FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Sustavi za praćenje i vođenje procesa

Povezivanje računala I mikrokontrolera USB protokolom

Zoran Tiganj

0036401310

Sadržaj

Uvod

Posljednjih godina USB (universal serial bus – univerzalna serijska sabirnica) sučelje postalo je jako popularno među krajnjim korisnicima. Glavni razlog za to je mogućnost uključivanja uređaja bez potrebe za resetiranjem računala (Plug and Play). No kod razvoja uređaja, razvoj za USB sučelje daleko je složeniji nego primjerice razvoj za RS232. Posebno veliki problem predstavlja nužnost izrade drivera za uređaj koji se preko USB porta povezuje s računalom. RS232 sučelje, komunikacija uređaja korištenjem RS232 sučelja i nekog serijskog protokola mnogo je jednostavnija. Problem predstavlja činjenica da se u novije vrijeme RS232 port ne ugrađuje na standardna računala (osobito prijenosna). Upravo ta činjenica jedan je od glavnih motiva da se i u manjim projektima kao komunikacijsko sučelje implementira USB.

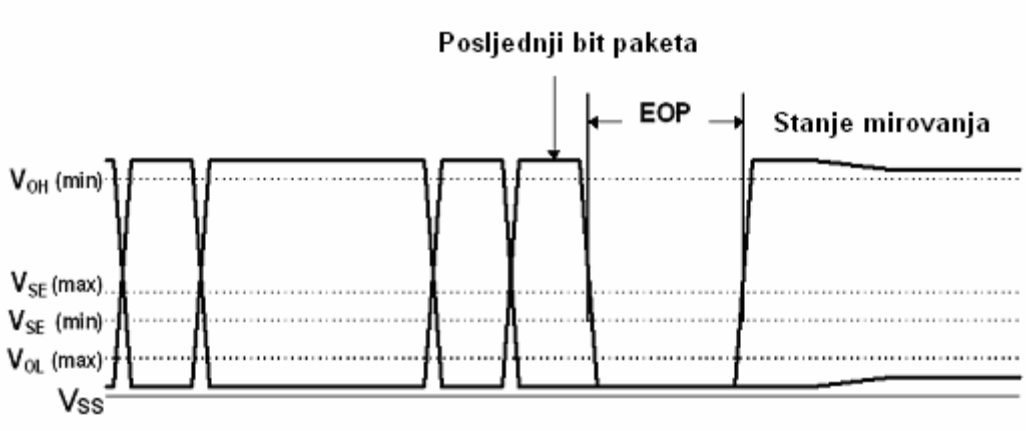
Implementaciju USB-a na vanjski uređaj s trenutno dostupnom tehnologijom moguće je izvesti na dva načina:

a) Korištenjem mikrokontrolera sa tvornički implementiranim USB sučeljem. Ovakav pristup zahtjeva poznavanje načina rada USB protokola. Potrebno je napisati ugrađeni softver (eng. firmware, u daljnjem tekstu koristiti će se izvedenica eng. izraza - firmver) za mikrokontroler, također je potrebno napisati upravljačke programe (eng. drivere u daljnjem će se tekstu zbog udomaćenosti koristiti eng. izraz) za računalo. Nedostatak ovakvog sustava je u problemu pronalaska odgovarajućeg mikrokontrolera. Na tržištu su naime ovakvi mikrokontroleri veoma rijetki (u maloprodaji na hrvatskom tržištu ne postoji ni jedan), a njihova cijena vrlo visoka u usporedbi sa klasičnim mikrokontolerima koji posjeduju RS232 sučelje. Ovi problemi osobitu težinu imaju kada se radi o amaterskim i ne serijskim sustavima.
b) Drugi način povezivanja vanjskog uređaja s glavnim računalom preko USB sučelja je korištenje univerzalnog konvertera između USB i nekog drugog (najčešće RS232) sučelja. U ovom slučaju nije potrebna izrada firmvera niti je potrebno poznavati način rada samog USB protokola. Uz ovakve konvertere proizvođač u pravilu dostavlja i drivere. Nedostatak ove metode jest visoka cijena samih konvertera, njihova relativno teška dostupnost kao i činjenica da ugradnja ovakvog konvertera povećava dimenzije cjelokupnog proizvoda.

Ideja ovog rada je ponuditi jeftinu i lako izvedivu alternativu gornjim metodama. Predložena metoda se sastoji od implementacije USB protokola u lako dostupan, jeftini mikrokontroler atmega8. Kontroler u sebi nema tvornički ugrađeno USB sučelje stoga je potrebno izvršiti emulaciju tog sučelja unutar firmvera. Glavni izazov predstavlja dostizanje odgovarajuće brzine koju USB protokol zahtjeva: LowSpeed - 1.5Mbit/s, FullSpeed - 12Mbit/s, HighSpeed - 480Mbit/s . Maksimalni radni takt mikrokontrolera atmega8 je 16Mhz. Ovakvim tipom mikrokontrolera moguće je implementirati samo LowSpeed protokol.

USB protokol je vrlo složen. Detaljan opis samog protokola nalazi se na adresi www.usb.org gdje se u dokumentu na oko 650 stranica objašnjavaju sve potankosti vezane uz protokol.
Jednostavniji (oko 30 stranica), ali ipak vrlo informativan vodič kroz USB komunikaciju nalazi se na adresi www.beyondlogic.org.

Ovdje ću pokušati pojasniti najelementarnije principe funkcioniranja komunikacije preko USB-a.
Fizički se USB sučelje sastoji od 4 priključka: (+5V, masa te DATA+ i DATA-). Maksimalna struja koju je USB može dati je 500mA što je vrlo često dovoljno za napajanje cijelog uređaja samo iz USB porta. Komunikacija između računala i vanjskog uređaja vrši se linijama DATA+ DATA- . Obje linije su dvosmjerne (eng. Bi-directional). Naponski nivoi su diferencijalni: kada je DATA+ u visokom stanju DATA- je u niskom i obrnuto. DATA+ i DATA- su u istom stanju samo na kraju paketa – u stanju mirovanja (eng. Idle state).

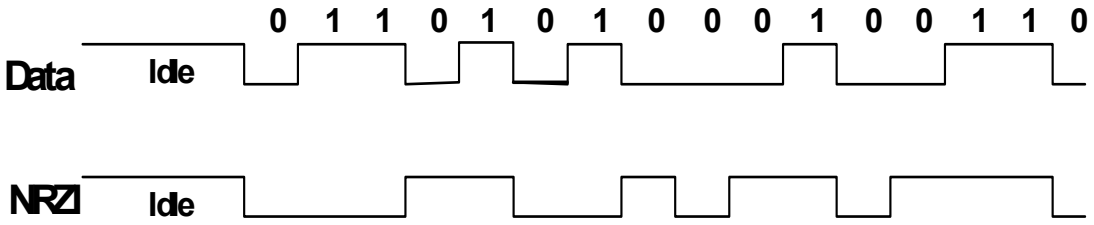


Slika 1

Spajanje i odspajanje USB uređaja uočava se detekcijom promjene impedancije koja se dogodi uslijed spajanja odnosno odspajanje USB uređaja. Kako bi se postigla promjena impedancije potrebno je u skladu s protokolom za LowSpeed uređaje na liniju DATA- dodati 1.5kohm otpornik dok je za FullSpeed uređaje potrebno otpornik dodati na DATA+ liniju.

Kada računalo (eng. host) detektira priključivanje novog uređaja počinje komunikacija među njima u skladu sa USB protokolom. Kako bi komunikacija bila sinkrona predajnik prije stvarnih podataka šalje sinkronizacijski uzorak (101010) kojeg slijede dvije nule nakon čega konačno slijede podatci.
Kako bi se sinkronizacija održala potrebno je svake milisekunde slati sinkronizacijski uzorak u slučaju full speed uređaja odnosno obje podatkovne linije spustiti na nulu u slučaju low speed uređaja. U hardverskoj implementaciji USB sučelja sinkronizaciju osigurava PLL sklop.

Obavijest o kraju prijenosa EOP (kraj paketa eng. end of packet) vrši se postavljanjem dviju uzastopnih nula na obje podatkovne linije.
Podatci između sinkronizacijskog uzorka i kraja paketa su kodirani po NRZI principu (slika 2), to znači da nakon svakih 6 uzastopnih jedinica biti ubačena jedna nula.



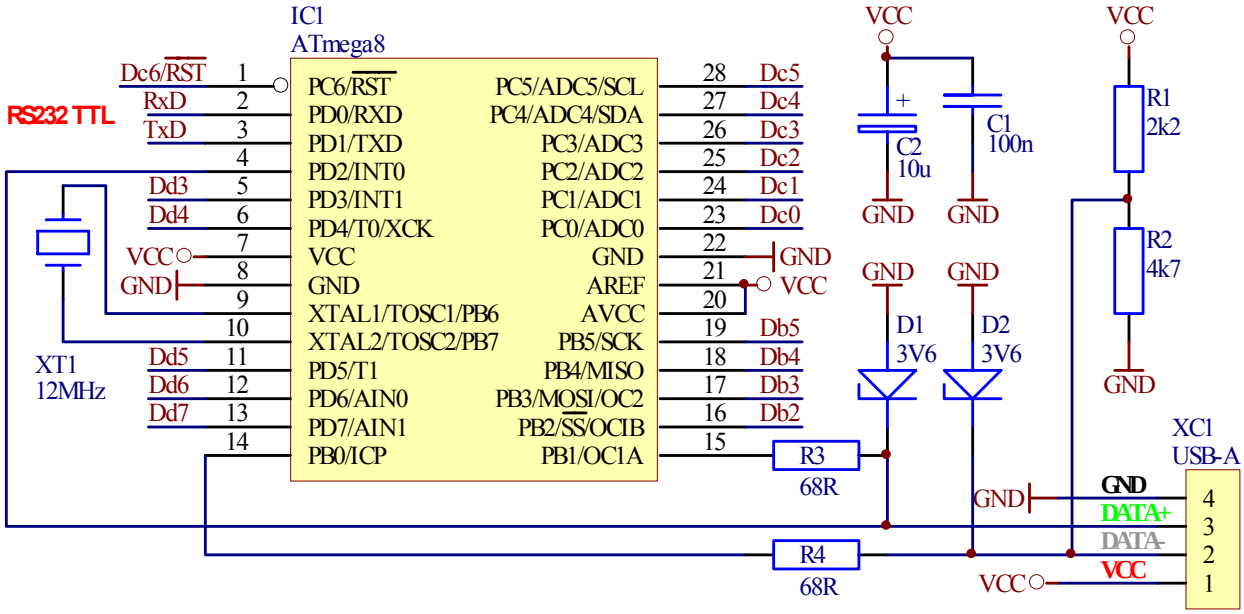
Slika 2

Svaki niz podataka dijeli se u više polja: sinkronizacijski uzorak, oznaka paketa (eng. packetID), adresno polje, zaključno polje (eng, endpoint field), podatci, provjera pogreške (eng. Cyclic redundancy check). Detalji o korištenju ovih polja pojašnjeni su na http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf.
USB protokol definira 4 tipa prijenosa podataka: kontrolni prijenos, prekidni prijenos, izokroni prijenos (eng. Isochronous) i masonvni prijenos (eng. bulk transfer). Svaki od ovih prijenosa je namjenjen uređajima drugačijih zahtjeva.

Prijenos korišten u realizaciji ovog projekta je kontrolni prijenos. Prvenstveno je namijenjen kontroli uređaja, ali u principu spada u prijenose opće namjene.
Svaki kontrolni prijenos sastoji se on nekoliko faza: faza postavljanja, podatkovna faza i statusna faza.
Podatci se u USB prijenosu prenose u paketima – po nekoliko bajtova u svakom paketu. Veličinu paketa određuju sami uređaji, dok specifikacije određuju samo limit. Za low speed uređaje veličina paketa je ograničena na 8 bajtova. Taj 8 bajtova velik paket + početak i kraj paketa moraju biti pohranjeni u spremnik (eng. buffer) uređaja pri svakom USB prijenosu. U hardverski implementiranom USB sučelju firmver automatski dekodira različite dijelove poruke i pohranjuje ih u spremnike. Ovo postavlja zahtjev na veličinu izlaznih i ulaznih spremnika koji moraju zadovoljiti standarde tražene protokolom. Dodatni zahtjev na firmver je da bude dovoljno brz da uspijeva raditi posao dekodiranja, spremanja kao i slanja podataka. Redovi veličine zahtijevanih brzina graniče s mogućnostima komercijalnih mikrokontrolera, stoga kod firmvarea treba biti maksimalno optimiziran. Jedan od aspekata minimizacije je svakako pisanje koda u strojnom jeziku – asembleru.

Električna shema sklopa

Na slici 3 je prikazan sam spoj mikrokontrolera atmega8 na USB sučelje. Samostalan, ovaj sklop predstavlja zapravo konverter sa RS232 na USB sučelje.



Slika3

Signalne linije DATA+ i DATA- spojene su na port B mikrokontrolera i to na PB0 i PB1. Priključak mikrokontrolera INT0 spojen je na DATA+ kako bi mikrokontroler mogao pokrenuti zahtjev za prekidom kada počne dobivati podatke. Za ispravan rad potreban je i poll up otpornik na priključak DATA- čija je uloga ranije pojašnjena.
Ostale komponente imaju ulogu da osiguraju ispravan rad mikrokontrolora – kristal za osiguravanje ispravnog izvora takta te kondenzatori za filtriranje izvora napajanja (sam USB port).
Vidljivo je kako ovo rješenje zahtjeva jako mali broj dodatnih komponenti što ga čini izrazito jeftinim. Ukoliko su potrebne neke druge funkcionalnosti sustava moguće je dodati dodatne komponente npr. ukoliko želimo primati infracrveni signal potrebno je dodati infracrveni senzor: TSOP1738. Ukoliko sklop želimo koristiti kao USB na RS232 konverter potrebno je dodati pretvornik naponskih nivoa (TTL na RS232) koji radi na principu nabojske pumpe MAX232. Ukoliko želimo upravljati LED diodama spajamo ih izravno preko odgovarajućih otpornika na pinove mikrokontrolera. Relejom je moguće upravljati korištenjem bipolarnog tranzistora ili FET-a kao sklopke.

Implementacija programske podrške na mikrokontroleru

Uvod

Firmver sadrži implementiran kompletan komunikacijski protokol. Preko firmvera mikrokontroler prima niz bitova u jednom USB paketu i pohranjuje ih u interne registre. Početak prijenosa označava prekid linije INT0 koja se brine da ispravno prepozna sinkronizacijski uzorak. Tijekom prijema iz razloga povećanja brzine provjerava se samo kraj paketa – EOP. Nakon prijema firmver dekodira pakete i analizira ih. Prvu analizu predstavlja provjera adrese – provjera da li je paket uopće namijenjen tom uređaju. Provjera adrese mora se obaviti jako brzo i predstavlja zapravo najzahtjevniji dio realizacije jer uređaj u vrlo kratkom vremenu mora odgovoriti signalom ACK (prihvaćam) pošiljatelju poruke. Nakon prijema provodi se NRZI dekodiranje koje se vrši u posebnim registrima čime se omogućava da za vrijeme dekodiranja mikrokontroler može i dalje primati poruke i odgovarati na njih jer se ulazni registri mogu osloboditi. Brzina dekodiranja nije ključna, ali je ključno da kada računalo traži od mikrokontrolera odgovor, mikrokontroler odmah odgovori signalom NAK (nisam još gotov) – ukoliko, naravno, proces dekodiranja još nije gotov. Iz ovog razloga je nužno da se dekodiranje provodi korištenjem drugih registara odnosno omogućavanjem firmveru da i dalje prima podatke. Firmver je također zadužen da se nakon što su podatci dekodirani poduzme odgovarajući postupak specificiran u samim podatcima npr. slanje podataka na RS232 i kontrolu prijenosa.

Opis firmvera

Firmver je podijeljen u nekoliko dijelova: prekidne potprograme, potprograme za dekodiranje, USB prijem, USB slanje, potprograme za dekodiranje traženih operacija, potprograme za izvođenje traženih operacija. U firmver je moguće i dodavati funkcije za obavljanje pojedinih specifičnih zadataka npr. za A/D konverziju (atmega8 ima pet 8 bitnih A/D konvertera) isl.

«Ext_int0» prekidna rutina

Vanjski prekid 0 je aktivan cijelo vrijeme dok se firmver izvršava. Prekidna rutina počima kada mikrokontroler dobije podatak preko USB linije. Prekid se događa na rastući brid signala na INT0 pinu mikrokontrolera (rastući brid označava i početak sinkronizacijskog uzorka). Uzorkovanje pojedinih bitova mora biti sinkronizirano te se događati na sredini samog bita. Ovo se postiže zahvaljujući sinkronizacijskom uzorku (koji je zapravo pravokutni signal - 101010). Trajanje pojedinog bita je samo 8 ciklusa signala takta. Kako je brzina podataka 1.5Mbit/s zaključujemo da nam takt mikrokontrolera od 12Mhz najbolje odgovara jer takvim taktom možemo uhvatiti točno sredinu bita. No isto tako za uzorkovanje podataka, pohranjivanje podataka u registre i provjeru da li je cijeli paket dobro primljen mikrokontroler ima samo 8 ciklusa stoga ovaj dio programske podrške – firmvera, kao što je već naglašeno, mora biti posebno optimiran.

Opis potprograma korištenih u firmveru

Reset:
Inicijalizira resurse mikrokontrolera: stog, serijske linije, USB registre, prekide…
Main: Glavna programska petlja se brine da se izvede potrebno prosljeđivanje primljenih podataka. Također provjerava da li je preko portova namijenjenih USB-u mikrokontroler dobio zahtjev za resetom u slučaju kojeg se glavni program brine da se reset i dogodi.

Int0Handler:
Prekidna rutina koja se pokreće prekidom na INT0. Predstavlja glavni odašiljačko/prijemni dio. Vrši samu emulaciju USB linija. Pohranjuje podatke u registre, prepoznaje paket ovisno o adresi, šalje odgovor računalu. Ovaj potprogram predstavlja zapravo glavni dio cijelog sustava.

MyNewUSBAddress:
Ovaj potprogram poziva prekidna rutina ukoliko postoji zahtjev računala za promjenom adrese.

FinishReceving
Kopira kodirani USB paket u pomoćne registre gdje će biti primijenjeno dekodiranje.

USBreset:
Inicijalizira USB sučelje (isto što i gašenje pa paljenje uređaja).

SendPreparedUSBAnswer:
Šalje prethodno spremljeni sadržaj registara na USB linije.

ToggleDATAPID:
Vrši zamjenu indetifikatora paketa između DATA0 i DATA1 – ova zamjena zahtijevana je specifikacijama samog protokola.

ComposeZeroDATA1PIDAnswer:
Priprema paket koji ne sadrži podatke i ponekad se koristi kao oznaka da nema dodatnih podataka spremnih za slanje.

InitACKBuffer:
Inicijalizira registar u RAM-u sa ACK paketom (paket kojim mikrokontroler odgovara računalu kada primi podatke). Ovaj paket se čuva u RAM-u jer ga je potrebno vrlo često slati i to odmah nakon što se podatci prime.

SendACK:
Šalje ACK paket preko USB linija

InitNAKBuffer:

Inicijalizira registar u RAM-u sa NAK paketom (paket kojim mikrokontroler odgovara da još nije obradio podatke koje je dobio). Ovaj paket se čuva u RAM-u jer ga je potrebno vrlo često slati i to odmah nakon što se podatci prime.

SendNAK:
Šalje NAK paket preko USB linija.

CompoeseSTALL:
Inicijalizira registar u RAM-u sa STALL paketom. Ovaj paket se čuva u RAM-u jer ga je potrebno vrlo često slati i to odmah nakon što se podatci prime.

DecodeNRZI:
Provodi NRZI dekodiranje. Podatci sa USB linija su NRZI kodirani, ovaj potprogram vraća dekodirane podatke.

BitStuff:
Dodaje ili eliminira dodatne bitove (eng. BitStuffing). Dodatne bitove USB dodaje računalo da osigura sinkronizaciju.
Pomoćni potprogram koji se koristi kod dodavanja bitova (eng. BitStuffing). Dodaje jedan bit izlaznim podatcima što povećava njihovu duljinu tako da višak treba prebaciti u drugi registar.

ShiftDeleteBuffer:
Pomoćni potprogram koji se koristi kod eliminacije dodatnih bitova (eng. BitStuffing).
MirrorInBufferBytes:
Mijenja redoslijed podataka primljenih preko USB-a jer su poslani podatci bili obrnutog redoslejda od traženog (LSB/MSB).

CheckCRCIn:
Primjenjuje CRC (eng. cyclic redundancy check) na primljeni paket podataka. CRC se dodaje radi zaštite točnog prijenosa podataka.

AddCRCOut:
Dodaje CRC polje izlaznom paketu podataka. CRC algotiram je specificiran USB protokolom.

CheckCRC:
Pomoćni potprogram koji se koristi kod dodavanja i provjere CRC.

LoadDescriptionFromROM:
Šalje podatke iz ROM memorije na USB izlaz.

LoadDescriptorFromROMZeroInsert:
Šalje podatke iz ROM memorije na USB izlaz, ali svaki jednaki byte dodaje kao nulu. Ovo se koristi kada se zahtjeva tzv. UNICODE format radi uštede ROM prostora.
LoadDescriptorFromSRAM:

Šalje podatke iz RAM memorije na USB izlaz.

LoadDescriptorFromEEPROM:
Šalje podatke iz EPROM memorije na USB izlaz.

LoadXXXDescriptor:
Vrši odabir memorije kada se traži odgovarajući odgovor (ROM, RAM ili EPROM).

PrepareUSBOutAnswer:
Priprema odgovor računalu, šalje ga u izlazne registre, dodaje odgovoru bitstuffing.

PrepareUSBAnswer:
Glavni potprogram za slanje odgovora računalu. Zavisno od argumenata koji se proslijeđuju potprogramu ovaj poduzima odgovarajuću akciju. Sam potprogram se dijeli na dva velika dijela:

- Dio za obrađivanje standardnih zahatjeva
- Dio za obrađivanje specifičnih zahtjeva

Standardni zahtjevi definirani su samim USB protokolom dok specifične zahtjeve definira proizvođač sustava.

Standardne USB funkcije su:
ComposeGET_STATUS:
ComposeCLEAR_FEATURE:
ComposeSET_FEATURE:
ComposeSET_ADDRESS:
ComposeGET_DESCRIPTOR:
ComposeSET_DESCRIPTOR:
ComposeGET_CONFIGURATION:
ComposeSET_CONFIGURATION;
ComposeGET_INTERFACE;
ComposeSET_INTERFACE:
ComposeSYNCH_FRAME;

Nekoliko primjera javno dostupnih funkcija kreiranih od različitih prodavača:
DoSetInfraBufferEmpty:
DoGetInfraCode:
DoSetDataPortDirection:
DoGetDataPortDirection:
DoSetOutDataPort:
DoGetOutDataPort:
DoGetInDataPort:
DoEEPROMRead:
DoEEPROMWrite:
DoRS232Send:
DoRS232Read:

DoSetRS232Baud:
DoGetRS232Baud:
DoGetRS232Buffer:
DoSetRS232DataBits:
DoGetRS232DataBits:
DoSetRS232Parity:
DoGetRS232Parity:
DoSetRS232StopBits:
DoGetRS232StopBits:
DeviceDescriptor:
ConfigDescriptor:
LangIDStringDescriptor:
VendorStringDescriptor:
DevNameStringDescriptor:

Softver na računalu:

Kako bi komunikacija računala i mikrokontrolera bila moguća na računalu mora biti instaliran odgovarajući softver.
Taj softver možemo podijeliti u tri dijela:

1) ***Upravljački program*** (eng. Driver) predstavlja najniži komunikacijski sloj. Driver korišten u ovom projektu je tipa otvorenog koda (u daljnjem tekstu će se zbog udomaćenosti koristiti eng. naziv open source) tako da je cjelokupni kod dostupan za proučavanje i mijenjanje. Driver je napisan u Windows2000 DDK (Driver Development Kit) okruženju. Razvoj drivera baziran je na jednom od primjera unutar DDK – IsoUsb. Driver je modificiran kako bi komunikacija s mikrokontrolerom bila moguća - originalnom su kodu dodani/prošireni djelovi vezani uz IOCTL jer računalo komunicira s mikrokontrolerom putem poziva IOCTL rutina. Kako bi se smanjila veličina samog drivera nepotrebni djelovi su obrisani. Driver radi pod svim 32 Windows operativnim sustavima osim pod Windows 95. Ime datoteke u kojoj se nalazi driver je "AVR309.sys". Kako bi se provela instalacija drivera operativni sustav zahtjeva instalacijsku skriptu. Ime skripte je „AVR309.inf" i kreirana je u programu Notepad. Tijekom instalacijskog procesa datoteka s driverima se kopira u sistemske direktorije i vrše se određene promjene u samom operativnom sustavu zatjevane od drivera. INF datoteka osigurava instalaciju DLL biblioteka na način da specificira odgovarajuče putove kako bi se DLL biblioteke mogle postaviti na potrebna mjesta unutar sustava računala.

2) ***DLL biblioteke*** predstavljaju skup funkcija koje komuniciraju s driverom, a mogu biti korištene u pisanju aplikacija. Na taj je način pisanje krajnje korisničke aplikacije znatno pojednostavljeno. Za ovaj projekt javno su dostupne funkcije koje služe za:

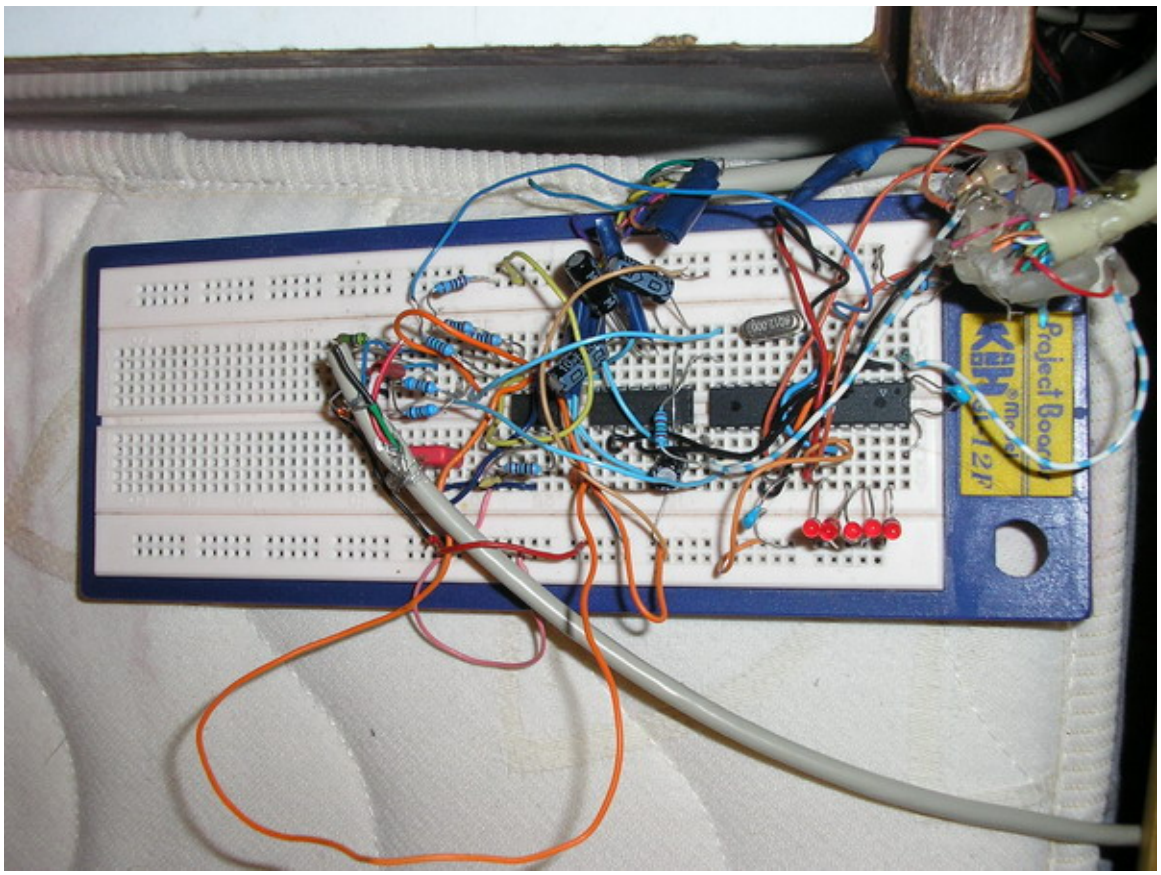
- slanje USB protokolom infracrvenog koda iz tzv. Infracrvenog registra mikrokontrolora
- promjenu smijera pojedinog priključka mikrokontrolera (ulaz/izlaz)
- dobivanje informacije o smijeru pojedinog priključka mikrokontrolera (ulaz/izlaz)
- slanje podatka na neki od portova mikrokontrolera
- primanje podataka s nekog od portova mikrokontrolera
- čitanje EPROM-a
- pisanje u EPROM
- slanje podataka preko RS232 priključka mikrokontrolera
- primanje podataka preko RS232 priključka mikrokontrolera
- određivanje brzine prijenosa (eng. baud rate) RS232 priključka mikrokontrolera
- dobivanje informacije o brzini prijenosa (eng. baud rate) RS232 priključka mikrokontrolera
- čitanje RS232 registra mikrokontrolera
- pisanje u RS232 registar mikrokontrolera
- podešavanje broja podatkovnih bitova za RS232 prijenos
- dobivanje informacije o broju podatkovnih bitova za RS232 prijenos
- postavljanje pariteta za RS232 prijenos
- dobivanje informacije o paritetu za RS232 prijenos
- definiranje broja stop bitova za RS232 prijenos
- dobivanje informacije o broju stop bitova za RS232 prijenos

Opis funkcija s popisom potrebnih argumenata, kao i deklaracije svih funkcija za programske jezike Delphi, Visual basic, C++ Builder / Microsoft Visual C++ nalaze se na adresi:
http://www.mikrocontroller.net/attachment.php/199876/AVR309_DLL_help.htm U sklopu ovog seminara korišten je Visual basic 6.0. Početna testiranja ispravnosti samog sustava vršena su gotovom open source aplikacijom napisanom programskom jeziku Delphi 7.

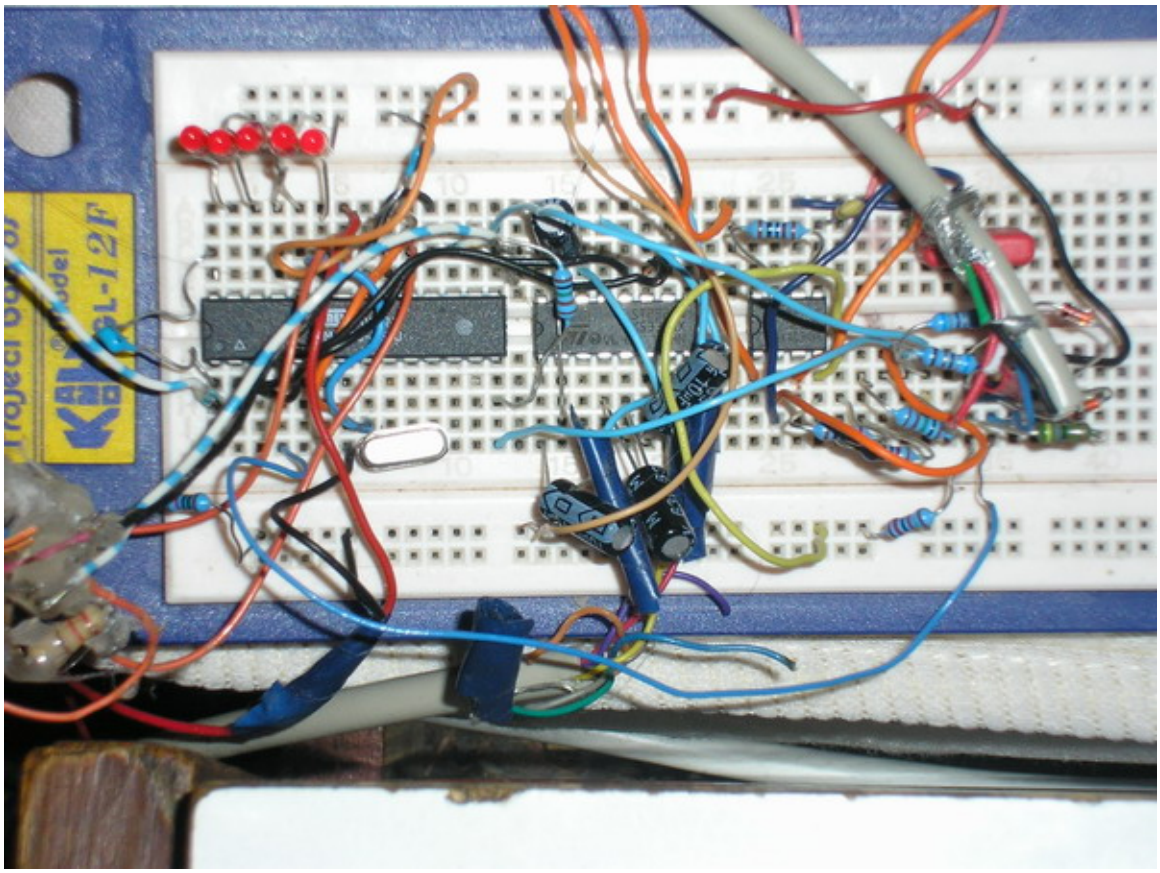
3) ***Korisnička aplikacija*** predstavlja softver koji korinik pokreće kada želi komunicirati s mikrokontroledom. Kao što je već spomenuto u sklopu ovog seminara korišten je Visual Basic 6.0 i Delphi 7. Zahvaljući bogatoj skupini funkcija unutar DLL biblioteke pisanje korisničkih aplikacija znatno je pojednostavljeno te se sama komunikacija s mikrokontrolerom preko USB sučelja svodi na pozivanje postojećih funkcija s odgovarajućim argumentima.

Rezultati:

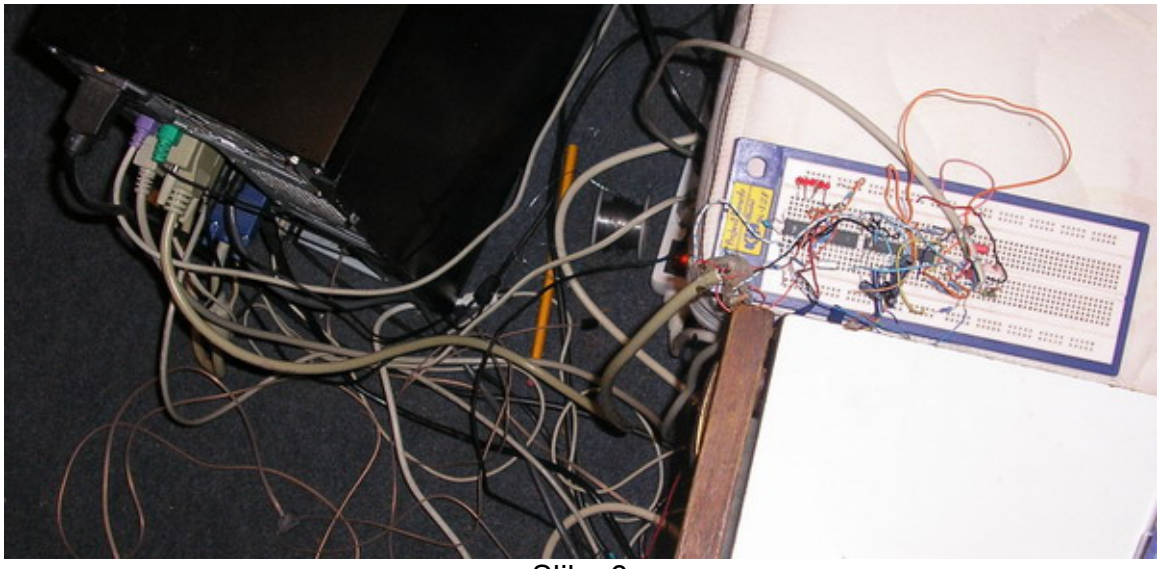
Shema prikazana slikom 3 realizirana je na testnoj pločici koja se inače koristi za razvoj sustava s mikrokontrolerom atmega 8 tako da se na pločici nalazi sučelje za serijski programator, priključci za serijski RS232 port te potrebne dodatne komponente. Izgled ispitne pločice - protoborda s elementima te cijelog sustava priključenog na računalo prikazan je na slikama 4, 5 I 6.



Slika 4



Slika 5



Slika 6

U slučaju da seminar bude izabran za prezentaciju samo onaj dio sklopa zadužen za USB komunikaciju (prema slici 3) biti će prebaćen na na drugi protoboard. Programska podrška sa strane računala kao što je već spomenuto napisao sam u programskom jeziku Visual basic 6.0. Razvijeni sustav moguće je

koristiti za vrlo sofisticirane sustave poput EEG akvizicije bilo koristeći A/D konvertere koji se nalaze u samom mikrokontroleru ili koristeći pinove mikrokontrolera kao ulaze pojedinih bitova vanjskog A/D pretvornika. Sustav je moguće koristiti i kao USB/RS232 konverter što je zapravo bila izvorna namjena. U takvoj primjeni neki drugi mikrokontroler bilo kojeg tipa koji obavlja posao unutar nekog drugog sustava može po dogovorenim parametrima prijenosa slati podatke na RS232 sučelje mikrokontrolera koji te podatke prosljeđuje računalu preko USB sučelja. Računalo vrši obradu podataka te prema potrebi ponovno mikrokontroleru preko USB-a šalje eventualne instrukcije koje ovaj preko RS232 sušelja šalje drugom mikrokontroleru.
Cijena kompletnog sustava u ovom tretnutku u maloprodaji na hrvatskom tržištu iznosi manje od 50kn (atmega8 30kn, USB kabel 10kn, te ostale komponente nekoliko kuna) čime sustav predstavlja veliku konkurenciju klasičnim RS232 na USB konverterima. Ipak za uređaje koji se proizvode u velikim serijama razlika u cijeni postaje zanemariva pa dominiraju klasična rješenja.

Literatura

1. http://www.usb.org
2. http://www.beyondlogic.org
3. http://www.mikrocontroller.net/attachment.php/199876/AVR309_DLL_help.htm
4. http://www.cesko.host.sk
5. http://www.avrfreaks.net