AKADEMIJA TEHNIČKO-VASPITAČKIH STRUKOVNIH STUDIJA
ODSEK NIŠ

Završni rad

Serijski port mikrokontrolera ATmega328

Predmet: Primena mikrokontrolera

Mentor:

dr Zoran Milivojević

Student:

Filip Stojanović

REr 56/17

April 2021.

AKADEMIJA TEHNIČKO-VASPITAČKIH STRUKOVNIH STUDIJA
ODSEK NIŠ

Završni rad

Serijski port mikrokontrolera ATmega328

Predmet: Primena mikrokontrolera

Mentor:

dr Zoran Milivojević

Student:

Filip Stojanović

REr 56/17

Članovi komisije:

1. Nataša Nešić
2. Danijela Aleksić
3. Zoran Milivojević

Zahvalnica

Zahvaljujem se svim profesorima na akademiji tehničko vaspitačkih strukovnih studija na odseku Niš, a naročito mom mentoru, profesoru dr Zoranu Milivojeviću na vođenju kvalitetne lako primenljive nastave koja me je podstakla da želim više od sebe i naučim šta sam mislio da neću moći.

Sadržaj

1. Uvod

Smatra se da je prvi elektronski računar bio ENAIC (Electronic Numerical Integrator and Computer) koji je proizveden u SAD-u 1965 godine. Od tada su računari doživeli veliki tehnološki napredak, prošli pet generacija i danas su za razliku od svojih predaka veoma kompaktni uređaji. U ovom radu istražićemo mikrokontroler ATmega328, specifično mogućnosti serijskog interfejsa za dvosmernu (potpuni dupleks) komunikaciju, koristeći mikrokontrolerski sistem Arduino Uno R3 baziran na ATmega328P.

Mikrokontroler je mali kompjuter koji sedi na integrisanom kolu, normalno radi u kontrolnoj petlji i njegova namena je upravljanje uređajima i procesima u ugrađenim sistemima (Engl. embedded systems) na čije događaje mora dati odgovore u pravom vremenu, on sadrži jedan ili više CPU (Central Processing Unit), memoriju i programabilne ulazno/izlazne periferije. Glavna razlika između modernih mikroprocesora i mikrokontrolera je to da su prvi optimizovani za brzinu i performanse kod računarskih programa dok su drugi optimizovani za integraciju većeg broja kola. Mikrokontroler može sadržati i oscilatore, timere, brojače, serijski port, analogno digitalni konvertor i druge dodatke za koje je nekada bio potreban niz posebnih integralnih kola.

Mikrokonktroleri su slični ali manje sofisticiraniji nego SoC (System on a Chip) kao što su Raspberry Pi računari. SoC može sadržati mikrokontroler kao komponentu ali integriše ga i sa naprednim perifernim uređajima poput GPU (Graphics Processing Unit), wifi modulima, eksterenim diskovima… Sa nekim modernim mikrokontrolerima teško je tačno definisati šta je SoC a šta je mikrokontroler jer oba mogu sadržati GPU i wifi module.

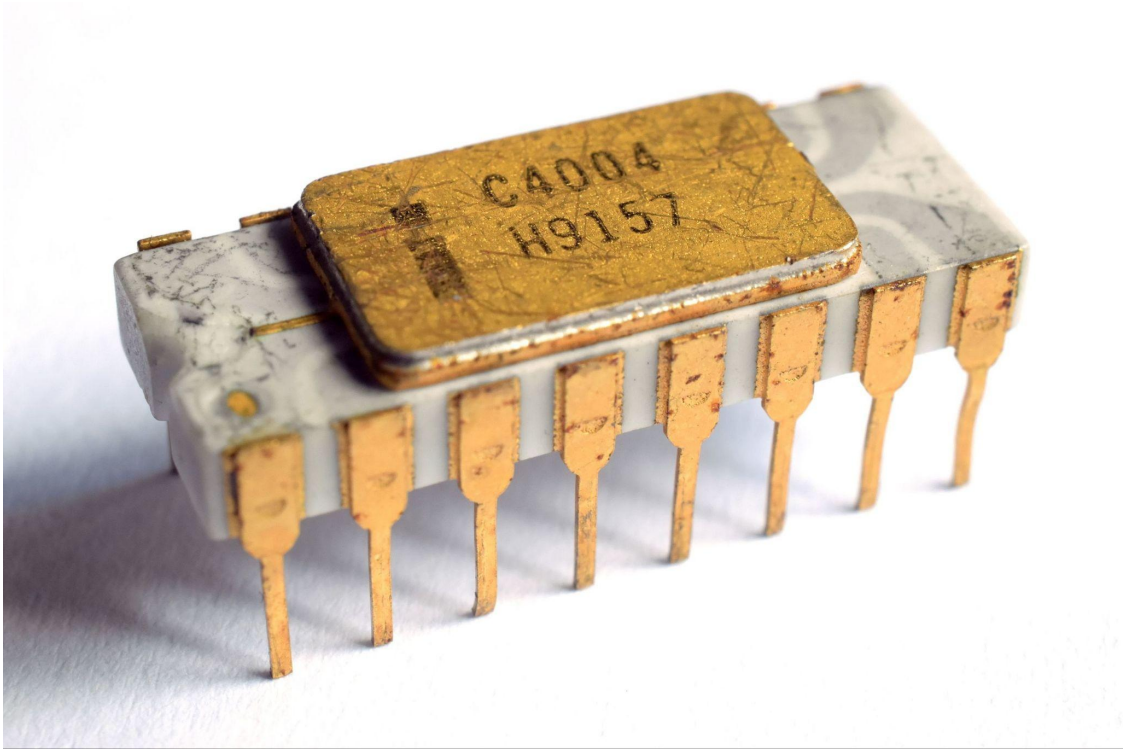
USART (Universal Synchronous and Asynchronous Receiver-Transmitter) je protokol za serijsku komunikaciju koji se koristi za slanje i primanje podataka po određenoj stopi bodova (Engl. Baud rate). Bod predstavlja broj prenetih signala kroz neku sredinu u sekundi - svaki signal može nositi jedan ili više bitova informacije. Tako da, ako imamo signal od 250 bodova a svaki signal ima po 4 bita informacija, onda nam je brzina prenosa 1000 bitova u sekundi (bit/s).

Arduino je kompanija koja proizvodi razvojni sistem Arduino IDE otvorenog tipa kao i jednoploče mikroračunarske sisteme otvorenog hardverskog dizajna sa ciljem da omogući brzu i jeftinu proizvodnju embedded uređaja. Ime je dobila po kafiću u Italiji u kome su osnivači izlazili. Arduino IDE razvojno okruženje primarno služi za programiranje Arduino i kompatiblnih Genuino ploča, međutim moguće je razvijati softver i za druge mikrokontrolere ali to obično zahteva zaseban programator.

2. Istorija mikrokontrolera

Začeci mikroprocesora kao i mikrokontrolera mogu se pronaći u MOSFETU (metal oxide silicon transistor), kompaktnom tipu operacionog pojačivača izumljenog od strane Bell Labs i javno prikazanim 1959. godine. Nakon začetka mikroprocesora, inovacija u računarstvu dešavala se toliko velikim tempom da je donet zakon nazvan po zapažanjima Gordon Moore-a (Murov zakon) koji glasi da se broj tranzistora u integralnom kolu udvostručuje približno svakih 18-24 meseca. Ovaj zakon važio je 55 godina ali mu se bliži kraj zbog trenutnih limitacija kvantnog računarstva.

2.1 Rani mikroprocesori



SL 2.1.1 Mikroprocesor Intel 4004 (CC Thomas Nguyen)

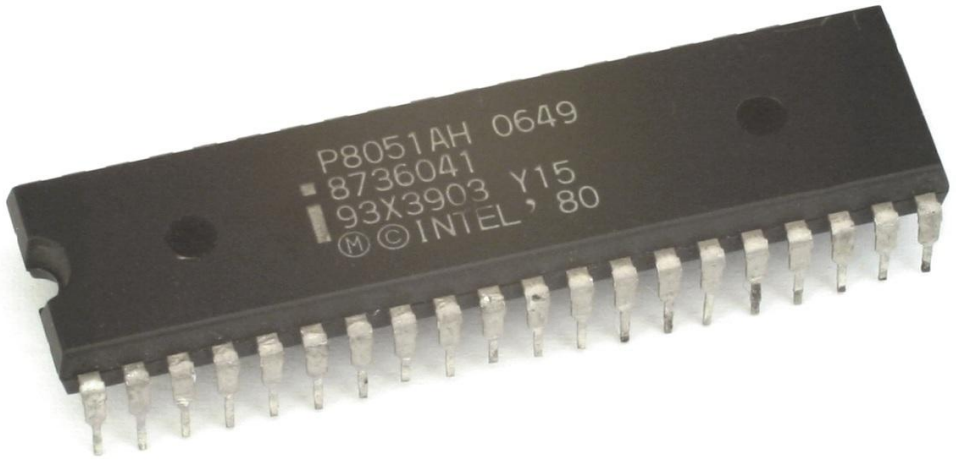
Prvi komercijalno dostupan (4 bit) mikroprocesor, Intel 4004 proizveden je u Sjedinjenim Američkim Državama 1971 godine. Prvobitno je bio korišćen u digitronima koje je proizvodila Japanska kompanija BUSICOM. Ubrzo su naredni čipovi rapidno postajali napredniji, intel je 1972 godine proizveo prvi 8 bit mikroprocesor, Intel 8008. Godinu dana kasnije, 1973 Toshiba je proizvela 12 bit mikroprocesor, TLCS-12. 1974 godine nastao je Intel 8080 koji je bio sastavljen od 6000 tranzistora, Zilog Z80 je ovom mikroprocesuru napravljen kao konkurencija. Po nekim

izvorima 1971 godine napravljen je prvi mikrokontroler, TMS1802NC, za čiju su kreaciju zaslužni inženjeri Gary Boone i Michael Cochran iz Texas Instruments kompanije što dovodi u pitanje šta je mikroprocesor a šta mikrokontroler. U cilju bolje klasifikacije, danas koristimo termin SoC kao oznaku za sofisticiranije čipove.

2.2 Rani mikrokontroleri

Jedan od prvih mikrokontrolera bio je Motorola 6801, dostupan 1974 godine i razvijen na bazi mikroprocesora Motorola 6800, tim koji je radio na ovom mikroprocesoru napravio je MOS Technology 6502, mikroprocesor na kome su bazirani Commodore PET i Apple 2 računari. Nakon njega ubrzo je javno predstavljena i TMS 1000 serija mikrokontrolera. NEC μCOM-16 bio je prvi 16 bit mikroprocesor, napravljen 1974 godine.

Tipičan rani mikrokontrolerski (mikroračunarski) sistem zahtevao je veliki broj dodatnih kola za rad, kao što su A/D pretvarači (Engl. A/D converters), brojači, oscilatori i drugo.



SL 2.2.1 Mikrokontroler Intel P8051 (CC Konstantin Lanzet)

Vremenom došlo je do integrisanja više komponenti u jedno kolo i tako su nastali moderniji mikrokontroleri. Jedan od prvih mikrokontrolera ovakvog tipa, koji je zbog svojih softversko-hardverskih karakteristika postao industrijski standard je Intel 8051, od koga je nastala MCS-51 serija mikrokontrolera koja je bila napravljena osamdesteih godina 20. veka ali zbog njene popularnosti napravljene su i modernije varijante. MCS-51 serija koristi NMOS, CISC (complex instruction set computer) instrukcioni set i nasledila je MCS-48 seriju koja je

koristila CMOS i kojoj je pripadao Intel 8048, prvi Intelov mikrokontroler napravljen 1976 godine. Intel 8086 je 16 bit mikroprocesor dostupan od 8og Juna 1978, od ovog modela nastala je x86 arhitektura koja je intelova najuspešnija linija procesora. Varijanta ovog mikroprocesora, Intel 8088 (dostupan od 1. Jula 1979) korišćena je u dizajnu prvog IBM kompjutera. 1979 godine napravljen je prvi 32 bitni mikroprocesor, Motorola 68000. 1985 godine stvoren je popularni Motorola 68HC11 model. 32 bitna arhitektura prevaziđena je tek 1991 godine, kada je napravljen prvi 64 bitni mikroprocesor, MIPS R4000. 64 bit arhitektura ostala je dominantna jer nije još bilo potrebe za 128 bitnim mikroprocesorima sem u eksperimentalnim svrhama.

Na prvim mikrokontrolerima ROM (Read Only Memory) memorija bila je podešena od strane proizvođača. Nakon pronalaska PROM i EPROM memorija postalo je moguće da se korisnički programiraju mikrokontroleri, PROM (Programmable ROM) memorija mogla je da se programira samo jednom dok je EPROM (Erasable Programmable ROM) mogla više puta.
Poznati tip memorije, UV-EPROM (Ultraviolet EPROM) mogo je biti brisan uperivanjem ultraljubičaste svetlosti ka transparentnom kvartznom prozoru u poklopcu paketa memorije, međutim ovo je moglo da traje i do 20 minuta tako da nije bilo idealno. 1993 godine predstavljena je EEPROM (Electrically Erasable ROM) memorija sa kojom je memorija mikrokontrolera (počevši sa mikročipom PIC16C84) mogla biti električno obrisana brzo i jeftino. Iste godine Atmel je predstavio i flash memoriju.

SL 2.2.2 ATmega328P mikroprocesor

AVR familiju mikrokontrolera je od 1996 godine razvijao Atmel (deo Microchip Technology od 2016 godine). To je serija mikrokontrolera sa RISC (Reduced Instruction Set**)** instrukcionim setom i modifikovanom harvard arhitekturom.

2.3 Istorija Arduino čipova

ATmega328P pripada Atmel porodici mikrokontrolera. Postoji par varijanti ovog mikrokontrolera: originalni ATmega328, picoPower ATmega328P koji koristi manje energije i moderniji ATmega328PB koji ima više komponenti i funkcionalosti na pinovima.

SL 2.3.1 Arduino UNO Rev 3 mikrokontrolerski sistem

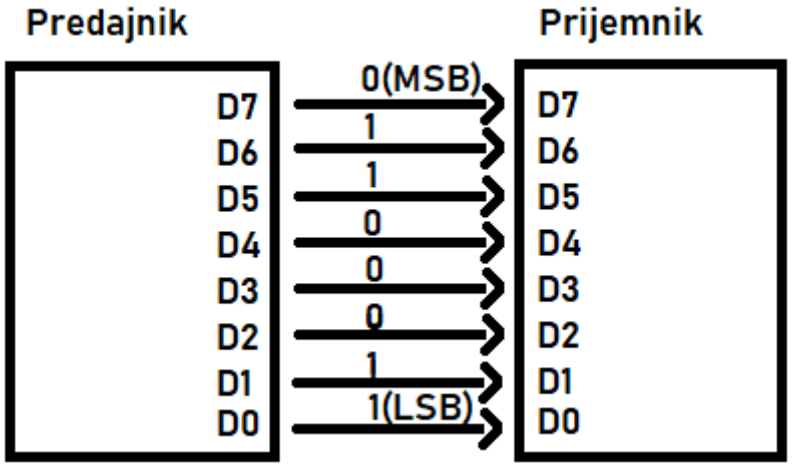
Arduino je započet 2005 godine u Italiji kao projekat grupe studenata koji su hteli da naprave jednostavne i jeftine alate sa kojima bi ljudi i bez inženjerskog iskustva mogli da prave digitalne projekte. Trenutno ima preko 17 zvaničnih ploča i većina je koristila kombinaciju flash i EEPROM memorija, prva ploča koju je Ardunio proizveo bila je “Serial Arduino”, dostupna 30. marta 2005. Još neke značajne ploče bile su LilyPad Arduino (dostupna 17. oktobra 2007) i Arduino Duemilanove (dostupna 19. oktobra 2008). Prva revizija popularne Arduino Uno ploče dostupna je od 24. septembra 2010 i bazirana je na ATmega328P mikokontroleru. Dodatne

funkcioninalnosti dostupne su za sve a pogotovo za ovu ploču korišćenjem takozvanih štitova (Engl. shield) i dodatnih biblioteka. Prva Arduino ploča bazirana na ARM arhitekturi je Arduino Due, dostupna od 22. oktobra 2012 godine. Od oktobra 2017 godine Arduno je u partnerstvu sa ARM i radi se na razvijanju Arduno ploča sa ARM mikroprocesorima. Od 2016 godine moguće je vizuelno programirati Arduino ploče upotrebom XOD programskog jezika.

3. Paralelna komunikacija

Data link predstavlja način povezivanja dve lokacije zarad upostavljanja digitalne komunikacije tj primo-predaje informacija. Format komunikacije određuje protokol koji je u sloju veze OSI referentnog modela. Postoje tri osnovna tipa komunikacije koji se mogu koristiti:

- Simplex, komunikacija samo u jednom smeru.
- Polu dupleks, komunikacija u dva smera ali ne i istovremena.
- Potpuni dupleks, istovremena komunikacija u dva smera.

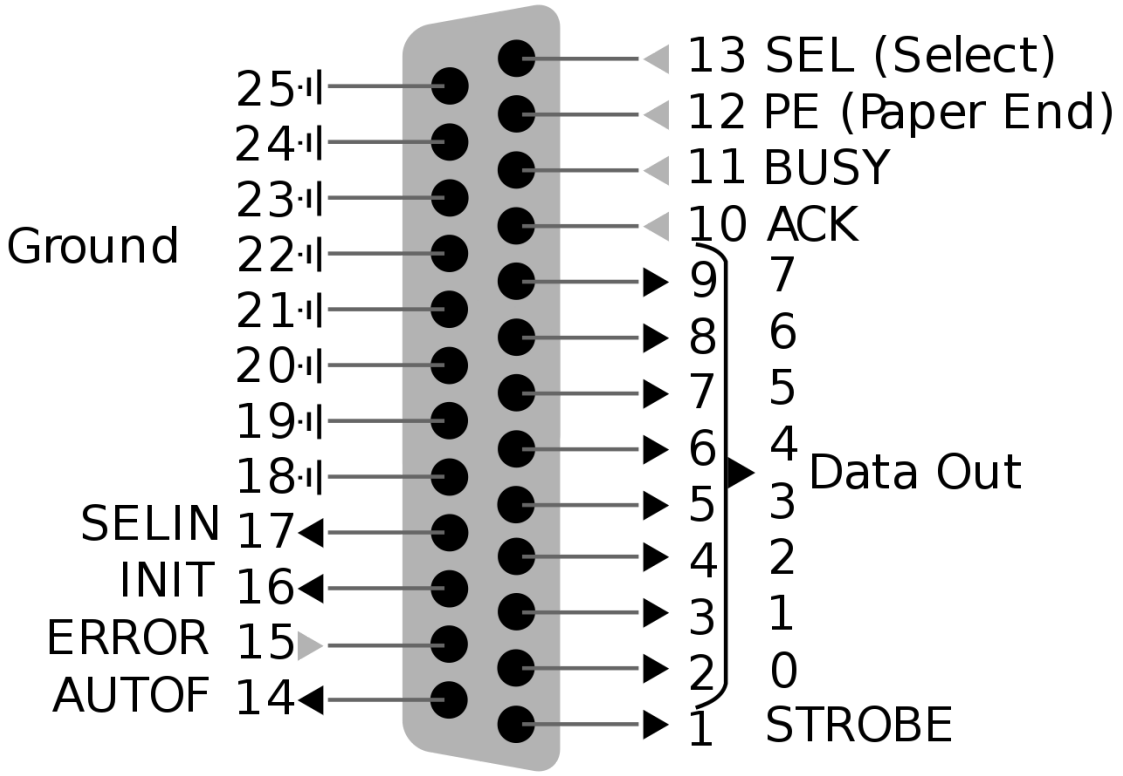


SL 3.1 Paralelni interfejs

Paralelna komunikacija je metoda prenosa podataka gde se više bitova transimituju istovremeno u toku jednog taktnog signala putem više linija prenosa (za razliku od transmitovanja bita po bita sekvencijalno kao u morsovom kodu i BPSK modulaciji). Ovo se dešava na fizičkom sloju OSI referentnog modela, dobro poznate tehnike modulacije kao što su PSM, PAM i MIMO (Multiple Input Multiple Output) šalju po nekoliko bitova u paraleli (svaka grupa bitova zove se simbol). Ove tehnike mogu se primeniti i za slanje celog bajta od jednom (na primer 256-QAM). Zbog potrebnih dodatnih linija za prenos, paralelna komunikacija je obično half duplex. Pored linija za prenos podataka može biti dodatnih linija na primer za taktni signal. Ova metoda prenosa primenjuje se kada se podaci trebaju poslati na kraćoj razdaljini (obično do 6 metara). Primeri uređaja koji koriste paralelnu komunikaciju su CNC mašine, integrisana kola, RAM, periferni busevi u računaru kao ISA, rane verzije SCSI-a, PATA (Parallel ATA), PCI (Peripheral Component Interconnect), IEEE-1284 štampači i drugi periferni uređaji.

Postoje dosta paralelnih portova, IEEE 1284 bio je de facto standard od 1970 do 2000 godine. Termin “paralelni port” se idalje najviše asocira sa printer portom tj Centronics portom ili paralelnim printer portom (na IBM PC računarima). Ovaj port je bio standardizovan devedesetih godina 20. veka.

3.1 Istorija paralelne komunikacije



SL 3.1.1 Pin out DB-25 paralelnog konektora (CC AndrewBuck)

IBM personalni računar napravljen je 1981. i imao je varijantu Centronics porta (koji je razvijen sedamdesetih godna 20. veka). Ovaj interfejs je standardizovao paralelnu komunikaciju sa DB25F konektorom - port sa 25 pinova koji ima 17 linija podataka za signale i 8 za uzemljenje. Sedamnest linija za signale podeljeno je na:

- 4 linija za inicijaciju komunikacije i izlaznu kontrolu
- 5 linija za notifikaciju o greškama i izlaznu kontrolu
- 8 linija za transmisiju podataka

Ovaj port je bio dominantni port koji je IBM koristio do 1987. kada je IBM napravio bidirekcioni interfejs IBM PS/2. Paralelni interfejs je bio adaptiran za upotrebu na više vrsta perifernih uređaja pored štampača. Jedna od prvih upotreba za paralelni port bili su dongle uređaji, optički drajvovi poput CD čitača, čitača disketa, skenera, eksternih modema,

gejmpadova i džojstika. Neki od ranih portabilnih mp3 plejera zahtevali su korišćenje paralelnog porta za prebacivanje pesama na uređaj. Drugi uređaji poput programera EPROM memorija i hardverskih kontrolera povezivali su se preko paralelnog porta a za povezivanje drugih uređaja postojali su adapteri za povezivanje - na primer da bi se povezivali SCSI uređaji.

Na sistemima baziranim na DOS-u logički paralelni portovi detektovani na BIOS-u bili su dostupni operativnom sistemu pod inkrementalnim imenima poput LPT1, LPT2, LTP3, ...) Ova imena su skraćenice za termine popupt line print terminal, local print terminal i line printer). LPT kao i par i drugih imena kao PRN, CON, AUX su zabranjeni nazivi dadoteka i fascikla u DOS operativnom sistemu i sistemima baziranim na njemu, čak i u Windows 11 operativnom sistemu ne može se napraviti dadoteka ili fasickla sa ovim imenima. Paralelna komunikacija je od uvek bila korišćena u integrisanim kolima, perifernim busevima i u memoriji. Direct control feature system/360, iz standard System/360 (1964 godina) varijante modela IBM računara i Laboratory Instrumentation bus IEEE-488 imali su 8-bitni port, IBM model 44, process-control varijanta imala je 32-bitni port. Busevi u računarima su prvobitno koristili paralelnu komunikaciju ali tokom vremena noviji računari prešli su na korišćenje serijske komunikacije za buseve ali su paralelni data linkovi videli uvećanu upotrebu u RF radio komunikaciji.

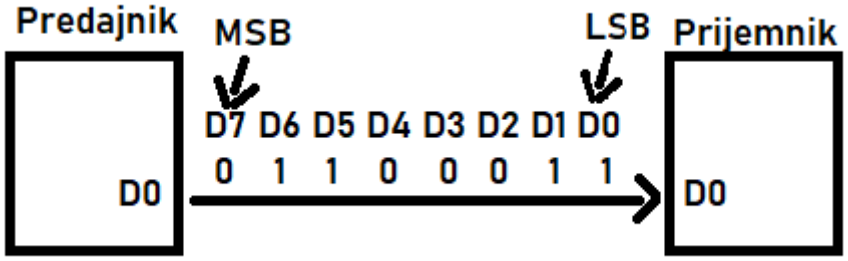
3.2 Clock skew

Clock Skew (timing skew) je pojava u paralelnoj komunikaciji koja ograničava brzinu prenosa transmisije na brzinu prenosa najsporije linije prenosa. Kada signali od istog izvora stignu do drugih komponenti u različito vreme, razlika između najbržeg i najsporijeg vremena se zove clock skew. Clock skew može napraviti da taktni signal signal prvo stigne na predajniku a zakasni na prijemniku (negativan skew) ili da prvo stigne na prijemniku a zakasni na predajniku (pozitivan skew). Nulti skew je kada signal stigne na vreme na primo-predajniku. Clock skew se dešava jer za razliku od idealnog scenarija svaka žica tj linija prenosa može imati (nenamerno) blago drugačije karakteristike tako da neki bitovi mogu stići pre drugih što može imati negativan uticaj na integritet poruke. Bit parnosti može pomoći u smanjenju ovog efekta ali i dalje se smatra da paralelna komunikacija nije pouzdana pri prenosu podataka na većim distancama jer sa većom distancom raste mogućnost korumpiranih poruka. Pod uslovom da su Ri i Rj dva susedna registra a Tci i Tcj vremena koja su potrebna signalu tih registra da stigne od izvora do destinacije clock skew se može definisati kao:

$$T_{skew\ i,j} = T_{ci} - T_{cj} \quad (1)$$

Clock skew može negativno uticati na komunikaciju. Može se desiti da se na drugom registru zamene inicijalno prebačeni podaci i time se korumpiraju podaci koji se već bili tu i uništi integritet poruke. Može se desiti da clock skew pozitivno utiče na kolo ako se njime smanji period satnog signala i time smanji vreme koje je potrebno da kolo funkcioniše korektno. Za postizanje pozitivnog clock skew-a razvijeni su algoritmi za optimizovanje sa podešivim varijablama sa kojima se može postići takozvani optimalni skew.

4. Serijska komunikacija



SL 4.1 Serijski interfejs

Serijska komunikacija je metoda prenosa podataka gde se jedan po jedan bit sekvencijalno transmituju preko jedne linije prenosa. Brzina prenosa preko te jedne linije može biti veoma visoka i može se lako koristiti i na većim distancama kao i jeftino implementirati potpuni dupleks. Pored bitova podataka uobičajeno se šalju dodatni bitovi. Primeri uređaja koji koriste serijsku komunikaciju su kompjuterska tastatura i miš, PS/2, nove verzije SCSI-a, USB, senzori, kamere, LCD monitori, GPS, GSM, RFID, itd.

Postoje dva režima serijskog prenosa: asinhroni režim serijske transmisije kod kojeg se koriste start (0) i stop (1) bitovi za obeležavanje početka i kraja transmije podataka, dodati bitovi smanjuju količinu bitova podataka u prenosu. Ova metoda transmisije podataka se koristi kada se podaci šalju sa prekidima umesto konzistentno, primer tipične start-stop transimije je ASCII (nasledio RS-232 format) koja je bila korišćena u pisaćim mašinama.

Drugi režim serijskog prenosa je sinhroni režim, on zahteva da se podaci kontinualno šalju i čitaju istom brzinom inače dolazi do korumpiranja poruka. On radi tako što se sinhronizuju brzine prenosa na tački slanja i prenosa koristeći taktnti signal, taj signal može biti poseban signal ili uključen u signal sa podacima u zavisnosti od formata prenosa. Brzina prenosa podataka u ovom režimu je efikasnija jer nema bitova za start i stop ali može se desiti potreba za resinhronizacijom jer vremenom taktni signali imaju tendenciju ka desinhronizaciji.

Podaci u serijskoj komunikaciji prebacuju se u “serijama” i svaka sadrži više bitova. Neki protokoli koji koriste sinhronu serijsku komunikaciju su SCSI, SPI i I2C. Sa druge strane asinhornu koriste CAN, RS-422, RS-485 i RS-232 (čije kablove sačinjavaju 25 žica ali samo dve se koriste za dvosmernu transmisiju podataka, druge se koriste za kontrolne signale).

Serijski prenos sastoji se od:

- Bitova sa podacima
- Sinhronizacionih Bitova
- Bitova parnosti

Standradne brzine prenosa u bodovima su: 300, 1200, 2400, 4800, 9600, …

SL 4.2 D-subminiature konektor (CC Jud McCranie)

Ima mnogo različitih serijskih konektora. D-subminature konektor ima 9 pinova u obliku slova D koji prebacuju podatke serijski. Drugi interfejsi poput Ethernet, FireWire, i USB koriste serijsku kokuminaciju ali termin serijski port se najčešće asocira sa D-9 konektorom ili hardver saglasan sa RS-232 / SCI standardom ili nekim drugim poput RS-485 ili RS-422. USB je zamenio većinu drugih konektora ali moguće je koristiti USB-to-serial konvertore kako bi se omogućila konekcija preko USB-a sa RS-232 i drugim klasičnim serijskim uređajima. Klasični serijski konektori se idalje koriste za uređaje koji zahtevaju proste konektore niskih brzina poput nekih naučnih instrumenata i POS sistema….

Arduino za konvertovanje podataka iz serijskih pinova RX i TX u USB format koristi poseban mikročip ATmega16U2, dosta modernih uređaja koriste USART integrisano kolo kako bi implementirali serijski port.

4.1 Istorija serijske komunikacije

Početkom 1980. godina, kada je IBM predstavio prvi IBM računar, IBM i druge kompanije su ubrzo napravile R232 dodatne ploče kako bi omogućile konekekcije računara ka spoljnim uređajima. Opadajuća cena i bolje performanse integrisanih kola dovele su do rasta upotrebe serijske komunikacije u poređenju sa paralelenom. Jedna velika prednost serijske komunikacije bila je korišćenje manje žica i pinova za komunikaciju što je inicijalno omogućilo i pravljenje kompaktnijih uređaja, ovo je rešilo mnoge probleme sa troškovima i dostupnošću pinova.

Primeri zamene paralelne komunikacije za serijsku su zamena IEEE 1284 štampačkih portova za USB, zamena parallel ATA sa serijskom ATA, zamena PCI za PCI Express. Uređaji za prebacivanje audiovizuelnih (AV) podataka kao što su digitalne kamere i profesionalni skeneri koji su ranije zahtevali posebne konektore kao SCSI HBA su većinski postali standardizovani upotrebom serijske komunikacije u potrošačkim uređajima. Serijski uređaji su se mogli videti kao /dev/tty* uređaji na unix sistemima i kao COM portovi (COM1, COM2, …) na DOS i Windows sistemima, nije moguće napraviti fascikle i dadoteke sa ovim imenima.

ISA kartice koristile su se u periodu od 1981 do 1997 godine, od početka upotrebe Windows 3.1 operativnog sistema ranih 1990. godina, Windows Device Driver preuzeo je od mikroprocesora ulogu upravljanja hardvera serijskog porta i primo predaje podataka koji su se putem njega slali, sklonivši teret procesiranje serijskih podataka u stvarnom vremenu sa mikroprocesora omogućen je rad više programa istovremeno na nivou OS-a, to je bio veliki napredak od MS DOS-a na kome je mogao raditi samo jedan program u datom trenutku. Krajem 1990. godina ISA kartice zamenjene su PCI karticama, migracija postojećeg koda napisanog za ISA kartice za ovu promenu zahtevala je dosta truda jer stari MS DOS programi su očekivali da serijski port radi sa fiksnim adresama i/o linija i prekida što više nije bio slučaj.

Jedan od prvih serijskih kompjuterskih komunikacionih uređaja bio je UART (Universal Asynchronous Receiver-Transmitter). Zbog napretka OS-a i izvršenja više programa simultano, CPU jedinice više nisu mogle da vrše kontrolu serijskog porta u stvarnom vremenu, UART je smišljen kao dodatak serijskom portu da bi se mitgirao ovaj problem i smanjio teret na računaru, UART je omogućio da način na koji aplikacije komuniciraju sa serijskim portom na modernim operativnim sistemima bude nevezan sa hardverom koji na niskom nivou implementira serijski

interfejs, na Windowsu podaci koji se šalju i primaju na serijskom portu više se ne prebacju karakter po karakter nego kao FIFO paketi. Neki rani niskobuđetni sistemi umesto UARTA koristili su pin na mikroprocesoru za slanje podataka tehnikom poznatom kao “bit banging”. Gordon Bell iz DEC-a dizajnirao je prvu verziju UART-a kome je upotreba bila povezivanje pisaćih mašina na interfejs komande linije kao i povezivanje uređaja u ranom internetu.

UART je postavši deo integrisanog kola serijskog porta standardizovao serijske portove, pre toga, kada se povezivalo na izlazni pin mikroprocesora za postizanje serijske komunikacije često su se koristili dizajni zatvorenog i nestandardnog tipa. Glavni napredak koji je UART doneo bila je automatska obrada signala iz analognih u digitalne preciznim uzorkovanjem (Engl. sampling).

UART prvobitno nije bio uređaj koji je stajao samo na jednom čipu ali vremenom je DEC napravio kompaktnu verziju. Western Digital je 1971. godine unapredio ovo u prvi javno dostupan UART model, WD1402A, još jedan popularan čip bio je SCN2651 iz signetics 2650 familije. Zavisno od proizvođača istorijski su korišćeni različiti termini za opis uređaja koji su imali UART sposobnosti. Intel je I8251 nazvao “programabilni komunikacioni interfejs”, MOS Technology 6551 je bio poznat kao “adapter za interfejs asinhrone komunikacije”. Termin “serijski komunikacioni interfejs” (Engl. SCI) iskorstila je Motorola 1975. godine za njihov start-stop uređaj sa asinhrono serijskim interfejsom koji je isto bio poznat kao UART. Zilog je napravio više jedinki “kontrolera za serijsku komunikaciju” (Engl. SCC).

Osamdesetih godina 20. veka napravljen je National Semiconductor 8250, popularan model UART-a koji je dosta puta kloniran. Jedna njegova upotreba bila je dodavanje asinhrone komunikacije preko adapterske kartice u prvobitnim IBM računarima. Devedesetih godina 20. veka noviji UARTovi razvijeni su sa baferima integrisanim u čipovima koji su omogućili podršku većih brzina transmisije bez gubitka podataka.

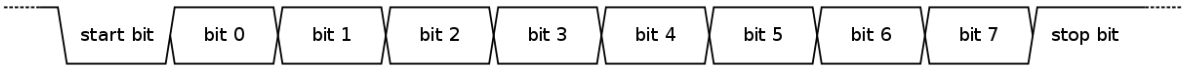
UART je podržavao samo asinhronu transmisiju podataka, sličan uređaj koji je nastao kasnije, USART (Universal Synchronous and Asynchronous Receiver-Transmitter) podržava i sinhronu i asinhronu transmisiju podataka. Serijske sposobnosti USART-a su prvobitno bile namenjene za protokole poput IBMovog STR (Synchronous Transmit-Receive), BSC (Binary Synchronous Communications), SDLC (Synchronous Data Link Control), kao i ISO-standard HDLC (High Level Data Link Control). U ranijim vremenima najbrži asinhroni modem mogo je raditi pri brzini od 300 bita u sekundi koristeći FSK (Frequency-Shift Keying) modulaciju dok su sinhroni modemi radili brzinama do 9600 b/s koristeći PSK (Phase-Shift Keying), IBM 5150 je primer uređaja sa maksimalnom brzinom prenosa od 9600 bodova u sekundi. Ovi protokoli su bili napravljeni tako da omoguće najbolji mogući protok sa tadašnjim analognim modemima.

Sinhrona transmisija mogla je da koristi malo preko 80% protoka asinhrone transmisije jer su start i stop bitovi bili nepotrebni.

Početkom 2000. godina većina računara kompatibilnih sa IBM-ovim računarom su zamenili RS-232 COM portove sa USB portovima koji mogu slati podatke mnogo brže. Zarad kompatibilnosti sa starijim uređajima napravljeni su eksterni uređaji gde čip konvertuje sa USB na UART za korisnike kojima je RS-232 i dalje bio potreban. Cypress Semiconductor i FTDI su jedni od najvećih komercijalnih distributera ovih čipova. Iako RS-232 portovi više nisu ugrađeni na modernim računarima, većina mikroprocesora ima UART stepen hardverski ugrađen u njima sa kojim se može komunicirati RS-232 ili RS-485 protokolima preko eksternih uređaja.

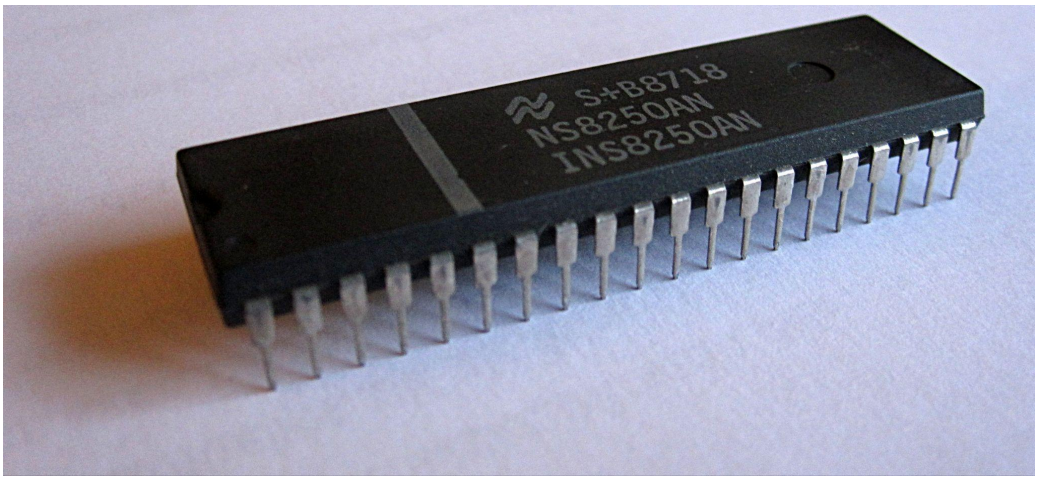
4.2 USART

USART (Universal Synchronous and Asynchronous Receiver-Transmitter**)** je vrsta uređaja za komunikaciju preko serijskog porta koji je zasebno integrisano kolo ili deo nekog postojećeg koje se može programski podesiti da radi u sinhronom ili asinhronom režimu (ista funkcionalnost kao i u UART-u). Takođe se mogu programski podesiti format i brzina prenosa.



SL 4.2.1 Format serijskog prenosa (CC IngenieroLoco)

USART konvertuje karaktere u asinhroni serijski format kao i da konvertuje nazad u karatere poštujući specifikaciju serijskog protokola. Koriste se kao serijski port i mogu povezati USB sa serijskim portom, u IBM računarima serijski portovi su implementirani sa jednim ili više UART.



SL 4.2.2 National Semiconductor 8250 UART (CC Nixdorf)

Rani model UARTA iz osamdesetih godina 20. veka bio je National Semiconductor 8250. Primeri uređaja koji koriste USART su automobili, smart i sim kartice, Intel 8251A i drugi mikrokontrolerski čipovi. USART se koristi u raznim protokolima kao što su RS-232 koji je 12 voltni sistem i RS-485 koji je 5 voltni sistem.

Bitovi se šalju jedan po jedan, tipično od najmanje značajnog bita do najznačajnijeg. U asinhronom radnom režimu ukoliko procesor ne prosledi karaktere USART-u do početka transmisije nekog okvira, transmisija tog okvira se otkazuje i dolazi do underrun greške. U sinhronom radnom režimu kroz liniju prenosa konstantno prolaze neki bitovi čak i ako nisu bitovi podataka jer će USART u zavisnosti od uređaja i protokola slati karakatere ili bitove kao indikaciju da je modem aktivan.

Slični protokoli za sinhronu serijsku telekomunikaciju postoje poput veoma rasprostranjenog IEEE 802.2 (Ethernet) protokola, link level u OSI referentnom modelu. USART stepeni se idalje standardno u mikrokontrolerima integrišu i koriste u ruterima koji se mogu povezati na eksternim CDU/DSU uređajima i najčešće koriste Cisco HDLC implemetanciju zatvorenog tipa ili IETF standard PPP (Point to Point Protocol) uređaje sa protokolima koji uokviraju podatke slično kao HDLC kao što je definisano RF 1662 standardom.

5. Serijska komunikacija Arduino Uno uređaja

Povezivanjem dodatnih uređaja na Arduino ploče omogućava se korišćenje dodatnih funkcionalnosti iz tih uređaja, na primer serijski se mogu povezati LCD displej, mikrokontroleri ESP8266, ESP32, bluetooth čipovi ili GSM modul i dodati IoT (Internet of Things) mogućnosti.

USART stepen ATmega328P mikrokontrolera omogućava mikrokontroleru da primi podatke preko serijskog porta dokle god 64 bajtni serijski bafer nije pun. USART stepen ATmega328P mikrontrolera obezbeđuje:

- Potpuni dupleks (nezavistan rad predajnika i prijemnika)
- Kontrolu parnosti
- Tri izvora prekida
- Asinhroni i sinhroni režim serijske transmisije

Brzine dostupne za prenos u Arduino serial monitoru idu od 300 bodova do čak 2000000 bodova; međutim, zbog loše optimizacije serijskog koda u Arduino bibliotekama teško je dobiti brzinu prenosa podataka preko 500 KB/s sem u kratkim intervalima ukoliko ne napišemo sopstvene funkcije ali tada gubimo mogućnost korišćenja nekih Arduino Serial metoda vezanih za serijski port.

Većina kompleksnih integrisanih kola (Engl. IC) za sinhronizaciju između delova kola koristi taktni signal. To je specifičan tip signala koji osciluje između visokog i niskog stanja i koristi se kao metronom radi koordinisanja rada strujnih kola. Jedini izuzeci koji ne koriste taktni signal su asinhrona kola poput asinhronih mikroprocesora.

Komunikacija sa spoljnim svetom je preko pinova PD0 (RX) i PD1 (TX), povezivanjem serijskog monitora pokrenuće se reset i program će početi rad ispočetka, ovo se događa jer arduno koristi DTR signal za reset, moguće je isključiti DTR signal u serijskim terminalima i time izbegnuti reset pri ponovnom povezivanju Arduino Uno mikrokontrolera.

ATmega16U2 čip u Arduino Uno sistemu omogućava povezivanje računara i serijskog porta, tačnije pinove 0 i 1 ATmega328P mikrokontrolera preko USB-a. Ranije verzije Arduino Uno i Ardunio Mega 2560 koristile su ATmega8U2 čip.

Preko serijskog porta se novi sketchevi prepisuju na Arduino Uno. Kada se ATmega328P čip na arduinu resetuje, bootloader proveri da li na serijskoj liniji ima novi program koji čeka da se

instalira - ako ima novog programa na serijskoj liniji bootloader ga prepiše preko postojećeg, u slučaju da nema, sistem se upali sa prethodnim programom i serijski port radi uobičajeno.

5.1 ATmega328P registri

Moguće je u Arduino IDE okruženju pristupiti registrima ATmega328 mikrokontrolera. Registri se podešavaju operacijama nad bitovima, primer bitshift i ili operacija nad bitovima se može videti u sledećem C++ programu:

```
#include <iostream>
#include <bitset>

int main() {
  int x=1; // 0b00000001
  x |= 0b11 << 2; // 12 (bitovi 11 pomereni za dva polja u levo)
  std::cout << x << ' 0b' << std::bitset<8>(x);
}
```

Izlaz ovog programa je: 13 0b00001101

• 5.1.1 UBRR0

Ovaj 12-bitni registar određuje brzinu serijskog prenosa, viših 4 bitova upisuju se u UBRR0H registar a zadnjih 8 bitova u UBRR0L registar. Oba uređaja moraju koristiti istu stopu bodova u serijskoj komunikaciji, generalno serijski uređaji će tolerisati grešku u stopi bodova do 5%, kako oba kraja mogu imati grešku u suprotnom smeru poželjno je da ni jedan nema grešku veću od 2.5%, u dokumentaciji mikrokontrolera ATmega328 data je tabela sa stopom grešaka za datu stopu bodova, u idealnoj situaciji za komunikaciju bez greška bila bi potrebna frekvencija oscilatora koja je deljiva sa 1.8432MHz

Tabela 5.1.1.1 podešavanje UBRR0 registra za frekvenciju oscilatora 16.0000MHz

Stopa bodova	U2X0 = 0		U2X0 = 1	
	UBRR0	Greška	UBRR0	Greška
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%

14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	17	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Maksimalna	1Mbps		2Mbps	

Dok je teorijski moguće ostvariti brzine i do 2M bodova u kratkim intervalima u režimu duple brzine (U2X0 = 1), u praksi je moguće jedino ukoliko se napišu sopstvene funkcije.

Formula sa kojom se može izračunati vrednosti registra UBRR0 za željenu stopu bodova:

$$UBBR0 = \frac{F_{CPU}}{BAUDRATE*16UL} - 1 \quad (2)$$

Ukoliko se koristi režim duple brzine (U2X0 = 1), koristi se sledeća formula:

$$UBBR0 = \frac{F_{CPU}}{BAUDRATE*8UL} - 1 \quad (3)$$

Primer upotrebe:

```
UBRR0=(F_CPU / (9600 * 16UL)) - 1;
```

- **5.1.2 UCSR0A**

Ovo je statusni registar kojim se mogu videti statusi serijskih prekida

Bit	7	6	5	4	3	2	1	0
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn
Read/Write	R	R/W	R	R	R	R	R/W	R/W
Initial Value	0	0	1	0	0	0	0	0

SL 5.1.2.1 UCSR0A registar

RXC0 se podesi na 1 kada ima nepročitanih podataka u prijemnom serijskom baferu

TXC0 se podesi kada se pošalju podaci i transmisioni bafer je ispražnjen

UDRE0 se podesi na 1 kada je transmisioni serijski bafer prazan i spreman za slanje

FE0 se podesi na 1 kada se dogodila greška pri primanju sledećeg bita iz bafera

DOR0 se podesi na 1 kada je prijemni serijski bafer pun i dogodio se data overrun

U2X0 se podešava na 1 kada se želi koristiti režim duple brzine u asinhronom režimu

UPE0 se podesi na 1 kada se dogodila greška pri proveri pariteta ukoliko je podešen

MPCM0 se podešava na 1 kada se želi koristiti mpcm na prijemniku

- **5.1.3 UCSR0B**

Ovo je kontrolni registar kojim se određuju prekidi za serijski prenos

Bit	7	6	5	4	3	2	1	0
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Initial Value	0	0	0	0	0	0	0	0

SL 5.1.3.1 UCSR0B registar

Postavljanjem RXCIE0 bita omogućava se RXC0 prekid pri primanju podataka

Postavljanjem TXCIE0 bita omogućava se TXC0 prekid pri slanju podataka

Postavljanjem UDRIE0 bita omogućava se UDRE0 prekid kada se isprazni serijski bafer

Postavljanjem RXEN0 bita omogućava se primanje serijskih podataka

Postavljanjem TXEN0 bita omogućava se slanje serijskih podataka

Postavljanjem UCSZ02 bita iz UCSR0B registra kao i UCSZ02 i UCSZ01 iz UCSR0C registra određuje se koliko se bitova podataka šalju

RXB80 je deveti najviši bit koji se mora pročitati pre ostatka preko UDR0 regitstra i prima kada se šalju 9 bitova podataka

TXB80 je deveti najviši bit koji se mora poslati pre ostata preko UDR0 registra i postavlja se kada se šalju 9 bitova podataka

Primer upotrebe:

```
UCSR0B |= (1 << RXCIE0) | (1 << RXEN0) | (1 << TXEN0);
```

- **5.1.4 UCSR0C**

Ovo je kontrolni registar koji određuje format serijske komunikacije usart stepena mikrokontrolera ATmega328, često se koristi u podrazumevanom stanju

Bit	7	6	5	4	3	2	1	0
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	0

SL 5.1.4.1 UCSR0C registar

Postavljanjem UMSEL01 i UMSEL00 bitova određuje se režim serijske komunikacije, moguća stanja su:

00 - Asinhroni režim

01 - Sinhroni režim

10 - Rezervisano stanje

11 - Master SPI (MSPIM)

Postavljanjem UPM01 i UPM00 bitova određuje se koja će se vrsta pariteta koristiti, moguća stanja su:

00 - Bez bitova parnosti

01 - Rezervisano

10 - Neparna parnost

11 - Parna parnost

Postavljanjem USBS0 bita određuje se koliko se stop bitova koriste, moguća stanja su:

0 - 1 stop bit

1 - 2 stop bita

Postavljanjem UCSZ02 bita iz UCSR0B registra i UCSZ01 i UCSZ00 bitova iz UCSR0C registra određuje se koliko se bitova podataka šalju, moguća stanja su:

Tabela 5.1.4.1 bitovi podataka za stanja UCSZ02, UCSZ01 i UCSZ00 bitova

UCSZ02	UCSZ01	UCSZ00	bitovi
0	0	0	5
0	0	1	6
0	1	0	7
0	1	1	8
1	0	0	Rezervisano stanje
1	0	1	Rezervisano stanje
1	1	0	Rezervisano stanje
1	1	1	9

UCPOL0 bit se koristi samo u sinhronom režimu i određuje kada će se u taktnom signalu podaci primati i slati, moguća stanja su:

0 - Podaci se šalju pri rastu logičkog stanja na signalu, podaci se primaju pri padu logičkog stanja na signalu

1 - Podaci se šalju pri padu logičkog stanja na signalu, podaci se primaju pri rastu logičkog stanja na signalu

- **5.1.5 UDR0**

Ovaj 8-bitni registar je registar podataka, koristi se i za primanje i za slanje podataka, mikrokontroler inteligentno upravlja ovim registrom ali trebalo bi imati na umu dok se upravlja ovim registrom da nije moguće čitanje i pisanje podataka istovremeno

Primer upotrebe:

```
while (!(UCSR0A & (1 << UDRE0))) {}
  UDR0=55;
```

- **5.1.6 DDRx**

DDRx registri odlučuju koji pinovi će biti ulazni a koji izlazni za port x, to mogu biti DDRB, DDRC i DDRD. Za svaki od ovih mogu se podesiti 8 bitova (sem za DDRC za

koji se mogu podesiti 7 jer port c ima 7 pinova, 6 kojih su priključeni i na ADC i 1 na reset), ima 23 ulazno izlaznih pinova koji se mogu ovako konfigurisati.

Bit 0 označava da je pin u izlaznom režimu sa visokom impedanskom a bit 1 označava da je pin u izlaznom režimu sa niskom impedanskom. Podrazumevano pinovi se podešavaju kao ulazni (0)

Primer upotrebe:

```
DDRD |= (1 << 2); // Podešava PD2 kao izlazni (1) pin
// Podešava PD6 kao ulazni (0) pin bez uticanja na druge bitove
DDRD &= ~(1 << 6);
```

- **5.1.7 PORTx**

PORTx registri odlučuju naponsko stanje na pinovima za port x, to mogu biti PORTB, PORTC i PORTD. Za svaki od ovih mogu se podesiti 8 bitova (sem za PORTC za koji se mogu podesiti 7 jer port c ima 7 pinova.

Kako efekat PORTx registara zavisi od DDRx registara, PORTx registri vezani su za voltažu i u ulaznom i u izlaznom režimu, pull up otpornik može slabom strujom snadbevati priključene komponente, u izlaznom režimu može snadbevati priključene komponente i do 40mA, ATmega328P koristi signale TTL nivoa

Tabela 5.1.7.1 stanje pina n za određena stanja bita n u DDRx i PORTx registarima

DDRx	PORTx	stanje pina
0	0	ulazni režim, isključen pull up otpornik
0	1	ulazni režim, uključen pull up otpornik
1	0	izlazni režim, 0 volta
1	1	izlazni režim, 5 volta

Primer upotrebe:

```
DDRD=0b01000100; // Podešava PD2 i PD6 kao izlazne pinove
PORTD=(1 << 6); // Postavlja visoko naponsko stanje na pin PD6
```

- **5.1.8 PINx**

PINx registri čitaju logično naponsko stanje na pinovima u ulaznom režimu za port x, to mogu biti PINB, PORTC i PORTD. Za svaki od ovih mogu se pročitati 8 bitova (sem za PORTC za koji se mogu podesiti 7 jer port c ima 7 pinova.

Kako su PINx registri 8 bitni, vrednosti pročitane iz ovih registara mogu jedino biti digitalne, 0 i 1

Primer upotrebe:

```
Serial.println(PIND >> 3 & 1); // Čita logičko stanje na PD3
```

- **5.1.9 EICRA**

Ovo je kontrolni registar kojim se podešavaju INT1 i INT2 prekidi.

Bit	7	6	5	4	3	2	1	0
(0x69)	–	–	–	–	**ISC11**	**ISC10**	**ISC01**	**ISC00**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

SL 5.1.9.1 EICRA registar

Postavljanjem ISC01 i ISC00 bitova određuje se kontrola prekida INT0 na pinu PD2 (2 na Arduino ploči)

Postavljanjem ISC11 i ISC10 bitova određuje se kontrola prekida INT1 na pinu PD3 (3 na Arduino ploči)

Moguća stanja su:

00 - zahtev za prekidom generiše se niskim naponskim stanjem

01 - zahtev za prekidom generiše se promenom logičkom stanja

10 - zahtev za prekidom generiše se opadajućom ivicom

11 - zahtev za prekidom generiše se rastućom ivicom

- **5.1.10 EIMSK**

Ovo je registar kojim se omogućavaju prekidi INT1 i INT0, može se videti kada se dogodio prekid u EIFR registru

Bit	7	6	5	4	3	2	1	0
0x1D (0x3D)	–	–	–	–	–	–	**INT1**	**INT0**
Read/Write	R	R	R	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

SL 5.1.10.1 EIMSK registar

Postavljanjem bitova na INT1 i INT0 polja određuje se dozvola INT0 i INT1 prekida

Moguća stanja su:

INT0 = 0 - zabranjen prekid sa INT0

INT1 = 0 - zabranjen prekid sa INT1

INT0 = 1 - dozvoljen prekid sa INT0

INT1 = 1 - dozvoljen prekid sa INT1

5.2 Arduino metode

Navešćemo neke od osnovnih i bitnih metoda pri korišćenju serijske komunikacije Arduino Uno sistema, pozivaju iz Serial biblioteke koja je automatski uključena u Arduino projekte. Biće dat primer upotrebe kao i potpis metode gde to jasnije predstavlja upotrebu metoda:

- ### 5.2.1 Serial.begin

  Serial.begin(stopa bodova, config(neobavezno))

  Pokreće serijsku komunikaciju, argument podešava brzinu prenosa bodova, podrazumevano se prvo proba režim duple brzine (U2X0 = 1). Podrazumevani config je SERIAL_8N1. Nakon upotrebe ove metode digitalni pinovi 0 (RX) i 1 (TX) se ne mogu koristi kao IO pinovi.

- ### 5.2.2 Serial.end

  Zaustavlja serijsku komunikaciju i omogućava uptrebu digitalnih pinova 0 (RX) i 1 (TX) kao IO pinove.

- ### 5.2.3 Serial.print

  Serial.print(podaci, brojevni sistem(neobavezno))

Šalje brojeve, karaktere i stringove preko serijskog porta enkodirane u ASCII formatu tako da se mogu lako pročitati, mogu se proslediti funkcije koje vraćaju stringove poput F (podaci). Ako se prosledi celobrojni broj može se proslediti brojevni sistem kao argument (BIN - 2, OCT - 8, HEX - 16, DEC - 10 (podrazumevano). Ukoliko se prosledi floating point broj može se proslediti broj decimalnih mesta (podrazumevana vrednost je 2). Vraća broj bajtova koji su poslati.

Primeri upotrebe i izlazne vrednosti:

```
Serial.print("10"); // Izlaz: 10
Serial.print(10); // Izlaz: 10
Serial.print(10, HEX); // Izlaz: A
Serial.print(1.0123, 0); // Izlaz: 1
Serial.print(1.0123, 1); // Izlaz: 1.0
Serial.print(1.0); // Izlaz: 1.00
```

- ### 5.2.4 Serial.println

Serial.println(podaci, brojevni sistem(neobavezno))

Radi isto što i print metoda i na kraju pošalje karaktere \r\n koji dodaju novi red.

Primer upotrebe i izlazne vrednosti:

```
Serial.println("Zdravo"); // Izlaz: Zdravo\r\n
```

- ### 5.2.5 Serial.write

Šalje brojeve, karaktere, stringove i nizove preko serijskog porta i ne enkodira ih, pravi razliku između "10" i 10. Ukoliko se prosledi pokazivač na niz treba se proslediti kao drugi argument koliko vrednosti da pročita iz niza. Vraća broj bajtova koji su poslati.

Primeri upotrebe i povratne vrednosti:

```
size_t num=Serial.write(65); // Šalje slovo A, num je 1
size_t pom=Serial.write("hello"); // Šalje hello, pom je 5
uint8_t p[3]={65, 66, 67}; Serial.write(p, 3); // Šalje ABC
```

- ### 5.2.6 Serial.available

Vraća broj bajtova koji su primljeni preko serijskog porta i čekaju da se pročitaju sa serijskog bafera.

Primer upotrebe i povratne vrednosti:

```
// Vraća 0 kada je prazan bafer, maksimalno 63
int num=Serial.available();
```

- **5.2.7 Serial.read**

Čita jedan bajt iz serijskog bafera, dostupna je readString metoda koja čita ceo bafer.
Primer upotrebe i povratne vrednosti:

```
uint8_t data=Serial.read(); // Popuje 1 bajt iz bafera
```

- **5.2.8 Serial.peek**

Vraća sledeći bajt (karater) iz serijskog bafera bez da ga izbaci iz bafera.

- **5.2.9 Serial.availableForWrite**

Vraća broj bajtova (karaktera) koji se mogu upisati u transmisioni serijski bafer bez da se blokira slanje. Serijska komunikacija u Arduino sistemima je asinhrona, što znači da ukoliko ima prostora u transmisionom serijskom baferu Serial.print() i Serial.write() metode će upisati karaktere tamo i vratiti povratne vrednosti pre nego što se bilo koji karakter pošalje. Ako je transmisioni serijski bafer pun onda će ove funkcije blokirati izvršenje drugih operacija dok ne bude bilo dovoljno prostora u baferu.
Primer upotrebe i povratne vrednosti:

```
int num=Serial.availableForWrite(); // Vraća 63 kada je prazan
```

- **5.2.10 Serial.flush**

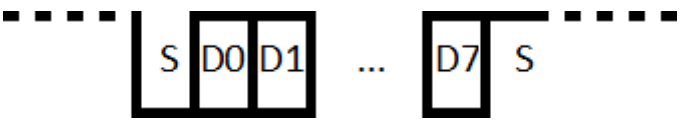
Sačeka da slanje serijskih podataka završi tj. da se isprazni transmisioni serijski bafer.

5.3 Format

U asinhronom režimu mora se odlučiti format transmisije. Dostupne opcije su:

- Potpuni ili polovični dupleks
- Broj bitova po karakteru (obično 8 bita ali neki raniji transmiteri su koristili 5, 6 ili 7 bit)
- Endian redosled bajtova
- Brzina bitova po sekundi prenosne linije (ekvivalentna stopi bodova kada svaki simbol predstavlja jedan bit). Neki sistemi mogu koristiti automatski detekciju brzine.
- Da li se koristi bit parnosti
- Parna ili neparna parnost, ukoliko se koristi bit parnosti
- Broj stop bitova (mora se poslati makar onoliko koliko prijemnik zahteva)

- Simboli oznake i razmaka (istorijski tok struje u ranoj telegrafiji, polaritet voltaže u EIA, RSA-232, polaritet promene frekvencije u FSK, itd)



SL 5.3.1 Podrazumevani format serijskog prenosa Arduino Uno

Arduino sistem većinu ovih stavki sam podešava ali moguće je podesiti neki parametre, za prenos podrazumevano se koristi 1 start bit, 8 bitova podataka, ni jedan bit parnosti i 1 stop bit, popularna stopa bodova je 9600, drugi formati se mogu podesiti putem UBRR0 i UCSR0C registra ali najčešće je u upotrebi format 9600/8-N-1. Brzina od 9600 bodova u ovom formatu znači da će se u svakoj transmiji poslati 10 bitova od kojih su 8 bitovi podataka, što znači da je brzina prenosa podataka zapravo 80% od brzine trasmisije, više bitova se šalju. Prvo se šalje bit najmanje težine (LSB), zadnje se šalje bit najveće težine (MSB), ovaj režim zove se little endian.

5.4 Softverski i hardverski serial

Arduino Uno ima hardversku podršku za serijsku komunikaciju preko pinova 0 i 1. Softverski se može emulirati serijska komunikacija i preko ostalih digitalnih pinova bit banging tehnikom pomoću dodatnih biblioteka poput SoftwareSerial (bazirana od verzije 1.0 na NewSoftSerial bilbiloteci). Korišćenjem SoftwareSerial biblioteke moguće je imati više softverskih serijskih portova sa brzinama do 115200 bps što je sigurna vrednost i za hardverski serial, moguće je koristiti i invertovati signale za uređaje koji to zahtevaju ali bolje je koristiti hardverski gde je to moguće jer SoftwareSerial blokira prekide dok vrši primo-predaju.

SoftwareSerial ima određena ograničenja:

- Ne mogu se istovremeno slati i primati podaci
- Ako se koristi više softverskih serijskih portova samo jedan port može u datom trenutku primati podatke.
- Na određenim pločama na primer Arduino Micro i Mega 2560 nema podrške na svim pinovima za prekide po promeni logičkog stanja.

Da bi se koristila ova bilbioteka mora se prvo uključiti u projekat

```
#include <SoftwareSerial.h>
```

Alternativna biblioteka koja podržava istovremeno primanje i slanje podataka je AltSoftSerial, ova biblioteka prevazilazi neke fundamentalne probleme sa SoftwareSerial bibliotekom, ali ima drugih sopstvenih ograničenja.

5.5 Preglednost podataka za ljude ili mašine

Za prikaz nula i jedinica u ASCII kodu se koriste brojevi 48 i 49, ovo je kontra produktivno kada želimo upoređivati brojeve umesto karaktere. U ovom radu poslati bitovi biće štelovani za lakše detektovanje i analiziranje na mašinama poput osciloskopa umesto prikazivanja na standardnom izlazu terminala. Metoda Serial.print() iskorišćena je ili izostavljena na više mesta umesto Serial.println() gde bi ona bila adekvatnija za ljudsko čitanje podataka.

6. Poređenje paralelne i serijske komunikacije

Paralelna i serijska komunikacija obe imaju svoje prednosti i mane. Osnovna razlika između njih je broj električnih konduktivnih linija koji se na fizičkom sloju osi referentnog modela koriste za prenos bitova. U paralelnoj komunikaciji može se poslati više bitova istovremeno, obično se šalje 2 do 8 bitova (jedan bajt) po prenosu što znači da je transmisija podataka paralelnom komunikacijom brža nego transmisija istih podataka serijskom komunikacijom koja šalje jedan bit podatka po prenosu. Za uzvrat, serijska implementacija ima svoje dobre strane, jedna velika prednost serijske komunikacije je smanjeni broj žica u kablovima i pinova potrebnih za prenos što značajno smanjuje veličinu i kompleksnost konektora i smanjuje cenu implementacije. Dizajneri uređaja poput pametnih telefona imaju veliku korist od konektora/portova koji su mali po veličini, izdržljivi a i dalje imaju visoke performanse.

Tabela 6.1 Poređenje paralelne i serijske komunikacije

Paralelna komunikacija	Serijska komunikacija
Skuplja implementacija jer koristi više linija prenosa	Jeftinija implementacija jer koristi jednu liniju prenosa
Veća brzina prenosa, prenosi više bitova po transmisiji ali je podložna clock skewu	Manja brzina prenosa, prenosti 1 bajt po transmisiji ali je konzistentnija
Mogućnost ometanja signala ako su žice međusobno preblizu jedna drugoj	Manje potencionalnih smetnji signalu
Primenljivija na kraćim distancama zbog većeg broja žica	Primenjljiva na bilo kojim distancama
Obično polu dupleks	Obično potpuni dupleks
Jednostavnija implementacija	Kompleksnija implementacija

Detaljna poređenja:

- Brzina: Na prvu pomisao, brzina prenosa paralelnog data linka jednaka je broju bitova (linija prenosa) koji se šalju u jednom momentu pomnoženo sa stopom prenosa bitova (brojem bitova koji svaka linija šalje u jedinici vremena) tako da u teoriji dodavanjem još jedne linije prenosa može se duplirati stopa prenosa podataka. U praksi, clock skew ograničava stopu prenosa bitova na brzinu prenosa najsporije linije prenosa što znači da brzina prenosa nije tačno proporcionalna broju linija prenosa. Zbog ovoga brzina prenosa

jedne linije prenosa u serijskoj komunikaciji može biti veća nego brzina prenosa individualnih linija prenosa u paralelnoj komunikaciji jer oba uređaja na “svim” linijama sigurno koriste istu stopu bodova i CPU bus.

- Pouzdanost: Crosstalk, pojava gde električni signal jedne transmisije utiče na drugi pravi smetnje u linijama prenosa paralelne komunikacije. Na manjim brzinama prenosa i distancama ovi efekti su zanemarljivi ali što je veće distanca prenosa to će Crosstalk i Clock skew imati veći uticaj na pakete prenete paralelnom komunikacijom, zbog toga serijska komunnikacija je pouzdanija na većim distancama.
- Kompleksnost: Paralelni prenos podataka se lako hardverski implementira. Napraviti paralelni port je relativno prosto jer sve što je potrebno je električna sklopka koja će prekopirati podatke na magistralu podataka, tako da je paralelna komunikacija logični izbor u projektima sa manje resursa. Većina podataka prebačenim serijskom komunikacijom mora se prvo konvertovati UART uređajem nazad u format podataka paralelne komunikacije pre nego što se kabli može povezati direktno na magistralu podataka.

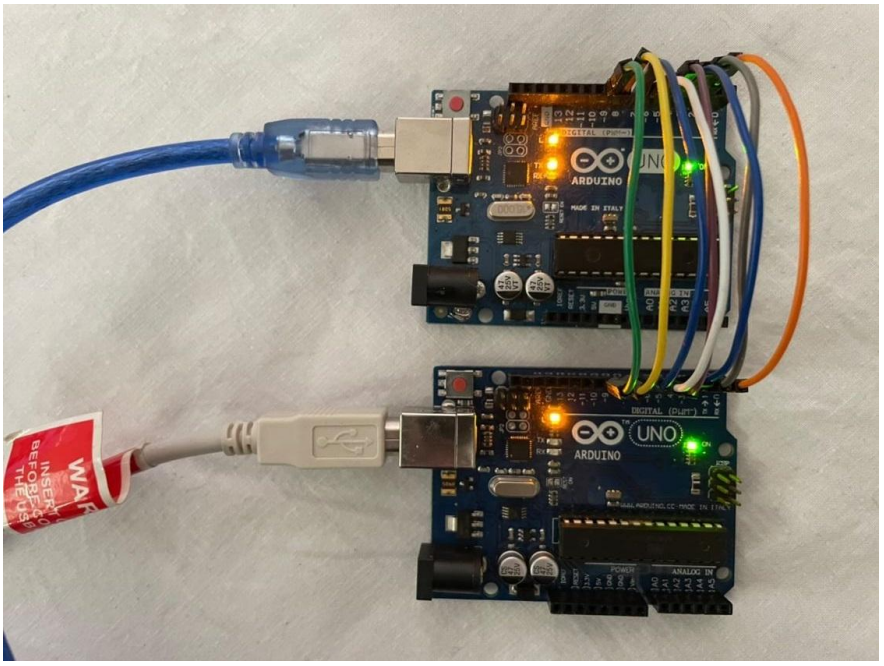
6.1 Korišćena oprema

Za svrhe ovog eksperimenta potrebno je

2x Arduino Uno R3

9x Konektora muško na muški

ATmega328 ploča ima 2 porta sa 8 pinova, PD i PB preko kojih bi se teorijski moglo povezati 8 linija za paralelnu komunikaciju. U praksi, Arduino Uno ploča kao i ploče slične njoj pinove PB6 i PB7 koriste za povezivanje eksternog oscilatora, zbog ovoga kao generične IO pinove možemo koristiti samo pinove PB0 - PB5, pritom je PB5 povezan na led diodu na Arduino Uno ploči. Kako ćemo pinove PD0 i PD1 koristiti za serijski prenos podataka do računara preko USB konekcije, za parelelnu komunikaciju možemo koristiti jedino preostale pinove iz oba porta, digitalne pinove 2 - 9. Za tačnije merenje brzine paralelnog prenosa ima potrebe za Arduino Mega pločom zbog veće dostupnosti pinova. Potrebno je i povezati GND pinove zarad izjednačavanja referente voltaže.



SL 6.1.1 Povezivanje paralelne komunikacije

6.2 Podešavanja predaoca

Podesićemo iste pinove na predaocu kao izlazne koje će na primaocu biti podešeni kao ulazni, od pina 9 će se slati MSB

```
void setup() {
  DDRB=0b11;
  DDRD=0b11111100;
}
```

Implementiraćemo najprostiji način slanja podatka, ovako možemo meriti brzinu jedino ako primaoc zna koji su podaci koji se treba primiti, u pravim uređajima postoje definisani standardi paralelne komunikacije

```
void loop() {
  for (int i=0; i < 256; i++) {
   PORTB=(i & 192) >> 6;
   PORTD=(i & 63) << 2;
  }
}
```

6.3 Podešavanja i merenja primalaca

Da bi izglatkali rezultate merenja koristićemo pokretni prosek, za njegovu implementaciju koristićemo queue strukturu podataka iz biblioteke ArduinoQueue

```
#include <ArduinoQueue.h>
```

Definisaćemo koliko elementa će biti u pokretnom proseku

```
#define __moving 5
```

Deklarisaćemo queue i inicijalizovaćemo prosek na 0 kome ćemo dodati i menjati vrednosti

```
ArduinoQueue<double> parallelQ(__moving);
double parallelAverage=0.;
```

Na početku izvršenja izvšićemo početna merenja pomerajuće sume

```
void setup() {
  Serial.begin(9600);
  for (int i=1; i <= __moving; i++) {
   parallelQ.enqueue(measureSpeed());
   parallelAverage += (parallelQ.getTail() - parallelAverage) / i;
  }
}
```

Funkcija koja će proceniti brzinu prenosa podataka preko pinova

```
double measureSpeed() {
  unsigned long start=micros();
  for (int i=0; i < 256; i++) {
   while (((PINB & 3) << 6 | (PIND & 252) >> 2) != i);
  }

  // 255 pomnoženo sa odnosom u sekundama
  return 255000000.0 / (micros() - start);
}
```

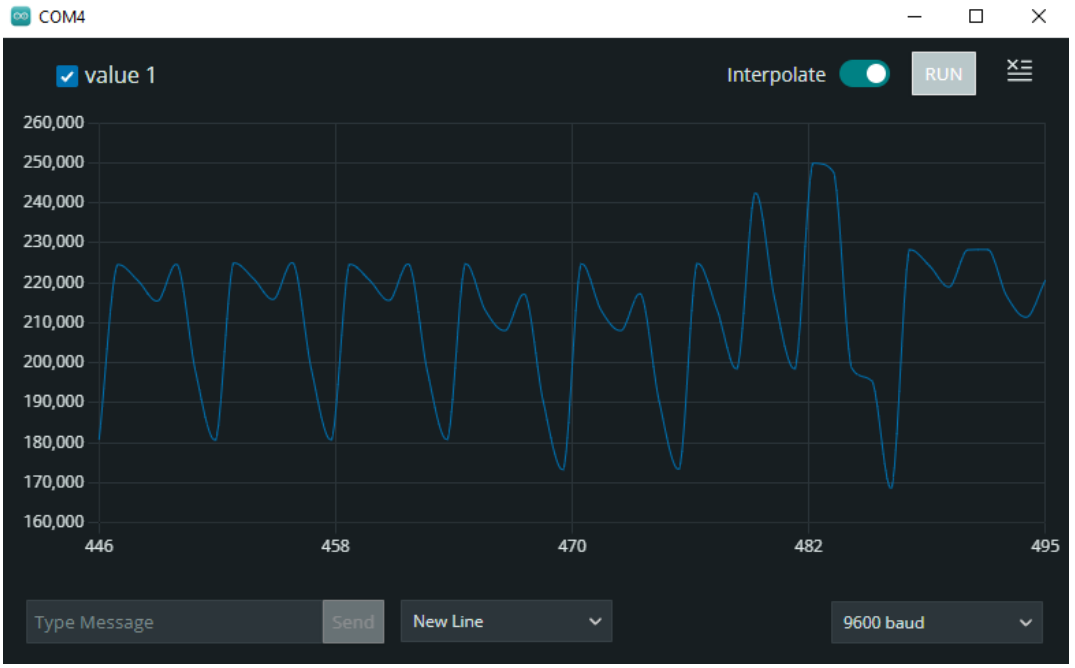
Ovo je funkcija kojom ćemo zameniti zadnju vrednost pomerajućeg proseka

```
double replaceInAverage(double &average, ArduinoQueue<double> &q,
double nValue) {
  average=(__moving * average - q.dequeue() + nValue) / __moving;
  q.enqueue(nValue);
  return average;
}
```

Na svakoj iteraciji loop petlje izvršiće se merenje paralelne brzine komunikacije i nova vrednost će biti dodata u pomerajući prosek. Za prikaz izmerenih vrednosti koristićemo Arduino Serial Plotter, moguće je prikazati više vrednosti u serial plotter programu razdvajanjem sa razmakom ili tabulatorom nakon prikazivanja prve vrednosti Serial.print() metodom, zadnju vrednost mora pratiti nova linija (\r\n), Serial.println() će dodati ove karaktere.

```
  void loop() {
    Serial.println(replaceInAverage(parallelAverage, parallelQ,
measureSpeed()));
  }
```

Rezultati merenja brzine paralelne komunikacije mogu se videti na osnovu stope bodova



SL 6.3.1 Arduino Serial Plotter

U toku testiranja 260,000 je bila najveća dostignuta vrednost, moguće je dostići više implementacijom paralelnog protokola. Zbog limitacija micros funkcije ovaj program ne može raditi duže od 70 minuta, nakon toga micros vrednost će se vratiti na 0 zbog overflow-a.

7. Serijska komunikacija između dva Arduino Uno uređaja

USB je limitiran na polu dupleks režim i ne možemo dvosmerno slati podatke računaru ali možemo izmeriti logično stanje na pinovima RX i TX kada se podaci ne čitaju na računaru

7.1 Korišćena oprema

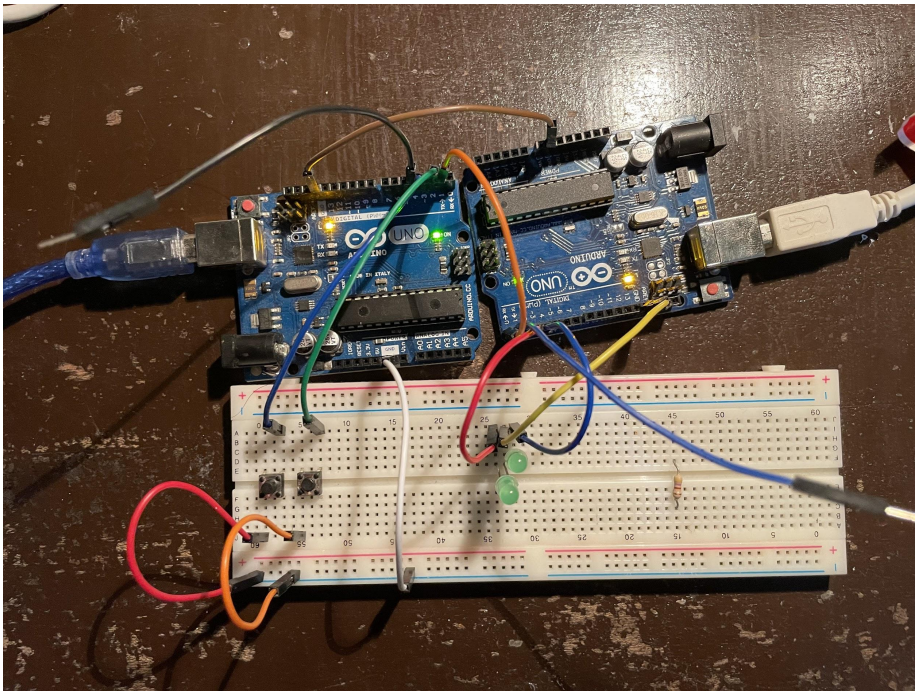
Za svrhe ovog eksperimenta potrebno je

2x Arduino Uno R3

2x Dugmeta

12x Konektora muško na muški

1x Protoploča



SL 7.1.1 Povezivanje dva Arduino Uno mikrokontrolera

RX i TX pinovi mikrokontrolera su povezani u toku izvršenja programa ali bi trebali biti odvezani prilikom prepisivanja softvera, u suprotnom neće biti moguće završiti prepisku

7.2 Podešavanje prijemnika

Na prijemniku ćemo koristiti metode dostupne uz Arduino. Da bi započeli komunikaciju sa stopom bodova od 9600 koristićemo metodu Serial.begin(). Ona se koristi u funkciji setup() koja

se pokreće prilikom svakog učitavanja programa i pokreće se sa memorijske lokacije 0 kada se vrši softverski reset tako da ćemo za simulaciju vraćanja u početno stanje u istoj postaviti svih 20 (uključujući analog pinove) pinova dostupnih na Arduino Uno ploči kao ulazne pinove

```
void setup() {
  Serial.begin(9600);
  for (uint8_t i=0; i < 20; i++)
   pinMode(i, INPUT);
}
```

Napisaćemo funkciju koja omogućava uzastopno čitanje serijskih podataka čekanjem prijema

```
uint8_t readw() {
  while (!Serial.available()) {}
  return Serial.read();
}
```

Napisaćemo funkciju koja će nakon početnog stanja menjati režim pina. Mogući režimi su input (0), output (1) i input_pullup (2) a parametri se primaju u serijki bafer preko RX pina.

```
void pinmode() {
  uint8_t pin=readw();
  pinMode(pin, readw());
}
```

Napisaćemo funkciju za promenu logičkog stanja na pinovima. Ova funkcija će zavisno od bitova u serijskom baferu izvršiti analogWrite koji prima vrednosti između 0 i 255 ili digitalWrite koji prima vrednosti 0 i 1

```
void pinwrite() {
  uint8_t pin=readw();
  uint8_t value=readw();

  if (readw())
   analogWrite(pin, value);
  else
   digitalWrite(pin, value);
}
```

Sledeće što nam je potrebno je funkcija koja će pročitati logičko stanje na pinovima i poslati ga preko TX pina, zavisno od bitova u serijskom baferu ova funkcija će izvršiti ili analogRead i poslati karakter između 0 i 255 ili digitalRead i poslati karakter 0 ili 1

```
void pinread() {
  uint8_t pin=readw();
```

```
  if (readw())
   Serial.print(analogRead(pin));
  else
   Serial.print(digitalRead(pin));
}
```

Za kontrolu toka napravićemo funkciju koja pravi razmak u radu programa u milisekundama. Arduino delay funkcija potpuno pauzira izvršenje programa (sem prekida) što može biti loše.

```
void delayf() {
  delay((readw() << 8) + readw());
}
```

Napisaćemo i funkciju koja će praviti razmak u radu programa u mikrosekundama - ova funkcija koristi timer integrisan u ATmega328P. Najveća vrednost sa kojom će ova funkcija proizveti precizan razmak je 16383, tako da za veće razmake treba koristiti običnu delay funkciju

```
void delaymicroseconds() {
  delayMicroseconds((readw() << 8) + readw());
}
```

write funkciju ćemo koristiti za slanje bajta bez konvertovanja u karakter

```
void writef() {
  Serial.write(readw());
}
```

Napisaćemo i naprednu funkciju koja će pročitati logičko stanje na pinu i invertovati ga

```
void pinchange() {
  uint8_t pin=readw()
  digitalWrite(pin, !digitalRead(pin));
}
```

Sledeća funkcija će staviti visoko logičko stanje na pin i nakon određenog vremena staviti nisko logičko stanje

```
void pinclick() {
  uint8_t pin=readw();
  digitalWrite(pin, HIGH);
  delayf();
  digitalWrite(pin, LOW);
}
```

Da bi iskoristili ove funkcije napravićemo niz funcs sa pokazivačima na njihove memorijske adrese, kao i memorijsku adresu reset funkcije (0)

```
void (*funcs[])() = {
  pinmode,
  pinwrite,
  pinread,
  delayf,
  delaymicroseconds,
  0,
  writef,
  pinchange,
  pinclick
};
```

U funkciji koja se konstantno pokreće u Arduinu proveravaćemo da li su poslati neki bajtovi preko serijskog porta i ako jesu proslediti taj bajt prethodno napravljenom nizu. Bajt 0 biće pinmode u funcs nizu. Ova implementacija pretpostavlja da će se na prijemniku uvek slati korektni podaci za svaku funkciju.

```
void loop() {
  (*funcs[readw()])();
}
```

7.3 Podešavanje predajnika

Na predajniku ćemo koristiti AVR C funkcije uz Arduino metode za demonstraciju funkcionalnosti, prijemink očekuje uvek primiti korektne podatke tako da svaka provera podataka dešavaće se na predajniku. Prvo ćemo izraćunati vrednost stope bodova koja se treba upisati u UBBR0 registar po datoj formuli i sačuviti vrednost preprocesorskom direktivom

```
#define UBRR_VALUE ((F_CPU / (9600 * 16UL)) - 1)
```

Definisaćemo globalne varijable za kontrolu toka i izvršenje programa

```
uint8_t cc=255;
bool execute=0;
```

Na početku izvršenja programa podesićemo stopu bodova preko UBBR_VALUE makro vrednosti kao i podesti stanje na pinovima i konfigurisati prekide

```
void setup() {
  UBRR0=UBRR_VALUE;
  // Omogućava USART transmisiju
  UCSR0B |= (1 << TXEN0);

  // Postavlja pinove 2 i 3 u input_pullup režim
```

```
  PORTD |= (1 << 2) | (1 << 3);

  // Postavlja pin 4 u izlaznom režimu
  DDRD |= (1 << 4);

  // Omogućava INT0 i INT1 prekida
  EIMSK |= (1 << INT0) | (1 << INT1);

  // Aktivira INT0 i INT1 prekide pri opadajućoj ivici
  EICRA |= (1 << ISC01) | (1 << ISC11);
}
```

Napisaćemo funkciju koja će se prva izvršiti i podesiti pinove 2 i 3 u izlaznom režimu

```
void pinmode() {
  uint8_t data[]={0, 2, 1};
  Serial.write(data, 3);

  uint8_t data2[]={0, 3, 1};
  Serial.write(data2, 3);

  execute=0;
}
```

Napisaćemo funkciju koja postavlja visoko naponsko stanje na pinu 2

```
void pin2on() {
  uint8_t data[]={1, 2, 1, 0};
  Serial.write(data, 4);
}
```

Napisaćemo funkciju koja prethodno postavljeno visoko naponsko stanje na pinu 2 vraća na nisko naponsko stanje

```
void pin2off() {
  uint8_t data[]={1, 2, 0, 0};
  Serial.write(data, 4);
}
```

Napisaćemo funkciju koja demonstrira komande pinchange koja invertuje logično stanje na pinu, delay koja pravi pauzu u izvršenju programa i demonstrira pinclick koji postavlja visoko naponsko stanje na pinu u trajanju datog vremena

```
void changepins() {
  // invertuje logično stanje na pinu 2
```

```
  uint8_t data[]={7, 2};
  Serial.write(data, 2);

  // pravi pauzu od 1000 milisekundi u izvršenju programa
  uint8_t data2[]={3, 0b00000011, 0b11101000};
  Serial.write(data2, 3);

  // invertuje logično stanje na pinu 2
  uint8_t data3[]={7, 2};
  Serial.write(data3, 2);

  // stavlja visoki napon na pinu 3 u trajanju od 1000 milisekundi
  uint8_t data4[]={8, 3, 0b00000011, 0b11101000};
  Serial.write(data4, 4);

  // pravi pauzu u izvršenju programa od 16000 mikrosekundi
  uint8_t data5[]={4, 0b00111110, 0b10000000};
  Serial.write(data5, 3);

  // invetuje logično stanje na pinu 2
  uint8_t data6[]={7, 2};
  Serial.write(data6, 2);

  // pauzira izvršenje programa zbog pauza
  execute=0;
}
```

Napisaćemo funkciju koja šalje bajt 33 i čita logičko stanje Arduino pina 2 postavljenog na visoko naponsko stanje u prethodnom koraku

```
void writef() {
  uint8_t data[]={6, 33};
  Serial.write(data, 2);

  uint8_t data2[]={2, 2};
  Serial.write(data2, 2);
}
```

Napisaćemo funkciju koja će izvršiti reset na predajniku i prijemniku

```
void reset() {
  cc=255;
```

```
  execute=0;
  Serial.write(5);
}
```

Napravićemo niz commands koji će sadržati memorijske adrese funkcija koje ćemo pozivati u redosledu preko loop funkcije

```
void (*commands[])() = {
  pinmode,
  pin2on,
  pin2off,
  changepins,
  writef,
  reset
};
```

U glavnoj funkciji na svakom izvršenju poslaćemo signal za proveru na pinu PD4 pre izvršenja svake komande

```
void loop() {
  PORTD &= ~(1 << 4);
  delayMicroseconds(100);
  PORTD |= (1 << 4);

  if (execute)
   (*commands[cc])();
  delay(10);
}
```

INT0 prekid koji će izvršiti narednu komandu

```
ISR(INT0_vect) {
  if (cc < 5 || cc == 255)
   cc++;
  execute=1;
}
```

INT1 prekid koji će ponoviti komandu

```
ISR(INT1_vect) {
  if (cc != 255)
   execute=1;
}
```

7.4 Tok komunikacije

sreg dozvoljava prekid

as electrical noise can be interpreted as zero or 1 unpredictably.

To avoid this problem, the ATmega328 has an interesting feature. It is a pull-up resistor

kontrolna petlja dakle očitava ulaze i zatim podešava izlaze u skladu sa svojim programom. Petlja se stalno ponavlja dok traje kontrola procesa

Biće moguće videti na osciloskopu

8. Zaključak

On 21 June 2018, the "world's smallest computer" was announced by the University of Michigan. The device is a "0.04mm3 16nW wireless and batteryless sensor system with integrated Cortex-M0+ processor and optical communication for cellular temperature measurement." It "measures just 0.3 mm to a side—dwarfed by a grain of rice. Microcontrollers usually contain from several to dozens of general purpose input/output pins (GPIO). Koriste se u najrazličitijim modernim uređajima. U robotima, telekomunikacionim uređajima, satelitima, automobilima, instrumentima, mobilnim telefonima, kamerama, kućnim uređajima kao što su mašine za pranje rublja, mikrotalasne rerne, kućnim pekarama hleba i drugde. Mikrokontroleri su od sedamdesetih godina prošlog veka imali brz razvoj. Sve više i više ranije odvojenih kola je integrisano, postoje i DSP (DSP) modeli podešeni za brze matematičke operacije sa proširenim setom instrukcija.[27] Micro-controllers have proved to be highly popular in embedded systems since their introduction in the 1970s. A typical home in a developed country is likely to have only four general-purpose microprocessors but around three dozen microcontrollers. A typical mid-range automobile has about 30 microcontrollers. They can also be found in many electrical devices such as washing machines, microwave ovens, and telephones.

Literatura

[1] ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash, https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf, Avgust 2021.

[2] Intel 8251A Programmable Communication Interface, http://www.nj7p.org/Manuals/PDFs/Intel/205222-002.pdf, Avgust 2021.

[3] Zoran Milivojević, *Mikrokontroleri arhitektura 805*, Niš, Punta, 2005.

[4] David Harris, Sarah Harris, *Digital Design and Computer Architecture Second Edition*, Morgan Kaufmann, Waltham, 2012.

[5] Simon Monk, *Programming Arduino: Getting Started with Sketches, Second Edition*, New York, McGraw Hill Education, 2016.

[6] Microprocessor chronology, https://en.wikipedia.org/wiki/Microprocessor_chronology, Oktobar 2021.

[7] Intel, *Embedded Microcontrollers*, Santa Clara, Semiconductor Product Groups, 1994.

[8] ATMEGA328PB, https://www.microchip.com/en-us/product/ATMEGA328PB, Februar 2022

[9] Colin A, *Programming for Microprocessors*, Boston, Butterworth, 1979

[10] ASCII Table, https://www.asciitable.com, Februar 2022.

[11] John Nussey, *Arduino For Dummies 2nd Ed*, John Wiley & Sons, Hoboken, 2018.

[12] Wrinkles and the MCS-51, https://www.intel.com/content/www/us/en/history/virtual-vault/articles/wrinkles-and-the-mcs-51.html, Februar 2022.

[13] Bernard Finn, *Exposing Electronics*, CRC Press, Washington, 2000

[14] PCIe, USB and Ethernet Serial Devices, https://www.brainboxes.com/files/pages/support/white-papers-and-presentations/Brainboxes%20Serial%20Products%20and%20Software%20Overview.pdf, Mart 2022.

[15] David R. Smith, *Digital Transmission Systems*, Springer, New York, 2004.

[16] Claus Kuhnel, *AVR RISC Microcontroller Handbook*, Boston, Elsevier Science, 1997.

[17] Arduino Hardware, https://www.arduino.cc/en/Main/Products, Oktobar 2021.

[18] List of common microcontrollers, https://en.wikipedia.org/wiki/List_of_common_microcontrollers, Oktobar 2021.

[19] Friedman, Eby G. (May 2001). "Clock Distribution Networks in Synchronous Digital Integrated Circuits"

[20] Sergio Benedetto, Ezio Biglieri, *Principles of Digital Transmission: With Wireless Applications*, Cambridgeshire, Springer, 2006.

[21] BITS, BAUD RATE, AND BPS Taking the Mystery Out of Modem Speeds, http://www.textfiles.com/apple/bitsbaud.txt, Januar 2022.

[22] Mazzei Daniele, Montelisciani Gabriele, Baldi Giacomo, Fantoni Gualtiero, *Changing the programming paradigm for the embedded in the IoT domain*, IEEE, 2015.

[23] Moving averages Rob J Hyndman, https://robjhyndman.com/papers/movingaverage.pdf, Maj 2022.

Sažetak / Abstract rada

Na kraju rada na posebnoj stranici potrebno je odštampati i naziv rada sa imenom kandidata sa sažetkom (*Abstract*) rada.

NAZIV RADA NA SRPSKOM
(MAX. DVA REDA)

Ime studenta / Mentor

Sažetak – Ove instrukcije su date kao smernice za pisanje diplomskih (završnih) radova u Visokoj tehničkoj školi u Nišu.

TITLE OF THE PAPER IN ENGLISH
(MAX. TWO LINES)

Name of student / Mentor

Abstract *– These instructions are for submitting a manuscripts for the final work in The College of Applied Technical Sciences in Nish and all students their manuscripts/works must be typewritten according to the text of these instructions.*

Biografija

Filip Stojanović rođen je 30.07.1998. godine u Nišu, Republika Srbija. Osnovnu i srednju školu završio je u Nišu. Nosilac je Vukove diplome za postignut uspeh u toku školovanja, kao i većeg broja broja diploma sa učešća na brojnim republičkim takmičenjima.

Visoku tehničku školu u Nišu, smer Savremene računarske tehnologije, upisao je školske 2017/18 godine. Za vreme studija učestvovao je u radu VTŠ Apps Tima Visoke tehničke škole i poseduje sertifikat o postignutim rezultatima.

Od 1.10.2014. do 1.12.2014. godine bila je zapošljena u preduzeću VTŠ Soft kao php programer.

Autor ili koautor je 5 radova objavljenih u zborniku radova Visoke tehničke škole u Nišu. Učestvovala je na domaćim i međunarodnim naučno-stručnim konferencijama sa radovima koji su publikovani u odgovarajućim zbornicima. Sa studentima Visoke tehničke škole u Nišu, 2014. godine je osvojila nagradu za najbolji rad na IEEESTEC konferenciji studentskih projekata.