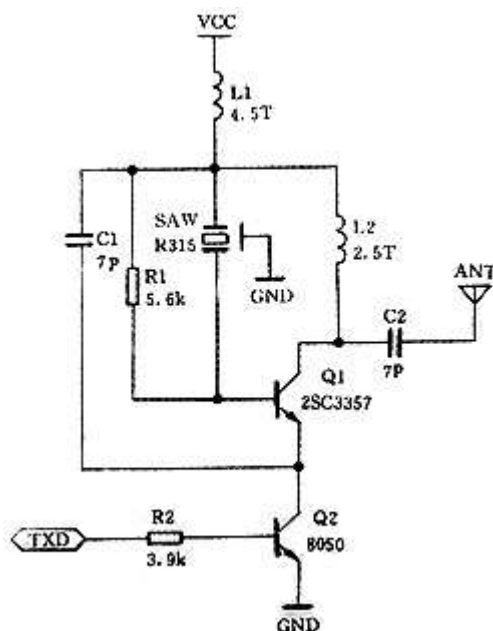
Posebno je zanimljiv natpis ATAD koji predstavlja palindrom reči „data”.

Sledeće tabele daju raspored pinova na modulima kada ih posmatramo sa prednje strane:

Da bi se povećala daljina prenosa podataka, na module treba zalemiti antenu, čija preporučena dužina za frekvenciju od 433 megaherca iznosi 17 centimetara ($\lambda/4$). Dok je za predajnik najpraktičnija omnidirekciona antena, što podrazumeva lemljenje ispravljenog komada žice, za prijemnik se preporučuje spiralna antena (može se formirati namotavanjem žice na valjkasti predmet).

S obzirom na primenjenu tehnologiju, ovi moduli imaju veći broj nedostataka.

Prvi se odnosi na korišćeni frekventni opseg koji je namenjen slobodnoj upotrebi, pa će signal biti slabiji zbog funkcionisanja drugih uređaja iz okoline. Sledeći problem je taj da je u pitanju jednosmerni prenos, kod koga je nemoguće proveriti da li je poslata informacija primljena. Zatim, u pitanju su uređaji koji su veoma osetljivi na šum iz okoline. Ukoliko je na Arduino priključen neki deo koji izaziva stalne pulsacije (na primer servo motor ili LE diode), može doći do velikih smetnji u prenosu podataka, što se može ublažiti korišćenjem dva odvojena izvora napajanja ili dodavanjem kondenzatora. Na kraju, brzine prenosa ove vrste radio veze, za današnje standarde daleko su ispod proseka.



Napredniji korisnici se dodatno mogu pozabaviti implementacijom algoritama za filtriranje šuma i šifrovanje podataka. Isto tako, moguće je kreirati sistem jednostavnog paketnog prenosa informacija koji bi omogućio simultani rad sa više različitih prijemnika i predajnika. Moguće je i istovremeno koristiti više prijemnika koji primaju signal od jednog predajnika.

Najčešće praktične primene ovih modula odnose se na daljinsku kontrolu otvaranja i zatvaranja vrata i prozora, kontrolu audio uređaja, hobističkih dronova, robota i tako dalje. Paradoksalno je da, na sajtovima koji prodaju ovu vrstu uređaja, prodavci često ističu visoku osetljivost uređaja kao vrlinu, dok ona zapravo predstavlja manu usled dejstva drugih izvora talasa u datom radio spektru.

Predajnik	
Model	FS100A, MX-FS-03V, XD-RF-5V, XD-FST
Radni napon	3/5-12 V (3 V / 9 mA, 12 V / 40 mA)
Snaga predajnika	10-40 mW
Modulacija	ASK
Rezonator	SAW (sound wave resonance)
Maks. odstup. frekv.	150 kHz
Brzina prijema	<9,6 Kb/s (2,4 Kb, srednja vrednost)
Daljina prenosa	do 100 m
Prijemnik	
Model	MX-RM-5V, MX-05V, XY-MK-5V
Radni napon	5 V / 4 mA
Osetljivost	-105 DB
Propusna moć	2 MHz
Metod prijema	OOK/ASK
Brzina prenosa	<9,6 Kb/s (2,4 Kb, srednja vrednost)

Zanimljivo je da se na različitim mestima mogu pročitati drastično različite informacije o daljinama prenosa. Negde ćete videti da moduli bez antena

međusobno komuniciraju tek na udaljenosti od par decimetara, dok pojedini izvori na internetu tvrde da je reč i o dometima preko 500 metara. Mi smo u konkretnom slučaju u radu bez antene bili loše sreće, pa smo stabilnu vezu uspjeli da ostvarimo tek na razmaku od nekih 30 centimetara.



Nakon postavljanja antena, sve je bilo mnogo bolje, pa se signal bez problema hvatao i kroz zidove. Po ovakvoj ceni teško se može naći bolji komplet za ostvarivanje bežične komunikacije za savladavanje gradiva, pošto je ceo princip povezivanja jednostavan i razumljiv, što se ne može reći za neke druge metode.

Modul predajnika		Modul prijemnika	
Pin	Funkcija		
1	DATA (ATAD)	1	Vcc (napon 5 V)
2	Vcc (napon 3-12 V)	2	DATA IN
3	GND	3	DATA IN
		4	GND

Postoje dve biblioteke za kontrolu ove vrste modula. Prva se naziva Virtual Wire i na njoj je zasnovan najveći deo projekata, pošto postoji već sedam ili osam godina. Novija, bolja i manje poznata biblioteka, nosi naziv rc-switch.

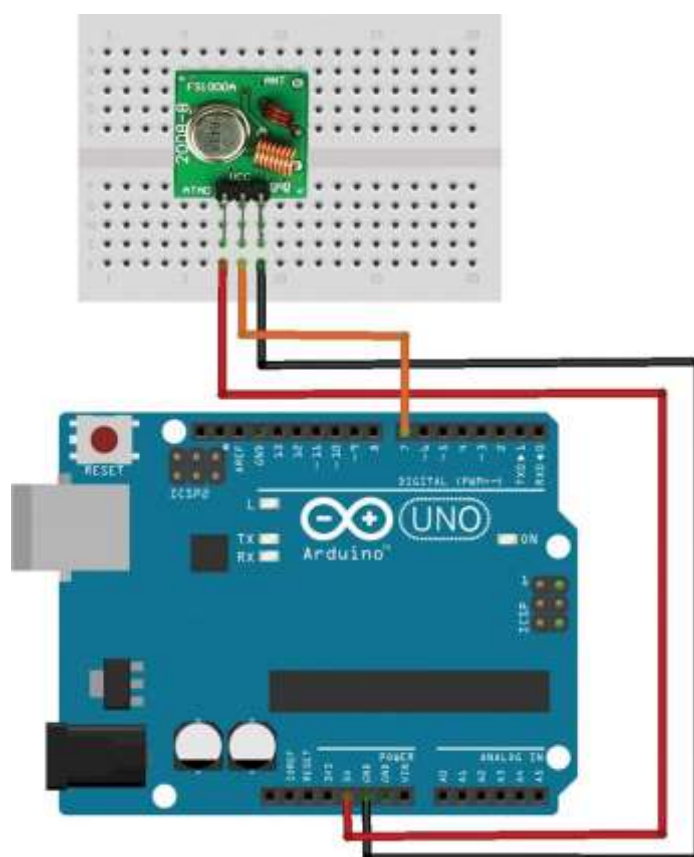
Slanje teksta bi izgledalo ovako:

```
#include <VirtualWire.h>
void setup(){
    vw_set_ptt_inverted(true); // postavljanje polariteta
    vw_set_tx_pin(7); // Arduino data pin za slanje
    vw_setup(2000); // brzina konekcije bps
}

void loop(){
    // pokazivac na tekst za slanje
    const char *tekst = "Svet kompjutera";
    vw_send((uint8_t *)tekst, strlen(tekst));
    vw_wait_tx();
    delay(1000);
}
```

Funkcija `vw_set_ptt_inverted()` postavlja uređaj u stanje potrebno za realizaciju push to talk modaliteta komunikacije. Još je potrebno odrediti preko kojeg pina će naš modul dobijati informacije i po kojoj brzini, što se postiže funkcijom `vw_setup()` koja ujedno inicijalizuje veze. Kao i uvek kada su u pitanju radio veze, brzina je u obrnutoj proporciji sa daljinom prenosa.

Unutar petlje, funkcijom `vw_send()` šaljemo podatke navodeći dva parametra. Prvi je niz bajtova, dok drugi ukazuje na broj karaktera u nizu. Posle toga, funkcija `vw_wait_tx()` čeka da se završi sa prenosom podataka, iza čega sledi pauza od jedne sekunde. Još da objasnimo značenje tipa `uint8_t`. U pitanju je nepredznačena celobrojna osmootna vrednost (unsigned integer) i kao takva je definisana unutar zaglavlja `stdint.h`. Suštinski je ekvivalentna vrednosti tipa `byte` (0-255), ali je korišćenje u obliku `uint8_t` bolji izbor kada se radi o prenošenju na neku drugu platformu koja podržava jezik C. Osim ove varijante moguće je koristiti i standardniji tip podataka `unsigned char`, ali namerno smo koristili ređi oblik, kako bi početnici lakše shvatili o čemu se radi ukoliko naiđu na takvu formulaciju.



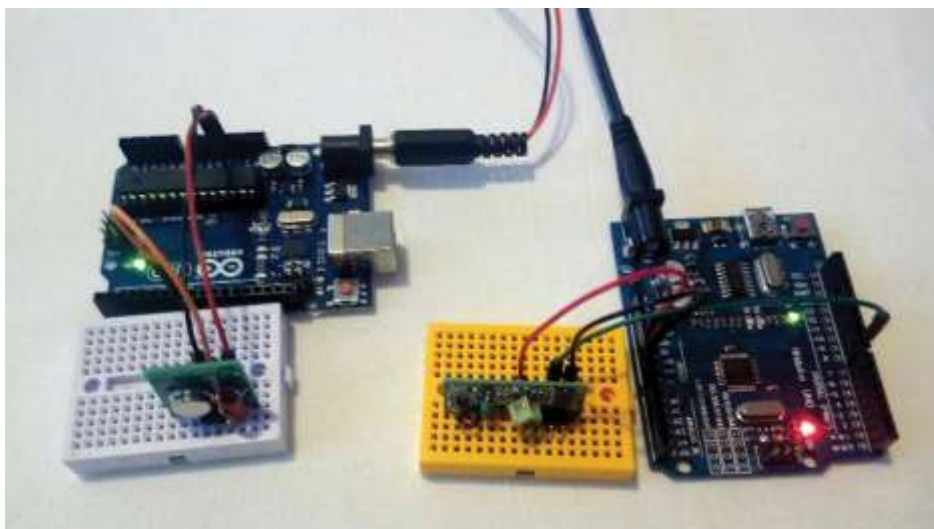
Program za prijem podataka izgleda ovako:

```
#include <VirtualWire.h>

void setup(){
  Serial.begin(9600); // inicijalizujemo serijski port
  vw_set_ptt_inverted(true); // invertovani 'push to talk'
  vw_set_rx_pin(8); // pin za prijem podataka
  vw_setup(2000); // brzina prijema
  vw_rx_start(); // zapocinjemo sa prijemom
}

void loop(){
  uint8_t bafer[VW_MAX_MESSAGE_LEN];
  uint8_t baferlen = VW_MAX_MESSAGE_LEN; // duzina
    podataka u baferu
  if (vw_get_message(bafer, &baferlen)) // ima li podataka?
  {
    for (int x = 0; x < baferlen; x++)
    {
      Serial.write(bafer[x]); // podatke smestamo u bafer
    }
    Serial.println(); // i saljemo na ispis
  }
}
```

U bloku Start prvo aktiviramo serijski port preko koga ćemo pratiti informacije u okviru modula Serial Monitor. Sledi deo sličan onome koji smo koristili za inicijalizaciju predajnika, s tim što ovde eksplicitno izdajemo naredbu `vw_rx_start` za započinjanje prijema podataka.

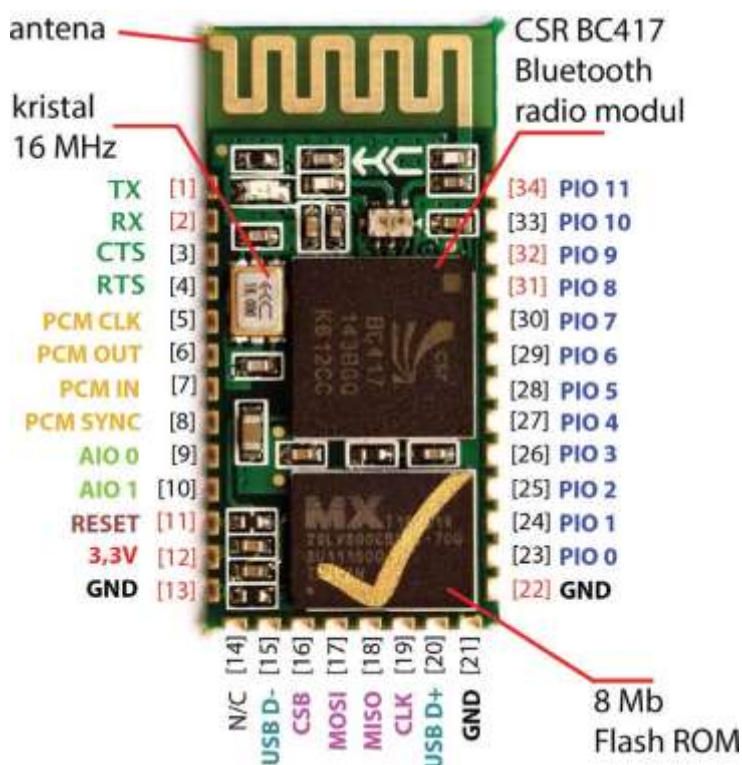


Na početku petlje Loop prvo definišemo bafer i određujemo kolika je njegova dužina. To se postiže korišćenjem makro izraza `VW_MAX_MESSAGE_LEN` koji izvorno ima vrednost 80. Sledi For..Next petlja koja preuzima podatke iz bafera i ispisuje ih u prozoru monitorskog modula.

Poglavica Plavi zub

Igor S. RUŽIĆ, Svet kompjutera, 2016

Sledeći način bežične komunikacije kojim ćemo se baviti odnosi se na dobro poznati Bluetooth sa svim svojim vrlinama i manama. Na tržištu postoji nekoliko sličnih vrsta modula, koji kod početnika unose zabunu, pa ćemo taj problem pokušati da demistifikujemo.



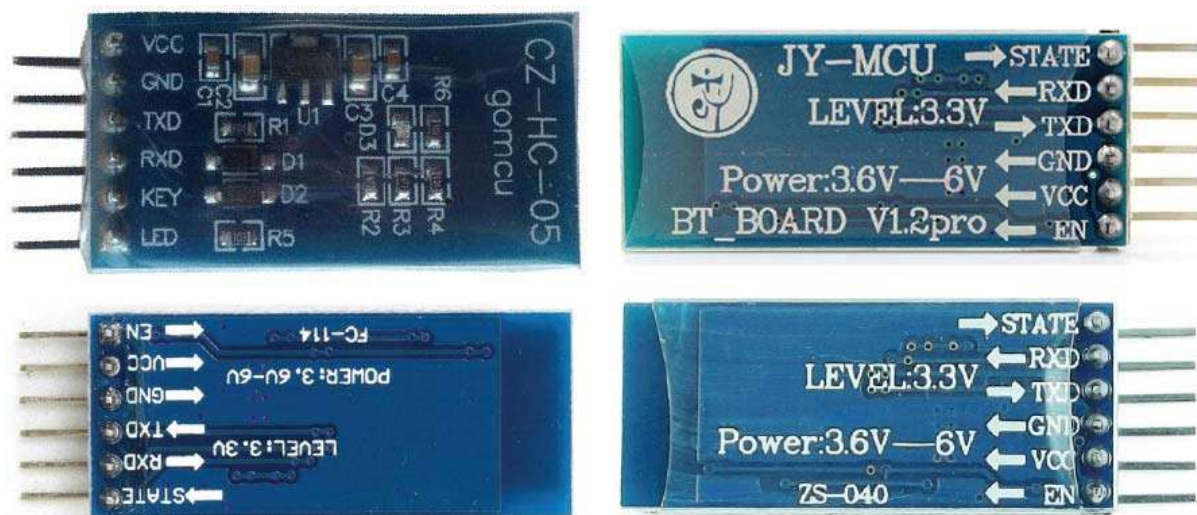
Najčešće se sreću moduli koji imaju oznaku HC iza koje sledi neki dvocifren broj od 03 do 09. Modeli HC-03 i HC-04 reklamiraju se kao podesni za rad u industrijskim uslovima, ali je teško reći u čemu se razlikuju od modela HC-05 i HC-06 koji su namenjeni kućnoj upotrebi. Moduli HC-03/HC-05 mogu po potrebi da obavljaju, kako funkciju slejva, tako i funkciju mastera, dok moduli HC-04/HC-06 mogu da rade samo u slejv režimu, jer im je fabrički onemogućeno da rade u režimu mastera, iako imaju isti hardver. Ko ima hakerske krvi, mogao bi da model HC-06 pretvori u HC-05 kopiranjem programa iz fleš memorije, ali neznatna razlika u ceni obesmišljava tu avanturu. Moduli sa oznakom HC-07 su u potpunosti kompatibilni sa HC-06, samo što koriste čipset CSR 41C6. Pod oznakom HC-08 kriju se noviji (i skuplji) moduli sa čipsetom TI CC2540, koji podržavaju standard Bluetooth 4.0 i troše veoma malo energije, dok je HC-09 (čipset Hanlynn HC1009) najnovija varijanta modula koji bi trebalo da posluži kao zamena za starije modele sa slejv funkcijom.

Mnogi korisnici Arduina će vam potvrditi da je najbolji izbor, kada se uzmu u obzir mogućnosti i cena, model HC-05 (na pločici ZS-040 košta oko dva i po evra), pa ćemo njega razmatrati u našem primeru. Srce modula je čip proizvođača Cambridge Silicon Radio sa oznakom CSR BC417143, koji se ranije često sretao u jeftinijim Bluetooth USB adapterima i čija cena u potpunosti odgovara amaterskim namenama. Pošto je reč o čipsetu koji je na tržištu već desetak godina, podržava nešto stariji Bluetooth standard V2.0+EDR (Enhanced Data Rate) pri brzinama do 3 Mbit/s. Radi u opsegu 2,4-2,48 gigaherca i omogućuje razmenu podataka među uređajima na razdaljini do deset metara. Modul troši najviše struje tokom procesa ostvarivanja veze, do 40 miliampera, dok je tokom rada potrošnja znatno niža i iznosi oko osam miliampera. Treba da napomenemo da su moduli HC-05 (proizvođača Wavesen) i EGBT-045MS (e-Gizmo Mechatronix Central) praktično isti proizvod, samo pod različitim nazivima.

	ZS-040	CZ-HC-05	FC-114	JY-MCU
1	STATE	LED	STATE	STATE
2	RXD	KEY	RXD	RXD
3	TXD	RXD	TXD	TXD
4	GND	TXD	GND	GND
5	VCC	GND	VCC	VCC
6	EN	VCC	EN	KEY

Postoje dva načina upotrebe ovog modula. Prvi je da ga koristimo zalemljenog na podnožje (ZS-040, FC-114, JY-MCU, CZ-HC-05), što je pogodno za postavljanje na prototipsku ploču. Ovo je obavezno prilikom korišćenja Arduino uređaja, koji rade na pet volti, pošto imaju elektroniku koja štiti uređaj od kvara. Drugi način je da potrebne žice zalemimo direktno na modul, što, zbog malog razmaka među izvodima (jedan milimetar), nije jednostavna operacija. Komunikaciju je moguće ostvariti putem USB 1.1 ili 2.0 interfejsa, kao i preko UART konekcije. Najuočljiviji elementi na pločici su dva čipa, od kojih jedan pripada kontroleru, dok je drugi namenjen čuvanju sistemskog programa i ima kapacitet od osam megabita. Kao što se može videti sa slike, modul poseduje sasvim pristojan skup funkcija. Pinovi 1-4 (TX, RX, CTS, RTS) zaduženi su za podršku UART interfejsa, odnosno za ostvarivanje serijske veze. Zatim slede pinovi 5-8, čiji nazivi započinju skraćenicom PCM i obavljaju funkciju serijskog interfejsa za prenos digitalnog zvuka. Pinovi 9 i 10 (AIO0 i AIO1) namenjeni su analognim U/I funkcijama. Na pinovima 15 i 20 (USB D+, USB D-) nalaze se linije diferencijalnih kanala za komunikaciju preko USB interfejsa, dok su pinovi 16-19 (CSB, MOSI, MISO, CLK) zaduženi za SPI interfejs. Na kraju, pinovi 23-34 imaju funkciju programabilnih U/I portova. Na žalost, firmware koji dolazi sa uređajem omogućava upotrebu samo onih pinova čiji su brojevi na ilustraciji označeni crvenom bojom. Ekstremniji korisnici na internetu mogu pronaći materijale koji govore o tome kako je ovaj modul moguće pretvoriti u mnogo moćniji uređaj koji košta desetak puta više.

Pomenuli smo različite tipove pločica na koje je moguće zalemiti naš Bluetooth modul. Da bi se izbegle komplikacije, najbolje je da se naruči kompletan proizvod na kome su svi kontakti zalemljeni za prateću pločicu. Prilikom naručivanja treba obratiti pažnju da postoje dve različite vrste, od kojih jedna ima nožicu izvoda označenu sa KEY, dok je druga označena kao EN.



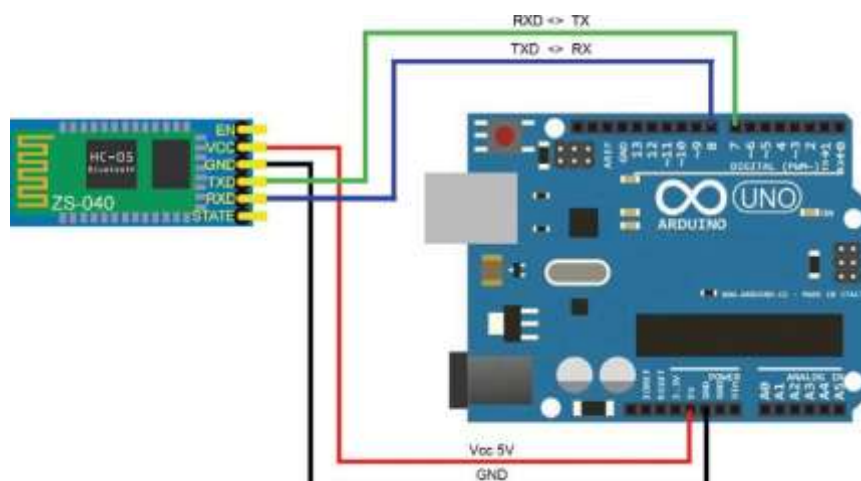
Pin sa oznakom STATE ima vrednost logičke jedinice kada je modul uključen, i logičku nulu kada je isključen. Njegova funkcija je da nas obavesti da li je modul aktivan ili ne. Pin EN (Enable) uključuje modul dovođenjem napona potrebnog za aktivaciju stanja logičke jedinice i isključuje kada dovodimo logičku nulu. Ova dva pina se, u najvećem broju slučajeva, ne koriste u praksi.

Tabela prikazuje nekoliko vrsta modula koji se najčešće nalaze na tržištu. Problem je u tome što svi oni imaju po više različitih verzija koje mogu da se razlikuju po oznakama i upotrebljenim komponentama. Na primer, pločica sa oznakom JY-MCU ima verzije koje imaju taster za ulazak u AT modus i one koje ga nemaju. Isto tako, postoje verzije sa četiri i šest nožica (sa oznakom „Pro”). Kod modula FC-114 mali taster nema nikakvu funkciju, pošto se uređaj posle uključivanja budi u AT režimu. Sve revizije ZS-040 imaju taster koji prebacuje uređaj u stanje prijema AT komandi.

Nekadašnji korisnici dial-up interneta se verovatno sećaju da su u okviru podešavanja modema morali da navedu kraću sekvencu ovih jednostavnih instrukcija, bez kojih uspostavljanje veze nije bilo moguće. Reč je o standardu za telefonske modeme sa početka osamdesetih, koji je potekao od firme Hayes, pa ćete često čuti i frazu „Hayes set komandi”. Njihov najvažniji deo odnosi se na komande koje započinju slovima AT (Attention) i koje su i danas u upotrebi u raznim telekomunikacionim uređajima. Neke od komandi koje prepoznaje modul HC-05 su:

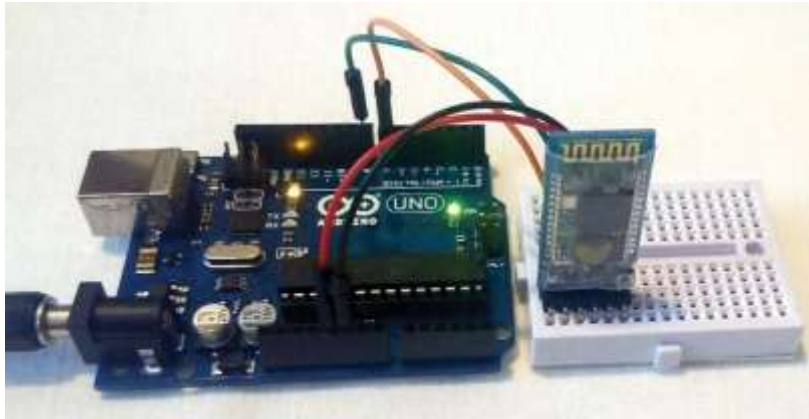
AT	provera ispravnosti
AT+RESET	reset uređaja
AT+ORGL	vrati na fabrička podešavanja
AT+ROLE	režim: master(0) slejv(1)
AT+VERSION	verzija firmvera u ROM-u
AT+NAME	naziv uređaja
AT+PIN	šifra za povezivanje
AT+BAUD	brzina prenosa
AT+RNAME	naziv udaljenog uređaja
AT+CMODE	definisanje modusa konekcije
AT+LOWPOWER	režim ekonomije el. energije
AT+UART	stop bitovi i paritet UART veze
AT+RESETPDL	resetuj spisak uparenih uređaja
AT+REMOVEPDL	udalji određeni uređaj
AT+CLASS	32-bitni indikator tipa uređaja
AT+PSWD	čitajte šifre

Treba napomenuti da se uz većinu ovih komandi (tačnije, iza njih) mogu nalaziti i znakovi „?” i „=”, od kojih prvi traži od modula da nam pokaže trenutno stanje parametara, dok se drugim zadaje njihova nova vrednost.



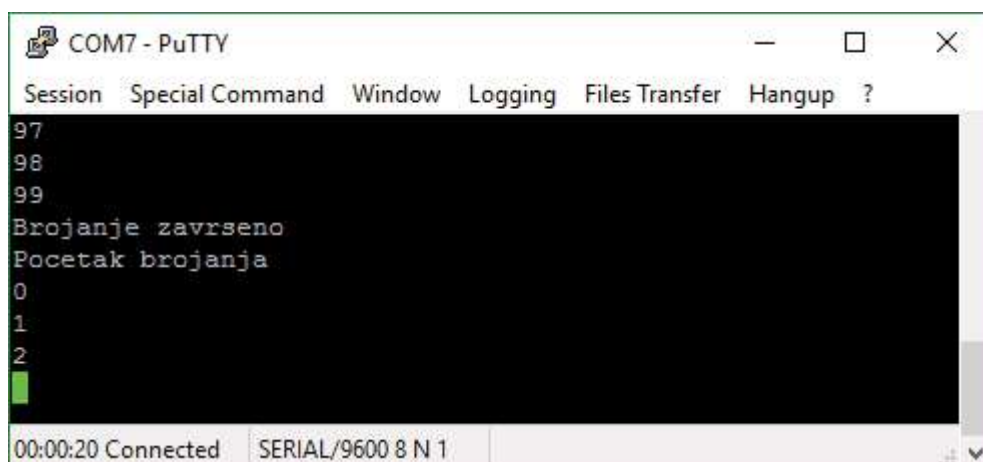
Postoje dva načina da se uđe u AT mod, pritiskom na mali taster u trenutku uključivanja, ili dovođenjem 3,3 volta na 34. pin modula u trenutku uključivanja.

Prvi način je jednostavniji, ali on ne zadržava stanje logičke jedinice na pinu 34 i omogućava pristup samo ograničenom skupu AT komandi. Drugim načinom dobijamo pristup punom setu komandi, sve dok održavamo logičku jedinicu na tom pinu, dok prelazak u stanje logičke nule označava i prelazak u ograničeni skup AT komandi.



Izvorne fabričke vrednosti kod ovih modula podrazumevaju brzinu komunikacije od 38400 bit/s, dok lozinka ima vrednost: „1234”. Ukoliko za komunikaciju sa modulom koristimo standardni serijski monitor iz Arduino razvojnog okruženja, biće potrebno i da postavimo opciju za slanje teksta (polje pored podešavanja brzine) na „Both NL & CR”, pošto bez toga AT komande neće biti ispravno prenesene. Osim njega, moguće je koristiti i bilo koji drugi program koji emulira funkcije starih dobrih terminala (kao što smo mi koristili PuTTY u primeru niže), samo je potrebno voditi računa o ovoj sekvenci kontrolnih karaktera na kraju teksta.

Moduli ne podržavaju mogućnost prenosa AT komandi putem Bluetooth veze, pa je potrebno pribеći nekoj od alternativnih metoda. Prva metoda se ogleda u korišćenju USB to TTL konvertera, koji priključujemo na Bluetooth adapter. Drugi način podrazumeva upotrebu samog Arduina koji priključujemo na naš računar. Potrebno je da žicama povežemo TXD nožicu modula sa softverskim RX pinom, čiji smo broj odredili prvim argumentom funkcije BTveza i analogno tome, RXD nožicu sa softverskim TX pinom.

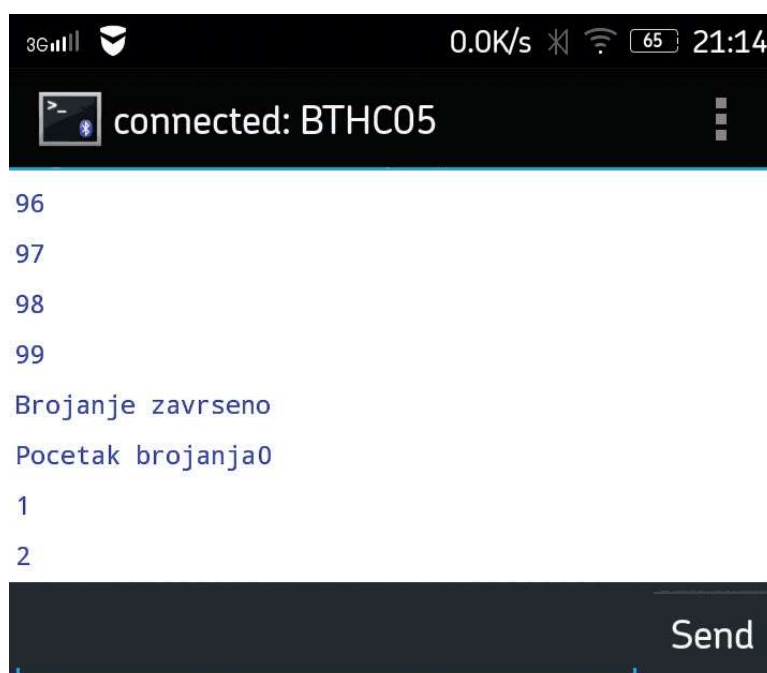


Sastavni deo „rituala” predstavlja i uvođenje uređaja u režim prijema AT komandi, držeći taster modula za vreme uključivanja. To postizemo tako što prvo izvučemo džemper žicu koja dovodi napon. Kada uključujemo modul, u

normalnom režimu rada svetleća dioda trepće ubrzano, dok u slučaju prelaska u AT režim, treptanje biva mnogo sporije. U režimu prenosa podataka, LED trepće po šemi koja oponaša ritam rada srca (heartbeat).

Sledeći skeč ostvaruje prenos AT komandi od serijskog monitora do Bluetooth modula.

```
#include <SoftwareSerial.h>
SoftwareSerial BTveza(7, 8); // softverski RX i TX pin
char znak = "";
void setup(){
  Serial.begin(9600);
  Serial.println("Unesite podrzanu AT komandu");
  BTveza.begin(38400); // default brzina za HC-05
}
void loop(){
  if (BTveza.available()) // pratimo podatke na BT{
    znak = BTveza.read(); //citaj tekst sa BT
    Serial.write(znak); //ispisi tekst na ser. monitoru
  }
  if (Serial.available()) // pratimo podatke na ser. monitoru
  {
    znak = Serial.read(); // citaj sa ser. monitora
    BTveza.write(znak); // posalji na BT
  }
}
```



Varijabla znak sadrži karakter koji je prenesen serijskom vezom. U okviru funkcije setup definišemo brzinu serijskog monitora na 9600 bit/s, kao i brzinu prenosa do Bluetooth uređaja koja je fabrički podešena na 38400 bit/s. Tu je još i linija kôda koji nam sugerše da počnemo sa unošenjem AT komandi. Unutar funkcije loop postoje dve IF naredbe od kojih je jedna zadužena za praćenje stanja na Bluetooth modulu, a druga prati vezu sa serijskim monitorom. Ukoliko se pojavi podatak za slanje, on biva prenesen do drugog uređaja. Nakon zadavanja AT komandi, Bluetooth modul vraća podatke o izvršenom zadatku. Sve vrlo jednostavno i razumljivo.

Vrlo sličan, pa i još jednostavniji mehanizam koristimo kada radimo u režimu prenosa podataka.

```
#include <SoftwareSerial.h>
SoftwareSerial BTveza(7, 8); // softverski RX i TX pin
void setup()
{
  BTveza.begin(9600);
}
void loop()
{
  BTveza.println("Pocetak brojanja: ");
  for (int broj = 0; broj < 100; broj++) {
    BTveza.println(broj);
    delay(500);
  }
  BTveza.println("Brojanje završeno!");
  delay(5000);
}
```

Scenarij je skoro isti kao u gornjem primeru, samo što sada ne koristimo komunikaciju sa serijskim monitorom, već unilateralno prenosimo podatke sa Arduina na bilo koji uređaj koji poseduje Bluetooth interfejs. Za demonstraciju smo odabrali prostu petlju koja na pola sekunde ispisuje brojeve od 0 do 99. Za ispis tekstualnih podataka u okviru prozora terminala zadužena je naredba println. Kao što možemo da šaljemo podatke sa Arduina na druge uređaje, tako isto možemo da primamo podatke iz spoljašnjeg sveta.

Arduino: Wi-Fi modul

Galaksija ESP8266

Igor S. RUŽIĆ, Svet kompjutera, 2016

Čip *ESP8266EX* je zanimljiv i donekle tajanstven proizvod ne baš poznate kineske kompanije *Espressif Systems*. Izrađuje se na osnovu tehnologije američke firme *Tensillica*, koja je pre nekoliko godina postala deo kompanije *Cadence Design Systems*. Reč je o čipu baziranom na 32-bitnoj RISC arhitekturi pod nazivom *Xtensa LX106 Diamond* i radi na (za mikrokontrolere) vrlo pristojnih 80 megaherca. Može da se overklopuje do 160 megaherca, ali se stabilnost ne garantuje. Ima 96 kilobajta radne memorije i 64 kilobajta memorije za instrukcije. Šta je sada ovo?! Korišćena arhitektura mikrokontrolera je zanimljiva po tome što proizvođačima opreme dozvoljava kreiranje sopstvenih instrukcija u okviru mikrokoda. Uz mikrokontroler ide eksterni čip fleš memorije kapaciteta od 512 kilobajta (retko se sreće), preko danas standardnog jednog megabajta, pa sve do četiri megabajta (teoretski maksimum 16 megabajta).



Prvi modeli pojavili su se na tržištu 2014. godine uz reklamu da se radi o zaokruženom rešenju za Wi-Fi komunikaciju. To znači da moduli mogu da se koriste samostalno, ali i kao sredstvo za povezivanje mikrokontrolera bežičnom Wi-Fi vezom. S obzirom da je cena ovih čipova u odnosu na ostale „Wi-Fi capable” konkurente od samog početka bila veoma niska, ne čudi činjenica da su vrlo toplo prihvaćeni od strane mejkera. Osim što je dobro rasprostranjen i popularan, *ESP8266* je učinio mnogo za celokupno IoT tržište primoravajući ostale proizvođače da znatno snize cene.

Vrste modula

Ono što zbunjuje početnike u radu sa ovom platformom jeste činjenica da postoji ceo niz različitih modula zasnovanih na istom čipu. *AI-Thinker* je jedan od proizvođača koji su među prvima počeli sa kreiranjem modula na bazi čipa *ESP8266*, pa su sticajem okolnosti postali omiljeni kod hobista. Pa da pogledamo nekoliko njih koji bi nam potencijalno mogli biti zanimljivi.

ESP-01



Prvi *AI-Thinker* modul sa čipom *ESP8266* koji se pojavio na tržištu, istovremeno je najpopularniji i najčešće korišćen, zahvaljujući ceni koja je niža od svih predstavnika u toj kategoriji. Omogućuje pristup do dva GPIO porta, što znači da može da se koristi za prikupljanje senzorskih podataka nezavisno od drugog mikrokontrolera. Dva reda sa po četiri pina nisu podesni za priključivanje na prototipsku pločicu, ali postoje jeftini dodatni moduli koji to omogućavaju. Dobar su izbor zbog toga što se često koriste u primerima koji se mogu naći na internetu, pa nije teško doći do gotovih rešenja. Međutim, nisu bez nedostataka. Pojedini izvodi glavnog čipa, zbog loših inženjerskih rešenja, nisu nigde priključeni, pa se nalaze u takozvanom „plutajućem” stanju, koje može prouzrokovati razne nepravilnosti u radu.

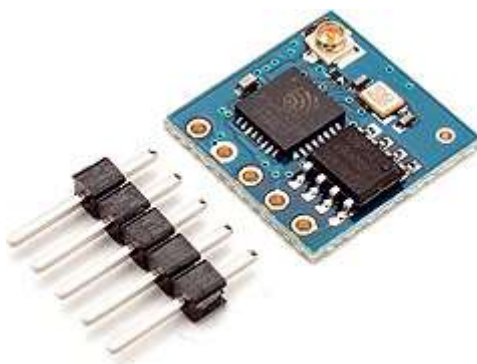
R.b.	Pin	Funkcija
1.	GND	uzemljenje
2.	TX	UART slanje podataka
3.	GPIO2	GPIO port 2
4.	CH_PD	aktiviranje modula
5.	GPIO0	GPIO port 0
6.	RST	eksterni reset signal
7.	RX	UART prijem podataka
8.	VCC	konektor napajanja

Osim toga, ovi moduli nemaju mogućnost dovođenja elektronike u stanje takozvanog „dubokog sna”, koje je vrlo važno za ekonomiju električne energije. Najčešća veličina fleš memorije za module crne boje iznosi jedan megabajt.

Espressif ESP8266EX	
Mikrokontroler	Tensilica Xtensa LX106 Diamond, 80 MHz (160 MHz overklok)
Memorija	96 KB programski RAM, 64 KB instrukcijski RAM, eksterna fleš memorija (512 KB - 4 MB)
Wi-Fi	802,11b/g/n WPA/WPA2 (WEP/TKIP/AES)
Interfejsi	GPIO (x16), SPI, I2C, I2S, UART(x2), PWM (x4), SDIO, IR Remote, ADC (x1) 10-bit
Radni napon	3-3,6 V
Potrošnja struje	80 mA, maks, 320 mA, <10 uA sleep

ESP-05

U pitanju je mali, pojednostavljeni modul koji ima svega pet nožica za povezivanje, kao i konektor po IPX standardu za priključivanje spoljašnje antene



. Njegov najveći plus predstavlja mogućnost jednostavnog korišćenja na prototipskoj pločici, dok mu se kao minus računa to što ne obezbeđuje nikakav pristup ka GPIO portovima mikrokontrolera. Sa *Arduinom* se povezuje putem UART interfejsa i služi za prenos podataka putem Wi-Fi mreže.

ESP-12E (NodeMCU 1.0)

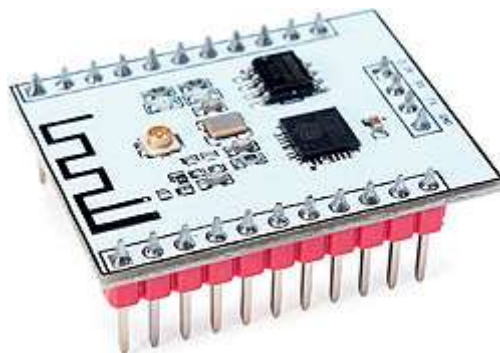
Ova vrsta modula, zbog rastojanja među konektorima od dva milimetra i nemogućnosti lemljenja pinova za priključivanje na prototipsku pločicu, nije naročito podesna za rad sa *Arduinom*.



Međutim, postoje i moduli koji na sebi imaju zalemljen *ESP-12* i istovremeno sadrže konektore sa iglicama razmaka 2,5 milimetara. Njihov plus u odnosu na ostale predstavljene module odnosi se na prisustvo 4096 kilobajta fleš memorije. Uz njih dolazi firmver pod nazivom *NodeMCU*, koji podržava skript programski jezik *Lua*, pa kao proizvod dobijamo funkcionalno i zaokruženo IoT razvojno okruženje. *Lua* je moćan i dosta jednostavan programski jezik, donekle sličan *JavaScriptu*, ali fleksibilniji od njega i znatno uprošćava programiranje, kako funkcija vezanih za Wi-Fi komunikaciju, tako i za rad sa priključenim sensorima. Ovi moduli obično na sebi imaju i čip *CH340G* koji omogućava USB<>UART povezivanje, što pojednostavljuje proces zapisivanja podataka u integrisanu fleš memoriju. Uz *ESP-07*, reč je verovatno o inženjerski najbolje rešenim modulima ove vrste, a sa cenom od približno 3,5 evra, predstavljaju odličan izbor

ESP-201

Ovaj modul osim antene na pločici, pruža mogućnost priključenja eksterne antene. U standardnom pakovanju obično dolazi sa jednostavnom antenom koja ima IPX interfejs.



Inicijalno se koristi spoljašnja antena, a u slučaju da želimo da koristimo integrisanu, potrebno je prelemiti SMD otpornik veličine 0 oma.

Modul na sebi ima 26 pinova od kojih su četiri namenjena povezivanju sa UART interfejsom. Nije baš najjasnije šta je konstruktore motivisalo da ta četiri pina okrenu prema dole i onemoguće postavljanje na prototipsku pločicu, ali je za veliki broj korisnika prva stvar koju urade njihovo odlemljivanje i postavljanje na vrh. Nazivi pinova su štampani sa donje strane pločice, ali je to od male koristi, pošto za vreme rada oni nisu vidljivi. Modul sadrži šest GPIO portova (GPIO6-GPIO11) koje nije moguće praktično koristiti. Uz hardversku modifikaciju na čipu sa fleš memorijom moguće je

osloboditi GPIO portove 9 i 10. Prilikom flešovanja pin IO15 mora biti u stanju logičke nule. Iako *ESP-201* na prvi pogled izgleda kao sjajan modul, on ima dosta slabosti konstruktorske prirode. Cena od 2,4 evra ga čini privlačnim za kupovinu, ali uprkos konektorima koji se mogu priključiti na prototipsku ploču, znatan broj konstrukcijskih grešaka uveliko smanjuje njegov potencijal

Adafruit HUZZAH

U pitanju je modul renomiranog proizvođača u svetu *Arduino* tehnike i baziran je na modulu *ESP-12*. Uz nešto višu cenu (ispod 10 dolara) nudi dosta jednostavnije upravljanje Wi-Fi modulom.



Od ostalih modela se razlikuje postojanjem dva mala tastera od kojih je jedan namenjen ulasku u *bootloading* režim, dok drugi funkcioniše kao *reset* taster. Korisniku je na raspolaganju devet GPIO portova (0, 2, 4, 5, 12, 13, 14, 15, 16), između ostalog, i sa funkcijama SPI i I2C interfejsa, dva UART interfejsa i jednim analognim ulazom (1,8 volti maksimalno). Na pločici se nalazi i regulator struje koji omogućuje napajanje do 500 miliampera. I ovde se koristi programski jezik *Lua* kao osnova za programiranje, ali je moguće koristiti i programiranje putem *ArduinoIDE* i raznih drugih programskih jezika.

Olimex ESP8266-DEV

Zanimljivo je da se sedište kompanije Olimex nalazi u nama susednoj Bugarskoj i da već duži niz godina uspešno opstaje u teškoj tržišnoj borbi sa dalekoistočnim proizvođačima elektronike (po svedočenju vlasnika firme Cvetana Usunova, dobrim delom zahvaljujući stimulaciji iz fondova EU).



Ovo je jedan od prvih Olimexovih modula koji se pojavio još 2014. godine i u međuvremenu je stekao dosta veliku popularnost.

Na pločici se nalaze tzv. UEXT (*Universal EXTension*) konektori za priključivanje na UART interfejs. Posедуje dva megabajta fleš memorije, što omogućava upotrebu u kompleksnijim projektima. Cena uređaja je oko pet evra, što se može smatrati vrlo prihvatljivim. Isti proizvođač po ceni od devet evra nudi razvojnu pločicu *ESP8266-EVB* koja već sadrži *ESP8266-DEV* modul i umnogome pojednostavljuje komunikaciju s njim.

Tu je i cela linija drugih modula (*ESP-02, ESP-03, ESP-04, ESP-06, ESP-08, ESP-09, ESP-10...*) koji su namenjeni ugrađivanju na druge uređaje i nisu pogodni za direktni rad sa *Arduinom*, pa im zbog toga nećemo posvetiti više prostora.

Naravno, i oni se, slično modulu *ESP-12E*, mogu zalemiti na dodatnu pločicu, ali to je već van opsega našeg teksta. Zbog ograničenosti prostora mesto u opisu nisu našli ni neki drugi moduli, uključujući jedan od najboljih (ali i nekoliko puta skuplji) *Sparkfun Thing*, koji ima konektor za eksternu Li-Po bateriju, koja mu uz ispravan dizajn projekta omogućava višegodišnje autonomno funkcionisanje

Povezivanje

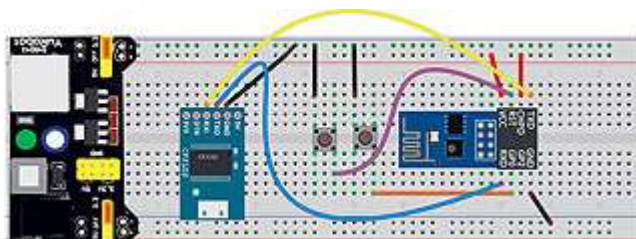
Iako se na internetu može videti velika količina materijala vezanog za rad sa platformom *ESP8266*, početnicima je prilično teško da pronađu pravu informaciju koja bi ih uputila u pravom smeru. Upravo zato ćemo se truditi da bolje objasnimo neke elementarne stvari bez kojih je dalji rad dosta komplikovan.

ESP-01	Povezivanje sa
GND	GND
Vcc	3,3V
Rx	Tx
Tx	Rx
CH_PD	3,3V
RST	GND (po potrebi)

Da bi modul *ESP-01* pro funkcionisao, potrebno je da na odgovarajuće nožice dovedemo napon od 3,3 volta i uzemljenje, nakon čega se aktivira crvena LED lampica. Odmah napominjemo da su ovi moduli projektovani za napon od 3,3 volta i da bi dovođenje napona od 5 volti bilo destruktivno. Za aktiviranje Wi-Fi režima rada, potrebno je da na nožicu *CH_PD* dovedemo napon od 3,3 volta, čime je postavljamo u stanje logičke jedinice. To će na modulu nakratko aktivirati plavu LED lampicu, a mi ćemo posle toga na svom računaru u spisku Wi-Fi predajnika videti novu stavku. Možemo se odmah priključiti na bežičnu mrežu, ali od toga neće biti neke preterane koristi, jer još nemamo vezu sa svojim Wi-Fi modulom.

U prethodnim nastavcima smo pominjali AT komande koje su najpoznatije po tome što su služile za podešavanje starih dobrih telefonskih modema. Najveći broj predstavljenih modula dolazi sa takozvanim AT firmverom, a u pojedinim

situacijama dolazi i bez bilo kakvog firmvera. Inače, AT komande se koriste za ostvarenje veze između mikrokontrolera i ESP modula putem serijskog interfejsa (ESP moduli se još nazivaju *Serial to Wi-Fi* uređajima).



Na taj način je moguće slati podatke i primiti ih od drugih uređaja. Da bismo mogli da dostavimo AT komande sa računara do samog modula, potrebno je da koristimo neki USB<>Serial adapter. U jednom od narednih brojeva ćemo se pozabaviti razlikama između pojedinih modela, a za sada samo da navedemo da su najčešće varijante bazirane na čipovima *FT232RL*, *CH340G* i *CP2102*. Za komunikaciju sa Wi-Fi modulima je obavezno da adapteri podržavaju rad sa naponom od 3,3 volta. Mi smo, čisto zbog toga što je najviše „breadboard friendly”, koristili modul *CP2102*, čiji se drajver može preuzeti sa lokacije goo.gl/Ons89Q.



Da bi posao učinili jednostavnijim, *ESP-01* smo utakli u podnožje koje omogućuje upotrebu prototipske pločice (cena oko jedan evro), i džamper žicama spojili njegov RXD port sa TXD portom *CP2102* (plava boja), a TXD sa RXI (žuta). Levi taster spaja uzemljenje sa pinom RST i služi za restart modula, dok desni taster spaja uzemljenje sa pinom GP0 i ima važnu funkciju prilikom flešovanja uređaja.

Ovu priliku koristimo da predstavimo veoma praktični modul za napajanje, prilagođen radu sa prototipskim pločicama, a koji smo koristili u okviru sheme povezivanja. Pošto *Arduino* poseduje ograničene kapacitete po pitanju struje (u konkretnom slučaju je potrebno 300 miliampera, a *Arduino* preko svog 3V3 pina isporučuje samo 50 miliampera), ponekad je potrebno koristiti dodatne izvore napajanja. Modul pod nazivom *MB102* je sjajan izbor kada su nam potrebni naponi 3,3 volta, pet volti ili oba istovremeno.



Sa donje strane se nalaze iglice preko kojih se modul fiksira za prototipsku ploču i to u područja predviđena za napon i uzemljenje. Pošto prototipske ploče imaju po dve strujne šine, ovaj uređaj je u stanju da na obe isporuči napon od nula, 3,3 ili pet volti, po želji. Veličina napona se određuje džamperima koji se nalaze na obe strane modula.

Uređaj se napaja ulaznim naponom u rasponu od 6,5-12 volti, dok je napajanje preko USB porta standardnih pet volti, ali isto tako, sa USB porta možemo da odvodimo napon od pet volti. Tu su i dva bloka od po četiri iglice preko kojih možemo preuzeti napon od 3,3 i pet volti.

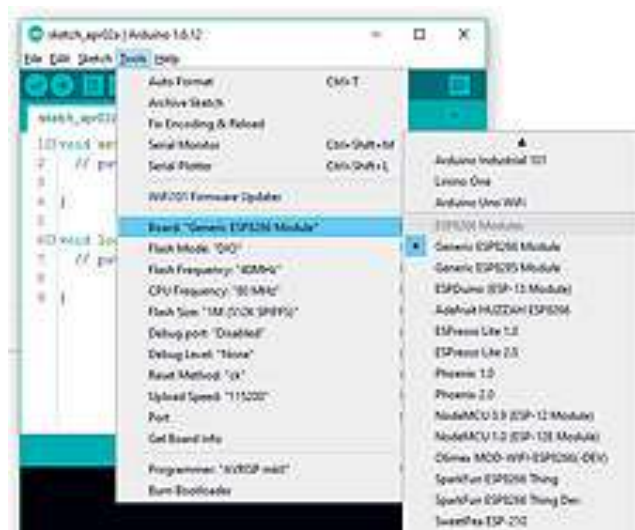
Za regulaciju napona koriste se integralna kola AMS1117 (jedno za napon od 3,3 volta, a drugo za pet volti). Ova kola nude solidan stepen zaštite od kratkih spojeva. Maksimalna jačina izlazne struje je 700 miliampera, ali je pametnije ako se ograničimo do nekih 500 miliampera.

Putem postojećeg prekidača možemo uključivati i isključivati uređaj bez potrebe da mu isključujemo dovod struje. Zelena LED lampica prikazuje da li je modul uključen. S obzirom na to da se mogu naručiti već po ceni od nekih 70 centi, moduli *MB102* su jednostavno *must have* u radu sa prototipskim pločama.

AT komande

Za pristupanje modulu putem AT parametara iz Arduino IDE, potrebno je da dodamo podršku za *ESP8266* platformu. U opciji menija *Menu/Preferences*, u okvir polja *Additional Boards Manager URLs*: upisujemo sledeću adresu:

http://arduino.esp8266.com/stable/package_esp8266com_index.json



Nakon toga prelazimo u meni *Tools/Board/Boards Manager*, i u polje za pretraživanje kucamo slova „ESP”, nakon čega dobijamo filtrirane stavke koje imaju vezu sa ovom vrstom modula. Odabiramo stavku sa oznakom *ESP8266 by ESP8266 Community* i kliknemo na *Install*. Posle preuzimanja potrebnih fajlova sa interneta, potrebno je ponovo da odemo u meni *Tools* i tamo u podopcijama *Board*: izaberemo uređaj pod nazivom *Generic ESP8266 module*. Primetićete da se u okviru menija pojavljuje veliki broj opcija vezanih za ESP platformu.

Nas, za početak, zanima *Serial Monitor*, kako bismo ostvarili komunikaciju sa modulom. Za to je potrebno poznavati broj COM porta, kao i brzinu konekcije. Najčešće vrednosti brzine su 115.200 i 9.600 bauda, ali znaju da budu i drugačije. U našem konkretnom slučaju, brzina je iznosila 74.880 bauda, što smo saznali tek nakon više pokušaja unosa teksta u terminal, menjajući brzine prenosa. Ukoliko su unesena podešavanja odgovarajuća, tekst će biti ispisan ispravno, a u suprotnom ćemo videti „hijeroglif”. Ukoliko nemamo nikakve informacije na ekranu, potrebno je da nožicu RST nakratko postavimo u stanje logičke nule putem našeg tastera ili jednostavno spajanjem tog pina sa linijom GND uz pomoć džemper žice.

Verziju trenutno postavljenog firmvera saznajemo kucajući AT komandu GMR:

AT+GMR

Spisak svih dostupnih AP dobijamo pomoću komande:

AT+CWLAP

Na željeni AP se priključujemo sa komandom:

AT+CWJAP="SSID_predajnika", „šifra”

Konekciju TCP tipa započinjemo na sledeći način:

AT+CIPSTART="TCP","192.168.1.77","123"

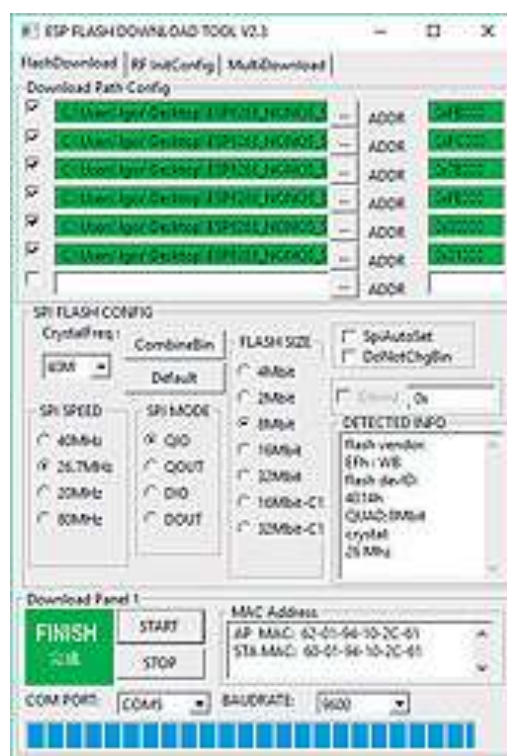
To jeste, uspostavljamo komunikaciju sa uređajem na ukazanoj IP adresi i preko ukazanog TCP porta.

Režim funkcionisanja modula saznajemo komandom:

AT+CWMODE?

Postoje tri režima funkcionisanja *ESP8266*. Prvi je režim klijenta, kada modul priključujemo na neku postojeću mrežu. Drugi modus se odnosi na kreiranje nove mreže gde modul obavlja funkciju tačke pristupa (AP). Treći modus je najfleksibilniji i predstavlja kombinaciju prethodna dva režima.

Prostor ne dozvoljava da se više posvetimo komunikaciji putem AT komandi, ali je princip isti onome što smo demonstrirali kada smo opisivali komunikaciju Bluetooth modulom. Celokupan spisak AT instrukcija za *ESP-xx* module se može pronaći na adresi goo.gl/ymOEba.



Nešto kao flash

U radu sa *ESP-xx* modulima, često je potrebno menjati sadržaj fleš memorije. Ukoliko napišemo neki program i prepisemo ga na fleš memoriju, za vraćanje na standardni AT firmver potrebno je da ponovo flešujemo inicijalni sadržaj.

Ranije je taj postupak bio vrlo jednostavan, ali su se tokom vremena pojavile nove verzije programa za flešovanje koje izdaje *Espressif* i koje su prilično komplikovane za upotrebu. U trenutku pisanja, najnovija verzija nosi oznaku 3.4.4 i može da se koristi za flešovanje modula baziranih na čipovima 8266, 8285 i *ESP 32*.

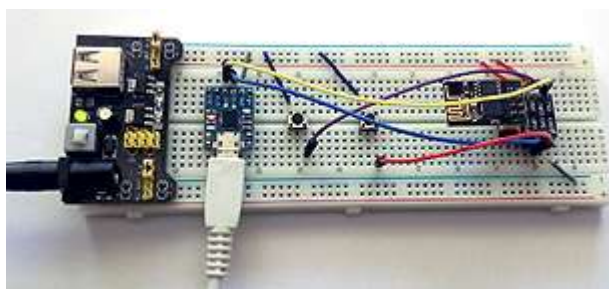
Verzije firmvera koji podržava rad sa AT komandama treba tražiti u fajlovima koji započinju tekстом „ESP8266_NONOS_SDK”, što znači da se radi o firmveru bez podrške operativnog sistema. Za razliku od njih, postoje verzije koje podržavaju minijturni operativni sistem pod nazivom *FreeRTOS*, ali o tome neki drugi put.

Prvo što treba da uradimo je da saznamo osnovne podatke o našem modulu. To se obavlja tako što pokrenemo *ESP Flash Download Tool* bez ikakvih dodatnih parametara (osim, naravno, naziva COM porta i brzine komunikacije). Nakon toga u prozoru *Detected Info* dobijamo informacije o veličini fleš memorije i frekvenciji kristala, koje ćemo posle upisati u odgovarajuća polja programa.

Naziv fajla	Lokacija	Adresa
blank.bin	\\ESP8266_NONOS_SDK\\bin	0xFB000
esp_init_data_default.bin	\\ESP8266_NONOS_SDK\\bin	0xFC000
blank.bin	\\ESP8266_NONOS_SDK\\bin	0x7E000
blank.bin	\\ESP8266_NONOS_SDK\\bin	0x7E000
boot_v1.6.bin	\\ESP8266_NONOS_SDK\\bin	0x00000
user1.1024.new.2.bin	\\ESP8266_NONOS_SDK\\bin\\at\\512+512	0x01000

Unutar fajla na lokaciji /bin/at_sdio/README.md nalazi se spisak memorijskih lokacija na koje je potrebno postaviti određene fajlove kako bi proces flešovanja bio pravilno odrađen. Različite veličine čipa sa fleš memorijom zahtevaju različite parametre. U našem konkretnom primeru modula *ESP-01* polja popunjavamo na sledeći način:

Kao poslednji korak, selektujemo ček boks kontrole ispred svakog od izabranih fajlova i, ako posle toga sva polja budu obojena u zeleno, znači da smo posao obavili kako treba i da možemo preći na proces zapisivanja fleš memorije. Pre nego što kliknemo na ekranski taster sa natpisom *Start*, potrebno je da prevedemo modul u režim zapisivanja. Za to je potrebno da pin GPIO0 dovedemo u stanje logičke nule i, dok se nalazi u tom stanju, istu stvar nakratko uradimo sa pinom RST. Na pratećoj fotografiji vidi se da je to urađeno uz pomoć malih tastera, dok je istu stvar moguće postići dovođenjem uzemljenja putem džamper žica, što je suštinski jednostavnije i praktičnije rešenje. Postoji i mogućnost da se flešovanje obavi putem AT komandi direktno preko oblaka, ali se na internetu može pročitati da je takav pristup često problematičan.



Na raspolaganju nam je veći broj programskih jezika preko kojih je moguće upravljati *ESP* modulima. Osim već pominjanog *NodeMCU*, tu su još *Espruino* i *Smart.js* koji omogućavaju programiranje putem jezika *JavaScript*, *MicroPython* je varijanta Python programskog jezika, dok *ESP8266 BASIC* i *ZBasic* jasno asociraju na svoj programski jezik. Za sada je najvažnije da se savladaju osnovni principi rada sa ovom vrstom modula, a kako korisnik bude napredovao, prelazak na druge programske jezike mu neće predstavljati veći problem. *ESP* moduli su vrlo zanimljiv instrument velikog potencijala koji jednostavno mora pronaći mesto u kolekciji svakog ozbiljnijeg mejkera.

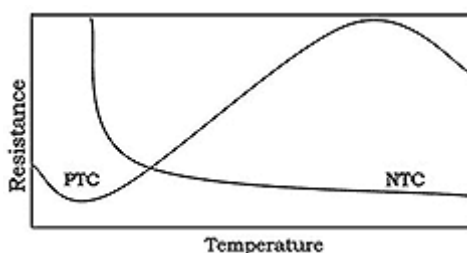
Arduino: Termistori

Od zamrzivača do furune

Dejan PETROVIĆ, Svet kompjutera, 2017

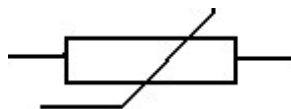


Termistori su otpornici kod kojih se otpornost menja u odnosu na spoljnu temperaturu. Razlikujemo termistore sa negativnim (NTC) i pozitivnim (PTC) temperaturnim koeficijentom, a to zavisi od toga da li će se vrednosti sa povećanjem spoljne temperature povećavati ili smanjivati. NTC se uglavnom koriste za merenje ekstremno niskih i visokih temperatura (od -55°C do 200°C). PTC se upotrebljavaju kada se očekuju nagle promene temperature, unapred definisane i uglavnom u rasponu od 60°C do 120°C .



Dolaze u raznim oblicima: radijalni (oblik diska), aksijalni (oblik valjka), ugrađeni u šraf ili u obliku sonde. Izrađuju se od raznih materijala, a oni koji nas interesuju kao spoljni omotač koriste staklo, silikon, ali to može biti i keramika i metal. S obzirom na to da su oni u suštini otpornici, prilikom prolaska struje

kroz njih, dolazi do određenog zagrevanja samih rezistora koji mogu, u zavisnosti od upotrebe, da utiču na preciznost.



Termistori se koriste svuda oko nas: mere temperature u kabini i motoru našeg četvorotočkaša, u klima uređajima, frižiderima i tako dalje. Termistori su zaduženi za očitavanje temperatura ključnih komponenti računara i možemo ih videti, recimo, ispod procesora ili na matičnoj ploči. Takođe se nalaze na obodu namotaja trafoa i prilikom većih opterećenja prvi stradaju. Tada je dovoljno malo zaseći papirni omotač na trafou do termistora i zameniti ga (autoru se takav peh desio na trafou Altekovih zvučnika). Tipova termistora ima mnogo, kao i njihovih namena, i kada bismo se dotakli svega, otišli bismo predaleko.

Mi smo za potrebe ovog teksta koristili NTC MF52 103 termistor. U pitanju je jedan od najčešćih koji se koriste u Arduino svetu i koji se mogu naći u online prodavnicama sa smešno niskom cenom (0,05 dolara). Da bi se dobila ispravna očitavanja, potrebno je znati određene parametre o datom termistoru. Prvi parametar je otpornost. Prema datasheetu (goo.gl/DR9TCL) za naš termistor vidimo da je na relativnoj sobnoj temperaturi (25°C) ta otpornost deset kilooma, sa dozvoljenom razlikom od oko deset procenata. Ovo ujedno znači i da nam treba jedan otpornik od deset kilooma, ali takođe znači da bi najispravnije bilo da se otpornost termistora izmeri unimerom na sobnoj temperaturi, s tim što su sada sobne temperature preko 30°C. Beta (B) konstanta predstavlja razliku između otpornosti i temperature u određenom temperaturnom opsegu (25/50°C > 10%) i u slučaju našeg termistora iznosi 3950. Preciznost našeg termistora je $\pm 0.25^\circ\text{C}$.

Prvo treba da izvučemo ispravnu vrednost otpornosti termistora u odnosu na napon i otpornik koji ćemo upotrebiti. Arduino ne može da meri otpornost, već samo napon. Omov zakon glasi $I=U/R$, gde je jačina struje (I) u amperima jednaka odnosu napona struje (U) u voltima i električnog otpora u omima (O).

Takođe znamo da analogno-digitalni konvertor Arduina ovaj napon konvertuje u vrednosti od 0 do 1023 (goo.gl/ynJTxo). Da bismo dobili konvertovano očitavanje iz analognog u digitalno, koristimo ovu jednačinu:

$$\text{AnalogToDigital (ADC)} = (U \text{ na analog pinu} / 5V) * 1023$$

Ako je očitavanje na analognom pinu 0 volti, onda je i dobijena vrednost 0, ali ako je na pinu pet volti, onda je vrednost 1023, tako da ovaj broj koristimo za izračunavanje temperature od dobijenih vrednosti kao V_{in} u sledećoj jednačini naponskog razdelnika:

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$

Gde je $V_{in}=1023$ (5V), V_{out} je dobijena analogno očitana vrednost termistora. $R_1=10000$ (10k?) i, na kraju, R_2 je vrednost koja nama treba. Da bismo dobili R_2 , jednačinu postavljamo ovako:

$$R_2 = R_1 \times \left(\frac{V_{in}}{V_{out}} - 1 \right)$$

Dobili smo digitalno očitanu vrednost termistora, a za dobijanje vrednosti u kelvinima (K) koristi se Štajnhart-Hartova jednačina (goo.gl/Bg1rQh) koja glasi:

$$\frac{1}{T} = A + B \ln(R) + C(\ln(R))^3$$

Ovde su A (1,009249522e-03), B (2,378405444e-04) i C (2,019202697e-07) konstante. Znamo da apsolutna nula u kelvinima iznosi -273,15°C, pa je jednačina za konverziju u celzijuse: $[^{\circ}\text{C}] = [\text{K}] - 273,15$.

U slučaju Arduina, skeč ide ovako:

```

int tempPin = 0;
int Vout;
float R1 = 10000; //otpornost na sobnoj temperaturi 10K ili 10000 oma
float R2, Tk, Tc; //otpornost, temperatura u K, temperatura u C
float Ac = 1.009249522e-03, Bc = 2.378405444e-04, Cc = 2.019202697e-07;
//konstante

void setup() {
  Serial.begin(9600);
}

void loop() {
  Vout = analogRead(tempPin);
  R2 = R1 * (1023.0 / (float)Vout - 1.0); // konvertovanje iz analogne u digitalnu
  Tk = (1.0 / (Ac + Bc*log(R2) + Cc*log(R2)*log(R2)*log(R2))); // temp. u kelv. (K)
  Tc = Tk - 273.15; //temperatura u stepenima celzijusa
  Serial.print("Temperatura: ");
  Serial.print(Tc);
  Serial.println(" °C");
  delay(2000);
}

```

Na ovaj način očitavanje temperature pratimo preko Serial Monitora, ali isto tako se vrednosti mogu prikazati na bilo kom ekranu koji se inače može povezati na Arduino.

Koliko su očitane vrednosti tačne? Nismo imali neophodne uređaje da ispitamo preciznost termistora, ali nam je pri ruci bio DHT22. Povezali smo uporedo sa termistorom i modul o kome smo već pisali i za koji znamo da je dosta precizan. Temperaturne vrednosti su se malo razlikovale, što možemo pripisati pomenutoj dozvoljenoj razlici od deset procenata. Razlika se može rešiti korigovanjem odnosa kelvin/celzijus u samom skeču.

Termistor možemo upotrebiti u bilo kom projektu gde nam je potrebna mala komponenta (0,5 sa 3,3 milimetra) koja će da odreaguje brzo, da traje dugo i da bude dosta precizna. Na kraju, kod stand alone projekata, gde je potrebno da se komponente zaleme, treba voditi računa da se lemljenje mora izvesti vrlo brzo i ne bliže od pet milimetara od samog termistora.



Svetlo



Kako spojiti LED diodu

[Ivan Matačić](#) 03.04.2015. [Elektronika](#)

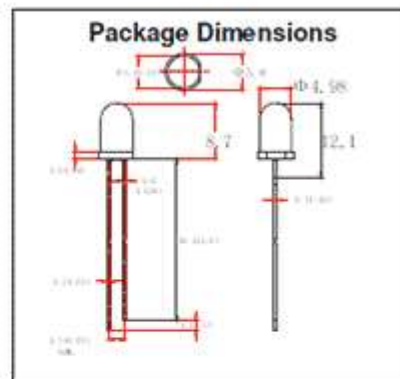
Želimo li spojiti LED diodu na neki izvor istosmjernog napona, to gotovo nikada nećemo moći učiniti bez odgovarajućeg otpornika. LED dioda je komponenta kojoj je potrebno ograničiti struju, a ne učinimo li to, kroz nju će poteći prevelika struja koja će je uništiti. U ovom članku ćemo saznati kako spojiti jednu LED diodu na neki izvor poznatog napona, kao npr. 5V u računalu, 12V u autu, te kako proračunati odgovarajući otpornik.

Osnovne karakteristike LED diode

Dvije najvažnije električne karakteristike LED diode koje moramo znati su Forward Voltage (V_f) i Forward Current (I_f). V_f nam govori koji je radni napon diode, a I_f koja joj je radna struja. Ove podatke ćemo pronaći u tehničkoj dokumentaciji, a izgledaju kao na donjoj slici. Kad uzimate u obzir ove podatke, uvijek gledajte vrijednost pod Typical (ili skraćeno Typ.), a ne pod Absolute Maximum Ratings. Ako Typical vrijednosti nisu navedene, uzmite neku srednju vrijednost između Min. i Max. Npr. u tablici na donjoj slici nije navedena Typ. vrijednost za Forward Voltage, već samo Min. i Max. Stoga za V_f uzmite vrijednost od $\sim 2.2V$.

Absolute Maximum Ratings at Ta=25°C

Parameter	MAX.	Unit
Power Dissipation	100	mW
Peak Forward Current ($\leq 1/10$ Duty Cycle, 0.1ms Pulse Wide)	100	mA
Continuous Forward Current	20	mA
Reverse Voltage	5	V
Operating Temperature Range	-40°C to +80°C	
Storage Temperature Range	-40°C to +80°C	
Lead Soldering Temperature [3mm(From solder joint to epoxy body)]	260°C for 3 Seconds	



Electrical Optical Characteristics at Ta=25°C

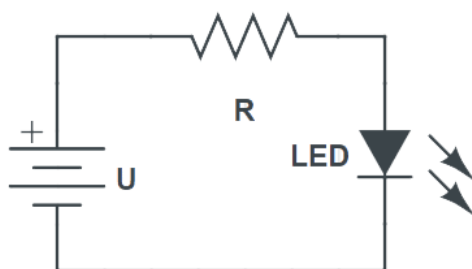
Part Number	Lens color	Source Color	Dominant Wavelength λ_d / nm $I_f = 20\text{mA}$ (Note 8)			Luminous Intensity I_v / mcd $I_f = 20\text{mA}$ (Note 5)			Forward Voltage / V $I_f = 20\text{mA}$			Viewing Angle / Deg (Note 6)
			Min.	Typ.	Max.	Min.	Typ.	Max.	Min.	Typ.	Max.	
WW05A3SRH4-N	Water Clear	Red	620	---	630	8200	10600	---	1.8	---	2.4	15
Reverse Voltage = 5V						Reverse Current $\leq 5\mu\text{A}$						

Preduvjet da bi LED dioda uopće mogla raditi jest da je napon izvora na koji spajate diodu veći od napona V_f diode. Imate li LED diode za koje ne znate ove podatke, a ne možete ih pronaći jer ne znate proizvođača i model, možete uzeti donje vrijednosti kao okvirne. Morate samo znati koje su boje, a to lako možete saznati po boji kućišta. Ako je kućište prozirno, uzmite najmanji V_f napon iz tablice i kad dioda zasvijestli, vidjet ćete o kojoj se radi. Neka vam pri tome I_f bude 20mA.

BOJA	V_f
Crvena	1.6 - 2.0V
Žuta	2.0 - 2.2V
Narančasta	2.1 - 2.2V
Plava	2.4 - 3.7V
Zelena	1.9 - 4.0V
Ljubičasta	2.7 - 4.0V
UV	3.1 - 4.4V
Bijela	3.2 - 3.6V

Otpornik za LED diodu

Otpornik ćemo serijski spojiti LED diodi kao na donjoj slici (element označen slovom R). Ovdje je otpornik spojen na anodu, no možete ga spojiti i na katoru, svejedno je.



Uz parametre V_f i I_f moramo znati koji je napon izvora U_i na koji ćemo spojiti LED diodu. Formulu za proračun otpornika ćemo lako izvesti iz Ohmovog zakona i ona izgleda ovako:

$$R = (U_i - V_f) / I_f$$

Dakle, da biste dobili vrijednost otpornika, od napona izvora U_i oduzmite V_f diode i dobivenu vrijednost podijelite s I_f diode. Kad uvrštavate podatke u ponudu, ne zaboravite podatak I_f iz mA pretvoriti u A.

Pokažimo izračun na sljedećem primjeru. Imate crvenu LED diodu i želite je spojiti na 12V. Budući da ne znate o kakvoj se LED-ici radi, uzet ćemo da je prema gornjoj tablici $V_f = 2V$, a $I_f = 20mA$ (0.02A). Uvrstimo podatke u formulu:

$$R = (U_i - V_f) / I_f = (12V - 2V) / 0.02A = \mathbf{500\Omega}$$

Da bismo LED diodu spojili na 12V, potreban nam je otpornik od 500Ω. Kako to nije standardna vrijednost, uzet ćemo najbližnju, a to je 510Ω ako imamo otpornike iz E24 niza, ili 470Ω ili 560Ω ako imamo otpornike iz E12 niza. Više o vrijednostima otpornika i njihovom označavanju možete pročitati u jednom od [prijašnjih članaka](#).

Na prvoj slici su tehničke karakteristike neke LED diode. Pogledajmo kako će izgledati izračun za otpornik ako bismo je htjeli spojiti na 5V:

$$R = (U_i - V_f) / I_f = (5V - 2.2V) / 0.02A = 140\Omega$$

Treba nam otpornik od 140Ω, a kako takvog nema, koristit ćemo onaj od 150Ω. Već smo [naučili](#) da otpornike osim otpora određuje i snaga koju mogu podnijeti. Sada kada imamo potrebnu vrijednost otpornika, možemo izračunati snagu koja će se razviti na njemu, kako bismo znali kolike snage otpornik treba biti. U prvom primjeru smo koristili otpornik od 510Ω, a I_f diode je bio 20mA. Iz Ohmovog zakona znamo da snagu računamo kao umnožak otpora i kvadrata struje, pa ćemo tako dobiti sljedeću jednadžbu:

$$P = R \cdot I^2 = 510\Omega \cdot 0.02A^2 = 0.204W$$

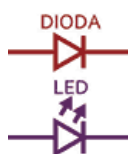
Na otporniku se troši 0.204W snage što znači da trebamo staviti otpornik od najmanje 1/4W (0.25W).

Svetleći svet

Tri boje za ceo spektar

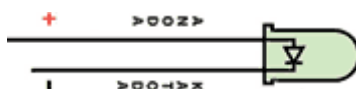
Igor S. RUŽIĆ, Svet kompjutera, 2016

Kao što se već može da se zaključi iz naziva, LE diode su varijanta „običnih” dioda čija je funkcija da dozvole kretanje elektriciteta samo u jednom pravcu. Kroz PN sloj struja protiče od anode (p) i kreće se prema katodi (n). Elektroni i šupljine se kreću jedni prema drugima i u trenutku sudara elektroni prelaze na niži energetska nivo, pri tome oslobađajući energiju u obliku fotona. Šematske oznake dioda i LE dioda su veoma slične i razlikuju se u dve male strelice koje simbolizuju izlučenje svetlosti.



Zbog svoje „diodne prirode”, LED lampice dozvoljavaju kretanje elektriciteta samo u jednom pravcu, od anode prema katodi. To znači da, ukoliko slučajno diodu okrenemo naopako, ona neće svetliti. Pozitivna stvar u svemu tome je da na ovaj način u standardnim uslovima nećemo izazvati pregorevanje elementa.

U svrhu lakšeg raspoznavanja, LE diode dolaze sa nožicama različitih dužina. Anoda ima dužu žicu od katode i na nju dovodimo pozitivni napon. Još jedan vizuelni način razlikovanja anode i katode odnosi se na malu izbočinu koja se nalazi na strani anode.



Intenzitet svetlosti LE dioda je direktno vezana za jačinu struje koja kroz njih prolazi. Što više struje, to više svetlosti. Međutim, ukoliko dovedemo više struje nego što je to dozvoljeno, lampica će jednostavno pregoreti. Ako malo obratimo pažnju na električne šeme, videćemo da uz svaku LE diodu ide prateći otpornik. Otpornici su pasivni elektronički elementi čija je funkcija da pruže otpor toku struje. Na taj način se ograničava maksimalna vrednost strujnog toka koja je bezbedna za naš uređaj.

Da bi odredili veličinu otpornika, potrebno je da poznajemo određene karakteristike LE diode sa kojom radimo. Iako svi proizvođači uz svoje proizvode navode tehničke specifikacije, prilikom kupovine preko interneta često dobijamo narudžbinu bez bilo kakvih pratećih informacija. Zato nam ne preostaje ništa drugo nego da koristimo standardne vrednosti, koje nisu uvek optimalne, ali su dovoljne za normalno funkcionisanje. Glavne karakteristike koje nas interesuju su: radni napon, maksimalna jačina struje i preporučena jačina struje.

Dva osnovna parametra svetlećih dioda su jačina struje i napon. Da bi videli svetlost, potrebno je da napon izvora energije bude veći od radnog napona. U sledećoj tabeli su prikazani podaci sa karakterističnim vrednostima radnog napona u zavisnosti od boje svetlosti:

Kao što možemo zaključiti iz priloženog, što je manja talasna dužina svetlosti, potrebno je više napona. Ukoliko ne posedujemo podatke o maksimalnoj jačini struje i preporučenoj jačini struje, okvirno možemo uzeti da se vrednosti za sve LE diode kreću u opsegu od 18-30 miliampera i pri tome se vodimo pravilom da onima koje imaju manje vrednosti radnog napona dajemo manje struje, dok onima kod kojih je ta vrednost veća, propuštamo više struje. Tako ćemo, recimo, za crvenu svetleću diodu uzimati da je potrebna jačina struje u rang 18-20 miliampera, za žutu 20-22 miliampera, a za zelenu i plavu 23-26 miliampera. Treba voditi računa o tome da maksimalne vrednosti skraćuju vek trajanja uređaja, pa se preporučuje da se koristi 80% predviđene jačine struje. Ukoliko smatrate da jačina svetlosti LE diode nije dovoljno intenzivna, možete pokušati sa promenom vrednosti otpornika za nekih 10% njegove trenutne vrednosti.

Računanje električnog otpora se izvodi preko starog dobrog Omovog zakona:

$$R = \frac{(U_i - U_{rn})}{I}$$

(R – otpor, U_i – napon izvora, U_{rn} – radni napon, I – jačina struje)

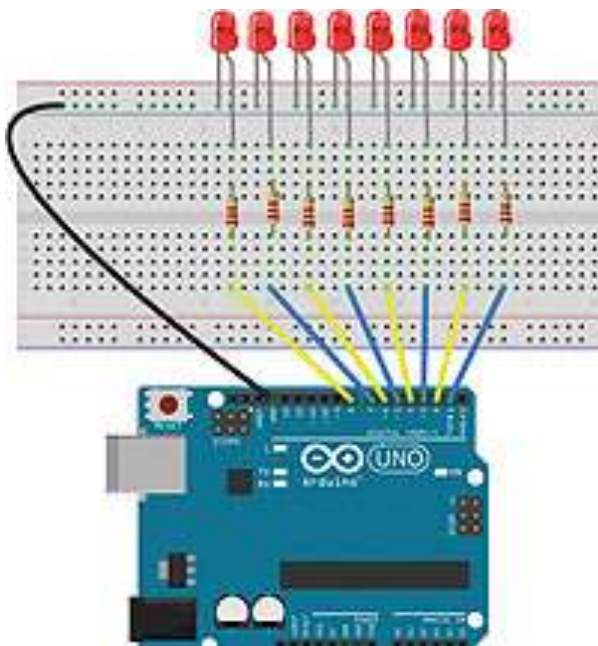
Uzmimo za primer da želimo da priključimo svetleću diodu crvene boje na naš *Arduino*, koji radi sa naponom od pet volti:

$$R = \frac{5v - 1,8v}{0,02A} = 160\Omega$$

Za radni napon crvene LE diode uzeli smo srednju vrednost koja se kreće u okviru 1,6-2 volta i oduzeli je od napona *Arduino* (pet volti), pa smo to podelili sa 0,02 ampera (20 miliampera) i dobili minimalnu vrednost otpornika od 160 oma.

Za plavu LE diodu imamo sledeću kalkulaciju:

$$R = (5\text{ V} - 3,3\text{ V}) / 0,025\text{ A} = 68\text{ oma}$$



Ovde smo za vrednost radnog napona uzeli 3,3 volta, a za jačinu struje 25 miliampera. U oba slučaja smo dobili vrednosti koje odgovaraju otpornicima sa standardnim vrednostima. Ukoliko smo proračunom dobili neku nestandardnu brojku (recimo, 500 oma), onda uzimamo sledeću veličinu koja odgovara standardnoj vrednosti (510 oma).

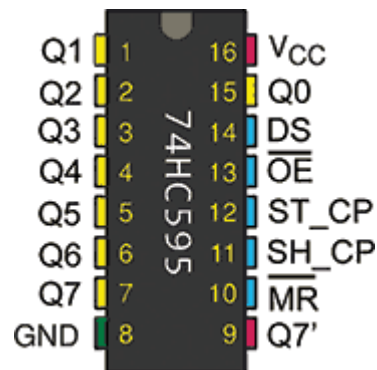
Povezivanje više svetlećih dioda na *Arduino* je trivijalan posao. Svaku anodu preko otpornika vežemo na neki od izlaza mikrokontrolera, dok katode priključujemo na uzemljenje. Pošto *Arduino (Uno)* raspolaže strujom jačine oko 200 miliampera, jednostavna matematika nam govori da smo u mogućnosti da povežemo do maksimalno deset LE dioda koje troše po 20 miliampera. Ukoliko

je potrošnja 25 miliampera, onda nam, teoretski, preostaje osam LE dioda. Jedan jednostavan primer koda za konfiguraciju sa osam LE dioda koje se pale i gase od početka do kraja bi mogao da izgleda ovako:

```
// niz portova za LED
int ledNiz[] = {8, 7, 6, 5, 4, 3, 2, 1}; int pinovi = 7; // ukupno
pinova (8)
int pauza = 100;
void setup() {
  for (int pin = 0; pin < pinovi; pin++) {    // svi pinovi na OUTPUT
    pinMode(ledNiz[pin], OUTPUT);
  }
}
void loop() {      // od prve do poslednje
  for (int pin = 0; pin < pinovi; pin++) {
    digitalWrite(ledNiz[pin], HIGH); // upali LED
    delay(pauza);
    digitalWrite(ledNiz[pin], LOW); // ugasi LED
  }
  // od poslednje do prve
  for (int pin = pinovi - 1; pin >= 0; pin--) {
    digitalWrite(ledNiz[pin], HIGH); // upali LED
    delay(pauza);
    digitalWrite(ledNiz[pin], LOW); // ugasi LED
  }
}
```

U prvoj liniji definišemo niz koji sadrži izlazne pinove *Arduina*, a njih je u našem slučaju osam (0-7). Kod sa funkcijom *setup()* uz pomoć *For..Next* petlje inicijalizuje sve pomenute pinove kao izlazne. U okviru *loop()* bloka imamo dve petlje koje uključuju i isključuju diode na portovima ukazanim nizom *ledNiz[]*.

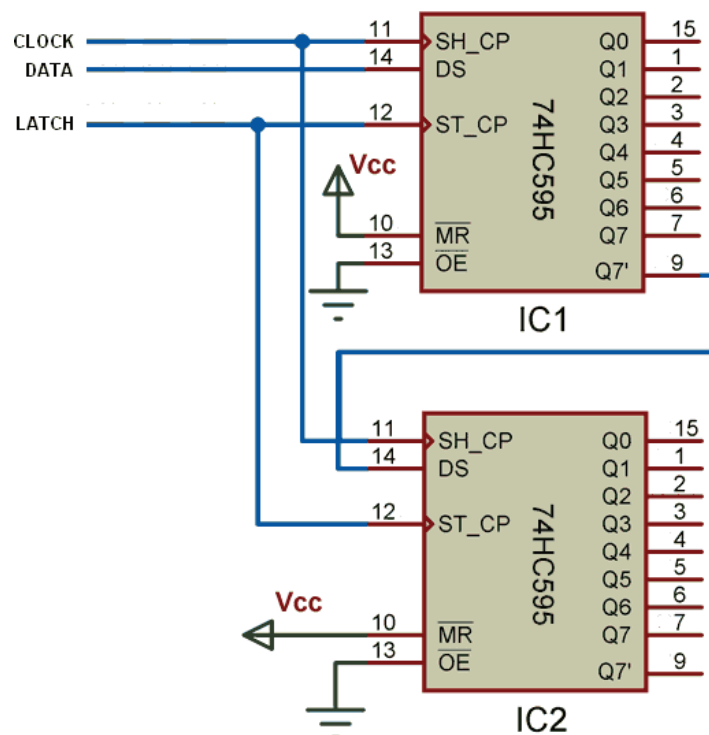
Šta da se radi u situacijama kada je potrebno priključiti veći broj lampica? Odgovor na to pitanje se ogleda u korišćenju integrisanih kola koja se nazivaju pomeračkim (*shift*) registrima.



Reč je o relativno jednostavnim napravama koje za svoju osnovu koriste *flip-flop* registre D tipa. Da sada ne ulazimo u detalje, ukratko ćemo reći da je *flip-flop* digitalni sklop koji je u stanju da pamti jedan bit informacije. Prateća šema prikazuje SIPO (*Serial-In, Parallel-Out*) šift registar sa četiri *flip-flopa* (FF1-FF4) koji na ulazu dobijaju podatke u serijskom obliku i transformišu ih u paralelni izlaz.

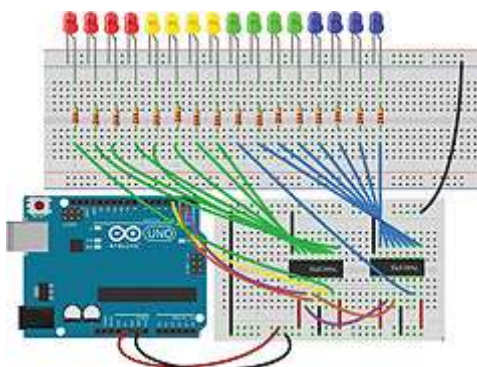


Za sinhronizaciju se koristi klok signal magistrale podataka. Upravo ovaj mehanizam nam omogućava da znatno povećamo broj izlaznih portova na račun transformacije serijskih podataka u paralelni oblik. Najpoznatiji predstavnik ove vrste integrisanih kola nosi naziv 74HC595 (varijanta u DIP kućištu ima dodatno slovo N) i obavlja funkciju 8-bitnog šift registra koji može da radi kako u SIPO, tako i u SISO režimu (*Serial-In, Serial-Out*). Osim njega, veoma često se kao 8-bitni šift registri koriste i čipovi sa oznakama 74HC594, 74HC164, 74HC165, 74HC299 (PIPO) i 74HC4094, koji imaju neke svoje specifičnosti. Pored 8-bitnih pomeračkih registara, postoje i integralna kola koja rade sa 4 i 16 bitova.



Pinovi 1-7, kao i pin 15, predstavljaju osam bitova paralelnog izlaza i imaju vrednost unutrašnjih *flip-flopova*. Zapravo, unutar čipa 74HC595 postoje dva bloka od po osam memorijskih ćelija. Prvi blok služi za popunjavanje podacima koji dolaze preko serijskog ulaza (ovo i jeste „pravi” šift registar), dok drugi blok (naziva se „*latch*”) pri pojavljivanju impulsa sinhronizacije ST_CP, preslikava trenutne vrednosti iz šift registra. Na taj način se sprečava treperenje koje može nastati pomeranjem bitova u šift registru. Serijski binarni podaci u čip stižu preko pina broj 14 (DS). Usled pomeranja, novi bit dolazi na mesto prethodnog. Pin 11 ili SH_CP (*Shift Register Input clock*), prelaskom iz stanja logičke nule u jedinicu, bit koji se nalazi na pinu DS smešta u registar, dok se istovremeno svi ostali bitovi registra pomeraju za jednu poziciju. Pin 12 (ST_CP) se za vreme upisivanja podataka nalazi u stanju logičke nule. Njegovim prelaskom u logičku jedinicu, bitovi se preslikavaju u „*latch*” registar. Pin 10 nosi oznaku MR (Master Reset) i ima funkciju brisanja podataka u šift registru bez uticaja na latch. Mi ga fiksno povezujemo na 5V.

Funkcija pina 13 (OE, *Output Enable*) je da dozvoli slanje stanja bitova *latch* registra na izlazne pinove. I ovde je standardno stanje logička nula, a najčešće se ovaj konektor fiksno povezuje na liniju GND, čime se omogućava konstantno prikazivanje podataka. Zanimljiva je funkcija pina 9 (Q7'), koji služi kao mesto izlaza serijskih podataka iz šift registra (*Serial Out*). Zahvaljujući njemu, moguće je iskombinovati veći broj 74HC595 kola u šift registar veličine 16, 24, 32, 40 ili još više bitova. To se postiže povezivanjem pina broj 9 na prvom čipu sa pinom 14 na sledećem.



Pošto je konstrukcija sklopa veoma slična, u našem praktičnom primeru ćemo pokazati kako se povezuje šesnaest LED lampica preko dva integralna kola 74HC595.

Iako nisu u pitanju idealne vrednosti, u primeru smo koristili otpornike od 220 oma, kojih smo imali na raspolaganju u većoj količini. Žutom džamper žicom smo povezali D3 pin *Arduina* sa pinom 14 (DS) levog čipa, narandžastom pin D2 sa pinom 12 (ST_CP), a ljubičastom pin D1 sa pinom 11 (SH_CP). Bela žica povezuje pin 9 levog čipa sa pinom 14 (DS) desnog čipa i na taj način omogućavamo serijski prenos podataka među njima.

```
int clock = 1; // SH_CP ulaz na 74HC595
int latch = 2; // ST_CP -//-
int data = 3; // DS -//-
void setup() { // svi pinovi su OUTPUT
  pinMode(clock, OUTPUT);
  pinMode(latch, OUTPUT);
  pinMode(data, OUTPUT);
}
void loop() {
  // upali sve
  digitalWrite(latch, LOW);
  shiftOut(data, clock, MSBFIRST, 0xFF);
  shiftOut(data, clock, MSBFIRST, 0xFF);
  digitalWrite(latch, HIGH); // B11111111-11111111
  delay(1000);
  // upali neparne
  digitalWrite(latch, LOW);
  shiftOut(data, clock, MSBFIRST, 0xAA);
  shiftOut(data, clock, MSBFIRST, 0xAA);
  digitalWrite(latch, HIGH); // B10101010-10101010
  delay(1000);
  // upali parne
```

```

digitalWrite(latch, LOW);
shiftOut(data, clock, MSBFIRST, 0x55);
shiftOut(data, clock, MSBFIRST, 0x55);
digitalWrite(latch, HIGH); // B01010101-01010101
delay(1000);
// ugasi sve
digitalWrite(latch, LOW);
shiftOut(data, clock, MSBFIRST, 0x00);
shiftOut(data, clock, MSBFIRST, 0x00);
digitalWrite(latch, HIGH);
delay(1000);
}

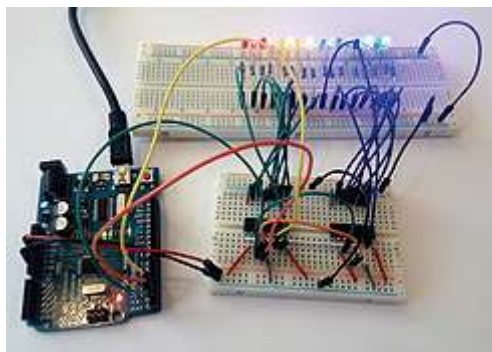
```

Prve tri naredbe određuju pinove koje nameravamo da koristimo, a onda, u bloku *setup()* svaki od tih pinova postavljamo u stanje izlaza. U okviru petlje *loop()* ponavlja se isti šablon četiri puta. Prvo postavljamo pin „*latch*” u stanje logičke nule, zatim sa dve komande *shiftOut()* šaljemo dva bajta prema našim šift registrima i na kraju, postavljanjem signala ST_CP u stanje logičke jedinice, prepisujemo podatke iz šift registara u „*latch*”, čime dovodimo do promene stanja na izlaznim pinovima čipa. Naredba *shiftOut*ima sledeću sintaksu:

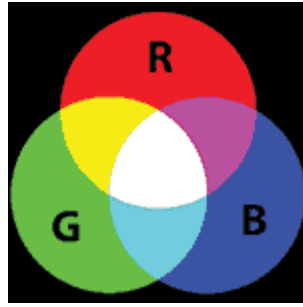
```
shiftOut(DS pin, SH_CP pin, redosled, vrednost)
```

Prva dva parametra ukazuju na to koji su pinovi *Arduina* povezani sa pomenutim nožicama integralnog kola. Argument *redosled* ukazuje na način pomeranja bitova. Može da bude MSBFIRST ili LSBFIRST i od njega zavisi da li će upisivanje bitova ići od najmanjeg ka najvećem ili obratno.

Argument *vrednost* predstavlja osmobitni broj koji šaljemo šift registru. Mi smo koristili heksadecimalni prikaz brojeva pošto on olakšava definisanje šablona binarnih brojeva. Ako bi hteli da napravimo raspored od po dve lampice koje svetle, iza kojih dolaze dve koje ne svetle, to bi se binarnim brojevima pisalo kao: B11001100, a heksadecimalnim kao 0xCC.



Višebojne (RGB) LED



Osim jednobojnih, postoje i svetleće diode koje mogu da prikažu svetlost u širokom opsegu boja. Njihova konstrukcija se sastoji u aditivnom kombinovanju crvene, zelene i plave svetlosti. Postoje modeli sa zajedničkom katodom i sa zajedničkom anodom, kao što je to prikazano na pratećoj ilustraciji. Sve „prave” LE diode imaju po četiri nožice: tri za boje i jednu koja predstavlja zajedničku katodu ili anodu.

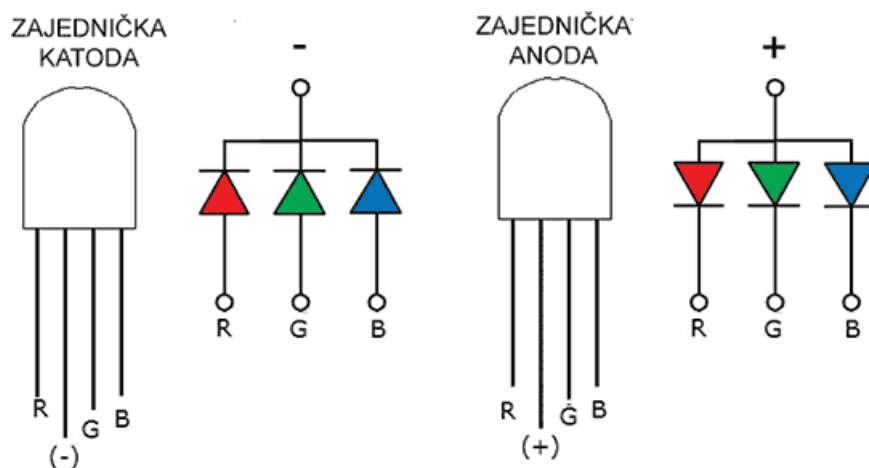
Moguće je kupiti LE diode koje emituju svetlost u bojama i imaju samo dve nožice, ali njih ne možemo programirati pošto u sebi već sadrže elektroniku i program za emitovanje po određenoj šemi.



Zajedničku anodu priključujemo na napon od pet volti, dok se pojedinačne nožice katoda putem otpornika priključuju na izlaze *Arduina*. Po analogiji, kod modela sa zajedničkom katodom nju priključujemo na uzemljenje, dok pojedinačne anode sa *Arduinom* spajamo preko otpornika. Kada imamo dve vezane elektroničke komponente i kada na prvoj imamo stanje logičke jedinice, struja iz prve komponente služi kao izvor struje za drugu komponentu, koja deluje kao otporno opterećenje prema uzemljenju. Zbog asocijacije na strujni izvor, ova pojava se na engleskom naziva *current sourcing*. U slučaju da je na izlazu stanje logičke nule, struja teče iz ulaza druge komponente u izlaz prve i tada izlaz prve komponente deluje kao ponor koji vuče struju iz druge komponente, koja ima ulogu otpornog opterećenja prema naponu napajanja. Ovo se naziva *current sinking*. Za sada nije potrebno da se opterećujemo ovim stručnim pojmovima, već ćemo jednostavno objasniti razliku između primene LED elemenata sa zajedničkom katodom i anodom. U slučaju da želimo da uključimo neku od tri LE diode na modele sa zajedničkom katodom, potrebno

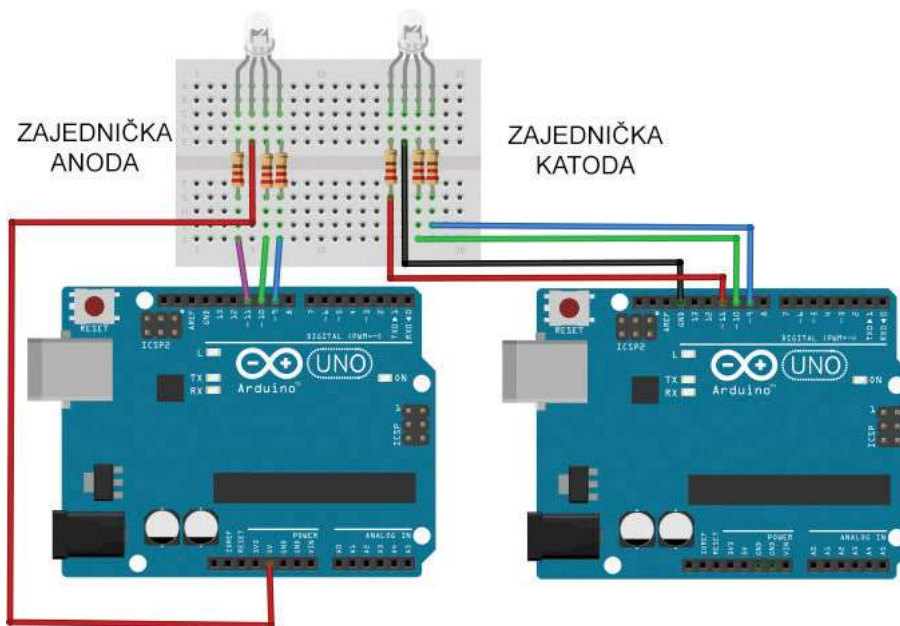
je da izlaznom *Arduino* pinu postavimo stanje HIGH (*current sourcing*), dok isključivanje postižemo slanjem signala LOW. Međutim, kod modela sa zajedničkom anodom, sve moramo da radimo suprotno. Lampicu uključujemo slanjem signala LOW, a isključujemo je signalom HIGH (*current sinking*). Dakle, u slučaju da radimo sa elementom koji ima zajedničku katodu, kôd bi mogao da izgleda ovako:

```
digitalWrite(11, HIGH); // ukljuci crvenu
digitalWrite(10, HIGH); // ukljuci zelenu
delay(1000);
digitalWrite(11, LOW); // iskljuci crvenu
digitalWrite(10, LOW); // iskljuci zelenu
delay(1000);
```



A u slučaju korišćenja LE diode sa zajedničkom anodom, ovako:

```
digitalWrite(11, LOW); // ukljuci crvenu
digitalWrite(9, LOW); // ukljuci plavu
delay(1000);
digitalWrite(11, HIGH); // iskljuci crvenu
digitalWrite(9, HIGH); // iskljuci plavu
delay(1000);
```



Funkcija *digitalWrite*, koju smo koristili u prethodnom kodu, poznaje samo dva stanja: uključeno i isključeno. To znači da uz njenu pomoć možemo prikazati samo ograničen broj boja (sedam). Ukoliko želimo da prikazemo neku od (teoretskih) 16,7 (256^3) miliona boja, potrebno je da koristimo funkciju *analogWrite*.

Ona se oslanja na tehniku PWM (*Pulse Width Modulation*), koja joj omogućava emulaciju 256 različitih nivoa napona, kao da je u pitanju analogni uređaj.

```
int crvena = 0;
int zelena = 0;
int plava = 0;
int pauza = 250;
void setup() { // izlazni pinovi
  pinMode(11, OUTPUT); // crvena
  pinMode(10, OUTPUT); // zelena
  pinMode(9, OUTPUT); // plava
}
void loop() {
  crvena = random(255);
  zelena = random(255);
  plava = random(255);
  analogWrite(11, crvena);
  analogWrite(10, zelena);
  analogWrite(9, plava);
  delay(pauza);
}
```

Kod je dovoljno jasan i svodi se na generisanje slučajnih vrednosti za sve tri boje, a zatim se te vrednosti šalju na izlazne pinove u pseudo-analognom PWM obliku putem funkcije *analogWrite()*.

Svetleće diode su zanimljivi elektronički elementi koji nisu teški za korišćenje, a istovremeno obezbeđuju dosta zabave i imaju širok spektar primena. Osim toga, videli smo da upotreba integrisanih kola za pomeranje sadržaja ne predstavlja nikakav bauk. U sledećem nastavku ćemo se pozabaviti tehnikama koje obuhvataju korišćenje uređaja kreiranih na osnovu svetlećih dioda, organizovanih u segmente i matrice.

TLC5940 LED driver

Extending PWM output pins with a
Texas Instruments

Julian Vogels in [Work](#) on June 17, 2013

Tags: [Arduino](#), [Control Circuit](#), [How To](#), [PWM](#), [TLC5940](#), [Tutorial](#)

This tutorial was originally written for sensorwiki.org, a fantastic resource for the design of gestural controllers that is supported by [IDMIL](#).

Introduction

Microcontrollers like the Arduino were designed to facilitate the use of electronics for designers and DIY enthusiasts. The interface provides a great starting point for a variety of electronic circuit designs. However, as the microcontroller is standardized, it is also limited in its use. That shows for example in the limited number of PWM (pulse width modulation) enabled output pins.

What can you do to extend the PWM capabilities of your Arduino? Just buy a bigger one? That is not necessary anymore after you have read this article. Here it is shown how to connect an Arduino microcontroller to a Texas Instruments TLC5940 LED Driver to connect a large number of LEDs, or even power-intensive devices such as star-mounted high power RGB LEDs or servo motors.

In the design of digital musical instruments (DMIs), this is particularly useful to provide different kinds of feedback to the performer while maintaining high extensibility at a lower cost.

Disclaimer: This information and the circuits are provided as is without any express or implied warranties. While every effort has been taken to ensure the accuracy of the information contained in this text, the authors/maintainers/contributors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Capabilities

The datasheet of the TLC5940 is available from Texas Instruments, amongst other useful information such as application notes and the option to request samples.

A selection of important features:

Number of channels: 16

Resolution: 12 bit (4096 steps)

Drive Capability: 0 mA to 120 mA (for $V_{CC} > 3.6$ V)

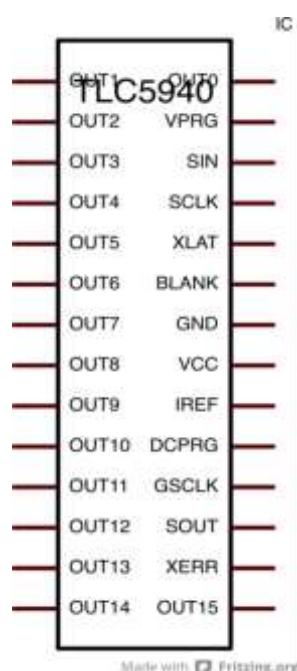
Connectable actuators

Many electrical components can be controlled using a PWM signal. Not only LEDs can be dimmed, but also Servo motors can be driven, as well as DC motors.

Daisy chaining

Daisy chaining means that you can wire multiple devices together in series. In our case, we can not only extend the PWM pins with one TLC5940 with 16 pins, but because of the daisy-chain ability even use multiple TLC5940s to output 32, 48 or 64 PWM signals.

Wiring



The wiring of the TLC5940 will occupy 4 PWM pins on the Arduino for the serial communication to the chip. Depending on your Arduino, you will have to look up the according pins that have to be connected. The [tlc5940arduino Arduino library](#) provides additional information (e.g., connecting an Arduino Mega). Please read the comments to each scheme, as there are a few mistakes in the showed pin setup.

Generally, the MOSI pin is connected to the TLC SIN, SCK to SCLK on the TLC, OC1A to XLAT, OC1B to BLANK and OC2B to GSCLK. In addition, the TLC's DCPRG should be connected to V_{CC} (rather than to GND) to disable the on-chip EEPROM dot-correction and enable the dot-correction from the DC-register in the device that can later be used with the Arduino library. The VPRG pin on the TLC can be connected to ground when the standard "greyscale" PWM register should be used, as opposed to connecting this pin to the Arduino's digital pin 8 to use the dot-correction functions in the library. This pin is optional and you can leave it on GND for now. The TLC's XERR pin can be used to check for thermal overloads if you connect it to digital pin 12 on the Arduino.

In order to daisy chain two or more TLC5940, connect the SOUT of TLC 1 to the SIN of TLC 2, and the SCLK, XLAT, BLANK and GSCLK and proceed in that manner for every additional TLC5940.

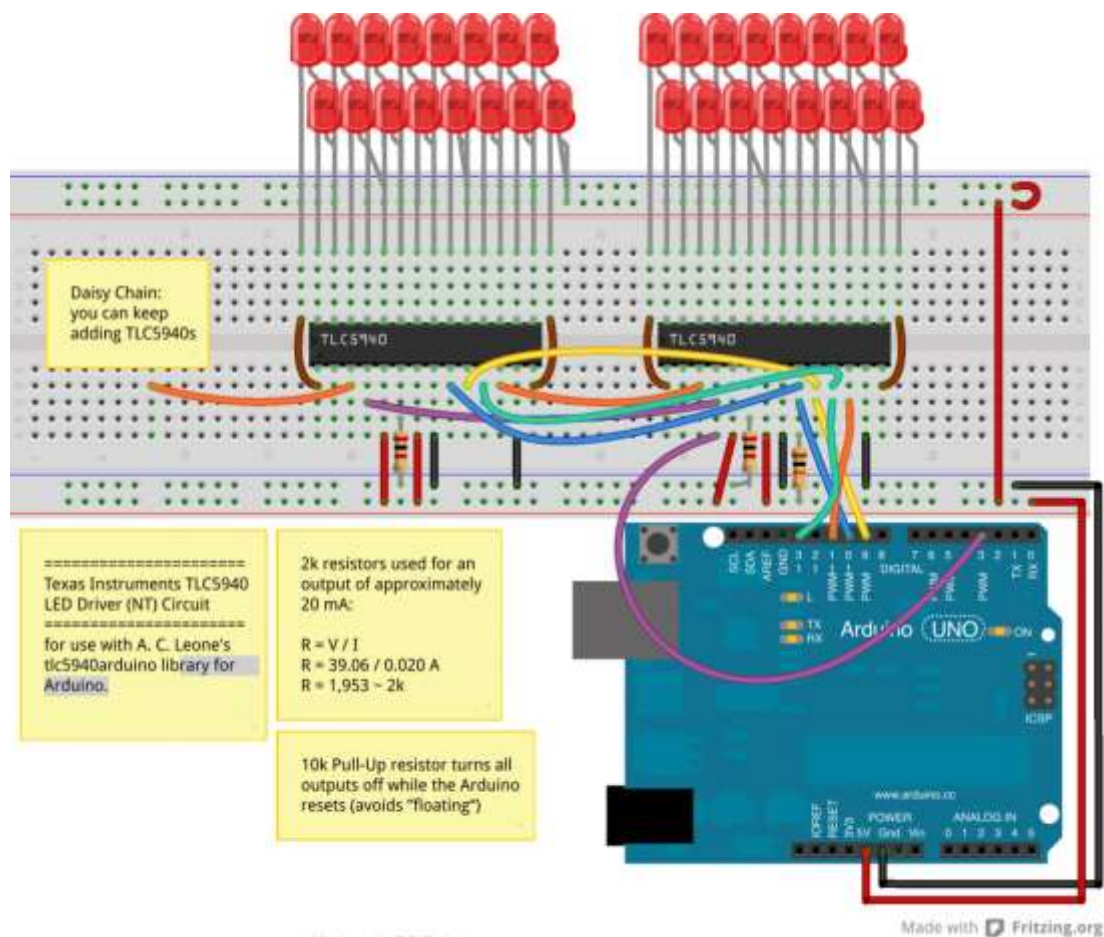
A 10k pull-up resistor connects the TLC BLANK to GND. This is necessary in order to turn off all outputs while the Arduino resets, so that they do not "float" (it would output the voltage difference between two not ground-referenced signals – basically noise). It is only necessary to add this resistor to the first TLC5940 in the daisy chain, as the BLANK pins are connected.

The IREF pin of every TLC5940 has to be connected to V_{CC} through a resistor. The resistor value has to be calculated according to the output current that is suitable for your application. If you want to connect components that draw 20 mA of current (such as standard LEDs), use Ohm's law to receive the resistor value:

$$\begin{aligned} R &= V / I \\ R &= 39.06 \text{ V} / 0.020 \text{ A} \\ R &= 1,953 \cong 2\text{k} \end{aligned}$$

For those who want to know where the number 39.06 comes from: As the output current of the TLC5940 is set by a current mirror by taking the reference current (that is determined by a resistor from an on-chip 1.24V voltage reference) and multiplying it with a nominal gain of 31.5 you get $1.24 \times 31.5 \cong 39.06$!

Please study this breadboard layout for connecting 32 LEDs to your Arduino. Note that output pins 0 and 15 of the TLC5940 are on the opposite site of the other output pins.



Control circuit

An Arduino microcontroller is limited in its output current to 40 mA, while it should probably not be driven at maximum. Overall, you should not draw more than 200 mA current from the Arduino as that is the processor chip package current.

If you need to drive high power consuming devices, you should design a control circuit and a work circuit. The control circuit, which is driven with a low current, will tell the work current when to let current flow to your connected devices. This is accomplished through the use of transistors. For every output pin that you want to control separately, you'll need a PNP transistor.

Note: Don't pick an NPN transistor, as the TLC5940 is a constant-current sink and the current has to flow towards the output pins. As a PNP transistor's base pin will connect the collector end emitter pins when current is drawn from is, as opposed to the behaviour of an NPN transistor, that switches when current is applied to the base pin.

Make sure that you get a PNP transistor that switches quickly and operates at the TLC5940's output current of 20 mA. You can verify this by looking on the graphs in the transistors datasheet.

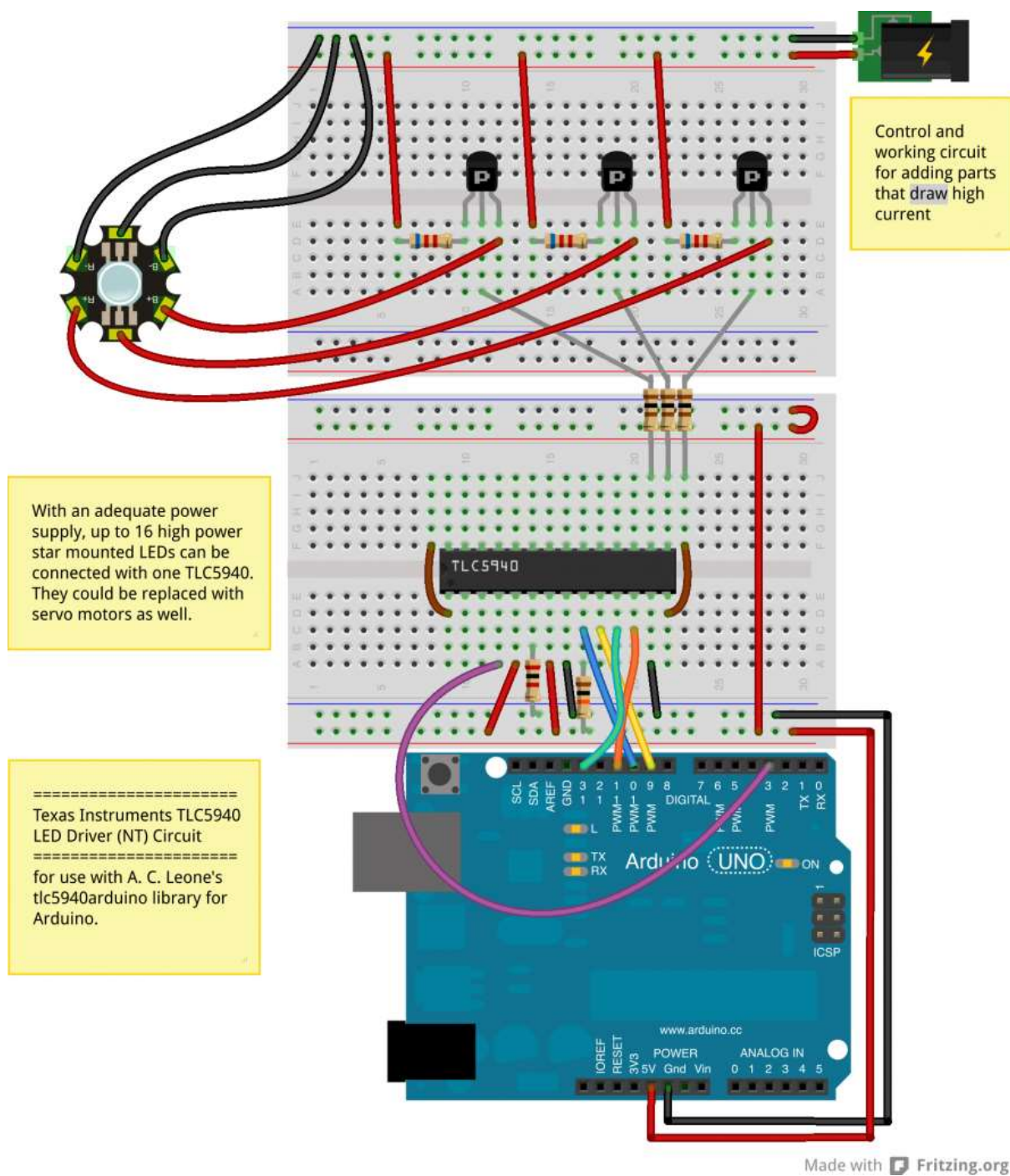
The work circuit is only connected with the Arduino through the transistors and operates at a higher current such as 400 mA. If you would connect a star-mounted high power RGB LED such as a [this model from Vollong](#), you will need to connect a resistor between the emitter and the diode. The resistance value is calculated as follows:

$$R = (5 \text{ V} - 2.5 \text{ V}) / (0.4 \text{ A}) = 6.25 \text{ Ohm}$$

$R = (\text{supply voltage} - \text{diode voltage}) / (\text{diode current})$

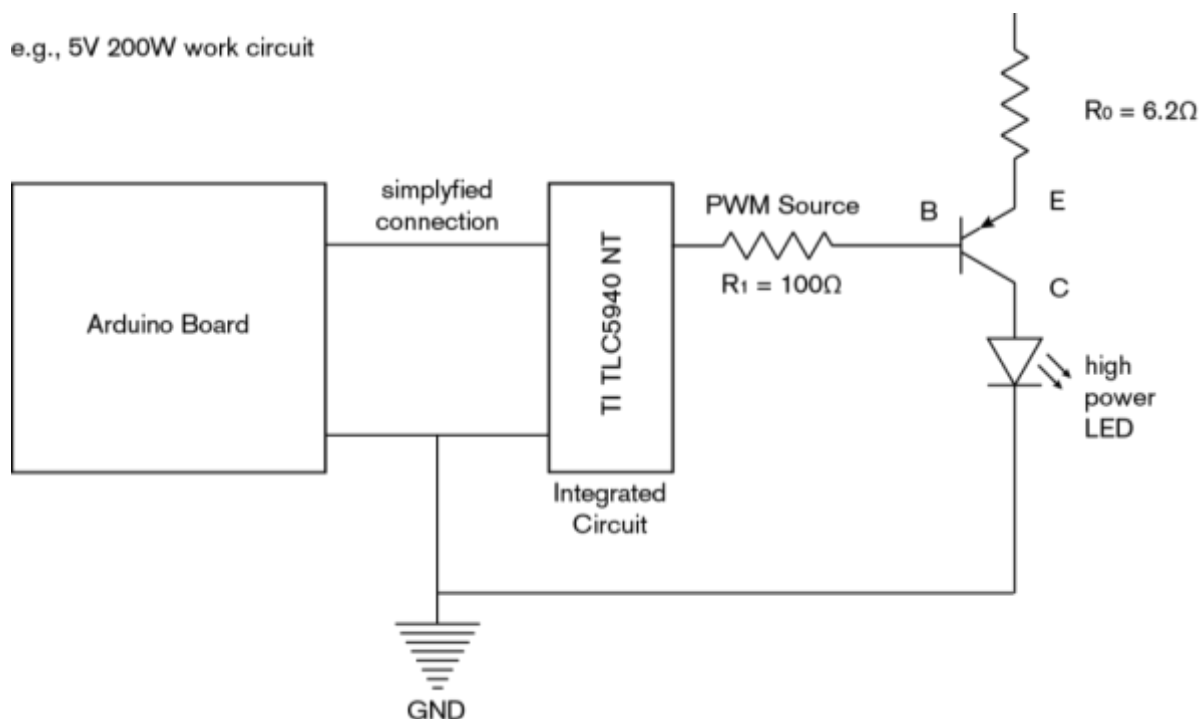
You should choose the resistor that is nearest to the off-rounded value of that number.

Choose the power supply according to the amount of Watt needed by the connected devices. You can calculate the power of each device by multiplying the voltage and the current and then sum the results.



These simplified schematics show a single high power LED connected via a PNP transistor to the TLC5940.

e.g., 5V 200W work circuit



Arduino Code

For the Arduino code, please refer to the well documented [tlc5940arduino Arduino library](#) written mostly by A. C. Leone. After putting the downloaded folder into the library folder inside the Arduino folder, example files will be available under the File > Example section in the Arduino IDE.

The Example file BasicUse.h will guide you through the most important library features. Basically, the TLC has to be initialized in the setup statement of the code (*Tlc.init()*). Then, you can set the value of each output pin in a for loop by accessing *Tlc.set(channel, value)* where channel is 0 to 15, and value is 0 to 4095. *Tlc.update()* is then used to actually send the set values to the TLC5940, whereas *Tlc.clear()* sets all values to zero without sending them.

An important thing to know is that if you want to use multiple TLC5940s, you have to set their quantity in the file "tlc_config.h" in the library's folder. Open the file with your favorite text editor and replace the value of the constant NUM_TLCS with the amount of TLCs you're using. Save the file and restart the Arduino IDE.

Servo motors have to be controlled differently than common LEDs. Fortunately, the tlc5940 arduino library provides a way of doing so without

having to change much code. The example file *Servos.ino* explains how you should connect a servo motor and shows the use of custom library functions such as `tlc_initServos()`.

Be aware that you cannot use LEDs and servo motors with the same TLC5940 (either if daisy-chained), as the use of the latter function will drop down the PWM frequency to 50 Hz (which will be significant for the LEDs).

Applications

The circuit is useful for connecting any large number of actuators to your device. For example, you could imagine giving visual feedback to user interaction on different interaction locations. An array of individually PWM controlled LEDs that can even be faded gradually can be accomplished using the TLC5940.

An array of individually controllable servo motors could be used for many purposes, as servo motors are very accurate, quite fast adjusting their angle and versatile due to the available servo accessories such as horns and rods.

Additional Information

Sparkfun sells a [TLC5940 Breakout board](#) for a reasonably low price (currently \$12.95 where the TLC5940 alone is at about \$8).

Summary

This document provided an introduction to the TLC5940 LED driver, details on its capabilities and applications and practical information on its implementation and use with the Arduino library.

Fritzing files

[Fritzing](#) is an open-source software distribution for designing breadboard layouts and much more. You can download these zipped Fritzing .fzz files to get a better understanding of how to wire it up.

[TLC5940 Breadboard Layouts](#)

The contained Fritzing file *TLC5940controlcircuit.fzz* requires the [Adafruit Fritzing Object library](#) as an exotic object is used (the high power LED).

WS2811 LED driver

Overview

These flexible RGB LED strips are an easy way to add complex lighting effects to a project. Each LED has an integrated driver that allows you to control the color and brightness of each LED independently. The combined LED/driver IC on these strips is the extremely compact WS2812B (essentially an improved WS2811 LED driver integrated directly into a 5050 RGB LED), which enables higher LED densities. In the picture on the right, you can actually see the integrated driver and the bonding wires connecting it to the green, red, and blue LEDs, which are on at their dimmest setting.

In contrast to the APA102C used in some of our [other similar LED strips](#), which uses a standard SPI interface (with separate data and clock signals), the WS2812B uses a specialized one-wire control interface and requires strict timing. See the bottom of this product page for a more detailed comparison of the WS2812B and APA102C.

We offer six different kinds of WS2812 LED strip with different LED densities and lengths. Our strips with **30 LEDs per meter** are available in three lengths:

- [1 meter, 30 LEDs](#)
- [2 meters, 60 LEDs](#)
- [5 meters, 150 LEDs](#)

We also offer denser WS2812 LED strips that have **60 LEDs per meter**:

- [1 meter, 60 LEDs](#)
- [2 meters, 120 LEDs](#)

Our [highest-density strip](#) has its WS2812 LEDs packed together as tightly as possible, resulting in 72 LEDs on a 0.5 meter strip (i.e. **144 LEDs per meter**).

The information on this page applies to all of the WS2812-based LED strips we sell.



LED side of the WS2812B-based addressable LED strips, showing 30 LEDs/m (top), 60 LEDs/m (middle), and 144 LEDs/m (bottom).

Features and specifications

- Individually addressable RGB LEDs (30, 60, or 144 LEDs per meter)
- 24-bit color control (8-bit PWM per channel); 16.8 million colors per pixel
- One-wire digital control interface
- 5 V operating voltage
- Each RGB LED draws approximately 50 mA at 5 V with red, green, and blue at full brightness
- 12 mm width, 4.6 mm thickness
- Flexible, waterproof silicone rubber sheath ([IP65 protection rating](#))
- Includes flexible silicone mounting brackets
- Black strip color
- Power/data connectors on both strip ends for easy chaining, and the input side includes an additional power and ground wire for alternate power connections
- Strips can be cut apart along the lines between each RGB LED segment to separate them into usable shorter sections
- Example code available for Arduino, AVR, and mbed

Using the LED strip



The connectors and power wires for addressable LED strips. On the left is the input end of the strip and on the right is the output end.

Each LED strip has three connection points: the input connector, the auxiliary power wires, and the output connector. These can be seen in the adjacent picture, from left to right: auxiliary power wires, input connector, output connector. The strip uses 3-pin JST SM connectors.

The **input connector** has three male pins inside of a plastic connector shroud, each separated by about 0.1". The black wire is ground, the green wire is the signal input, and the red wire is the power line.

The **auxiliary power wires** are connected to the input side of the LED strip and consist of stripped black and red wires. The black wire is ground, and the red wire is the power line. This provides an alternate (and possibly more convenient) connection point for LED strip power.

The **output connector** is on the other end of the strip and is designed to mate with the input connector of another LED strip to allow LED strips to be chained. The black wire is ground, the green wire is the signal output, and the red wire is the power line.

All three black ground wires are electrically connected, and all three red power wires are electrically connected.



www.pololu.com

A close-up of the JST SM connectors for our addressable LED strips.

Included hardware

These LED strips ship with flexible silicone brackets and screws. Strips with lengths of 1 meter or greater include five brackets and ten screws *per meter*. Our 0.5 meter high-density strip ships with a total of two brackets and four screws. The brackets fit over the waterproof sheath and can be used to mount the LED strip. The LED strip also ships on a plastic reel.



The 1m, 2m, and 5m addressable LED strips include five mounting brackets per meter; the 0.5m strip includes 2 total brackets.



A 2-meter, 60 LED addressable RGB LED strip on the included reel.



Controlling an addressable RGB LED strip with an Arduino and powering it from a 5V wall power adapter.

Connecting the LED strip

To control the LED strip from a microcontroller, two wires from the input connector should be connected to your microcontroller. The LED strip's ground (black) should be connected to ground on the microcontroller, and the LED strip's signal input line (green) should be connected to one of the microcontroller's I/O lines. The male pins inside the input connector fit the female terminations on our [premium jumper wires](#) and [wires with pre-crimped terminals](#). If you are connecting the LED strip to a breadboard or a typical Arduino with female headers, you would want to use [male-female wires](#).

We generally recommend powering the LED strip using the auxiliary power wires. Our [5 V wall power adapters](#) work well for powering these LED strips and a [DC Barrel Jack to 2-Pin Terminal Block Adapter](#) can help you make the connection between the adapter and the strip. However, you might need a [wire stripper](#) to strip off some more insulation from the power wires. It is convenient that the power wires are duplicated on the input side because you can connect the auxiliary power wires to your 5 V power supply and then the power will be available on the data input connector and can be used to power the microcontroller that is controlling the LED strip. This means you can power the microcontroller and LED strip from a single supply without having to make branching power connections.

Warning: The WS2812B seems to be more sensitive than the TM1804 on our original LED strips. We recommend taking several precautions to protect it:

- Connect a capacitor of at least 100 μF between the ground and power lines on the power input.
- Avoiding making or changing connections while the circuit is powered.
- Minimize the length of the wires connecting your microcontroller to the LED strip.
- Follow generally good engineering practices, such as taking precautions against electrostatic discharge (ESD).

- Consider adding a 100 Ω to 500 Ω resistor between your microcontroller's data output and the LED strip to reduce the noise on that line.

If the strip does get damaged, it is often just the first LED that is broken; in such cases, cutting off this first segment and resoldering the connector to the second segment brings the strip back to life.

Making a custom cable

If you do not want to use our [premium jumper wires](#) to connect to the LED strip's input, it is possible to make a custom cable.

One option for making a custom cable is to cut off the unused output connector on the last LED strip in your chain. This can then be plugged into the input connector of the first LED strip. The wires on the output and input connectors are 20 AWG, which is too thick to easily use with our crimp pins and housings, but you could solder the wires to header pins.

Alternatively, you can get your own JST SM connectors and make a custom cable using those. The parts you would need to get are the SMP-03V-BC and the SHF-001T-0.8BS, which are described in the [SM Connector datasheet](#) from JST. These can be purchased from several places, and we got them from [Heilind](#). You will also need some 22–28 AWG [stranded wire](#) and a [wire stripper](#). We do not know of a great way to crimp wires onto the JST crimp pins, but we were able to successfully do it using our [narrower crimping tool](#) and [pliers](#). (With the wider crimping tool, it is hard to avoid crimping parts of the pin that should not be crimped.) Before crimping, use pliers to bend the outer set of tabs a little bit so that they can hold on to the insulation of the wire. This makes it easier to position the crimp pin and the wire. Next, you should be able to follow the instructions on the [crimping tool product page](#) to crimp the wire. After that, you will probably need to squeeze the crimp pin with pliers to get it to fit into the JST plug housing. On the other end of the cable you could make a custom connector using our [crimp pins](#) and [crimp connector housings](#), which will allow you to plug it directly into a breadboard or 0.1" header pins.

Current draw and voltage drop

Each RGB LED draws approximately 50 mA when it is set to full brightness and powered at 5 V. This means that for every 30 LEDs you turn on, your LED strip could be drawing as much as 1.5 A. Be sure to select a power source that can handle your strip's current requirements.

There is some resistance in the power connections between the LEDs, which means that the power voltage near the end of the strip will be less than the voltage at the start of the LED strip. As the voltage drops, RGB LEDs tend to look redder and draw less current. This voltage drop is proportional to the current through the strip, so it increases when the LEDs are set to a higher brightness.

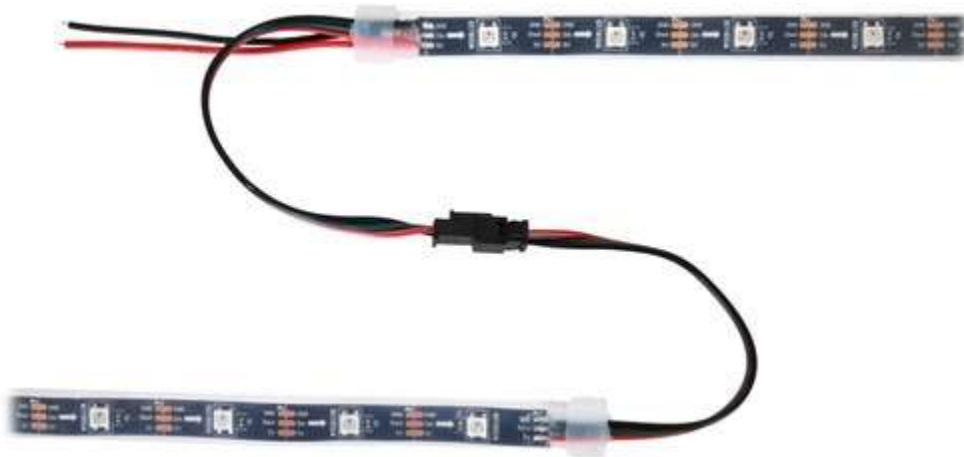
We tested the current draw and voltage drop of some LED strips by setting all the LEDs to full brightness, and these were the results:

- The 30 LED 1 m strip drew 1.5 A and had a voltage drop of 0.2 V.
- The 60 LED 2 m strip drew 2.9 A and had a voltage drop of 0.8 V.
- The 150 LED 5 m strip drew 4.1 A and had a voltage drop of 2.0 V.
- The 60 LED 1 m strip drew 3.0 A and had a voltage drop of 0.6 V.
- The 120 LED 2 m strip drew 4.7 A and had a voltage drop of 1.4 V.

The voltage drop was computed by measuring the voltage difference between ground and power on the input end of the strip, then doing the same measurement on the output end, and subtracting the two values.

Chaining

Multiple LED strips can be chained together by connecting input connectors to output connectors. When strips are chained this way, they can be controlled and powered as one continuous strip. Please note, however, that as chains get longer, the ends will get dimmer and redder due to the voltage drop across the strip. If this becomes an issue, you can chain the data lines while separately powering shorter subsections of the chain.



Two addressable RGB LED strips connected.

We recommend chains of LEDs powered from a single supply not exceed 180 total RGB LEDs. It is fine to make longer chains with connected data lines, but you should power each 180-LED section separately. If you are powering each section from a different power supply, you should cut the power wires between the sections so you do not short the output of two different power supplies together.

Cutting

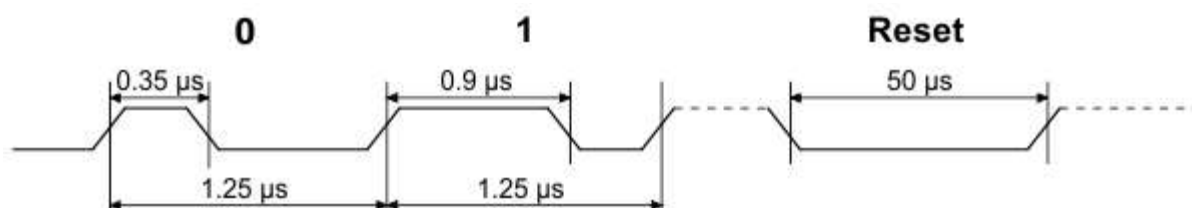
The LED strip is divided into segments, with each segment containing one RGB LED. The strip can be cut apart on the lines between each segment to separate it into usable shorter sections. The data connection is labeled **DO**, **Dout**, **DI**, or **Din**, the positive power connection is labeled **5V**, and the ground connection is labeled **GND**. Each LED in the picture below is at the center of its own segment; there are little scissors drawn on the PCB silkscreen where the segments can be cut.



Protocol

These LED strips are controlled by a simple, high-speed one-wire protocol on the input signal line. The protocol is documented in the [WS2812B datasheet](#) (266k pdf) and also below.

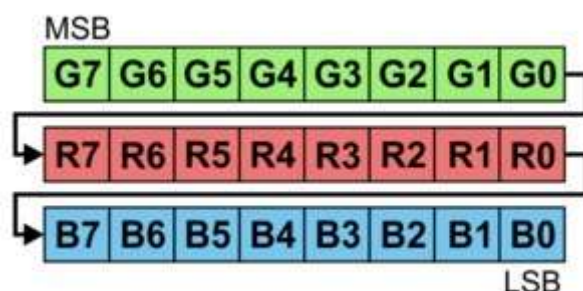
The default, idle state of the signal line is low. To update the LED colors, you need to transmit a series of high pulses on the signal line. Each high pulse encodes one bit: a short pulse ($0.35\ \mu\text{s}$) represents a zero, while a long pulse ($0.9\ \mu\text{s}$) represents a one. The time between consecutive rising edges should be $1.25\ \mu\text{s}$ (though in our tests, the strips worked with cycle times up to approximately $6\ \mu\text{s}$). After the bits are sent, the signal line should be held low for $50\ \mu\text{s}$ to send a reset command, which makes the new color data take effect (note: it is possible for low pulses as short as $6\ \mu\text{s}$ to trigger a reset). The pulse widths do not have to be precise: there is a threshold that determines whether the pulse is a 0 or a 1, and a wide range of pulse widths on both sides of the threshold will work.



SK6812/WS281x RGB data timing diagram.

The color of each LED is encoded as three LED brightness values, which must be sent in GRB (green-red-blue) order. Each brightness value is encoded as a series of 8 bits, with the most significant bit being transmitted

first, so each LED color takes 24 bits. The first color transmitted applies to the LED that is closest to the data input connector, while the second color transmitted applies to the next LED in the strip, and so on.



24 bits represent the color of one SK6812/WS281x LED in an addressable RGB LED strip.

To update all the LEDs in the strip, you should send all the colors at once with no pauses. If you send fewer colors than the number of LEDs on the strip, then some LEDs near the end of the strip will not be updated. For example, to update all 30 LEDs on a 1-meter strip, you would send 720 bits encoded as high pulses and then hold the signal line low for 50 μ s. If multiple strips are chained together with their data connectors, they can be treated as one longer strip and updated the same way (two chained 1-meter strips behave the same as one 2-meter strip).

Each RGB LED receives data on its data input line and passes data on to the next LED using its data output line. The high-speed protocol of the WS2812B allows for fast updates; our library for the Arduino below takes about 1.1 ms to update 30 LEDs, so it is possible to update 450 LEDs faster than 60 Hz. However, constant updates are not necessary; the LED strip can hold its state indefinitely as long as power remains connected.

Implementing the protocol on a microcontroller

Since this LED strip does **not** use a standard protocol, a software bit-banging approach is usually needed to control it from a microcontroller. Because of the sub-microsecond timing, the bit-banging code generally needs to be written in assembly or very carefully optimized C, and interrupts will need to be disabled while sending data to the LED strip. If the interrupts in your code are fast enough, they can be enabled during periods where the signal line is low.

Note: The minimum logic high threshold for the strip data line is 3.5 V, so you should use level-shifters if you want to control these strips from 3.3 V systems. In our tests, we were able to control them with 3.3 V signals from an mbed, but using the strip out of spec like this could lead to unexpected problems.

Sample code

To help you get started quickly, we provide sample code for these microcontroller platforms:

- [PololuLedStrip Arduino library](#) (also works with our Arduino-compatible [A-star modules](#))
- [Example AVR C code](#)
- [PololuLedStrip mbed library](#)

Additionally, the [Adafruit NeoPixel library](#) for Arduino should work with these strips since the NeoPixels are based on the WS2812B.

Comparison with TM1804 LED Strips

These WS2812B-based strips are similar in many ways to our older high-speed TM1804 LED strips (items [#2543](#), [#2544](#), and [#2545](#)). The WS2812B's timing parameters are very similar to those of the high-speed TM1804 LED strips, so you can use the same code to control either of them and you can chain one type to the other. However, the two types of strips have different, incompatible connectors, and the order of the red and green channels in the protocol is swapped: the TM1804 colors are sent in red-green-blue order while the WS2812B colors are sent in green-red-blue order.

The TM1804 is just an LED driver and it requires a separate RGB LED to be placed on the strip. Since the WS2812B combines the LED and the driver in a single package, it can be packed more densely, which is why we are able to offer strips with 60 LEDs per meter.

Unlike the TM1804 strips, these LED strips do not have an adhesive backing, but they do include mounting brackets as described above.

Comparison with APA102C LED Strips

Like the WS2812B, the APA102C used in [some of our newer LED strips](#) also combines an RGB LED and driver into a single 5050-size package, allowing them to be packed as densely as 144 LEDs per meter. However, while the WS2812B uses a one-wire control interface with strict timing requirements (timing requirements so strict that it is typically impractical to have interrupt-based events running on the controlling microcontroller while it is updating the WS2812B LEDs), the APA102C uses a standard SPI interface, with separate data and clock signals, that lets it accept a wide range of communication rates; the trade-off is that two I/O lines are required to control it instead of just one.

The APA102C provides a 5-bit color-independent brightness control that is not available on the WS2812B. This feature can be used to vary the intensity of each pixel without changing its color, and it enables much subtler variations at the low end of the LEDs' brightness range.

In addition, the APA102C uses a much higher PWM (pulse-width modulation) frequency for controlling each color channel—about 20 kHz, compared to around 400 Hz on the WS2812B. As a result, APA102C LEDs can be less prone to flickering when recorded with a camera and are more suited to applications like persistence-of-vision (POV) displays. (The color-independent brightness is modulated separately at about 600 Hz).

For further comparison of the ICs, see the [WS2812B datasheet](#) (266k pdf) and [APA102C datasheet](#) (1MB pdf).

While our WS2812B strips and APA102C strips are physically very similar, they are **not** functionally compatible with each other. The easiest way to tell them apart is to look at the strips' end connectors and the connections between each LED segment: WS2812B strips have three connections (power, data, and ground), while APA102C strips have four (power, clock, data, and ground). On strips with 30 LEDs/m, you can also check whether "WS2812B" or "APA-102C" is printed next to each LED.

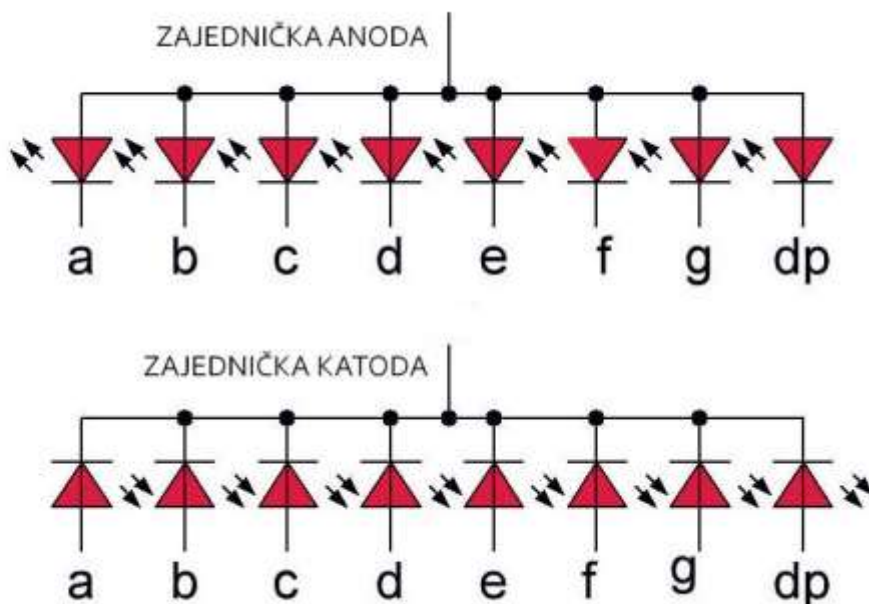
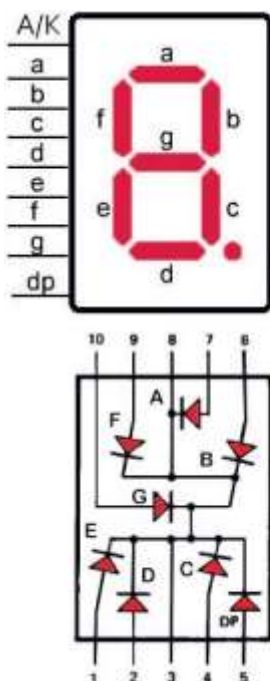
Brojke i slova

Ispisivanje teksta i cifara na jednostavan način

Igor S. RUŽIĆ, Svet kompjutera, 2017

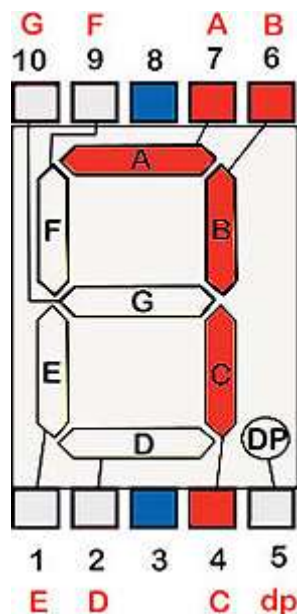
Sedmosegmentni displeji se vrlo često koriste u situacijama kada nam je potreban prikaz isključivo numeričkih informacija. Nisu skupi, malo troše i pružaju informaciju koja je jasno čitljiva i sa veće udaljenosti. Naziv duguju svom dizajnu u obliku broja osam, sastavljenom od sedam segmenata LE dioda. Tačnije, u pitanju je osam dioda, pošto se još jedna koristi za prikazivanje decimalne tačke, ukoliko je to potrebno.

Slično kolor LE diodama iz prošlog broja, sedmosegmentni displeji mogu biti građeni na osnovu zajedničke katode i zajedničke anode. Kod displeja sa zajedničkom anodom, sve anode pojedinačnih LED se ujedinjaju u jednu jedinstvenu anodu, to jest katodu, kod modela sa zajedničkom katodom. U pitanju je čisto stvar polariteta, a za korisnika je najvažnije da zna da se indikatori sa zajedničkom anodom priključuju na pozitivni napon, dok oni sa zajedničkom katodom idu na uzemljenje.



Na tržištu je moguće pronaći displeje koji prikazuju jedan, dva, tri ili četiri broja istovremeno. Mi ćemo za demonstraciju rada koristiti najprostije jednocifrene LED displeje sa zajedničkom anodom. Filozofija rada sa njima je veoma slična radu sa LE diodama. Pošto je u pitanju osam LED elemenata, potrebno je da ih sa džemper žicama priključimo na Arduino preko nezaobilaznih otpornika. Od prodavca smo dobili informaciju da radni napon segmenata iznosi 1,8 volti, dok je potrebna jačina struje u rasponu 15-20 miliampera. Formula koju smo koristi

u prethodnom nastavku nam daje sledeći rezultat: $R = (5\text{ V} - 1,8\text{ V}) / 0,015\text{ A} = 213\text{ oma}$, pa će otpornici od 220 oma biti dobar izbor za dati primer.

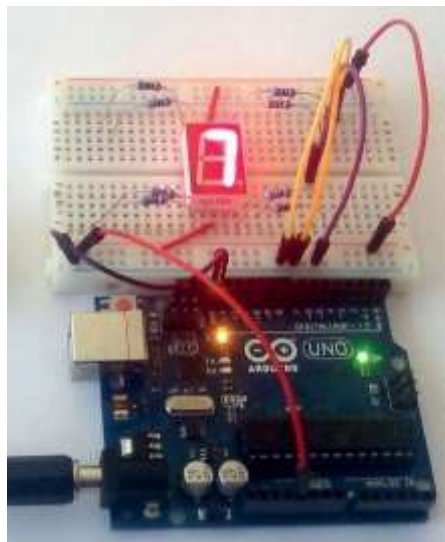


Naveli smo da je naš displej sa zajedničkom anodom. Da bismo na njemu prikazali neki broj, potrebno je da uključimo segmente koji reprezentuju taj broj, a u našem slučaju to znači da te pinove treba da povežemo sa uzemljenjem, a pinove 3 i 8 sa naponom od pet volti.

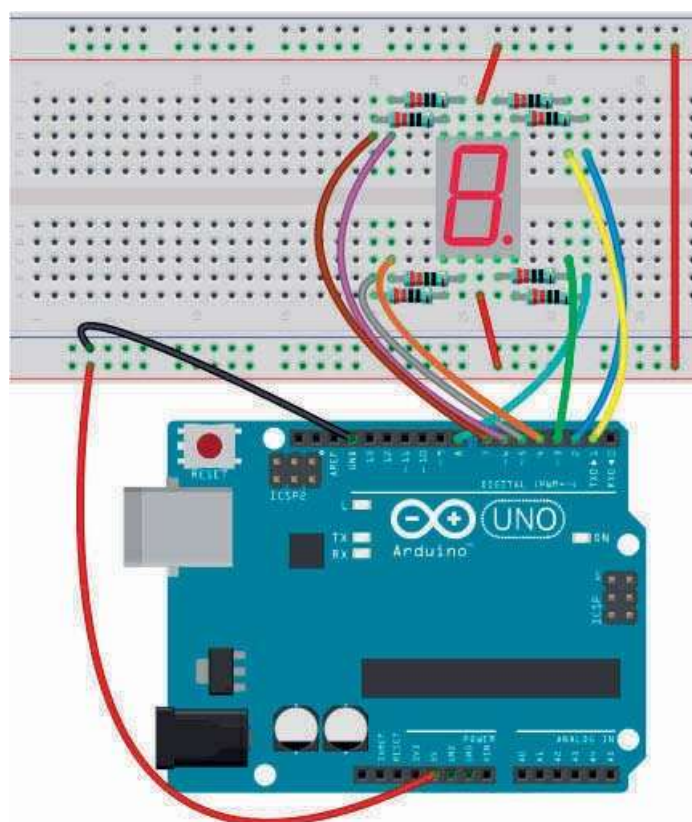
Pin	Displej	Arduino
7	segment A	D1
6	segment B	D2
8	zajednička anoda	5V
4	segment C	D3
5	DP	D8
2	segment D	D4
1	segment E	D5
3	zajednička anoda	5V
9	segment F	D6
10	segment G	D7

Hajde da to objasnimo na primeru broja 7, i to bez upotrebe mikrokontrolera (*Arduino* na slici služi jedino za napajanje displeja). Shema nam govori da je

potrebno dovesti uzemljenje do pinova 4, 6 i 7, što će izazvati pojavljivanje svetlosti u segmentima C, B i A.



Nije teško pogoditi da je za dinamičko prikazivanje brojeva potrebno povezati ove pinove sa izlazima Arduina.



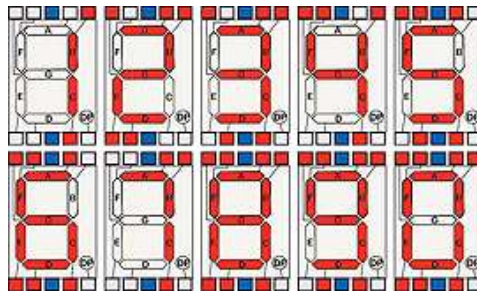
Pre nego što pređemo na pisanje skeča, da vidimo kako bismo na najjednostavniji način mogli da prikažemo broj 7 iz gornjeg primera, pod uslovom da smo displej povezali onako kako je to prikazano u tabeli.

```

void setup() {
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  digitalWrite(1, 0);
  digitalWrite(2, 0);
  digitalWrite(3, 0);
}

```

Pošto su nam potrebna samo tri segmenta (a, b, c) koji su povezani sa pinovima 1, 2 i 3 respektivno, njih prvo postavljamo u stanje izlaza, a zatim ta tri pina putem komande *digitalwrite* postavljamo u stanje logičke nule, što za uređaje sa zajedničkom anodom znači da ih uključujemo. Sledi primer prikazivanja brojeva u automatskom režimu:



```

//zajednicka anoda
byte brojevi[10][7] = {
  { 0, 0, 0, 0, 0, 0, 1 }, // = 0
  { 1, 0, 0, 1, 1, 1, 1 }, // = 1
  { 0, 0, 1, 0, 0, 1, 0 }, // = 2
  { 0, 0, 0, 0, 1, 1, 0 }, // = 3
  { 1, 0, 0, 1, 1, 0, 0 }, // = 4
  { 0, 1, 0, 0, 1, 0, 0 }, // = 5
  { 0, 1, 0, 0, 0, 0, 0 }, // = 6
  { 0, 0, 0, 1, 1, 1, 1 }, // = 7
  { 0, 0, 0, 0, 0, 0, 0 }, // = 8
  { 0, 0, 0, 0, 1, 0, 0 } // = 9
};

void setup() {
  //svi potrebni portovi su...
  for (byte x = 1; x < 8; x++) {
    pinMode(x, OUTPUT); // ...izlaznog tipa
    ++x;}
}

void loop() {
  for (byte x = 0; x < 10; x++) {

```

```

        delay(250);
        ispisi_broj(x);
    }
    delay(2000);
}

void ispisi_broj(byte broj) {
    byte pin = 1; //pocetni pin
    //petlja za proveru svih segmenata
    for (byte segment = 0; segment < 7; segment++) {
        digitalWrite(pin, brojevi[broj][segment]);
        pin++; //idemo na sledeci pin }
    }
}

```

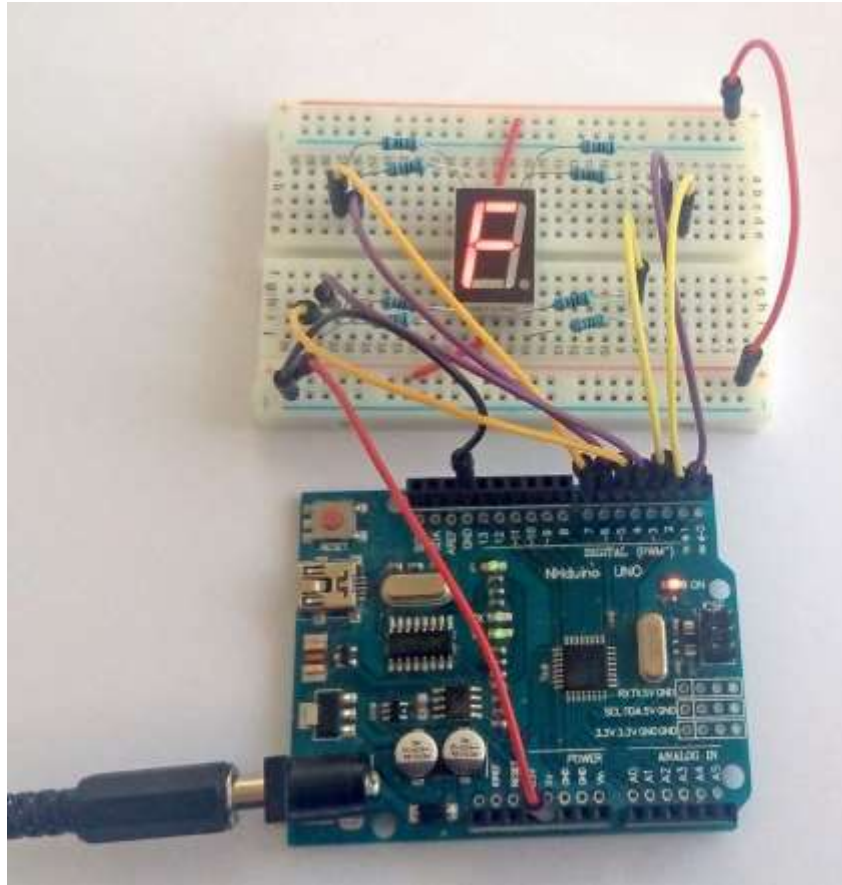
I ovde se radi o dosta jednostavnom kôdu, gde u okviru matrice *brojevi* definišemo izgled svih deset arapskih brojeva. U slučaju da koristimo displej sa zajedničkom katodom, potrebno je postaviti inverzne vrednosti:

```

//zajednicka katoda
byte brojevi[10][7] = {
    { 1, 1, 1, 1, 1, 1, 0 }, // = 0
    { 0, 1, 1, 0, 0, 0, 0 }, // = 1
    ...
    { 1, 1, 1, 0, 0, 1, 1 } // = 9
}

```

U delu *setup* uključujemo izlaz na svim korišćenim portovima. Petlja *loop* svakih 250 milisekundi ispisuje po jedan broj od 0 do 9 i zatim pravi pauzu od dve sekunde pre ponavljanja. Najvažniji deo skeča se nalazi u okviru funkcije *ispisi_broj*, koja kao argument prima broj koji treba ispisati. Vrednost varijable *pin* definiše početnu vrednost od koje počinjemo brojanje korišćenih pinova. Zatim se, kroz petlju od sedam koraka (varijabla *segment*) očitava red matrice koji odgovara poziciji našeg broja.



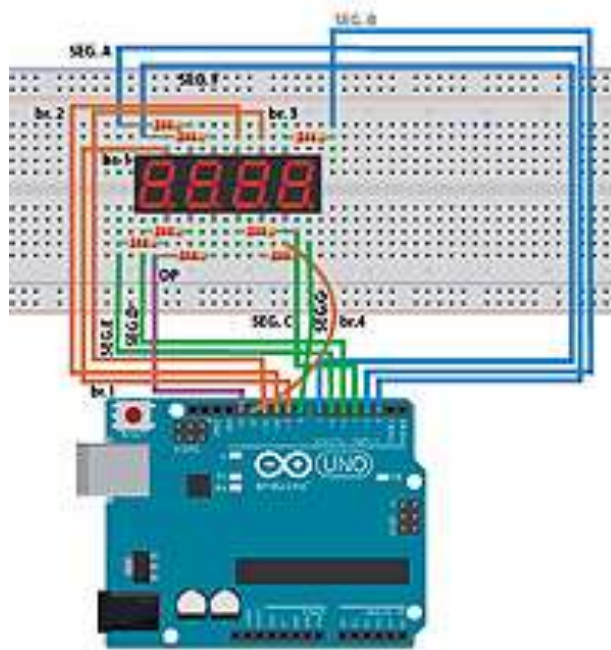
Uvođenje heksadecimalnih brojeva je vrlo jednostavna stvar. U pitanju je brojni sistem sa osnovom 16, koji pored deset cifara koristi i latinična slova od A do F. Da ne bismo zauzimali mesto kôdom koji bi bio vrlo sličan prethodnom, samo ćemo reći da dimenzije višedimenzionalnog niza brojeva treba postaviti na [16][7] i dodati sledeće redove

```
{ 0, 0, 0, 1, 0, 0, 0 }, // = a
{ 1, 1, 0, 0, 0, 0, 0 }, // = b
{ 1, 1, 1, 0, 0, 0, 0 }, // = c
{ 1, 0, 0, 0, 0, 1, 0 }, // = d
{ 0, 0, 1, 0, 0, 0, 0 }, // = e
{ 0, 1, 1, 1, 0, 0, 0 }, // = f
```

Naravno, prilikom pozivanja petlje za ispisivanje brojeva, umesto broja 10 koristimo vrednost 16.

Ovo nije jedini način na koji se mogu zapisati oblici brojeva. Umesto dvodimenzionalnog niza sa vrednostima 0 i 1, mogli smo koristiti zapisivanje u obliku sedmobitnog binarnog broja ili još prostije, decimalnim i heksadecimalnim brojem. Tako bi, recimo, broj 1 imao oblik B0110000, što je isto kao i decimalna vrednost 48 ili heksadecimalno 0x30. Vrednosti u takvom obliku možemo smestiti u jednodimenzionalni niz i pozivati ih preko indeksa pozicije. Moguće je, na kraju krajeva, formirati brojeve navođenjem sekvence

naredbi *digitalWrite* za svaki pojedinačni segment, kao što smo mi uradili prilikom ispisivanja broja 7 u prethodnom delu teksta.



Videli smo kako funkcioniše povezivanje jednocifrenog displeja, a sada ćemo videti kako treba povezati one sa više cifara. Oni su u stvari kombinovani od više pojedinačnih modula i svaki od njih ima po jednu zajedničku anodu ili katodu, dok su pinovi namenjeni segmentima zajednički za sve cifre zajedno. Možete se zapitati: „Kako onda kontrolišemo kojem od brojeva pripadaju koji segmenti?” Odgovor na to pitanje se krije u prikazu trenutno aktivnih brojeva u nekom kratkom vremenskom intervalu.

```
const int pinovi_segm[] = { 13,8,7,6,5,4,3,2 };
const int cifara= 4; //broj cifara na displeju
const int pinovi_cifre[cifara] = { 9,10,11,12 }; //pinovi za cifre
const int brojevi[10] = {252, 96, 218, 242, 102, 182, 62, 224, 254, 246};
//definicije brojeva 0-9
void setup(){ //inicijalizuj pinove na izlaz
  for (int i=0; i < 8; i++) {
    pinMode(pinovi_segm[i], OUTPUT);
  }
  for (int i=0; i < cifara; i++) {
    pinMode(pinovi_cifre[i], OUTPUT); }
}

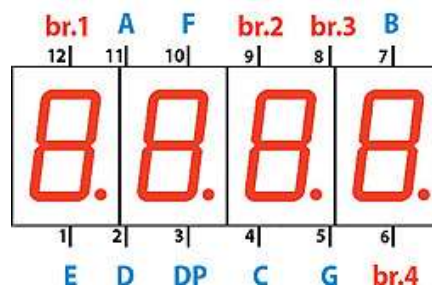
void loop(){ //ispisi brojeve 0-1023
  for (int x=0; x < 1024; x++) {
    prikazi_broj(x);
    delay (200);}
  delay (5000);
}

void prikazi_broj( int broj){
  if (broj == 0) { //ako je 0, ispisi
```

```

        ispis_broja( 0, cifara-1);}
    else {
//sve cetiri cifre
    for ( int cifra = cifara-1; cifra >= 0; cifra--) {
        if (broj > 0) {
            ispis_broja( broj % 10, cifra); //ostatak deljenja
sa 10
            broj = broj / 10; //sledeca cifra        }
        }
    }
}
void ispis_broja( int broj, int cifra){
//ispis broja na odredjenoj poziciji
digitalWrite( pinovi_cifre[cifra], HIGH ); //broj je vidljiv
//petlja za citanje segmenata za prikaz broja
for (int segment = 1; segment < 8; segment++) {
    boolean jedinica = bitRead(brojevi[broj], segment);
//jedinica = ! jedinica; //ako je zajednicka anoda!
digitalWrite( pinovi_seg[segment], jedinica);
}
delay(5); //drzi upaljeno 5 milisekundi
digitalWrite( pinovi_cifre[cifra], LOW ); //iskljuci
}

```



Koncentrišimo se samo na najvažnije delove kôda. Funkcija *prikazi_broj* kao argument prima broj koji treba prikazati. Ako broj nije nula, petlja *For* će da obradi sve postojeće cifre. Svaki pojedinačni broj koji treba prikazati dobijamo preko ostatka deljenja sa 10. Na primer, ukoliko 251 podelimo sa 10, dobijamo 25 i ostatak 1. Jedinicu zapisujemo na displej i prelazimo na deljenje 25 sa 10, iz čega dobijamo ostatak 5, da bi nam na kraju ostao sam broj 2. Dobijeni brojevi se šalju funkciji *ispis_broja*, koja uključuje vidljivost pina i zatim kroz petlju *For..Next* čita bitove kojima popunjava segmente na displeju. Ukoliko koristimo module sa zajedničkom anodom, tada ćemo skinuti komentar sa linije koja invertuje bitove u petlji. Posle toga, pravimo pauzu od pet milisekundi i zatim isključujemo prikaz cifre. Pošto se sve to odvija brzo, ljudsko oko ne može primetiti razliku i čini nam se da su sve cifre prikazane konstantno.

Postoji i dosta jednostavniji metod koji koristi biblioteku pod nazivom *SevSeg*, i u tom slučaju kôd izgleda ovako:

```
#include <SevSeg.h>
SevSeg segm_disp; //inicijalizujemo biblioteku
void setup() {
    byte cifara = 4;
    byte pinovi_segm[] = {2, 3, 4, 5, 6, 7, 8, 13};
    byte pinovi_cifre[] = {9, 10, 11, 12};
    segm_disp.begin(COMMON_CATHODE, cifara,
        pinovi_cifre, pinovi_segm);
    segm_disp.setBrightness(50);
}
void loop() {
    for (int x = 0; x < 1024; x++) {
        segm_disp.setNumber(x, 3);
        segm_disp.refreshDisplay();
        delay (200); }
    delay (5000);
}
```

Skrećemo pažnju na to da na pratećem crtežu većina otpornika ima vizuelni produžetak koji omogućuje njihovo pravilno razmeštanje na prototipskoj pločici.

Integralno kolo MAX7219



U prošlom poglavlju smo se upoznali sa pomeračkim integralnim kolom *74HC595*, koje nam je omogućilo da znatno povećamo broj svetlećih dioda priključenih na naš *Arduino*. Sada ćemo se upoznati sa još jednim zanimljivim čipom koji se veoma često koristi u radu sa segmentima i matricama svetlećih dioda.

Nožica	Oznaka	Funkcija
1	DIN	serijski ulaz podataka
2, 3, 5, 6, 7, 8, 10, 11	DIG n	8 linija za brojeve
12	LOAD	Load-Data Input
13	CLK	klok signal, maks. 10 MHz
14, 15, 16, 17, 20, 21, 22, 23	SEG x	sedam segmenata i DP (decimal point)
18	ISET	podešavanje intenziteta
19	Vcc	napajanje kola, 5V
24	DOUT	serijski izlaz
4, 9	GND	uzemljenje

Reč je o posebnom integrisanom kolu koje poseduje izlaze za rad sa 8 cifara na displeju, od kojih svaka ima osam segmenata (sedam plus decimalna tačka).



Na šematskom prikazu čipa ljubičastom bojom su označeni pinovi (DIN, LOAD i CLK) koji se priključuju na *Arduino* putem SPI interfejsa, dok je plavom bojom označen pin DOUT, koji služi za slanje podataka prema sledećem čipu. U takvim slučajevima, na svim čipovima se koriste zajednički signali za LOAD i CLK. Umesto da za svaki pin koristimo otpornik, ovde to činimo preko samo jednog otpornika koji se sa jedne strane povezuje sa nožicom 18 (I_{set}), a sa druge na liniju napajanja. Optimalna vrednost otpornika varira u zavisnosti od radnih karakteristika displeja. Sledeća tabela daje pregled nekih standardnih vrednosti (u kiloomima) :

Struja (mA)	Radni napon (V)		
	2,0	2,5	3,0
40	11,8	11,0	10,6
30	17,1	15,8	15,0
20	28,0	17,9	24,5

U projektima se uz *MAX7219* najčešće koriste i dva prateća kondenzatora, jedan od sto nanofarada i drugi od deset mikrofara, koji povezuju napon od pet volti sa uzemljenjem. Čip podržava samo displeje sa zajedničkom katodom.

Na linije zajedničkih katoda povezujemo linije DIG n, dok se segmenti svih brojeva vežu na magistralu koja vodi od pinova SEG x. Navodimo programski primer časovnika na osnovu gotovog modula sa osam cifara zasnovanih na čipu *MAX7219*, uz korišćenje biblioteke *LedControl*. Takvi moduli obično koštaju nešto malo više od jednog evra i nude mogućnost povezivanja više displeja u jedan niz.

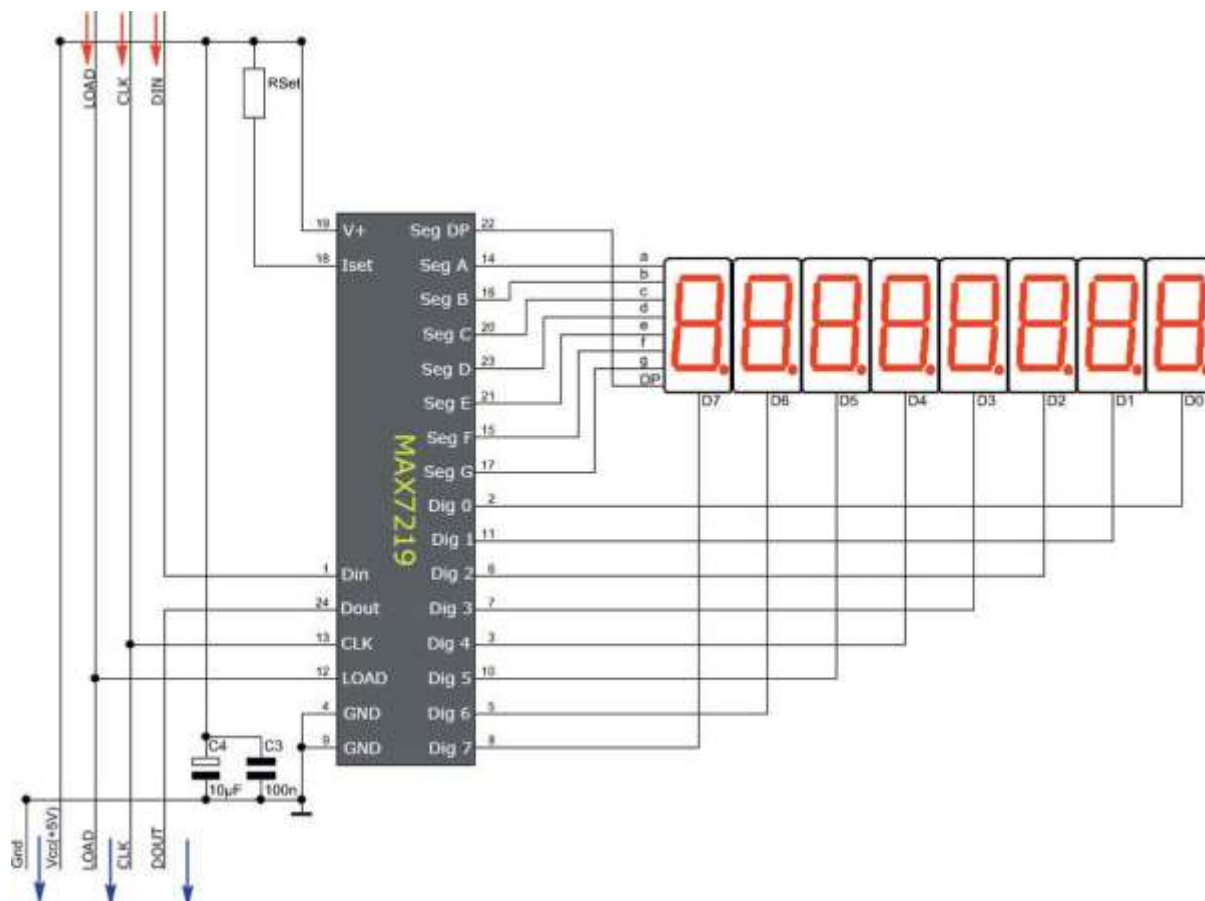
```
#include <LedControl.h>
//DIN, CLK, Load/CS, broj cifara
LedControl s7 = LedControl(10, 13, 11, 8);
int stoA, stoB, sekA, sekB, minA, minB, casA, casB;
void setup()
{
  pinMode(10, OUTPUT); //DIN
  pinMode(11, OUTPUT); //LOAD
  pinMode(13, OUTPUT); //CLK
  s7.shutdown(0, false); //aktiviramo displej
  Anuliranje(); //pocetne vrednosti na nulu
  s7.setIntensity(0,5); //intenzitet svetla
}
void loop()
{
  Casovnik(); //pocinjemo sa merenjem
}
void Casovnik()
{
  delay(10); //interval 1/100 sekunda
  s7.setDigit(0, 0, stoA++, false); //povecaj za 1
  if (stoA == 10) { //ako dodje do 10
    stoB++; //povecavamo drugu decimalu
    stoA = 0; //a prvu vracamo na 0
  }
  if (stoB < 9) { //druga decimala <9?
    s7.setDigit(0, 1, stoB, false); //ne, ispisi
  } else {
    stoB = 0; //da, ispisi 0
    s7.setDigit(0, 1, stoB, false);
    Sekunde(); //i prelazi na sekunde
  }
}
void Sekunde() {
  sekA++; //dodaj sekundu
  if (sekA == 10) { sekB++; sekA = 0; } //doslo do 10?
```



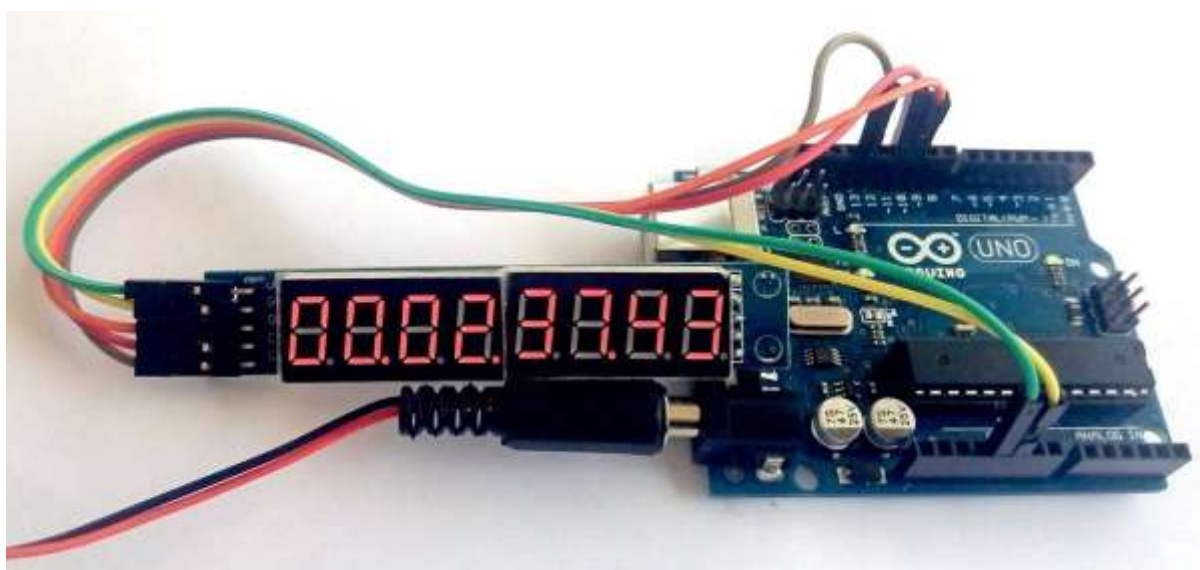
```

s7.setDigit(0, 2, sekA, true); //ispisi prvu decimalu
if (sekB < 6) { //druga decimala dosla do 60?
s7.setDigit(0, 3, sekB, false); //ne, ispisi
} else { sekB = 0; sekA = 0; //da, vrati na 0
s7.setDigit(0, 3, sekB, false); //ispisi...
Minute(); //prelazi na minute
}
}
void Minute(){
minA++; //sve isto kao gore
if (minA == 10) {minB++; minA = 0;}
s7.setDigit(0, 4, minA, true);
if (minB < 6) {
s7.setDigit(0, 5, minB, false);
} else { minB = 0; minA = 0;
s7.setDigit(0, 5, minB, false);
Casovi(); //prelazimo na casove
}
}
void Casovi(){
casA++;
if (casA == 10) {
casB++; casA = 0;}
s7.setDigit(0, 6, casA, true);
if (casB < 6) {
s7.setDigit(0, 7, casB, false);
} else { //doslo do kraja, sve na 0
casB = 0; casA = 0; Anuliranje();}
}
void Anuliranje(){ //na pocetne vrednosti
s7.setDigit(0, 0, 0, false);
for (int dig = 1; dig < s7.getDeviceCount(); dig++) {
s7.setDigit(0, dig, 0, (dig % 2 == 0) ? true : false);
}
}

```



Naredbom `LedControl(10, 13, 11, 8)` govorimo *Arduinu* da se radi o prikazu osam cifara, a da će biti korišćeni data pinovi 13(CLK), 11(LOAD) i 10(Din). Zatim, definišemo po dve varijable za praćenje vremenskih intervala na dve decimale, i to tako što je niža decimala označena sa A, dok je viša označena slovom B.

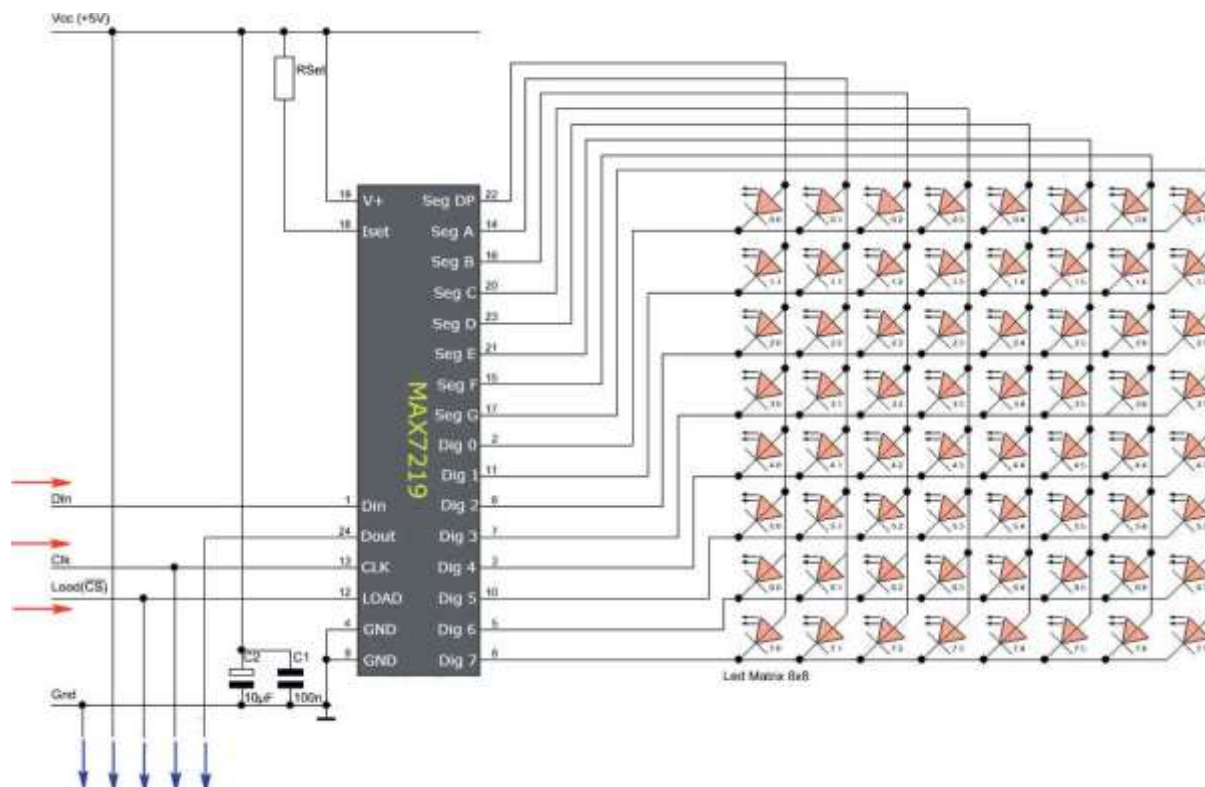


U `Setup()` bloku naredbom `shutdown` uključujemo displej, izvodeći ga iz režima uštede energije zadavanjem parametra `false`. Zatim vršimo inicijalizaciju ekrana i na kraju naredbom `setIntensity` postavljamo osvetljenost displeja (moguće

vrednosti 0-15). Imajte u vidu da svetliji ekran troši više struje.

Blok *loop()* poziva našu funkciju *Casovnik*, koja počinje sa brojanjem stotinki i kada dođe do 99, prelazi na sekunde, pa kada izbroji 60 sekundi na minute, zatim i na časove.

Na isti način se ovaj čip može koristiti i za prikazivanje informacija na matricama do 8x8 tačaka, a takve matrice se, opet, mogu povezati u kompleksnije blokove koji omogućavaju formiranje prezentacionih panela.



Korišćena je biblioteka *MaxMatrix*, koju je moguće preuzeti sa linka goo.gl/a1N4t4. Nju treba da postavimo unutar foldera C:\Users\ime_korisnika\Documents\Arduino\libraries.

```
#include <MaxMatrix.h>
#include <avr/pgmspace.h>
const unsigned char PROGMEM karakter[] = {
  3, 8, 0, 0, 0, 0, 0, //space
  1, 8, 95, 0, 0, 0, 0, //!
  ...
  4, 8, 62, 65, 65, 62, 0, //0
  3, 8, 66, 127, 64, 0, 0, //1
  4, 8, 98, 81, 73, 70, 0, //2
  ...
  4, 8, 126, 17, 17, 126, 0, //A
  4, 8, 127, 73, 73, 54, 0, //B
```

```

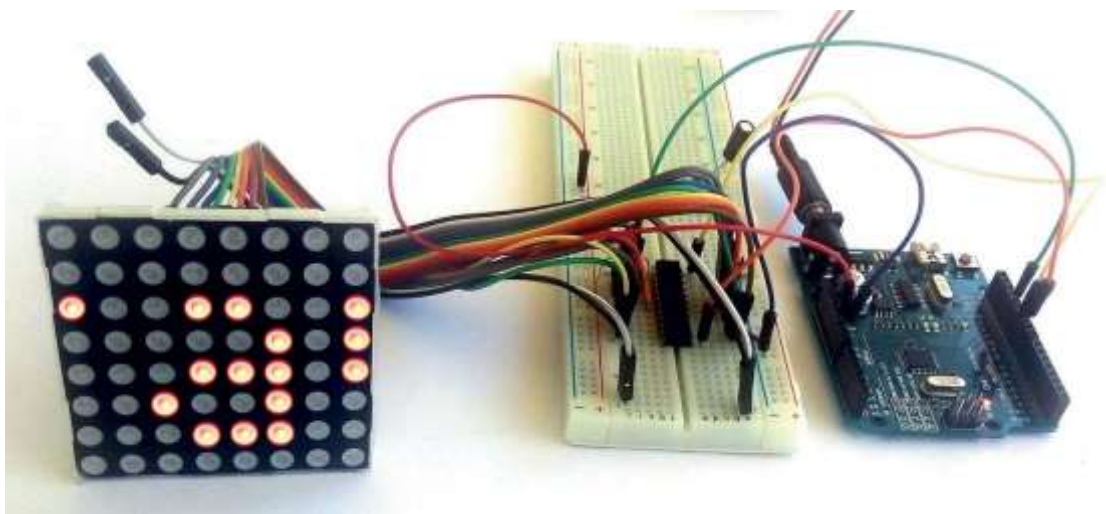
4, 8, 62, 65, 65, 34, 0, //C
...
5, 8, 99, 20, 8, 20, 99, //X
5, 8, 7, 8, 112, 8, 7, //Y
4, 8, 97, 81, 73, 71, 0, //Z
...
...
5, 8, 68, 40, 16, 40, 68, //x
4, 8, 156, 160, 160, 124, 0, //y
3, 8, 100, 84, 76, 0, 0, //z
...
};
char tekst[]="Veliki pozdrav za sve citaoce casopisa Svet
komputera!";
byte bafer[10]; //radni bafer
MaxMatrix m8x8(10, 11, 13, 1); //DIN, CS, CLK, broj modula
void setup(){
m8x8.init(); //inicijalizacija modula
m8x8.setIntensity(0); //jacina svetlosti 0-15
}
void loop(){
ispisi_tekst(tekst, 50);
}
void skrolovanje_levo(char znak, int brzina){
if (znak < 32) return; //ako nije vidljiv, izlazak
memcpy_P(bafer, karakter + 7*(znak-32), 7); //kopiraj
definiciju
m8x8.writeSprite(32, 0, bafer);
m8x8.setColumn(32 + bafer[0], 0);
for (int x=0; x<bafer[0]+1; x++)
{ //pomeramo bafer ulevo
delay(brzina); //pauza
m8x8.shiftLeft(false, false); //pomeri levo
}
}
void ispisi_tekst(char* znak, int brzina){
while (*znak != 0){ //dok ne dodjemo do kraja niza
skrolovanje_levo(*znak, brzina);
znak++; //sledeci znak
}
}
}

```

	K O L 0	K O L 1	K O L 2	K O L 3	K O L 4	K O L 5	K O L 6	K O L 7
RED 0	●	●	●	●	●	●	●	●
RED 1	●	●	●	●	●	●	●	●
RED 2	●	●	●	●	●	●	●	●
RED 3	●	●	●	●	●	●	●	●
RED 4	●	●	●	●	●	●	●	●
RED 5	●	●	●	●	●	●	●	●
RED 6	●	●	●	●	●	●	●	●
RED 7	●	●	●	●	●	●	●	●

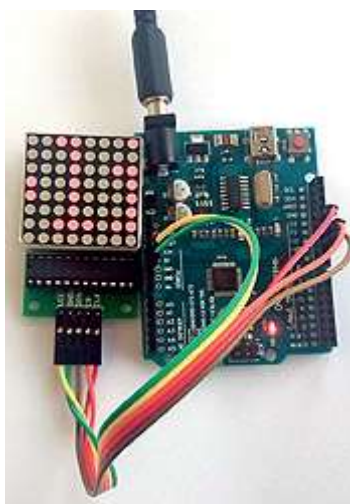
Varijabla *karakter* sadrži definiciju prvih 127 znakova skupa ASCII, i to tako što se za zapis definicije znaka koristi pet bajtova. Prva dva bajta svakog reda govore biblioteci stvarne dimenzije znaka. Recimo, broj 1 ima širinu od tri i visinu od osam piksela (dve nule s desna se ne računaju). Gornji levi ugao displeja predstavlja najniži bit prve kolone piksela, što važi i za svaku preostalu kolonu. Da bismo uštedeli na prostoru izbacili smo veći deo definicije karaktera, a kompletan skeč je moguće preuzeti sa adrese pastebin.com/u3Fse7iB.

Reč *PROGMEM* govori kompajleru da sadržaj varijable smesti u fleš-memoriju, pošto je fiksnog karaktera, a da bismo je koristili, potrebno je da u program uvrstimo fajl `<avr/pgmspace.h>`. Varijabla *tekst* sadrži niz znakova koji nameravamo da ispišemo na displeju, dok *bafer* ima funkciju pomoćne varijable.



U narednoj liniji kreiramo instancu objekta sa parametrima pinova korišćenih za DIN, CS i CLK signale i sa ukazivanjem broja korišćenih panela. Zatim, objekat *m8x8* u bloku *Setup* inicijalizujemo i postavljamo jačinu svetlosti na najmanju moguću. Petlja *loop()* sve vreme poziva funkciju *ispisi_tekst*, prosleđujući joj kao argumente zadani tekst i brzinu

skrolovanja (pomeranja). Najvažniji deo skeča predstavlja funkcija *skrolovanje_levo*, koja preuzima pojedinačne znakove i vrši njihovo pomeranje. Ako je ASCII vrednost poslanog znaka manja od 32, sledi izlazak iz funkcije zato što se ti karakteri ne mogu prikazati na ekranu. Zatim, od ASCII vrednosti oduzimamo 32 kako bismo lakše pronašli definiciju karaktera i to kopiramo u bafer. Slede funkcije *writeSprite* i *setColumn*, koje ispisuju znak iz bafera.



Postoje i matrice sačinjene od LE dioda u boji, za čije korišćenje je potrebno konstruisati sklop sa upotrebom tri integrisana kola *MAX7219* (po jedno za svaku od tri boje), ali zbog relativne kompleksnosti i nedostatka prostora, nismo ih obradili u okviru ovog teksta.

Vizuelizacija informacija

Rad sa ekranima je uvek spadao među najzanimljivije teme kada su u pitanju računari, pa je odličan za početak priče o hardverskom povezivanju *Arduina* sa spoljašnjim uređajima. Na tržištu postoji veliki broj različitih vrsta displeja za sve moguće namene, a mi imamo nameru da obradimo one jeftinije i, praktično svakome, pristupačne varijante.

Stari dobri Hitachi

Uređaji bazirani na kontroleru LCD ekrana *Hitachi HD44780* bili su veoma popularni među proizvođačima fiksnih telefona, štampača, faksova, aparata za kafu i drugih sličnih naprava, sve negde do pred kraj prethodne decenije kada su im na zamenu došli tada već dovoljno jeftini LCD/LED displeji sa svojim superiornijim mogućnostima. Ipak, i dalje postoji veliki broj primena u kojima su ovi ekrani više nego dobar izbor za prikazivanje informacija

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
CG RAM (I)				0@P`P												一タミxp
CG RAM (I)				!1AQa9												。アチムāq
CG RAM (I)				"2BRbr												「イツ×pθ
CG RAM (I)				#3CScs												」ウテモεω
CG RAM (I)				\$4DTdt												、イトハμω
CG RAM (I)				%5EUeu												・オナJεū
CG RAM (I)				&6FUfv												ヲカニヨρΣ
CG RAM (I)				'7GW9w												フキヌラgπ
CG RAM (I)				(8HXhx												イクネリヲ×
CG RAM (I))9IYiy												おケリル"y
CG RAM (I)				*:JZjz												エコハレJチ
CG RAM (I)				+;K[k<												オサヒロ°ヲ
CG RAM (I)				,<L¥1												ハシフワ4円
CG RAM (I)				-=M]m>												ユスヘンも÷
CG RAM (I)				.>N^n→												ヨセホ°ん
CG RAM (I)				/?O_o€												ッソマ°ō■

Prvo što treba znati je činjenica da je Hitachi HD44780 kontroler koji obrađuje informacije koje se potom prikazuju na displeju, a samih displeja ima prilično veliki broj. Najskromnija varijanta prikazuje osam karaktera u jednom redu, dok one izdašnije nude i po dvadeset karaktera u četiri reda. Postoje i varijante koje kombinovanjem više kontrolera prikazuju i do 80 znakova u osam redova, ali njihova cena prelazi granice naših interesovanja. Među ljubiteljima elektronike su najpopularniji modeli sa 16 karaktera u dva i 20 karaktera u četiri reda. Ovi prvi se mogu pazariti već po ceni od nekih 1,3 evra dok veći model košta nekih 3,3 evra (najniže cene na sajtu Ebay u vreme pisanja teksta, sa besplatnom dostavom). Ekrani se najčešće označavaju oznakom u kojoj se kombinuje broj redova i kolona. Tako, recimo, model 16 × 2 nosi oznaku 1602, dok je 20 × 4 model označen kao 2004. Moguće je kupiti displeje u različitim bojama pozadine. Najčešći su oni u plavoj, zelenoj i žutoj, dok se ređe mogu naći i crveni i beli.

Kontroler HD44780 podržava rad sa 208 unapred definisanih karaktera i pored toga ima mogućnost definisanja osam dodatnih znakova od strane korisnika. Svaki karakter je predstavljen matricom od 5 × 8 tačaka. Raspored karaktera u tabeli izgleda ovako:

- 0x00-0x07 korisnički definisani simboli
- 0x20-0x7F ASCII simboli
- 0xA0-0xFF nacionalno specifični simboli

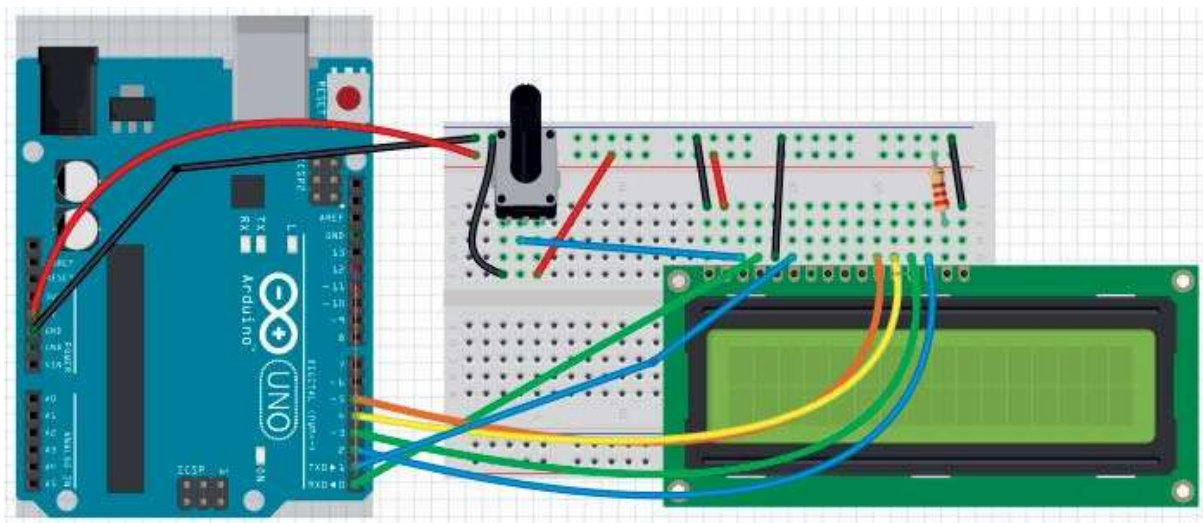
Prilikom kupovine preko interneta sa Dalekog istoka, praktično svaki put ćete dobiti displej koji pored standardnih ASCII simbola iz donjih sedam bita tabele dodatno sadrži simbole iz japanskog nacionalnog pisma. Kontroleri koji

podržavaju japanske znakove nose dodatnu oznaku UA00, dok oni sa oznakom UA02 sadrže karaktere iz zapadnoevropskih pisama i rusku ćirilicu. Postoje i varijante sa oznakom UBxx sa karakterima po želji naručioca, ali njih nije moguće naći u slobodnoj prodaji. Što se tiče korisničkih karaktera, njih je potrebno upisivati na uređaj svaki put nakon njegove inicijalizacije, pošto se oni smeštaju u CGRAM (Character Generator RAM), a ne u neki oblik fleš memorija. Na lokaciji goo.gl/sz9EYU se nalazi stranica za generisanje korisničkih znakova za ovu vrstu ekrana.

Kod modela sa oznakom 1602 svaki od dva reda ima po 40 bajtova za podatke. U prvoj liniji karakteri su smešteni od lokacije 0x80 do 0xA8 a u drugoj od 0xC0 0xE8. Unutar tog područja je moguće pomerati pokazivač na prvi karakter, tako da se postiže efekat horizontalnog skrolovanja. Komplikacije u organizaciji video memorije nastaju kao posledica činjenice da je HD44780 prvobitno kreiran za korišćenje u režimu 20×4 karaktera.

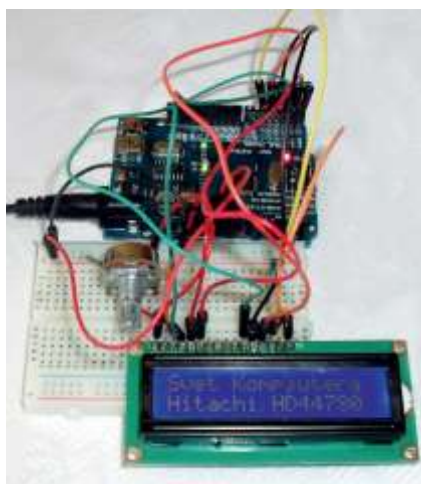
Prilikom kupovine displeja obično dobijate nožice sa pinovima koje je potrebno zalemiti u predviđene otvore. Funkcije tih pinova su sledeće (oznake se često razlikuju kod različitih modela):

1 Vss	- uzemljenje
2 Vdd	+5V
3 VO	kontrast
4 RS	izbor registra
5 RW	čitanje/pisanje
6 Eklok	sinhronizacija upisivanja
7-14	D0 – D7magistrala podataka



Prva dva pina (kao i ona dva poslednja vezana za osvetljenje) imaju funkciju napajanja uređaja i o njima se nema šta naročito reći. Pin broj 3 je zadužen za kontrolu kontrasta ekrana i najčešće je povezan sa potencijetrom. Pin broj

četiri se naziva RS (Register Select) i preko njega definišemo da li je kontroler u modusu prenosa komandi ili podataka (ukoliko je na ovom pinu prisutno stanje logičke nule onda kontroler prima komande, a u suprotnom operiše sa podacima). Pin broj 5 nosi naziv RW i njegova je funkcija da odredi da li nameravamo da podatke upisujemo na ekran ili da ih od tuda čitamo. U ogromnom broju slučajeva ovaj pin se stavlja u stanje logičke nule povezivanjem sa uzemljenjem, što znači da će se ekran koristiti samo za ispisivanje. Pin broj šest je namenjen vremenskoj sinhronizaciji prenosa podataka.



Ekрани bazirani na kontroleru HD44780 imaju mogućnost da se povezuju kako preko svih osam linija za podatke, tako i preko samo četiri gornje (D4-D7) data linije. Iako logika govori da korišćenje dvostruko šire magistrale dovodi do dvostruko većeg prenosa podataka, to u praksi nije tako. Razlike su skoro zanemarljive, a nama u radu sa 4-bitnim modusom ostaju četiri neiskorišćena Arduino pina koji se mogu upotrebiti mnogo pametnije. Da ne govorimo o smanjenju broja kablova koji kvare preglednost projekta. Kada se koriste samo četiri gornja bita magistrale podataka, svaki pojedinačni bajt se prenosi iz dva puta, odnosno u formi nibla (naziv za polovinu bajta).

Da bismo povezali displej sa Arduinoom Uno, koristićemo podatke iz sledeće tabele:

LCD Pin	Funkcija	LCD pina	Arduino Uno
1	GND		GND
2	5V		5V
3	Kontrast		GND
4	RS (Register Select)		Digital0
5	RW (Read/Write)		GND
6	E (Clock Enable)		Digital1

7	D0-	
8	D1-	
9	D2-	
10	D3-	
11	D4	Digital2
12	D5	Digital3
13	D6	Digital4
14	D7	Digital5
15	Anoda (osvetljenje)	5V
16	Katoda (osvetljenje)	GND

Tamo gde su ukazane vrednosti GND i 5V napon dovodimo sa linija napajanja prototipske ploče. Kada sve odradimo kako treba, preostaje nam još da dodamo malo programskog kôda za ispisivanje poruke na ekranu:

```
#include <LiquidCrystal.h> // biblioteka LCD
// definišemo pinove koje koristimo
LiquidCrystal lcd(0, 1, 2, 3, 4, 5);
void setup() {
  lcd.begin(16, 2); //broj stubaca i redova
  lcd.print(„Svet kompjutera”);
  lcd.setCursor(0, 1);
  lcd.print(„Hitachi HD44780”);
}
void loop() {}
```

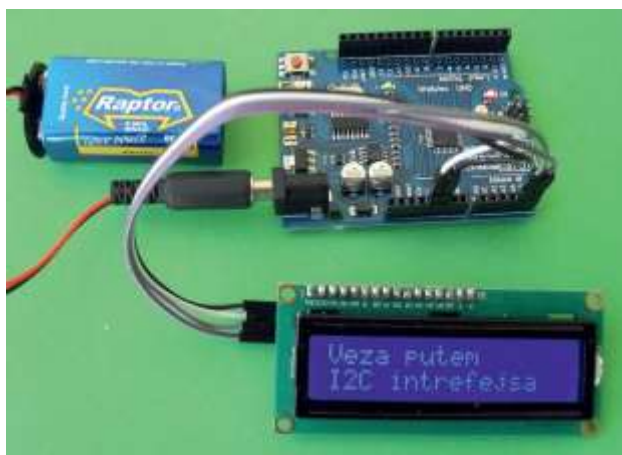
Zahvaljujući biblioteci LiquidCrystal, izbegnuto je žongliranje sa komandama i registrima LCD kontrolera, pa je rad sa ekranom sasvim jednostavan. Prva linija kôda našem skeču dodaje pomenutu biblioteku, dok u sledećoj liniji vršimo inicijalizaciju interfejsa sa vrednostima iz naše tabele sa sledećim rasporedom parametara:

```
LiquidCrystal lcd(RS, Enable, D4, D5, D6, D7);
```

Kao argumente upisujemo brojeve Arduino data pinova povezanih sa pinovima LCD kontrolera ukazanim u primeru. Funkcijom begin vršimo inicijalizaciju rezolucije displeja, dok funkcija print ispisuje sadržaj na ekran. Kao što samo ime kaže setCursor postavlja kursor na ukazujuću poziciju. Prvo se zadaje broj kolone, a zatim i broj reda. Biblioteka LiquidCrystal sadrži dvadesetak funkcija.

Može još jednostavnije

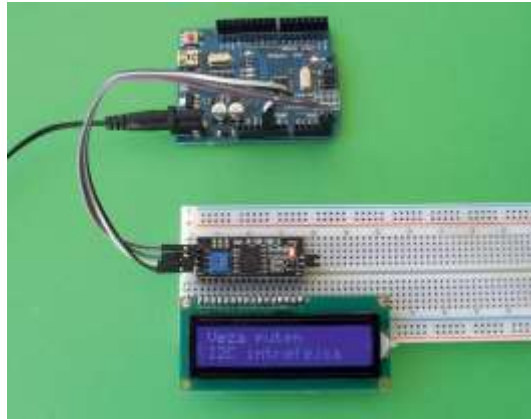
Složit ćemo se da prethodni primer nije težak ni za one koji se samo površno razumeju u elektroniku. Povezivanje žica na prototipskoj ploči ne traje više od pet minuta i od dodatnih delova su nam potrebni samo jedan potencijometar (od pet do deset kilooma) i jedan otpornik veličine 470 oma (može i neki drugi iz tog opsega), sa namenom da štiti pozadinsko osvetljenje.



Međutim, postoji još jednostavniji metod povezivanja ove vrste displeja na naš *Arduino*. Za tu svrhu je potrebno da nabavimo dodatni modul baziran na čipu *PCF8574* (*port ekspander*, SLIKA 6) koji će nam omogućiti povezivanje uređaja putem I2C standarda za serijski prenos podataka. Cena ovih modula je nešto manja od jednog evra, što i nije mnogo novca u odnosu na komfor koji nude. Za povezivanje su nam potrebne samo četiri žice koje spajamo na sledeći način:

Arduino	I2C modul
Gnd	Gnd
5V	Vcc
Analog 4	SDA
Analog5	SCL

SDA je linija I2C interfejsa za prenos podataka, dok SCL označava liniju za sinhronizaciju (klok). Bazično postoje dva načina spajanja LCD displeja sa modulom. Prvi je da na prototipsku ploču postavimo oba elementa tako da im se iglice poklapaju, dok je drugi (i dosta češći) način da jednostavno zalemimo ova dva uređaja.



```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x20, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
void setup() {
  lcd.begin(16,2);
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print(„Veza putem”);
  lcd.setCursor(0, 1);
  lcd.print(„I2C intrefejsa”);
}
void loop()
{}
```

Kôd je sličan prethodnom primeru, sa tom razlikom da ovde koristimo biblioteku Wire.h za komunikaciju putem I2C protokola, kao i biblioteku LiquidCrystal_I2C.h za podršku ispisivanju na displeje ove vrste. Ovde treba napomenuti da je najpametnije rešenje korišćenje biblioteke pod nazivom NewliquidCrystal (goo.gl/hV10sG), koja ima niz prednosti nad originalnom bibliotekom. Ako u radu sa displejem budemo dobijali svakoake greške, treba pokušati sa korišćenjem ove biblioteke i verovatno će problemi biti rešeni. Prvi parametar konstruktora pod nazivom lcd ima vrednost 0x20 i to će uglavnom funkcionisati ispravno. Međutim, u pojedinim slučajevima će biti potrebno postaviti drugu vrednost. Najlakši način da utvrdimo I2C adresu uređaja je da pokrenemo sledeći program i otvorimo prozor serijskog monitora u Arduino IDE.

```
#include <Wire.h>
void setup() {
  Serial.begin (9600);
  Wire.begin();
  for (int adr = 8; adr < 120; adr++) {
    Wire.beginTransmission (adr);
```



```

if (Wire.endTransmission () == 0) {
  Serial.print („Adresa: „);
  Serial.print („0x");
  Serial.println (adr, HEX);
}
}
}
void loop() {}

```

Ostali argumenti će najčešće biti onakvi kakvi su prikazani u našem primeru.

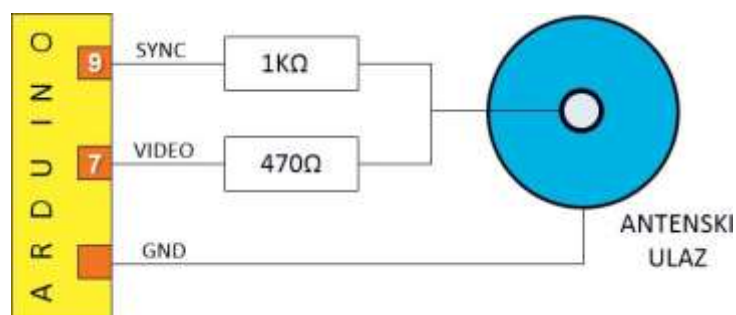
Ukoliko vaš I2C adapter na sebi ima tri polja sa oznakama A0, A1 i A2, to znači da lemljenjem ovih izvoda možete fiksirati adresu interfejsa i to po sledećoj tabeli:

Adresa	A0	A1	A2
0x20000			
0x21100			
0x22010			
0x23110			
0x24001			
0x25101			
0x26011			
0x27111			

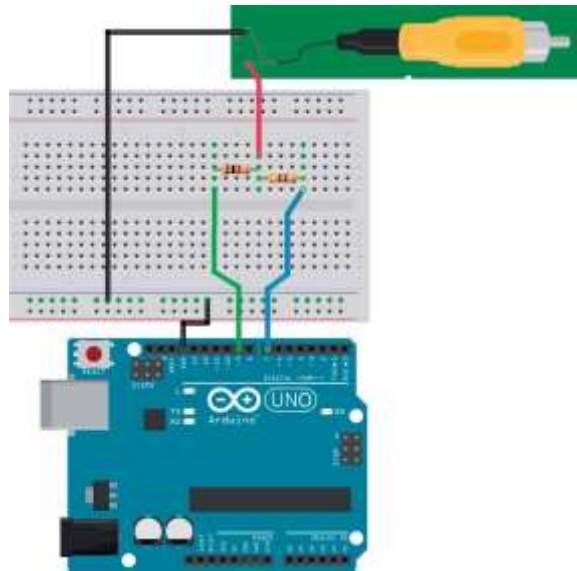
Često će se desiti da nakon prvog startovanja program ne ispisuje ništa na ekranu. Zapravo, on ispisuje, ali mi to ne primećujemo, pošto potencijometar za kontrolu kontrasta nije podešen kako treba.

Slika na TV

Prvoborci računarske revolucije sa početka osamdesetih godina dvadesetog veka svakako se sećaju računara ZX80 i ZX81, kao i domaće Galaksije koji su sliku generisali uz pomoć sirove snage mikroprocesora koja je radila posao umesto namenskog hardvera.



Isti princip je iskorišćen prilikom kreiranja biblioteke za Arduino i od nas se očekuje da, osim same biblioteke, napravimo fizički interfejs između televizora i Arduina. Izrada interfejsa je više nego jednostavna. Potrebna su nam samo dva otpornika, kao i jedan koaksijalni kabl za priključivanje antene sa konektorom za kompozitni signal. Alternativno, moguće je pre otpornika postaviti po jednu diodu i na taj način dodatno osigurati Arduino od struje sa TV uređaja.



Kod Arduino Una se za signal sinhronizacije koristi digitalni pin 9, dok signal videa dolazi preko digitalnog pina 7. Kod modela Mega2560 za istu namenu se, respektivno, koriste digitalni pinovi 11 i 7. Standardna rezolucija definisana bibliotekom iznosi 128×96 piksela. U slučaju da se u projektu koristi mikrokontroler sa kilobajtom RAM, potrebno je koristiti naredbu

```
TV.begin(_PAL,128,56)
```

zbog manjeg zauzeća memorije.

Poslovi ovakve vrste zahtevaju maksimalnu efikasnost, pa nije iznenađujuće kada u izvornom kôdu biblioteke nailazimo na optimizovane rutine pisane u asemblerskom jeziku.

Korišćenje biblioteke ćemo ilustrovati jednostavnim primerom:

```
#include <TVout.h>
#include <fontALL.h>
TVout TV;
void setup ( ) {
TV.select_font(font6x8);
TV.begin(_PAL);
```

```

}
void loop () {
  for(int x=0; x<5; x++) {
    TV.print(10,x*10,"Svet Kompjutera");
    TV.delay(50);
    TV.clear_screen();
  }
}

```

Prve dve #include direktive su namenjene dodavanju biblioteka TVout i fontALL gde ova potonja daje mogućnost upotrebe rasterskih fontova različitih dimenzija. U okviru Setup bloka izabiramo font 6 × 8 piksela i dajemo do znanja da ćemo koristiti signal po PAL standardu koji je karakterističan za Evropu (SAD i Japan koriste standard NTSC).

Sledi petlja za ispisivanje teksta. Biblioteka sadrži solidan skup funkcija za rad sa monohromatskom grafikom i tekstom, pri tome dajući utisak kao da pred sobom imamo stari osmobitni računar. Pomoću nje je napisano desetak video igara koje na lep način demonstriraju kako je moguće ostvariti i ono što se na prvi mah čini nemogućim. Uz malo više zalaganja i nešto dodatnog hardvera je moguće implementirati i rad sa bojama.

Vizuelizacija informacija

Nastavljamo temu iz prošlog broja i upoznajemo se sa ostalim načinima za prikazivanje informacije upotrebom različitih tipova displeja za platformu Arduino.

Nokia do Tokia



Oni koji se sećaju vremena dolaska mobilne telefonije u naše krajeve svakako će se setiti i modela Nokia 5110 koji je uz Nokia 3210 (posle i 3310) predstavljao najpopularniji telefon na tržištu. Ovi telefoni su za prikazivanje podataka koristili Filipsov displej PCD8544 koji generiše crno-belu sliku rezolucije 84 × 48 tačaka, što je dovoljno za prikazivanje šest redova od po

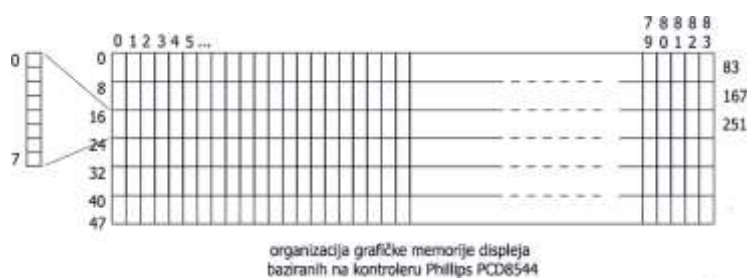
četrnaest karaktera sa matricom 5×8 tačaka. Danas se ovakvi displeji, montirani na module pogodne za priključivanje na Arduino, mogu nabaviti već po ceni od oko 1,5 do dva evra, što ih čini odličnim izborom za projekte gde je potreban prikaz jednostavnije monohromne grafike, animacije ili tekstualnih podataka uz minimalnu potrošnju energije (oko 200 mikroampera).

Prvo što treba znati u radu sa ovom vrstom ekrana je činjenica da se oni proizvode sa nekoliko različitih rasporeda pinova konektora. Ovde navodimo one najčešće:

Najcesci rasporedi konektora za displeje PCD8544

Pin	Tip1	Tip2	Tip3
1	RST	VCC	GND
2	CE	GND	VCC
3	DC	SCE	CLK
4	DIN	RST	DIN
5	CLK	D/C	D/C
6	VCC	DN(MOSI)	CS
7	LIGHT	SCLK	RST
8	GND	LED	LED

Osnovna razlika među njima je ta da kod jednih (mahom sa crvenom štampanom pločicom) na pin napajanja pozadinskog osvetljenja dovodimo signal GND, dok kod drugih (plavih) tu povezujemo napajanje od 3,3 volta.



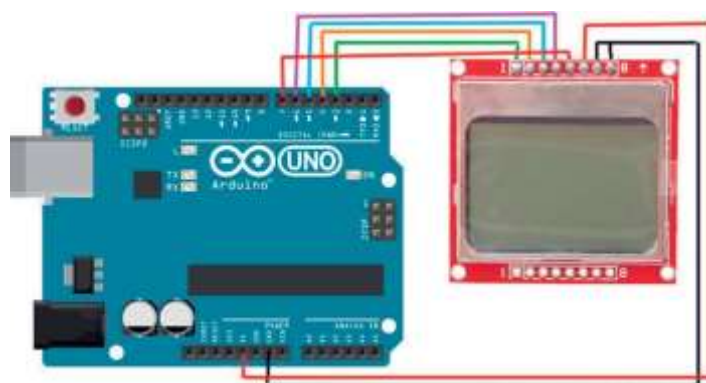
Da bismo pojednostavili stvar, kreirali smo tabelu koja prikazuje različite oznake pinova sa njihovim funkcijama. Vidimo da su funkcije slične sa onima koje smo koristili kod displeja 1602.

Oznaka	Funkcija
VCC	napajanje modula
GND	uzemljenje
CE, SCE, CS	aktiviranje cipa (Chip Enable)

RST	restart
DC, D/C	rezim: Data/Command
DIN, SDIN, DN(MOSI)	ulazni kanal podataka (Data IN)
CLK, SCLK, SCLC	klok sinhronizacija prenosa
LIGHT, LED, BL	napajanje pozad. osvetljenja

Preostaje nam da džamperima povežemo ekran sa linijama napajanja i data pinovima Arduina. Ukoliko želimo da oslobodimo jednu od linija za prenos podataka, moguće je povezati ekranski pin RST sa RESET pinom na Arduino (obratiti pažnju na napomenu u donjem kodu). Isto tako, ukoliko ne nameravamo da koristimo pozadinsko osvetljenje, jednostavno isključimo žicu koja dovodi uzemljenje na pin LIGHT.

Arduino	LCD
D3	1 (RST)
D4	2 (CE)
D5	3 (DC)
D6	4 (DIN)
D7/RESET	5 (CLK)
5V	6 (VCC)
GND	7 (LIGHT)
GND	8 (GND)

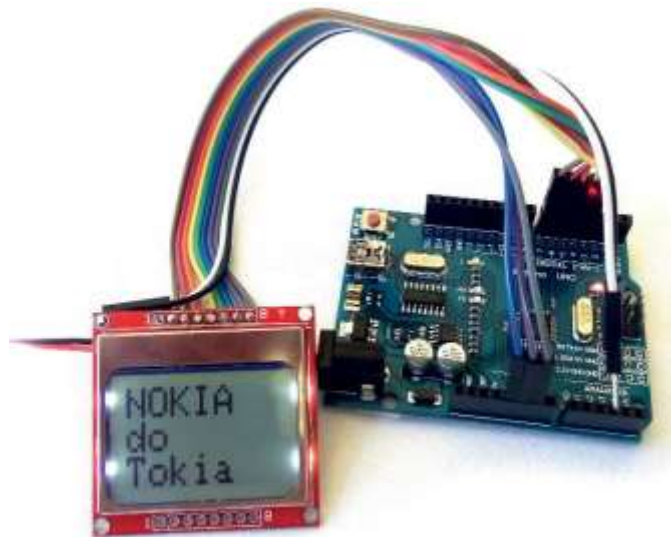


Postoji nekoliko različitih programskih biblioteka koje podržavaju ovu vrstu ekrana. Najpopularnija je (mada i dosta spora) ona koju potpisuje u Arduino svetu poznata firma Adafruit i uz nju je potrebno instalirati biblioteku pod nazivom Adafruit GFX Library. Funkcijom `Adafruit_PCD8544(3, 4, 5, 6, 7)` određujemo koji pinovi displeja su korišćeni za povezivanje sa Arduino i to u sledećem poretку: `Adafruit_PCD8544(SCK, SDIN, D/C, CS, RES)`.

Jedan primer korišćenja ove biblioteke bi izgledao od prilike ovako:

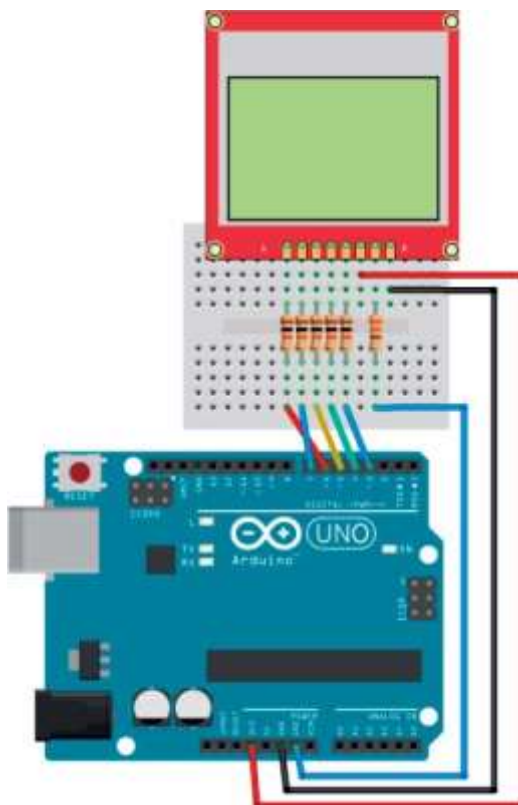
```
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
Adafruit_PCD8544 display = Adafruit_PCD8544(3, 4, 5, 6, 7);
// stavljamo 0 umesto 7 ako koristimo pin RESET
void setup() {
  display.begin(); // inicijalizacija
  display.clearDisplay();
  display.display();
  display.setContrast(45); // kontrast ekrana

  delay(1000);
  display.setTextSize(2); // velicina teksta
  display.setTextColor(BLACK); // boja
  display.setCursor(0,0); // pozicija
  display.println("NOKIA\nDo\nTokia");
  display.display();
}
void loop() {
}
```



Početak programa je posvećen inicijalizaciji ekrana, a posle toga slede komande vezane za ispis teksta. Obavezno je navođenje funkcije `display()` nakon komandi za ispisivanje bilo kakvog sadržaja. Primećujemo i funkciju `setContrast()` čiji je zadatak da programski podesi oštrinu slike, koja inicijalno

Kada u narednom poglavlju budemo govorili o povezivanju TFT displeja putem hardverskog SPI interfejsa, videćemo kako funkcioniše mehanizam koji možemo primeniti i kod ekrana sa kontrolerom PCD8544. Za to će nam biti potrebna biblioteka koju možemo preuzeti sa lokacije goo.gl/IHCGwK. Ukoliko vam je potrebna veća brzina, svakako uzmite u razmatranje tu varijantu.

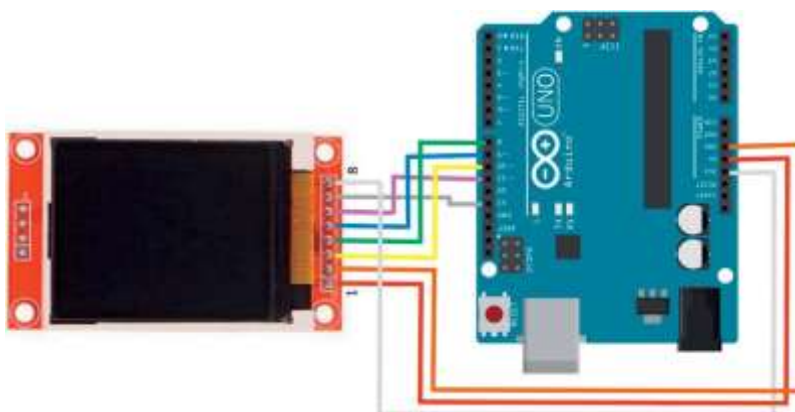


155 Arduino hrestomatija za digitalne umetnike i dizajnere

Boje su u nama

Kada je u pitanju Arduino platforma, na raspolaganju nam je desetak modula sa TFT displejima u boji različitih dimenzija, rezolucija i cena koje se (u zavisnosti od modela) kreću između tri i petnaest evra.

Mi ćemo u ovom primeru koristiti modul: 1.8TFT SPI 128*160 v1.1, koji je baziran na displej kontroleru pod nazivom ST7735R. Osim modela koje pokreće taj čip, često se sreću i oni koji su zasnovani na kontroleru ILI9163.



Generalno, ovaj prvi je bolji izbor zbog bolje podrške po pitanju programskih biblioteka. Arduino IDE ima ugrađenu biblioteku za ispis na TFT ekranima, ali je ona jako spora, pa se preporučuje izbor alternativnih biblioteka koje donose ubrzanje od dva do deset puta. Moduli vrlo često osim TFT displeja koji prikazuje 262144 boje na sebi imaju i slot za standardne SD memorijske kartice. Kao što je obično slučaj sa vrlo jeftinim stvarima, o njima nećete od prodavca dobiti mnogo informacija, pa ćete morati da se snalazite kako znate i umete. Situacija je slična kao kod opisanih Nokia displeja, pa i ovde imamo više modela sa različitom bojom, rasporedom elemenata i izvoda.

Najčešći su modeli sa osam pinova (plus četiri dodatna za SD konektor), dok je moguće pronaći i one sa deset, jedanaest ili čak šesnaest pinova. U našem konkretnom slučaju raspored izgleda ovako:

pin displej	funkcija
1 VCC	napajanje modula
2 GND	uzemljenje
3 CS	izbor uredjaja
4 RESET	reset
5 A0	linija komandi
6 SDA	signal podataka
7 SCK	klok signal
8 LED	napajanje svetla

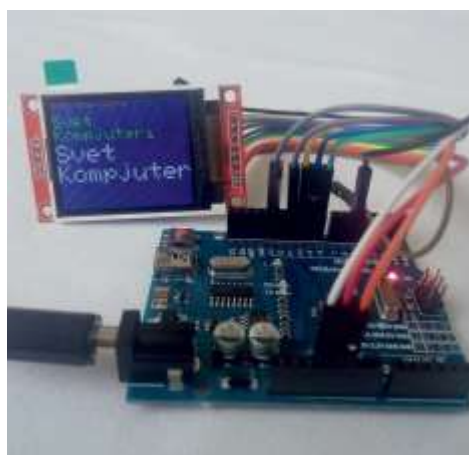
Na zadnjoj stranici modula nalazi se mesto označeno sa J1 i koje predstavlja selektor napona sa kojim će uređaj funkcionisati. Ukoliko dva izvoda nisu spojena lemom, on će raditi na pet volti, dok u slučaju spajanja radi na 3,3 volta. Pri ovome morate biti oprezni jer ukoliko koristite displej sa spojenim konektorom J1 na naponu od pet volti, on će pregoreti. Napajanje osvetljenja displeja obavezno ide na 3,3 volta (ili na GND kod nekih modela). Inače, različiti naponi su stvar koja izaziva glavobolju. Zbog nedostatka dokumentacije smo prisiljeni da informacije tražimo po internetu. Na dosta mesta ćemo pročitati da je rad sa naponom od pet volti (iako ga po specifikaciji podržava) štetan po sam uređaj i da doprinosi njegovom preranom kvarenju. Zato se preporučuje postavljanje otpornika na linijama prenosa podataka. Mi ćemo u našem primeru, jednostavnosti radi, koristiti direktno povezivanje bez otpornika, ali u slučaju realizacije nekog praktičnog projekta, ne košta nas mnogo da ih postavimo.









U oznaci naziva modula smo videli da se koristi akronim SPI, sa značenjem Serial Peripheral Interface. Arduino podržava kako hardverski SPI interfejs tako i onaj koji se emulira softverskim putem. SPI je protokol koji omogućava simultanu dvosmernu komunikaciju između dva ili više uređaja od kojih je jedan u režimu Master, dok su ostali povezani kao Slave. Za njegovu punu implementaciju potrebne su četiri linije prenosa podataka, kao što prikazuje sledeća ilustracija:



Kod modela Arduino Uno, sledeći pinovi obavljaju funkciju hardverskog SPI interfejsa:

pin	skraćenica	puni naziv
13	SC(L)K	Serial Clock
12	MISO	Master In, Slave Out
11	MOSI	Master Out, Slave In
10	SS	Slave Select



boja	Arduino	LCD
	+5V	1 (VCC)
	GND	2 (GND)
	D10 (SS)	3 (CS)
	D8	4 (RESET)
	D9	5 (A0)
	D11 (MOSI)	6 (SDA)
	D13 (SCK)	7 (SCK)
	3,3V	8 (LED)

SCK je zadužen za klok sinhronizaciju prenosa, MOSI prenosi podatke od mastera prema slejvu, MISO, opet, od slejva prema masteru, a SS određuje sa kojim se od slejv uređaja trenutno komunicira. Treba znati da je SDA analogno funkciji MOSI. Kao što vidimo, MISO je ostao neiskorišćen, pošto je po svojoj prirodi ispis na ekran komunikacija u samo jednom smeru. Dakle, shema povezivanja ekrana i Arduina Uno bi izgledala ovako:

Hardverski SPI:

pin	TFT	Arduino
1	VCC	+5V
2	GND	GND
3	CS	10
4	RESET	8
5	A0	9
6	SCK	13
7	SDA	11
8	LED	+3,3V

Prilikom implementacije softverskog SPI interfejsa moguće je koristiti proizvoljne digitalne izlaze. Jedan takav slučaj bi mogao da izgleda ovako:

Softverski SPI

pin	TFT	Arduino
1	VCC	+5V
2	GND	GND
3	CS	2
4	RESET	3
5	A0	4
6	SCK	5
7	SDA	6
8	LED	+3,3V

Treba odmah reći da upotreba hardverskog SPI ima za posledicu veoma brže iscrtavanje na ekranu pa bi je trebalo koristiti kad god je to moguće. Softverski SPI je od koristi kada su linije hardverskog SPI iz nekog razloga zauzete. Navodimo primer jednostavnog koda zasnovanog na standardnoj TFT Arduino biblioteci:

```
#include <TFT.h> // Arduino LCD biblioteka
#include <SPI.h> // podrška SPI interfejsa
#define RST 8 // pin za reset
#define A0 9 // pin prenosa podataka
#define CS 10 // pin izbora uredjaja
// inicijalizacija biblioteke
TFT displej = TFT(CS, A0, RST);
void setup() {
    displej.begin(); // uvek ide na pocetku
    displej.background(50, 0, 0); //tamno plava
    displej.stroke(0, 0, 255); // crvena boja ispisa
    displej.setTextSize(1); // najmanji font
    displej.text("Svet Kompjutera\n", 0, 0);
    displej.stroke(0, 255, 0); // zelena
    displej.setTextSize(2); // veci font
    displej.text("Svet\nKompjutera\n", 0, 15);
    displej.stroke(255, 255, 255); // bela
    displej.setTextSize(3); // jos veci font
    displej.text("Svet\nKompjutera\n", 0, 50);
}
void loop() {}
```

Prvo u projekat uključujemo kôd iz biblioteka TFT i SPI, da bismo zatim definisali na koje pinove Arduina su priključene tri navedene linije prenosa. Onda te vrednosti prosleđujemo kao argumente funkcije za inicijalizaciju biblioteke. Funkcija Begin se poziva pre početka ispisivanja na ekran (inicijalizacija). Zatim sledi funkcija za postavljanje boje pozadine i nakon nje blok linija za ispis tri niza znakova na ekranu. Svakom nizu karaktera zadajemo veličinu fonta i boju ispisa. Treba napomenuti da se boje definišu u formatu BGR umesto mnogo češćeg RGB. Kao i u slučaju kada smo pisali o ekranu Nokia i ovde je preporučljivo korišćenje biblioteka pod nazivom Adafruit ST7735 i Adafruit GFX (ova druga je potrebna za funkcionisanje prve) koje u sebi sadrže rašireni set komandi za crtanje i ispis teksta.

• • •

Zvuk



Kretanje



CEPBO MOTOPH

U čemu je među njima razlika?

1. Metalni ili plastični zupčanici (METAL GEAR vs. PLASTIC GEAR)?

Servo motori sa metalnim zupcima su jači i mogu duže da podnesu teže uslove rada od plastičnih. Najčešće u oznakama ovih servo motora stoji dodatno "MG".

Međutim oni povremeno kod određenih prijemnika izazivaju smetnje i habaju se tokom vremena. Preporuka je da se oni u servo motoru menjaju povremeno uz čišćenje kućišta i zamenu maziva.

Takođe i oni "MG" servo imaju dodatni problem da proizvođači varaju. Tj. veoma često usred metalnih zupčanika "stoji" i jedan plastičan, pa je servo "MG" oslabljen sa dodatnom plastikom.

2. Servo sa kugličnim ležajem (lager, kugl-lager (germ.)) ili sa kliznom čaurom (biksnom) (BALL BEARING vs. BUSHING)?

Svaki servo ima izlaznu osovину koja prolazi kroz servo kućište. Servo radi mnogo tačnije i lakše ako ima kuglični ležaj. Klizna čaura se vremenom se ošteti pa se pojavljuju zazori. Uvek kada je moguće, koristiti servo motore sa ležajevima.

3. Digitalni ili analogni servo (DIGITAL vs. ANALOG)? Može li digitalni servo na moj prijemnik?

Odgovor. Na svaki prijemnik može da se namesti i digitalni i analogni servo. Pa u čemu je razlika?

Sličnosti:

I digitalni i analogni servo motori mogu da imaju dobar deo istih komponenti. To znači da i jedni i drugi mogu da imaju isti elektromotor, iste zupčanike i isti potencijometar. Tu ne postoje nikakve razlike.

Razlike:

Ono u čemu se razlikuju je način obrade dolazećeg signala (komande za pomeraj servo-a).

Analogni servo motor ima posebno projektovani čip za kontrolu motora.

Digitalni servo ima mikroprocesor i fet pojačavače za kontrolu.

Nailaskom signala komande, analogni servo šalje impulse motoru (50 impulsa u sekundi). Kod digitalnog servoa mikroprocesor šalje 300 impulsa u sekundi motoru i motor se pokreće brže.

To ima svoje dobre i loše strane.

Digitalni se centrira perfektno, drži svoju poziciju i brže reaguje. Nažalost neki digitalni servoi neće trajati duže nego analogni zato što oni "guraju" struju u servo motor u svakom položaju, pa zato lakše i pregore.

Analogni servo su manje tačni, više dostupni i trajniji od digitalnih.

Digitalni servo motori troše mnogo više struje (za njih je potrebno u model ugraditi najveću bateriju (ako se koristi baterijsko napajanje...) koja može da stane zbog povećane potrošnje).

Digitalni servo daju pun obrtni moment na svim uglovima dok analogni ne.

4. Obični (standardni) motor ili motor bez jezgra (ili možemo reći sa drugačijim rotorom) (STANDARD MOTORS vs CORELESS MOTORS)?

Obični elektro motori imaju stator i rotor. Rotor je namotan na gvozdeno jezgo. Zbog toga su rotori teški i inertni. Rotor je pričvršćen sa obe strane.

Coreless motori su dizajnirani na istim principima kao ovi prvi, ali su sastavljeni drugačije. Rotor je male težine. Namotaji su napravljeni u cilindru bez gvožđa i pričvršćeni su samo na jednom kraju rotora.

Zato što su mnogo lakši (nema gvožđa u sredini) pri komandi reaguju mnogo brže, manje su inertni, mnogo brže usporavaju, precizniji su i mogu da generišu više snage za istu veličinu.

5. Brzina servo motora i obrtni moment (HIGH SPEED vs. HIGH TORQUE)

Za većinu brži i snažniji motor je bolji. Ustvari servo motori velikih brzina su onoliko dobri koliko čovek može brzo da reaguje.

Veliki obrtni moment je bitan za veće modele ili modele sa velikim površinama za upravljanje gde se stvaraju velike sile (kao što su 3D modeli).

Brzina servo motora se izražava u sekundama za pomeraj od 60 stepeni.

Postoje motori od 0.05s - 0.2s a možda i više za 60 stepeni pomeraj. Ovde treba primetiti da ne daju svi proizvođači tačno pri kom naponu na servo motoru je koja brzina. Nije isto da li je 0.12s za 4.8V ili za 6V napona.

Takođe obrtni moment zavisi od priključenog napona. Motor je snažniji na većem naponu.

Obrtni moment se meri u kg/cm il u oz/in (ovo je unca po inču). Odnos je sledeći: 1 kg=35.27 unci, 1 inč=25.4mm.

Kako se to meri obrtni moment servo motora?

Na servo motor se zakači poluga (ruka) od 1cm. Proverava se koliku težinu taj servo motor može da drži bez pokretanja. Pa ako može da izdrži 2.63kg, onda je to motor od 2.63 kg/cm ili 36.1 oz/in (unci po inču).

TLC5940 PWM driver

Servo motors have to be controlled differently than common LEDs.

Fortunately, the `tlc5940` arduino library provides a way of doing so without having to change much code. The example file *Servos.ino* explains how you should connect a servo motor and shows the use of custom library functions such as `tlc_initServos()`.

An important thing to know is that if you want to use multiple TLC5940s, you have to set their quantity in the file "**tlc_config.h**" in the library's folder. Open the file with your favorite text editor and **replace the value** of the constant **NUM_TLCS** with the amount of TLCs you're using. Save the file and restart the Arduino IDE.

Be aware that you cannot use LEDs and servo motors with the same TLC5940 (either if daisy-chained), as the use of the latter fuction will drop down the PWM frequency to 50 Hz (which will be significant for the LEDs).

Promena Arduino PWM frekvencije

Here are some usage examples of the function:

```
// Set pin 9's PWM frequency to 3906 Hz (31250/8 = 3906)
// Note that the base frequency for pins 3, 9, 10, and 11 is 31250 Hz
```

```
setPwmFrequency(9, 8);
```

```
// Set pin 6's PWM frequency to 62500 Hz (62500/1 = 62500)
// Note that the base frequency for pins 5 and 6 is 62500 Hz
setPwmFrequency(6, 1);
```

```
// Set pin 10's PWM frequency to 31 Hz (31250/1024 = 31)
setPwmFrequency(10, 1024);
```

Please keep in mind that changing the PWM frequency changes the Atmega's timers and disrupts the normal operation of many functions that rely on time (`delay()`, `millis()`, `Servo` library).

```
/**
 * Divides a given PWM pin frequency by a divisor.
 *
 * The resulting frequency is equal to the base frequency divided by
 * the given divisor:
 * - Base frequencies:
 *   o The base frequency for pins 3, 9, 10, and 11 is 31250 Hz.
 *   o The base frequency for pins 5 and 6 is 62500 Hz.
 * - Divisors:
 *   o The divisors available on pins 5, 6, 9 and 10 are: 1, 8, 64,
 *     256, and 1024.
 *   o The divisors available on pins 3 and 11 are: 1, 8, 32, 64,
 *     128, 256, and 1024.
 *
 * PWM frequencies are tied together in pairs of pins. If one in a
 * pair is changed, the other is also changed to match:
 * - Pins 5 and 6 are paired on timer0
 * - Pins 9 and 10 are paired on timer1
 * - Pins 3 and 11 are paired on timer2
 *
 * Note that this function will have side effects on anything else
 * that uses timers:
 * - Changes on pins 3, 5, 6, or 11 may cause the delay() and
 *   millis() functions to stop working. Other timing-related
 *   functions may also be affected.
 * - Changes on pins 9 or 10 will cause the Servo library to function
 *   incorrectly.
 *
 * Thanks to macegr of the Arduino forums for his documentation of the
```

* PWM frequency divisors. His post can be viewed at:
* <http://forum.arduino.cc/index.php?topic=16612#msg121031>
*/

```
void setPwmFrequency(int pin, int divisor) {  
  byte mode;  
  if(pin == 5 || pin == 6 || pin == 9 || pin == 10) {  
    switch(divisor) {  
      case 1: mode = 0x01; break;  
      case 8: mode = 0x02; break;  
      case 64: mode = 0x03; break;  
      case 256: mode = 0x04; break;  
      case 1024: mode = 0x05; break;  
      default: return;  
    }  
    if(pin == 5 || pin == 6) {  
      TCCR0B = TCCR0B & 0b11111000 | mode;  
    } else {  
      TCCR1B = TCCR1B & 0b11111000 | mode;  
    }  
  } else if(pin == 3 || pin == 11) {  
    switch(divisor) {  
      case 1: mode = 0x01; break;  
      case 8: mode = 0x02; break;  
      case 32: mode = 0x03; break;  
      case 64: mode = 0x04; break;  
      case 128: mode = 0x05; break;  
      case 256: mode = 0x06; break;  
      case 1024: mode = 0x07; break;  
      default: return;  
    }  
    TCCR2B = TCCR2B & 0b11111000 | mode;  
  }  
}
```

Processing



Mapping projection



Vebografija:

1. [servos – TLC5940 + Arduino](#)
2. [TLC5940 PWM LED Driver IC](#)
3. [100+ Arduino Projects](#)
4. [12 servos controller low cost](#)
5. [15 Arduino Projects that you will love to see](#)
6. [16-channel PWM / Servo Controller | PWMSERVO | HobbyTronics](#)
7. [2 PIR Sensor controlling servo movement](#)
8. [200+ Arduino Projects List For Final Year Students](#)
9. [4 Segment LED Clock - Multiplexing slowed down - YouTube](#)
10. [555 Timer : Universal PWM Controller](#)
11. [5940:PWM LED driver](#)
12. [74HC595 IC](#)
13. [A DIY digital Arduino clock designed for and by teachers | Electronics Infoline](#)
14. [A long range FM Transmitter circuit | Electronics Infoline](#)
15. [ACES: TEI3M](#)
16. [Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface \[PCA9685\] ID: 815 - \\$14.95 : Adafruit Industries, Unique & fun DIY electronics and kits](#)
17. [Adafruit Learning System](#)
18. [Aimagin Blogspot – How to Drive Stepper Motors and RC Servo Motors](#)
19. [Arduino - ShiftOut 74HC595 IC](#)
20. [Arduino – Primeri | Tehničko i informatičko obrazovanje](#)
21. [ARDUINO | alselectro](#)
22. [Arduino 20 Servos, 4 Pins, Variable Refresh Rate | Let's Make Robots!](#)
23. [Arduino and Microphone \(Sound Sensor\) -](#)
24. [Arduino and processing interact. - All](#)
25. [Arduino and the TM1637 4-digit seven-segment display | Code, the Universe and everything...](#)
26. [Arduino Basics: PIR Sensor \(Part 2\)](#)
27. [Arduino Blog](#)
28. [Arduino Bluetooth Basic Tutorial](#)
29. [Arduino Bluetooth Control - Android Apps on Google Play](#)
30. [Arduino Christmas Light Controller - All](#)
31. [Arduino CNC](#)
32. [Arduino Explained](#)
33. [Arduino for Projects - Tutorials - Latest News and UpdatesUse Arduino for Projects](#)
34. [arduino free download - SourceForge](#)
35. [Arduino Lessons | Technology Tutorials](#)
36. [Arduino Motion Following 2 x HC-SR04 Ultrasonic - YouTube](#)
37. [Arduino Motor Party](#)
38. [Arduino Parts, Tutorials, News for DIY electronics | Brainy-Bits Canada](#)

39. [Arduino Playground - MegaServo48](#)
40. [Arduino pro naprosté debily \(?\) » Retročip](#)
41. [Arduino Radar Project - YouTube](#)
42. [Arduino to Arduino Serial Communication](#)
43. [Arduino Tutorial - Learn electronics and microcontrollers using Arduino!](#)
44. [Arduino Tutorial: Ultrasonic Sensor HC SR04 distance meter with a Nokia 5110 LCD display - educ8s.tv - Watch Learn Build](#)
45. [Arduino UNO Controlling 20 Servos With 15 bit Precision And Low Jitter | Lamja.com](#)
46. [Arduino Uno microcontroller - HomoFaciens](#)
47. [Arduino with HC-05 \(ZS-040\) Bluetooth module – AT MODE | Martyn Currey](#)
48. [Arduino With HC-05 Bluetooth](#)
49. [Arduino with PIR Motion Sensor | Random Nerd Tutorials](#)
50. [arduino-info - MegaQuickRef](#)
51. [Automatika.rs | DC Elektromotori | Mehatronika](#)
52. [Automatika.rs | Kako upravljati uređajima pomoću Androida? \[ARDUINO\] | Projekti](#)
53. [Automatika.rs | Projekti](#)
54. [BeginnersGuide - Python Wiki](#)
55. [BeginnersGuide/Overview - Python Wiki](#)
56. [BlueTooth-HC05-HC06](#)
57. [Brokking.net - Your Arduino Balancing Robot \(YABR\) - Home.](#)
58. [BubbleBee - Cpp za početnike](#)
59. [Can Android/Arduino Combo Accelerate The Global Electronics DIY Revolution? - TechBitar](#)
60. [Can't Get I2C to Work on an Arduino Nano? \(Pinout Diagrams\) | Big Dan the Blogging Man](#)
61. [Chimera: \\$60 DLP High-Res 3D Printer - All](#)
62. [Control a DC motor with Arduino and L293D chip — Lucky Larry](#)
63. [Controlling 12 servos with I2C Servo Controller IC](#)
64. [Controlling Many Servo\(s\) Using IC TLC5940 | sw0rdm4n](#)
65. [Crossover](#)
66. [Curved Circuit Board Art: Make a Touchless Touch-Switch LED Lamp - All](#)
67. [Dash Board - Programming Electronics Academy](#)
68. [Datasheets & application notes - Datasheet Archive Search Engine](#)
69. [Dejan Nedelkovski - YouTube](#)
70. [Digital/Analog Clock - Arduino + PaperCraft - All](#)
71. [DIY Motion Sensing Ceiling Fan/light ***5\\$ Energy Saver!!!*** - do it yourself](#)
72. [DS3231 AT24C32 TM1637 4digits | Easy Arduino](#)
73. [EEE COMMUNITY](#)
74. [Ep.53 Arduino Projects - Potentiometer Servo Control & Memory - YouTube](#)
75. [Explore by newest | OpenHardware.io](#)

76. [FirstPCB - the Best Valuable Prototype & Batch PCB Production](#)
77. [FlowStone Graphical Programming Software](#)
78. [Fritzing Projects](#)
79. [Getting Started with ArduBlock](#)
80. [Getting started with BTE13-010 - Arduino Mini clone](#)
81. [Getting Started with ESP8266 WiFi Transceiver \(Review\) | Random Nerd Tutorials](#)
82. [GlobalSpec - Searchable Engineering Catalogs on the Net](#)
83. [Google Code Archive - Long-term storage for Google Code Project Hosting.](#)
84. [Google Code Archive - Long-term storage for Google Code Project Hosting.](#)
85. [Google Code Archive - Long-term storage for Google Code Project Hosting.](#)
86. [Graphical Resistance Calculator](#)
87. [grbl/grbl - C - GitHub](#)
88. [HC-05 Bluetooth Android/PC](#)
89. [How To - Expand Digital Inputs with the 74HC165 + Test Code - YouTube](#)
90. [How to Build a Servo Motor Circuit \(with Arduino\) -Use Arduino for Projects](#)
91. [How to chose Best Graphics Card for your PC – How much Proce ... | Electronics Infoline](#)
92. [How To Configure and Pair Two HC-05 Bluetooth Modules as Master and Slave | AT Commands - HowToMechatronics](#)
93. [How to make high-tech LED decorations for the holidays | Electronics Infoline](#)
94. [How to Make MIT app for controlling servo Motor using arduino and bluetooth module. - YouTube](#)
95. [How To Mechatronics](#)
96. [How To Select an Inductor | Electronics Infoline](#)
97. [How-To: Daisy Chain Arduinos via Serial](#)
98. [Industrijska napajanja za monažu na panel](#)
99. [infostan - Google преграда](#)
100. [Inlaid Logic : A giant circuit board inlay made out of waterjet electronic traces - All](#)
101. [Installing an Arduino Bootloader - learn.sparkfun.com](#)
102. [Interactive Hanging LED Array - learn.sparkfun.com](#)
103. [Jianping Electronics 34 servos](#)
104. [K4S, a Keyboard for Arduino to use with Scratch -Use Arduino for Projects](#)
105. [Kako spojiti LED diodu - PetVolta.com](#)
106. [Kalkulator za proračun otpornika za LED diode - Održavanje vozila - Besplatno!](#)
107. [Kinect - 2.007 Arduino Nano Carrier](#)
108. [L298N Motor Driver Controller Board - All](#)
109. [LASER DIODES – worldstartech.com](#)
110. [Laser scanners for 2D/3D profile measurement | Micro-Epsilon America](#)

111. [Let's Make Robots!](#)
112. [Literatura - PIC mikrokontroleri](#)
113. [Long Range Ultrasonic Distance Sensor - All](#)
114. [Make Your Own Temperature Sensor and Email Alarm | Electronics Infoline](#)
115. [Matrix & Sprite Arduino Libraries, for a many-LED display!](#)
116. [Micro Servo Robot | Let's Make Robots!](#)
117. [Mini Maestro 24-Channel USB Servo Controller \(Assembled\) | 1356 | Pololu](#)
118. [Mini Pan-Tilt Kit - Assembled with Micro Servos ID: 1967 - \\$18.95 : Adafruit Industries, Unique & fun DIY electronics and kits](#)
119. [MIT app inventor Control Multiple Arduino's PWM pins - YouTube](#)
120. [MITappinventor: Course on MIT app inventor Arduino](#)
121. [MOAR OUTPUT](#)
122. [motor - How can the Arduino Uno support up to 12 servos if it only has 6 digital PWM pins? - Electrical Engineering Stack Exchange](#)
123. [motor-speed-and-direction](#)
124. [Motors | Hobbyist.co.nz](#)
125. [Multiplexing with TLC5940](#)
126. [MySensors - Motion Sensor](#)
127. [Naslovnica - Croatian Makers](#)
128. [Nastavna sredstva | M&G Dakta](#)
129. [Network Five Arduinos \(or more\) using I2C - TechBitar](#)
130. [Petlja for - Osnove programiranja u jeziku C++](#)
131. [PetVolta.com](#)
132. [PINOUT all](#)
133. [PIR Motion Sensor Tutorial - do it yourself](#)
134. [Pogledajte kako 3D robot štampač pravi stolice \[VIDEO\] - Inženjerski portal Balkana](#)
135. [power supply - Multiple Voltage Sources using a single Ground - Electrical Engineering Stack Exchange](#)
136. [Programming for Everybody \(Getting Started with Python\) - University of Michigan | Coursera](#)
137. [Pull-up / pull-down otpornik / e-radionica.com LEARN hrvatski](#)
138. [Pure Data to Arduino Over Serial](#)
139. [PWM Outputs- TLC5940 Tutorial](#)
140. [PYTHON](#)
141. [Random Nerd Tutorials](#)
142. [RC boat with NRF24L01+ and Arduino | Electronics Infoline](#)
143. [rcservo - Controlling more than 12 Servos with the Arduino Servo library - Robotics Stack Exchange](#)
144. [Related video](#)
145. [Reset your circadian clock "biological body clock" with Arduino - All](#)

146. [robotics - Arduino Uno with Micro Maestro 24-ch Servo Controller Tutorial - Stack Overflow](#)
147. [S4A](#)
148. [Serial Communication - learn.sparkfun.com](#)
149. [Simple three-servo hexapod walker using the Pololu Micro Maestro | Let's Make Robots!](#)
150. [Škola elektronike](#)
151. [Soapbox Robotics | Compiler](#)
152. [Static Jolt Electronics - Where projects come to life...](#)
153. [SVET KOMPJUTERA - LAKI PINGVINI - Stonoga zvana GPIO](#)
154. [termo_servo](#)
155. [The Arduino Course - Programming Electronics Academy](#)
156. [The Custom Geek | A place to share my inventions.](#)
157. [The Mind of Bill Porter](#)
158. [The Secret Project!](#)
159. [The World Famous Index of Arduino & Freeduino Knowledge](#)
160. [TimerOne & TimerThree Arduino Libraries](#)
161. [Tlc5940/Servos.pde](#)
162. [TLC5940NT LED driver IC \(Kao novo - nekorisćeno\)](#)
163. [Top 4 Graphical Programming Software Comparison - Makeblock](#)
164. [tutorials:extending_pwm_output_pins_with_a_texas_instruments_tlc5940_led_driver · SensorWiki.org](#)
165. [Učenje programiranja programiranjem robota i igara | Pogled kroz prozor](#)
166. [Upravljanje step motorom](#)
167. [Using PL2303 based USB to TTL converter with Arduino | Kaushlesh Chandel](#)
168. [VarSpeedServo - a modified Servo library with speed control](#)
169. [Vezivanje komponenata | Arduino & C#](#)
170. [Voice Controlled Arduino Drone - All](#)
171. [WIKI Arduino 7 segment LED timer with 74HC595 module - Geeetech Wiki](#)
172. [www.thecustomgeek.com/files/ 4 Seg LED Clock 2.pde](#)
173. [Headtracking in the browser on Vimeo](#)
174. [Face Detection Software / Facial Recognition Source Code API SDK Login](#)