1. Uvod

Na začetku 21. stoletja smo ljudje soočeni z zelo hitro se spreminjajočem informacijskim okoljem. Smo v informacijski dobi. Človek, znanje, informacije postajajo vedno pomembnejše v primerjavi s kapitalom. Nove tehnologije ustvarjalo nova delovna mesta, nove potrebe, novo povpraševanje. Uporaba svetovnega spleta (interneta), kot enega najpomembnejših dejavnikov večanja produktivnosti v gospodarstvih sveta, postaja nujnost pri sodobnem načinu poslovanja. Ogromne količine podatkov lahko zdaj sprejemamo in prikazujemo s pomočjo svetovnega spleta po celem svetu (v primerjavi s preteklostjo) z zanemarljivimi stroški.

Za uporabo, obdelavo in vzdrževanje teh podatkov pa potrebujemo učinkovite baze (skladišča) podatkov. To nam omogočajo različna orodja kot so Oracle, SQL... Nakup takih, visoko profesionalnih orodij predstavlja veliko investicijo, tako da si manjši subjekti tega ne morejo privoščiti. Za te manjše subjekte so potrebne drugačne bolj racionalne rešitve.

Danes si lahko vsakdo preko telefonskega priključka postavi spletni strežnik. Programsko opremo za strežnik je mogoče dobiti povsem brezplačno. Trenutno je najbolj razširjen spletni strežnik Apache, katerega bomo uporabljali tudi mi, ki ga dobimo v različici za Linux ali MS Windows. V našem primeru ga bomo uporabili za brezplačni Linux operacijski sistem. Spletni strežnik omogoča stalno prisotnost na svetovnem spletu in s tem lastno identiteto. Preko spletnih iskalnikov (Google, Matkurja, Yahoo, Altavista) imamo tako že omogočen prvi kontakt.

Če želimo sprejemati podatke iz web-a ali jih prikazovati na web-u moramo izgraditi primerno bazo podatkov. Ta baza podatkov nam omogoča enostavno vzdrževanje, ažuriranje in analiziranje podatkov. Ker spet težimo k temu, da imamo čim nižje stroške, uporabimo MySQL podatkovno bazo, ki jo prav tako dobimo povsem brezplačno. Internet pa ni uporaben samo za prenos podatkov, ampak ima še veliko drugih možnosti uporabe. Na vmesne člene, kot so strežniki in podobne naprave lahko namestimo tudi različne

aplikacije. Te aplikacije lahko končni uporabnik uporablja sedeč za domačim računalnikom, kot da bi sedel za napravo na kateri se bi pač takšna aplikacija drugače izvajala. Aplikacije na strežnikih so lahko tudi takšnega tipa, da komunicirajo z zunanjimi napravami, jih usmerjajo, ter dajejo uporabniku povratne informacije, kaj se je z napravo zgodilo. V tem primeru lahko rečemo, da uporabnik teleoperira z zunanjo napravo. V našem primeru bomo preko interneta krmilili tekoči trak, kateri bo glede na vhodne podatke na prosto programljivem krmilniku Mitsubishi FX izvajal razne aplikacije.

Svetovni splet, med drugim, sestavlja množica HTML spletnih strani. Izdelava in oblikovanje HTML strani je preprosta. Za izgradnjo sistema uporabnik - web strežnik - baza podatkov – zunanja naprava, smo torej uporabili Apache web strežnik, MySql podatkovno bazo, dva programa napisana v programskem jeziku C za vpis v podatkovno bazo in pošiljanje podatkov na krmilnik, ter PHP skriptni jezik za izgled in prenos podatkov preko interneta na strežnik. Na zahtevo uporabnika mora web strežnik "vedeti" kako in v katero bazo "pogledati". Tako kot mora web strežnik vedno "poslušati" kaj bo uporabnik zahteval od njega, mora biti tudi baza podatkov pripravljena za uporabo.

Ker je komunikacija preko interneta zelo primerna za takšno delo, poleg tega pa je dostopna kjerkoli po svetu, smo se odločili narediti sistem za krmiljenje tekočega traku preko spletnega brskljalnika.

2. Opis aplikativne in aparaturne opreme

2.1 Linux operacijski sistem

Ker celotna doplomska naloga deluje na Linux operacijskem sistemu, mu bomo namenili nekaj besed. Za ta sistem smo se odločili predvsem zato, ker je zanesljiv in je brezplačen za uporabo, saj težimo k temu, da bi zgradili sistem s čim manj stroški.

Linux je Unix in ni Unix. Je Unix, saj upošteva določila **POSIX** [1] ter se nasploh obnaša kot Unix, in ni Unix, saj je bil od začetka napisan na novo in ne vsebuje niti ene same programske vrstice iz sistema AT&T Unix. Linux je avtorsko delo **Linusa B. Torvaldsa** in drugih sodelavcev, ter se lahko prosto razširja pod pogoji, navedenimi v GNU Public License (GPL) [2]. Zavedati se moramo, da je razvoj Linuxa odprt in distribuiran, medtem ko je razvoj večine ostalega programja zaprt in centraliziran. To pomeni, da je trenutna razvojna verzija vedno javno dostopna (z zamikom tedna ali dveh) in jo lahko kdorkoli uporablja. Rezultat tega je, da izdaja, ki prinaša novo funkcionalnost, skoraj vedno vsebuje tudi napake, po drugi strani pa to pomeni tudi izjemno hiter razvoj, tako da so napake najdene in odpravljene zelo hitro, dostikrat v nekaj urah, saj se z njimi ukvarja veliko ljudi.

2.1.1 Lastnosti in uporaba

Navkljub tričetrt milijona programskih vrstic v jedru, še vedno ostaja mnogo možnosti za dodatke in izboljšave, zato bomo navedli samo ključne lastnosti.

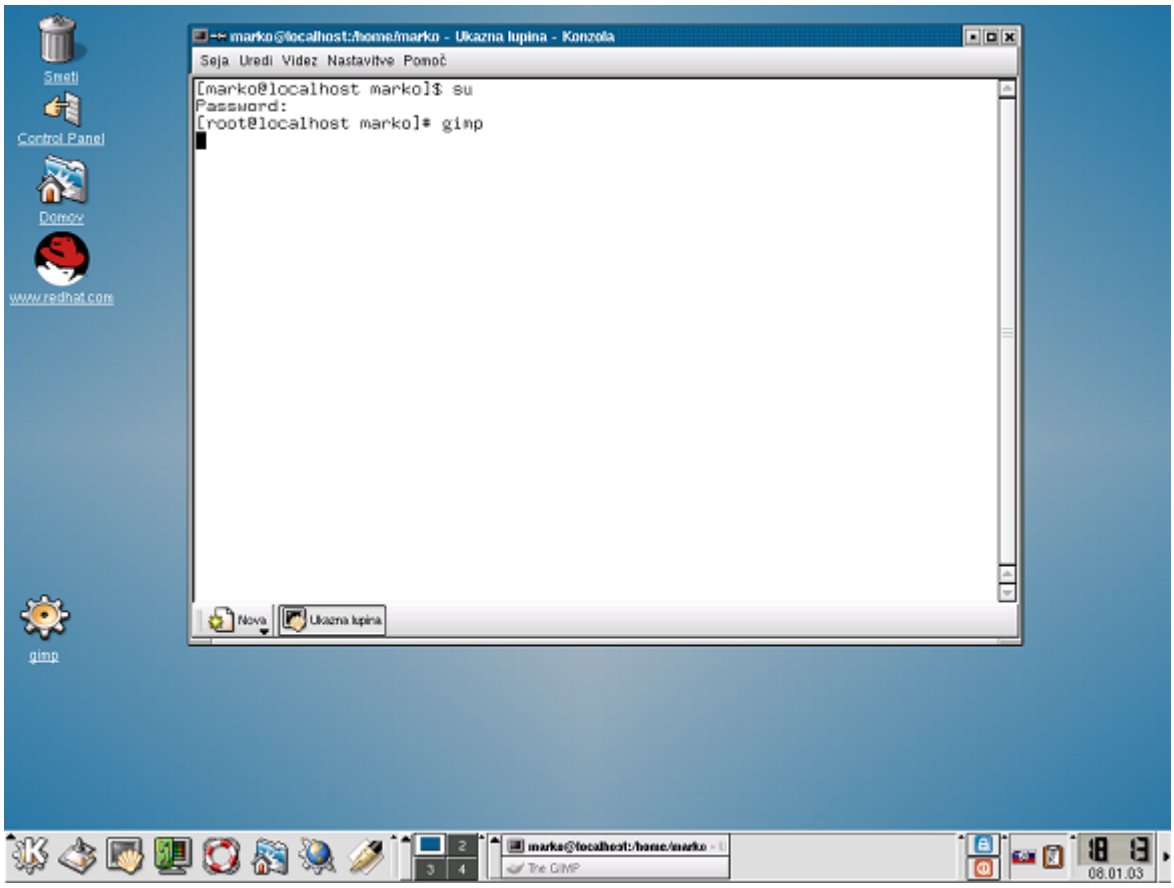
Ključne značilnosti Linux operacijskega sistema so [3]:

- večopravilni: več programov teče naenkrat,
- večuporabniški: več ljudi lahko hkrati dela na istem računalniku (in nobenih omejitev na enega uporabnika),
- večarhitekturni: teče na več različnih mikroprocesorjih, ne le na Intelovih,

- večprocesorski: podpora za SMP obstaja za procesorje Intel in SPARC (za ostale pa je v delu),
- teče v zaščitenem načinu na 80386 in više,
- vsebuje zaščito pomnilnika med procesi, tako da en sam program ne more ogroziti stabilnosti celotnega sistema,
- nalaganje programov na zahtevo: Linux prebere z diska v pomnilnik samo tiste dele programa, ki se jih dejansko potrebuje,
- deljene pomnilniške strani. Več programov lahko uporablja isti pomnilnik. Ko program poskusi pisati v pomnilnik, Linux stran (blok 4KB) pomnilnika prepiše nekam drugam. Ta način prinese dve prednosti: višjo hitrost in manjšo porabo pomnilnika,
- shranjevanje virtualnega pomnilnika na disk po eno stran naenkrat (paging), namesto shranjevanje celotnega procesa (swapping); na ločeno particijo ali pa v datoteko, ali pa na obe. Dodani prostor za swap (prostor na disku se iz zgodovinskih razlogov še vedno imenuje swap) je možno dodati tudi med delom. V celoti more Linux uporabiti 16 področij za swap po 128 MB vsako, skupaj 2 GB. Če je potrebno, se more ta omejitev popraviti s spremembo nekaj vrstic v izvorni kodi jedra,
- skupni pomnilniški fond za uporabniške programe in diskovni medpomnilnik, tako da se lahko ves prosti pomnilnik uporablja za predhranjene vsebine diska, po potrebi, ko programi zahtevajo več pomnilnika, pa se ta dinamično prilagodi,
- dinamično povezane deljene knjižnice in seveda tudi statične knjižnice,
- izpis vsebine pomnilnika (core dump) ob nepravilnem zaključku programa, kar dovoljuje analizo z iskalnikom napak ne le med tekom programa, ampak tudi potem, ko se je zrušil,
- večinoma združljiv na ravni izvorne kode s standardi POSIX, System V in BSD,
- vsa izvorna koda je dostopna, vključujoč celotno jedro, gonilnike, razvojna orodja in uporabniške programe; vse se sme tudi prosto razširjati. Kopica komercialnih programov je bila prirejenih za Linux, za katere izvorna koda ni na voljo; vse, kar pa je bilo prosto, vključno celoten osnoven operacijski sistem, ostaja prosto,

- podpira mnoge neangleške ali posebne tipkovnice, enostavno pa je tudi dodati podporo za nove,
- več virtualnih konzol: več (do 64) neodvisnih prijav prek konzole, med katerimi lahko preklapljate s kombinacijo tipk (neodvisno od strojne opreme za prikaz),
- posebni datotečni sistem, UMSDOS, omogoča, da Linux namestimo na datotečnem sistemu MS-DOS,
- datotečni sistem CD-ROM bere vse standardne zapise CD-ROM,
- podpora za omrežje TCP/IP, vključujoč ftp, telnet, NFS itd.

Kot vidimo iz vseh napisanih lastnostih, Linux že dolgo ni več samo »okno za pisanje«, ampak je primerljiv z vsemi ostalimi operacijskimi sistemi, le da ima to prednost da je brezplačen (slika 2.1).

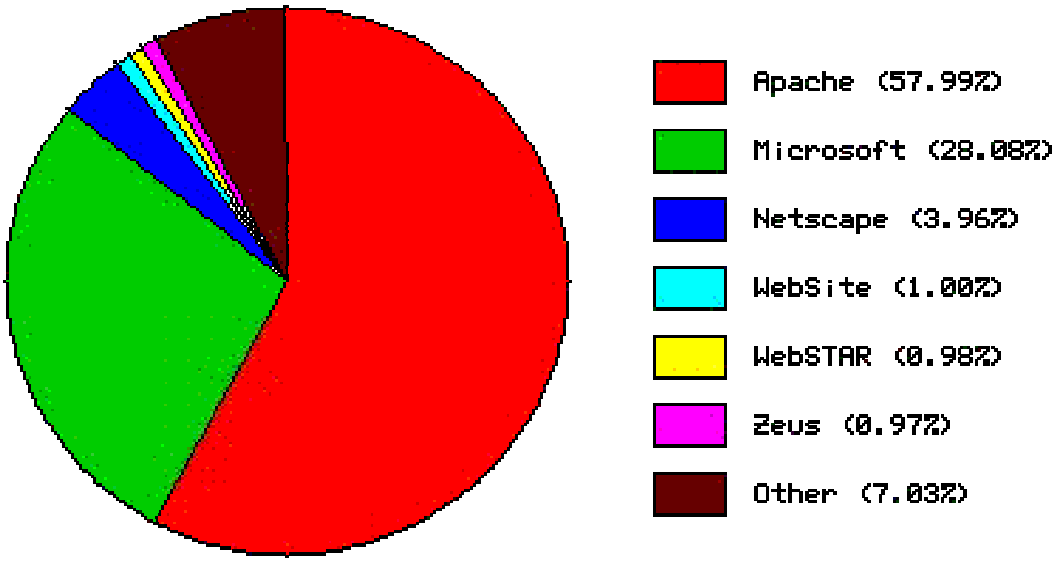


Slika 2.1 : Delovna postaja Linux RedHat

Razlogov za uporabo Linuxa, tudi v šolah, je vse več. Spodbudni so predvsem premiki na področju internacionalizacije delovnega okolja KDE in razvoj različnih paketov za podjetja, šole, kar je posebej pomembno za množično uporabo. Na področju strežnikov in varnosti, kjer v celoti gledano, Linux nima prave konkurence, je škoda, da ga ne uporabljamo pogosteje. Za realizacijo zadostujejo računalniki v "odpisu", stari pet ali več let. Vsa programska oprema, vključno z operacijskim sistemom, je prosto dostopna preko interneta. Najpomembnejše pri vsem je, da sistem deluje zanesljivo in varno.

2.2 Apache WEB strežnik

HTTP (Web) strežnik je eden izmed najbolj nujnih. Apache strežnik je brez dvoma najmanj tvegana odločitev iz večih razlogov. Prepričljivi podatek je, da je trenutno več kot polovica HTTP strežnikov na internetu prav iz družine Apache (slika 2.2) [4]. Tako velik uspeh je pogojen z razpoložljivostjo za mnoge platforme, fleksibilnostjo konfiguracije, robustnostjo, dobro dokumentacijo, povezljivostjo s podatkovnimi bazami, varnim delovanjem... Za poenostavljeno konfiguracijo so na razpolago GUI vmesniki. Možna je integracija sistema za enkripcijo OpenSSL, skriptnih jezikov PHP in Tcl, programskih jezikov Python, Perl in Java, ter baze podatkov, na primer MySQL in PostgreSQL. Kombinacija Apache, MySQL, PHP je nepisani standard, ob katerem ne bo neizpolnjenih želja glede sodobnega in dinamičnega strežnika spletnih strani.



Slika 2.2 : Tržni deleži med 2.349.895 strežniki (november 2000)

2.2.1 Opis in konfiguracija

Čeprav se podrobnosti pri posameznih distribucijah razlikujejo, se v grobem namestijo enako. Apache strežnik inštaliramo z ukazom **rpm –ivb apache1.xxx-r4.rpm.** Tako v podimenikih /etc/httpd oz. /etc/apache (imenik /etc je mesto, kjer lahko najdemo nastavitvene datoteke za večino sistemskih programov) najdemo datoteko **httpd.conf**, ki je nastavitvena datoteka za Apache.

Datoteka ni prazna, temveč so v njej privzete vrednosti za večino nastavljivih spremenljivk in praktično je najbolje, da vse izmed njih pustimo pri miru, razen kadar imamo dober razlog za spreminjanje. Privzete vrednosti so namreč dobre za veliko večino scenarijev, le nekaj izmed njih je dejansko takih, ki jih je treba spremeniti v strežniku, da je pripravljen za strežbo.

Prva izmed njih je spremenljivka **ServerName**, ki nastavi ime, s katerim se strežnik predstavlja brskalnikom. To ime ne more biti karkoli, temveč mora ustrezati zapisu v imenskem strežniku DNS ali naslovu IP, če strežnik nima takega zapisa. Če si isti naslov deli več domen, lahko uporabimo katerokoli izmed njih (pač tisto, ki nam najbolj ustreza).

Naslednja spremenljivka, ki jo lahko spremenimo, je **ServerAdmin**. Le-ta določa elektronski naslov skrbnika spletnega strežnika. Ta naslov se pojavlja na straneh, ki jih strežnik ustvari ob napakah in zato na tem mestu vnesemo naslov, na katerega želimo dobivati sporočila uporabnikov o napakah v strežniku.

Naslednji korak, ki ni nujen, vendar je ponavadi zaželen, je da dodamo datoteko **index.php** med datoteke, ki se izvedejo, kadar brskalnik poda naslov, ki kaže na imenik namesto na točno določeno datoteko. To storimo tako, da dodamo datoteko **index.php** na seznam v direktoriju: **/var/www/html/diploma**

Vrstni red datotek je pomemben, ker bo Apache serviral prvo datoteko, ki jo bo našel v imeniku. Potem, ko smo nastavili te spremenljivke, je Apache že pripravljen za strežbo spletnih strani. Edino, kar moramo še narediti je, da strežnik ponovno zaženemo.

Ta korak je nekoliko odvisen od izbrane distribucije. Ker uporabljamo RedHat Linux distribucijo je dovolj, da izvedemo ukaz **service httpd restart**, drugod pa je še potrebno napisati /etc/rc.d/init.d/apache restart.

Zdaj naš strežnik streže statične spletne strani, vendar za pisanje spletnih strani še ne moremo uporabljati skriptnih jezikov kot so Python, PHP in Perl. Jezik, s katerim

želimo ustvariti dinamično ustvarjene strani, moramo najprej namestiti in konfigurirati. Kako, si lahko ogledamo na zgledu skriptnega jezika **PHP**, katerega bomo uporabljali za pisanje spletnih strani.

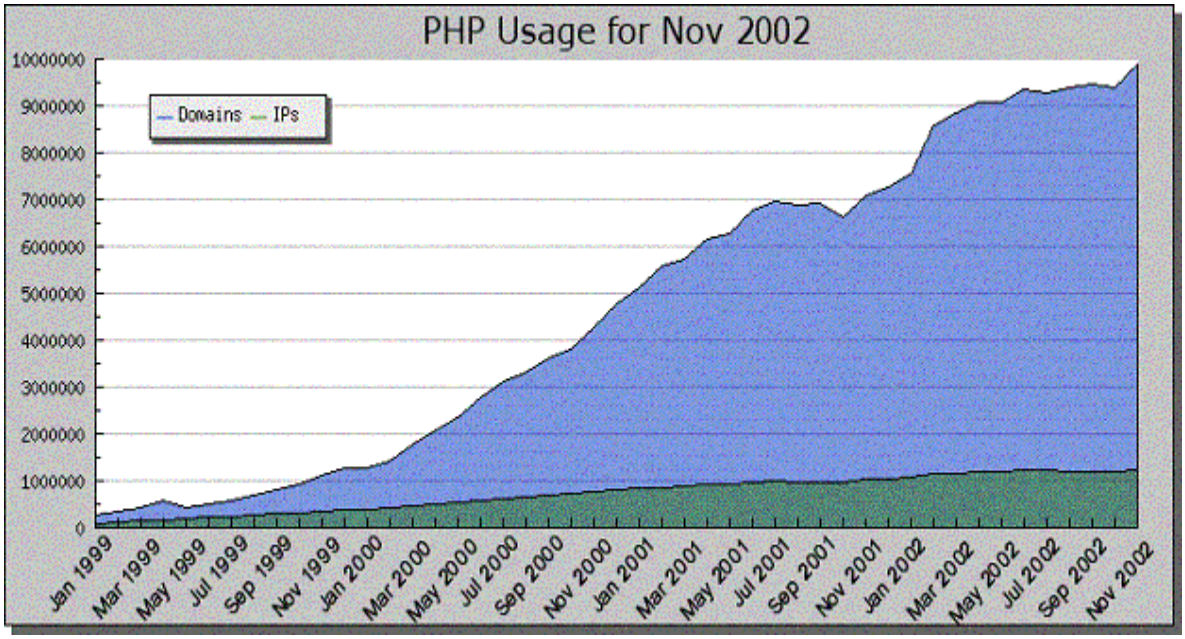
2.2.2 Opis in uporaba PHP skriptnega jezika

PHP je skriptni jezik, ki ga vključimo v opisni jezik HTML, kar nam omogoči izdelavo dinamičnih spletnih strani, ki so osnova za široko paleto storitev, ki jih danes najdemo v spletu. Sama inštalacija je zelo enostavna, saj ga inštaliramo z enim samim ukazom **rpm –ivb phpxx.rpm.** Ker je PHP kar nekakšen naravni izbor za pripravo dinamičnih spletnih strani, ki jih streže spletni strežnik Apache - res pa je na voljo tudi za druge strežnike, Apache pa ima močno podporo še za kup drugih programskih dodatkov - bomo v nadaljevanju povedali, za kaj je PHP najbolj zanimiv in pogosto uporabljen (slika 2.3) [5].

Strokovnjaki pravijo, da je PHP *HTML-embedded scripting language*. Po naše bi se reklo, da je PHP skriptni jezik, ki je vključen v opisni HTML jezik [6]. Vse skupaj se odvija v spletnem strežniku. Nekdo ″prideska″ na našo spletno stran, spletni strežnik zaradi končnice datoteke (.php ali .php3) zažene interni tolmač PHP, ki to stran obdela in kot rezultat vrne opisni jezik HTML, ki ga zna prikazati (skoraj) vsak spletni brskalnik.
V naših primerih bo vsakokrat poudarjeno, kdaj se začne in konča koda PHP. Spletni strežnik za začetek kode PHP uporablja znak <? za konec pa ?>. Vse kar je med oznakami za začetek in konec, se šteje za kodo PHP, ki jo obdela naš strežnik.

Vsakdo, ki vsaj malo pozna programski jezik C, bo opazil, da je PHP podoben C-ju, vendar v nekaterih primerih PHP ponuja veliko več kakor C in je dosti bolj prilagodljiv. Vse spremenljivke se začnejo z znakom $ (dolar). V PHP-ju spremenljivk ni treba vnaprej deklarirati, niti ni treba navesti, katerega tipa so. Različnih podatkovnih tipov je v primerjavi s kakim drugim jezikom malo, pa vendar za potrebe v spletnih straneh povsem zadoščajo. Komentarji v kodi se označujejo enako kakor v C-ju. Uporabimo lahko kombinacijo /* in */ ali pa //. Programiranje v PHP je najbolj razumljivo, če si predstavljamo, da je namesto monitorja naša izhodna naprava okno v spletnem brskalniku. Za izpis podatkov na našo izhodno napravo smo uporabili funkcijo **echo()**. Obstaja še

nekaj funkcij, ki smo jih vzeli iz jezika C (**printf()**, **print()**, **sprintf()**, **for()**...) in se vedejo, kakor bi C-programer pričakoval.



Slika 2.3 : Uporaba PHP-ja pri 9,866,075 domenah in 1,232,818 IP naslovih

Primer programa **index.php**:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2">
</head>

<? /* začetek kode PHP, od tu naprej dokument sproti
pripravlja strežnik */
echo "<body>";
echo "<h1>Živijo svet!</h1>";
echo "</body>";

/* konec PHP kode */
?>
</html>
```

Testni programček napišemo v php skriptu na direktoriju /var/www/html/diploma, nato pa v spletnem brskljalniku vpišemo http://localhost/diploma/index.php in na zaslonu se nam mora pojaviti izpis :

Živijo svet!

Če se nam to izpiše, potem nam pravilno deluje tako Apache web strežnik, kot PHP skriptni jezik in lahko nadaljujemo s povezavo z MySQL podatkovno bazo.

Najprej moramo spoznati osnovne ukaze MySQL baze. Kako se povezati z MySQL strežnikom in kako od strežnika zahtevati nek podatek, saj bomo na web strani uporabljali vnos in prikaz podatkov iz baze.

2.3 MySQL podatkovna baza

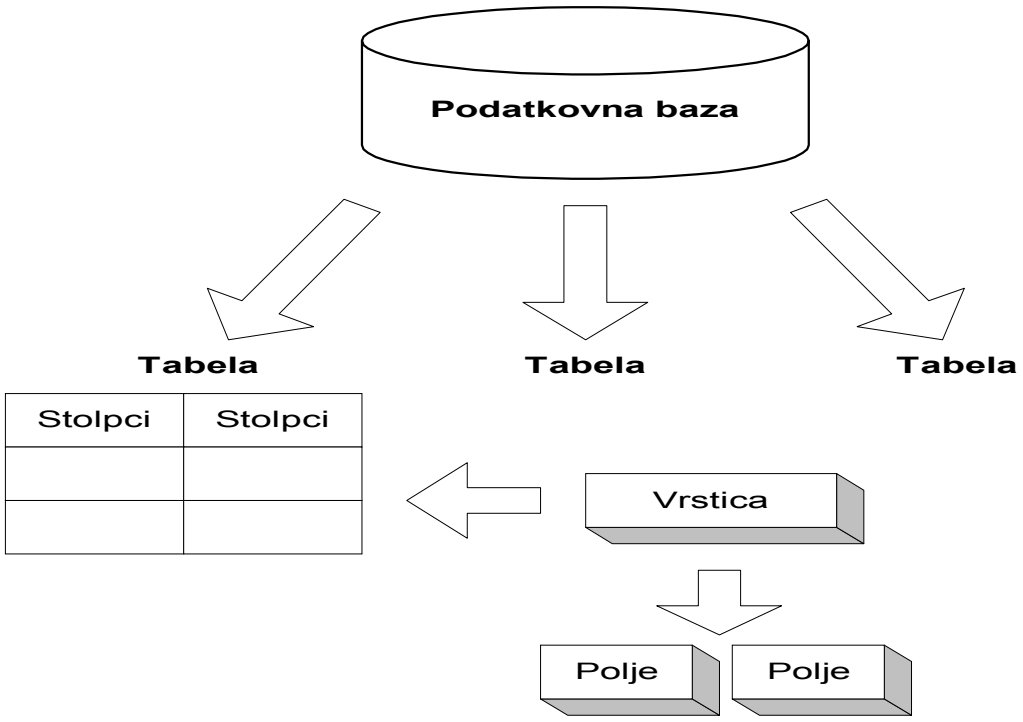
MySQL je strukturirana zbirka podatkov. To lahko zajema vse od preproste nakupovalne liste, galerije slik, do obsežnega števila informacij v družbenem omrežju. Načrtovalci, skrbniki, programerji in analitiki uporabljajo orodja, ki olajšajo opravila, prikrijejo kompleksnost informacijskih sistemov in omogočajo hitro ter učinkovito implementacijo novih zamisli in nujnih posegov. Mednje spadajo prav gotovo tudi sistemi za upravljanje baz podatkov (SUBP) [7], ki omogočajo shranjevanje in dostop do podatkov, organizacijo ter manipulacijo s podatki, izvajanje varnostnih opravil itd.

SUBP MySQL se uporablja za dodajanje, dostop in procesiranje podatkov shranjenih v bazi**.** Računalniki so zelo dobri pri obravnavanju velike količine podatkov. Zaradi tega ima upravljanje s podatki osrednjo vlogo pri računanju, kot samostojni pomožni program, ali pa kot del drugih aplikacij. **MySQL** je relacijski SUBP. Relacijska baza hrani podatke v različnih "separate" tabelah in ne shranjuje vseh podatkov le na eno. To dodaja hitrost in fleksibilnost. Omenjene tabele so povezane z relacijami, tako da je mogoče po želji združiti podatke iz več tabel. Del kratice SQL pri **MySQL** pomeni "Structured Query Language" - najbolj pogost standardiziran jezik, ki se uporablja za dostopanje do baze podatkov. **MySQL** je Open Source programska oprema. Odprti vir lahko vsak uporablja in spreminja. Kdorkoli ima možnost povleči **MySQL** iz interneta in ga brezplačno uporabljati. Za začetek je potrebno biti seznanjen z nekaj terminologije (slika 2.4):

- baza (database) - skupek vseh tabel, funkcij in nasploh vseh persistentnih podatkov, ki se nahajajo na strežniški strani,
- tabela (table) - zaključena celota večih podatkov z enako strukturo - enakimi vrsticami,

- vrstica (row) - samostojen vpis v tabeli, ki sestoji iz polj,
- polje (field) - najmanjša enota informacije znotraj ene vrstice; s polji je moč matematično in drugače manipulirati znotraj jezika SQL, zato morajo imeti polja določen tip (številski, textovni...).

Najvišja enota, s katero se lahko manipulira znotraj jezika MySQL, pa ni baza, pač pa tabela.



Slika 2.4 : Anatomija podatkovne baze

2.3.1 Administracija baze

Da lahko začnemo uporabljati bazo, jo moramo najprej štartati in sicer to naredimo z ukazom: **/sbin/service mysqld start**

Če je vse v redu, se nam na Linux konzoli izpiše da se je baza pravilno štartala in da jo lahko začnemo uporabljati.

Najprej moramo poskrbeti, da bomo imeli dostop do zbirke, se pravi uporabniško ime, geslo in dovoljenje skrbnika sistema za delo z zbirko. Ker bomo dostopali in

uporabljali bazo samo mi in noben drug uporabnik, se prijavimo kot root uporabnik z ukazom:

>**mysql –u root -p**

Izpiše se nam:

>Enter password: ****** (vpišemo root geslo za dostop do baze)

Ko to naredimo, vstopimo v bazo, na zaslonu se nam izpiše:

```
[root@localhost marko]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.23.49

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Sedaj smo prijavljeni na bazo mysql. Prepričamo se lahko z ukazom :

>**show databases;**

Izpiše se nam:

```
mysql> show databases;
+----------+
| Database |
+----------+

| mysql    |
| test     |
+----------+
3 rows in set (0.10 sec)
```

Sedaj moramo najprej narediti novo bazo (katero bomo vseskozi uporabljali) in ustrezne tabele v njej. Z ukazi MySQL se to iz ukazne vrstice naredi takole:

mysql>**create database diploma;**

Izpiše se nam:

Query OK, 1 row affected (0.06 sec)

Z ukazom CREATE DATABASE DIPLOMA smo naredili novo bazo, v kateri se bodo nahajale tabele s podatki. Pisanje z veliki črkami je dostikrat uporabljeno zaradi lažjega branja, ni pa obvezno, ker MySQL pri tem ne dela razlik med malimi in velikimi črkami. Temu sledi ime baze, ki jo želimo ustvariti, tudi to je lahko napisano z malimi ali velikimi črkami. Na koncu pa je podpičje, ki je obvezno na koncu vseh SQL stavkov. Če slučajno pritisnemo Enter preden napišemo podpičje, bo kazalec skočil v novo vrstico (namesto, da

bi se ukaz izvedel), kjer lahko nadaljujemo ukaz oz. če smo samo pozabili napisati podpičje, dopišemo le-tega in stisnemo Enter.

Naredili smo novo bazo, v kateri bomo sedaj kreirali našo tabelo, v katero bomo vpisovali podatke iz web-a in krmilnika.
Z ukazom:

mysql>**show databases**;

```
mysql> show databases;
+----------+
| Database |
+----------+
| diploma  |
| mysql    |
| test     |
+----------+
3 rows in set (0,10 sec)
```

bomo dobili seznam vseh obstoječih MySQL baz. Med njimi mora biti tudi nova baza z imenom "**diploma**". Ker pa lahko podatke vnašamo le v tabele, tabele pa lahko postavljamo le znotraj baz, moramo priti v eno izmed baz. Izbrali bomo seveda našo bazo "diploma". Bazo "diploma" izberemo z ukazom:

mysql>**use diploma;**

```
mysql> use diploma;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

s katerim dejansko vstopimo v bazo **diploma**. Na ta način lahko izberemo katerokoli bazo v MySQL strežniku, le besedo **diploma** zamenjamo z imenom željene baze.

S tem smo v glavnem naredili vse, kar je potrebno pri izdelavi administrativnega dela oz. za postavitev baze. Sedaj je potrebno kreirati tabelo, katero bomo kasneje uporabljali. Opisali bomo tudi nekaj najpogostejših ukazov, ki se uporabljajo za samo operiranje s tabelo in bazo.

2.3.2 Kreiranje tabel in uporaba

Definirali smo bazo in se lahko lotimo kreiranja tabel. Tabele lahko kreiramo čisto po lastnih željah.V našem primeru bomo kreirali tabelo **test**.

Tabela bo imela enaindvajset polj. Večina jih bo tipa integer, saj bomo za naše potrebe uporabljali samo nule (0) in enice (1).
Tabelo začnemo kreirati v ukazni vrstici:

mysql>create table test (Port1 **int**, Port2 **int**, Port3 **int**, Port4 **int**,
→ Polje5 **int**, Polje6 **int**, Polje7 **int**, Polje8 **int**,
→ Polje9 **int**, Polje9 **int**, Polje10 **int**, Polje11 **int**,
→ Polje12 **int**, Polje13 **int**, Polje14 **int**, Polje15 **int**,
→ Polje16 **int**, datum_vnosa **text**, Error **int**, Pe **int**,
→ Ack **int**, Selec **int**, id **int not null primary key auto_increment**);

Izpisalo se bo nekaj podobnega temu:

Query OK, 1 row affected (0.06 sec)

Pisanje v večih vrsticah se uporablja zaradi preglednosti, nikakor pa ni obvezno.
Z zgornjim ukazom naredimo tabelo z imenom **test**, ki bo imela tri lastosti. Ti stoplci so vedno navedeni v oklepaju. Lastnost pa so:

- Port1 **int**

Port1 je ime lastnosti, temu sledi tip, ki je v tem primeru **int**, kar pomeni integer (celo število). Če v oklepaju ne navedemo dolžine številke ki jo potrebujemo, nam baza sama določi dolžino 11 (v tem primeru je največja vrednost števil od -2147483648 do 2147483647),

- datum_vnosa **text**

Je tipa text, kar pomeni (niz (besedilo), ki nima fiksne dolžine), dolžina pa je omejena na 200 znakov,

- id **int not null primary key auto_increment**

id je ime lastnosti, temu sledi tip, ki je v tem primeru **int**, kar pomeni integer. Temu sledijo parametri. Parameter NOT NULL pove, da lastnost »id« ne bo vsebovala vrednosti NULL. S parametrom »auto_increment« bomo dosegli, da se ob vsakem dodajanju novega vnosa v tabelo, temu vnosu priredi vrednost id-ja za eno večja kot je pri prejšnjem vnosu. Da to omogočimo pa moramo zapisati tudi »primary key«. Pri tem lahko seveda lastnost »id« zamenjamo s katerokoli lastnostjo, ki je število.

Če hočemo sedaj pogledati če smo tabelo pravilno definirati, lahko to storimo z ukazom:
mysql>**describe test;**

Izpiše se nam:

```
mysql> desc test;
+-------------+---------+------+-----+---------+----------------+
| Field       | Type    | Null | Key | Default | Extra          |
+-------------+---------+------+-----+---------+----------------+
| Port1       | int(11) |      |     | 0       |                |
| Port2       | int(11) |      |     | 0       |                |
| Port3       | int(11) |      |     | 0       |                |
| Port4       | int(11) |      |     | 0       |                |
| Polje5      | int(11) |      |     | 0       |                |
| Polje6      | int(11) |      |     | 0       |                |
| Polje7      | int(11) |      |     | 0       |                |
| Polje8      | int(11) |      |     | 0       |                |
| Polje9      | int(11) |      |     | 0       |                |
| Polje10     | int(11) |      |     | 0       |                |
| Polje11     | int(11) |      |     | 0       |                |
| Polje12     | int(11) |      |     | 0       |                |
| Polje13     | int(11) |      |     | 0       |                |
| Polje14     | int(11) |      |     | 0       |                |
| Polje15     | int(11) |      |     | 0       |                |
| Polje16     | int(11) |      |     | 0       |                |
| datum_vnosa | text    |      |     |         |                |
| Error       | int(11) |      |     | 0       |                |
| Selec       | int(11) |      |     | 0       |                |
| Pe          | int(11) |      |     | 0       |                |
| Ack         | int(11) |      |     | 0       |                |
| id          | int(11) |      | PRI | NULL    | auto_increment |
+-------------+---------+------+-----+---------+----------------+
22 rows in set (0.00 sec)
```

Če so vsa polja takšna kot smo si želeli, potem lahko začnemo vstavljati ali brati podatke iz tabele. To storimo z naborom ukazov, ki jih je ogromno, mi pa bomo uporabljali samo en del. Vse te ukaze lahko uporabljamo kot samostojne ali pa v povezavi z drugimi programskimi jeziki (PHP, C...).

Insert

Poglejmo si kako v našo tabelo vstavimo novo vrstico:

INSERT INTO test (Port1, Port2, Port3…,datum_vnosa) VALUES (1,0,1…"sysdate ") ;

Torej, kot vidimo smo preprosto navedli vsebino polj, ki jih ima vsaka vrstica. Velja omeniti, da se nizi v SQL omejijo s pomočjo dvojnih zgornjih narekovajev.

Select

Seveda je glavni namen podatkovne baze možnost opravljanja vseh možnih vrst poizvedb. Za poizvedbe se uporablja ukaz **Select**, ki nam omogoča popoln nadzor nad vračanjem vrednosti, hkrati pa nudi možnost zelo kompleksnih poizvedb s pomočjo povezav med različnimi polji različnih tabel in podobno.

Najbolj enostavna oblika stavka SELECT je sledeča:

*SELECT * FROM test ;*

Stavek nam bo vrnil vsa polja vsake vrstice, ki je v tabeli **test**. Zaporedje polj bo enako kot je bilo podano ukazu »create«. Kakšno bo zaporedje vrstic, ki bodo vrnjene ni definirano, zato se na to ne smemo zanašati.

Če bi radi iz vsake vrstice dobili le posamezna polja bi bil ukaz SELECT naslednji:

SELECT Port1 FROM test WHERE Port1=0 ;

Tukaj je še nešteto možnosti z ukazom SELECT, ki pa jih pogledamo v priročnikih [8].

Delete

Ker včasih želimo kaj iz tabel tudi zbrisati, uporabljamo za to ukaz **Delete**. Uporaba je relativno enostavna:

DELETE FROM test ;

Ta stavek bo izbrisal iz tabele »test« vse vrstice. Brišemo lahko tudi posamezna polja, to pa naredimo tako, da uporabimo WHERE stavek:

DELETE FROM test WHERE Port1=0 ;

Update

Če želimo določene vrstice le spremeniti, uporabimo ukaz **Update**:

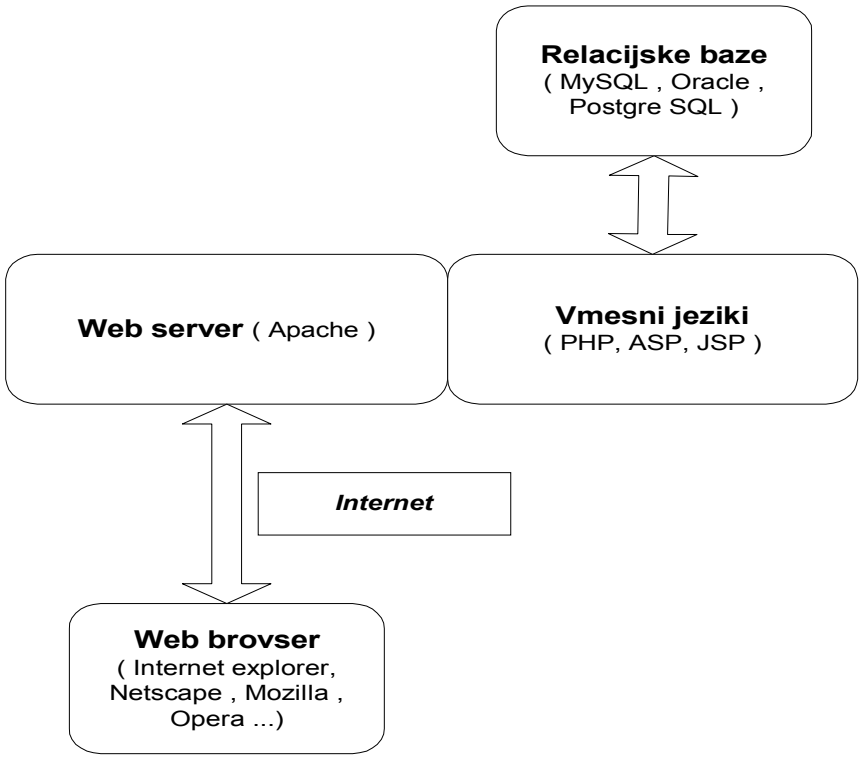
UPDATE test SET Port1 = 1 ;

Zgornji ukaz bo polje Port1 postavil na vrednost 1.

To bi bilo nekaj najpogostejših ukazov, s katerimi operiramo v MySQL bazi. Teh ukazov je še zelo veliko, vendar vseh ne bomo predstavljali.

Zakaj bi uporabljali **MySQL**?

MySQL je zelo hiter, zanesljiv in enostaven za uporabo. Poleg tega ima zelo praktičen niz značilnosti, ki so razvite v tesnem sodelovanju z uporabniki. **MySQL** je bil na začetku narejen za hitrejše upravljanje zelo velikih baz podatkov od ostalih rešitev. Ta povezanost, hitrost in varnost je tisto kar naredi **MySQL** zelo primernega za dostop do baz podatkov na internetu (slika 2.5).



Slika 2.5 : Arhitektura web aplikacij

2.4 Mitsubishi prosto programljivi krmilnik serije FX

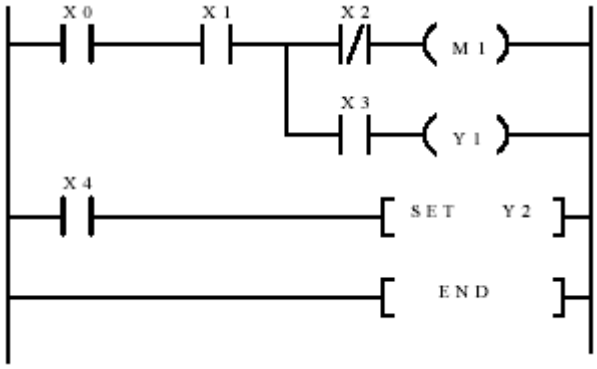
Za krmiljenje tekočega traku smo uporabili PPK Mitsubishi FX, saj je nameščen v laboratoriju za opravljanje laboratorijskih vaj. Na vhod PPK krmilnika smo pošiljali podatke, ki smo jih vpisali na web strani. Glede na vhodne podatke, pa smo napisali program za PPK, ki bo za različne vrednosti na vhodu izvajal različne aplikacije.Gre predvsem za krmiljenje tekočega traku. Ta tekoči trak lahko krmilimo ročno ali avtomatsko, odvisno od vhoda, vrti pa se lahko levo ali desno.

2.4.1 Programiranje in uporaba

Za programiranje prosto programrljivega sistema **Mitsubishi** služi program **medoc** [9], ki ga najdemo na šolskem računalniku v direktoriju **C:\MEDOC**. Poženemo ga z ukazom **medoc.exe** (ali samo **medoc**).

Na zaslonu, ki se pokaže, imamo kurzor, s katerim se pomikamo po ukazni vrstici (s kurzorskimi tipkami). Izbiranje poteka s tipko **<ENTER>** ali s kurzorsko tipko dol. Iz spodnjih nivojev se vračamo v višje s pritiskom na kurzorsko tipko gor.Najprej izberemo **Start** in v njem **New_Proj**. Zdaj je potrebno vpisati tip PPS, ki je v našem primeru **FX**. Izbira poteka s kurzorskima tipkama gor in dol. Ko smo postavili kurzor na **FX**, pritisnemo **<ENTER>** in vpišemo ime projekta (npr. dipl).S kurzorsko tipko gor se nato pomikamo na višje nivoje in ko je kurzor spet na **Start**, se pomaknemo na **Edit**. Ko izberemo to opcijo, imamo na voljo orodja za programiranje in dokumentiranje. Izberemo opcijo **Ladder** in s pritiskom na tipko **<F2>** vstopimo v polje KON. Zdaj moramo napisati program za PPK, ki ga v tem primeru zapišemo s kontaktno shemo, katere vpis je mogoč po pritisku na tipko **<F7>**. S pritiskom na tipke od **<1>** do **<9>** in **<F9>** izbiramo elemente.

Za primer si oglejmo naslednjo kontaktno shemo (slika 2.6):



Slika 2.6 : Primer kontaktne sheme

Delamo po naslednjih korakih:

1. Pritisnemo **<1>** in vpišemo **X0 <ENTER>**

2. Pritisnemo **<1>** in vpišemo **X1 <ENTER>**

3. Pritisnemo **<5>** - za vejitev

4. Pritisnemo **<2>** in vpišemo **X2 <ENTER>** (negiran vhod)

5. Pritisnemo **<7>** in vpišemo **M1 <ENTER>**

6. S kurzorsko tipko se pomaknemo desno za navpi•no •rto vejitve (ena vrstica navzdol)

7. Pritisnemo **<1>** in vpišemo **X3 <ENTER>**

8. Pritisnemo **<7>** in vpišemo **Y1 <ENTER>**

9. Pritisnemo **<6>** - ravna črta za eno mesto

10. Pritisnemo **<1>** in vpišemo **X4 <ENTER>**

11. Natipkamo **SET <ENTER> Y2 <ENTER>**

12. Natipkamo **END <ENTER>**

Program je zdaj napisan. S pritiskom na **<F7>** in nato **<F2>** se vrnemo v ukazni menu. Pomaknemo se na ukaz **Test** in pritisnemo **<ENTER>**. Če se izpiše *No errors found*, je program OK. Program shranimo tako, da izberemo **Save**. Nato se s pritiskom na kurzorsko tipko gor (2x) pomaknemo v višji nivo in izberemo opcijo **Transfer**. Tukaj izberemo opcijo **Medoc>PLC** in program se naloži. Pri tem mora biti stikalo **RUN/STOP** v položaju **STOP.** S kurzorsko tipko za gor (2x) se vrnemo v osnovni menu in izberemo **Edit** in nato **Ladder**. Pritisnemo **<F2>** in nato še **<F8>**. Zdaj lahko opazujemo delovanje

PPK na zaslonu PC-ja. V spodnji vrstici se prikazujejo registri, ki so uporabljeni v na zaslonu nahajajočih se ukazih. Stikalo **RUN/STOP** pomaknemo v položaj **RUN** in postavljamo vhode z žicami ali s tipkami na plošči PPK (slika2.7). Nato s tipko **<F8>** in nato še z **<F2>** zapustimo okno s **KON** ter se s kurzorsko tipko gor pomaknemo v osnovni menu, kjer izberemo funkcijo **Quit** in nato **Yes**. V oknu za **KON**, kamor pridemo z **<F2>** (v **Edit**, **Ladder**) so najpomembnejši ukazi:

<F7> - spremenimo vrstico, dodamo vrstico

<F6> - izbrišemo vrstico

<F5> - vstavimo vrstico

<F8> - opazujemo delovanje PPK na zaslonu PC

Simboli so:

X - vhodi (X0,...,X7)

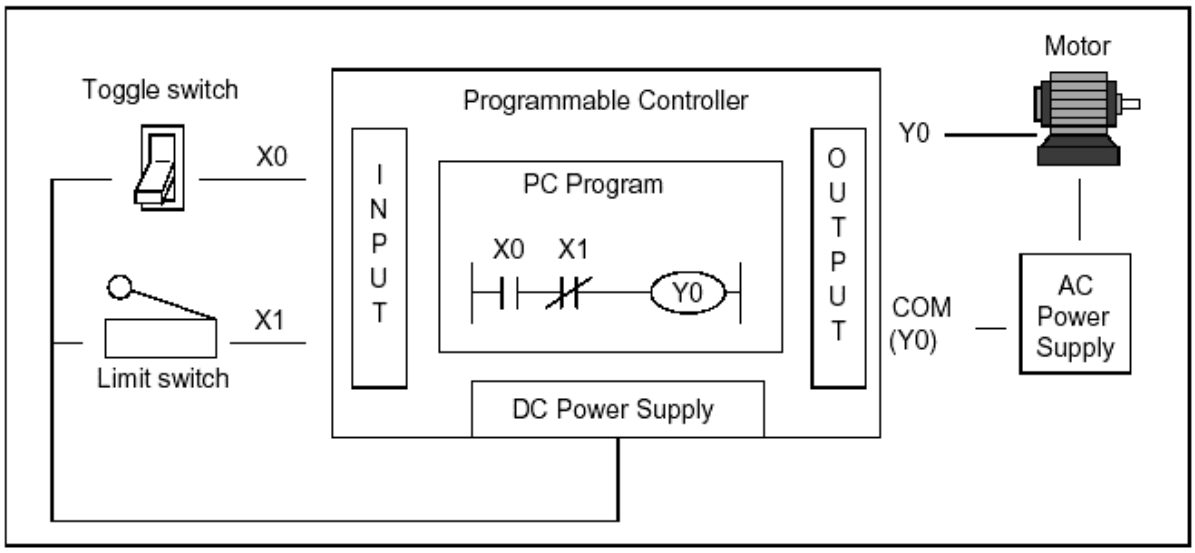
Y - izhodi (Y0,...,Y7)

M - merkerji (M0,...,M499)

D - podatkovni 16 bitni registri (D0,...,D199)

K - konstanta

H - šestnajstiška (heksadecimalna) konstanta



Slika 2.7 : Primer ročnega vodenja PPK

2.4.2 Delo s paralelnimi vrati LPT1

Paralelna vrata LPT1 [10] smo uporabili za dostop do PPK Mitsubishi FX. LPT1 ima ponavadi naslov 0x378 (naslov vrat je heksadecimalni(slika2.9)). Poslane podatke z weba, smo zapisali v podatkovno bazo, od tam pa smo jih programsko prenesli na 25-delne pine paralelnih vrat LPT1 (slika2.8). Pošiljali smo samo štiri bite (D0-D3).

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out PaperEnd	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer nSelect-In	In/Out	Control	Yes
18 - 25	19-30	Ground	Gnd		

Slika 2.8 : Razporeditev pinov 25-polnega konektorja paralelnih vrat

Address	Notes:
3BCh - 3BFh	Used for Parallel Ports which were incorporated in to Video Cards and now, commonly an option for Ports controlled by BIOS. - Doesn't support ECP addresses.
378h - 37Fh	Usual Address For LPT 1
278h - 27Fh	Usual Address For LPT 2

Slika 2.9 : LPT naslovi

Osnovne naslove imenujemo podatkovna vrata ali podatkovni register, ki pa se uporablja za prenos podatkov po podatkovni liniji: pini 2-9 (slika2.10). Ti pini so ponavadi (domača uporaba-printer) samo za pisanje. Če pa beremo z vrat, pa dobimo zadnji zlog, ki je bil poslan. V našem primeru smo uporabili linije od (Data0 – Data3).

Offset	Name	Read/Write	Bit No.	Properties
Base + 0	Data Port	Write (Note-1)	Bit 7	Data 7 (Pin 9)
			Bit 6	Data 6 (Pin 8)
			Bit 5	Data 5 (Pin 7)
			Bit 4	Data 4 (Pin 6)
			Bit 3	Data 3 (Pin 5)
			Bit 2	Data 2 (Pin 4)
			Bit 1	Data 1 (Pin 3)
			Bit 0	Data 0 (Pin 2)

Slika 2.10 : Podatkovna vrata

Statusna vrata (osnovni naslov + 1) se uporabljajo samo za branje (slika2.11). Vsak podatek, ki se pošlje na ta vrata se ignorira. Statusna vrata so sestavljena iz 5-ih vhodnih linij (Pini 10-15), IRQ statusnega registra in dveh rezerviranih bitov. Ker nam PPK vrača štiri bite o stanju na njegovih vhodih, smo uporabili statusne bite od (Bit3 – Bit6).

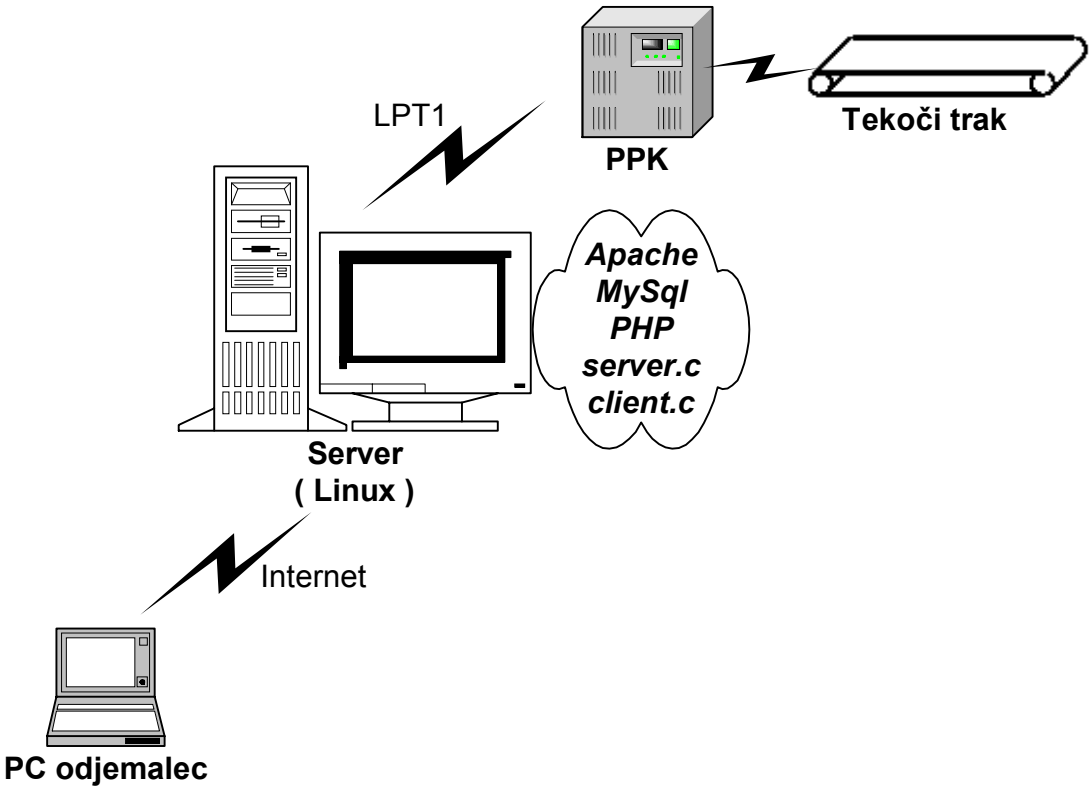
Base + 1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

Slika 2.11 : Statusna vrata

2.5 Zaključek

Kot je bilo že rečeno v uvodu, je naloga te diplomske naloge izdelati povezavo med Webom - MySQL podatkovno bazo in PPK Mitsubishi FX-om (slika2.12). Da smo lahko naredili to povezavo je bilo potrebno veliko programskih orodij, da je povezava stekla in pravilno delovala. Do sedaj smo opisali predvsem del, kako se posamezni programi inštalirajo, administrijajo ter pravilno nastavijo, da jih lahko uporabljamo.

V drugem delu diplomske naloge pa bomo predvsem skušali podrobno opisati praktični oz. programski del. Predstavili bomo programje, ki smo ga napisali za komunikacijo Web – MySQL - PPK .



Slika 2.12 : Delovanje sistema Web – MySQL - PPK

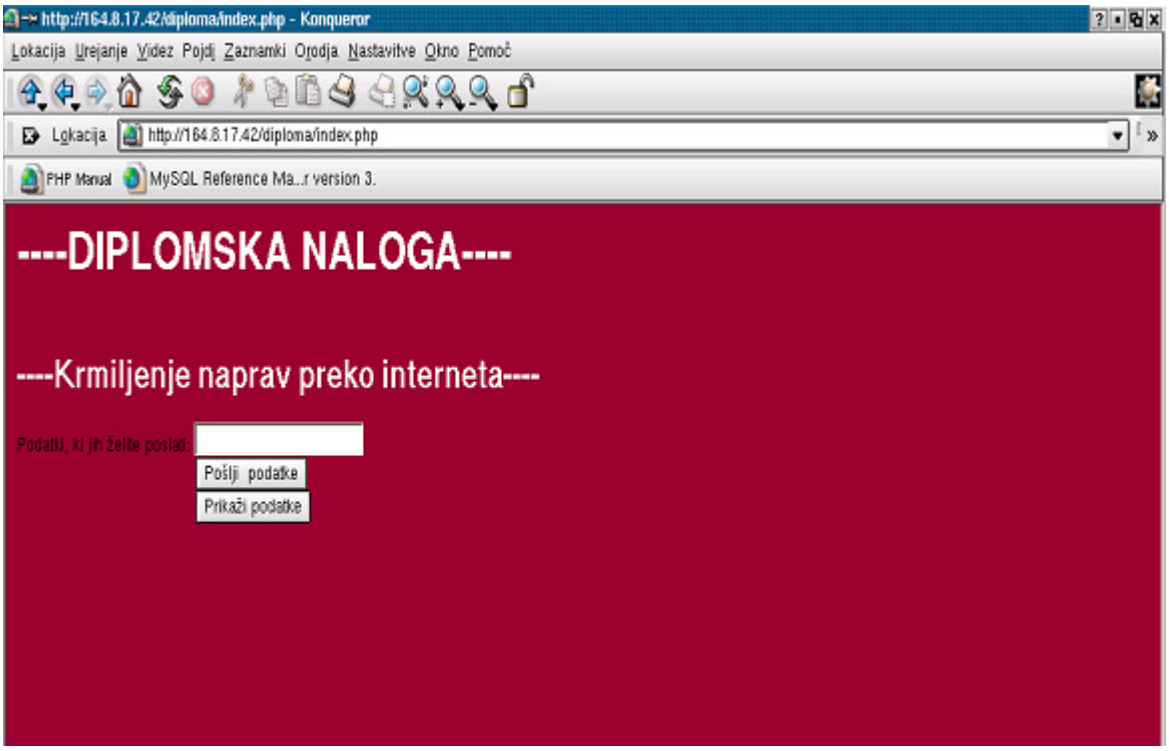
3. Povezava WEB – MySQL – PPK Mitsubishi FX

Web – MySQL – PPK Mitsubishi FX je povezava oz. aplikacija, s katero krmilimo preko interneta tekoči trak. Da smo lahko to realizirali, smo uporabili že zgoraj našteto programsko opremo, sedaj pa bomo opisali oz. razložili delovanje programov, ki smo jih napisali v različnih programskih jezikih (C , PHP , Medoc). Sama povezava je razdeljena na tri dele : - web aplikacija (PHP)

- branje in pisanje podatkov v MySQL podatkovno bazo (server.c , client.c)
- pošiljanje podatkov na LPT1 vrata in PPK Mitsubishi

3.1 Vnos podatkov na Web strani

Da smo lahko podatke vnašali na web strani (slika3.1), smo morali uporabiti Apache strežnik ter PHP skriptni jezik, da smo lahko napisali dinamično spletno stran.



Slika 3.1 : Web stran za pošiljanje podatkov

Na sami spletni strani imamo eno vnosno polje, katero nam služi za prenos vpisanih podatkov preko interneta na serverski računalnik. Prenašamo lahko šestnajst bitov naenkrat, čeprav dejansko uporabljamo samo štiri. Vsi ostali biti so rezervirani biti, če bi imeli priključenih več mikrokrmilnikov na glavnem računalniku. Ker vidimo tukaj samo izgled spletne strani bomo opisali tudi program, s katerim izvedemo to spletno stran.

3.1.1 Program napisan v PHP skriptnem jeziku

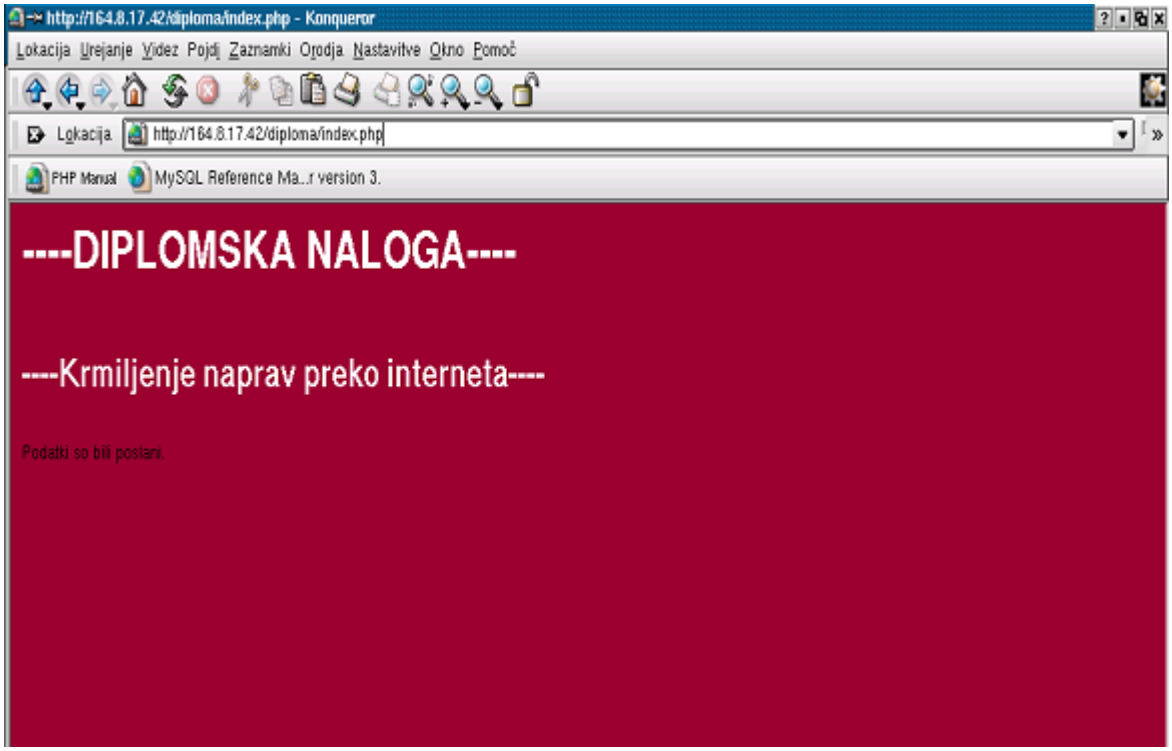
Ker je PHP program sestavljen iz dveh delov, in sicer iz dela s katerim vnašamo podatke na glavni računalnik in v podatkovno bazo, ter dela s katerim beremo podatke iz podatkovne baze in jih potem prikazujemo na spletni strani, bomo tukaj opisali samo del za vnos podatkov. Program se imenuje **index.php** in je shranjen na Apache serverju in sicer na direktoriju **/var/www/html/diploma.**

/* Tukaj uporabimo html kodo, za sam izgled spletne strani. Na začetku spletne strani se nam izpiše ----DIPLOMSKA NALOGA----, ter malo nižje ----Krmiljenje naprav preko interneta----. Lahko izbiramo velikost črk, ter barvo ozadja. */

```
-------------------------------------------------
<html>
<body bgcolor="#990033" text="#000000" >
<p><font face="Arial,Arial,Helvetica">
<b><font color="#FFFFFF" size="8">----DIPLOMSKA NALOGA----
</font></b></font></p><br></br>
<b><font color="#FFFFFF" size="6">----Krmiljenje naprav preko interneta--
--</font></b></font></p><br></br>
-------------------------------------------------

/* Sedaj pride na vrsto PHP koda. Uporabimo ukaz switch, saj lahko potem
uporabimo različne case ukaze. V tem delu bomo opisali ukaz »case
Execute« za pošiljanje podatkov. */

-------------------------------------------------
<?php
      switch ($HTTP_POST_VARS['Action']) {

            /* S tem ukazom pokličemo program client.c ki je shranjen na
            istem direktoriju na Apache serverju, kot ta program
            index.php! Podatke, ki jih tukaj vpišemo, prenesemo na
            client.c program in ta potem naprej operira z njimi. */


            case "Execute" :
                  $command = "./client " . $HTTP_POST_VARS['param'];
                  $output = `$command`;
```

```
        /* Izgled strani po upešnem prenosu podatkov */
        /* Ko podatke pošljemo, se nam na ekranu izpiše
        nasljednja vrstica */
        echo "<P>Podatki so bili poslani.<P>";
        echo $output;
        break;
/* Seveda če hočemo pošiljati podatke, moramo napisati kako
mora stran izgledati na prvem mestu pred prenosu! To storimo
z ukazom default, ki pa je le en del switch ukaza.  */

default:
/* Izgled strani pred pošiljanjem ... */
echo "<form action=\"index.php\" method=\"POST\">\n" .
     "<input type=hidden name=Action value=Execute>\n" .
     "Podatki, ki jih želite poslati:\n" .
     "<input type=text name=param><br>\n" .
     "         
               
      <input type=Submit name=submit value=\"Pošlji
      podatke\">\n" .
     "</form>\n" . }
?> /* Konec php kode  */
</body>
</html>
```

Po izvedbi programa **index.php** oz. po prenosu podatkov, se nam na spletni strani izpiše stavek »Podatki so bili poslani« (slika3.2). To pomeni, da ni prišlo do napak in da je prenos podatkov uspel, saj bi nam v nasprotnem primeru strežnik javil napako.



Slika 3.2 : Web stran po poslanih podatkih

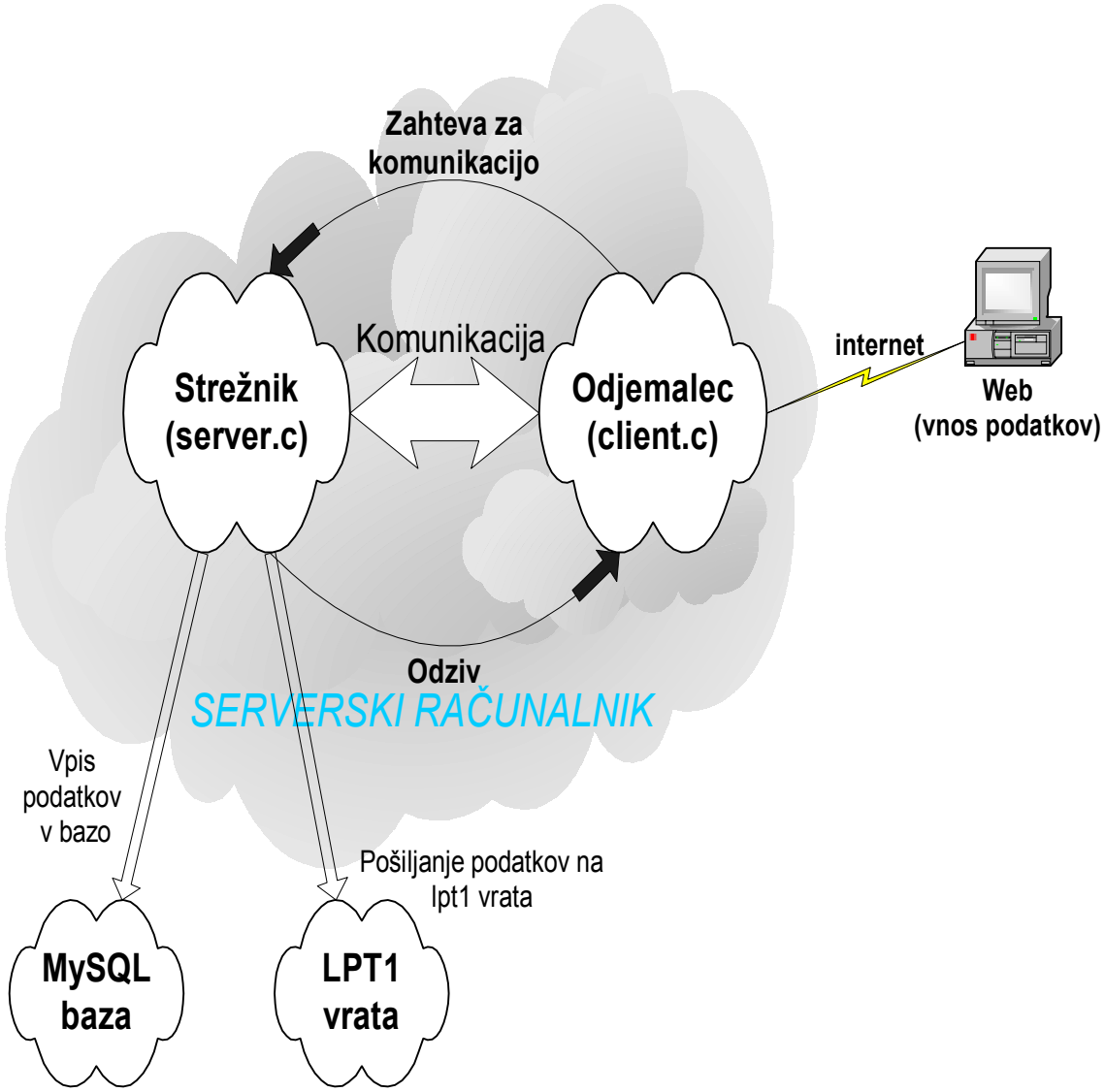
To bi bil program za pošiljanje podatkov iz web-a. Sedaj pa sledi povezava med programoma odjemalec (**client.c**) – strežnik**(server.c)**, saj podatke, ki smo jih vpisali na web strani prenesemo na program »client.c«, ki pa potem naprej komunicira s programom »server.c«, ki pa operira s podatkovno bazo.

3.2 Komunikacija programa odjemalec – strežnik

Komunikacija odjemalec - strežnik je razmeroma preprosta. Komunikacija med njima poteka preko TCP/IP povezave. V grobem bi lahko podatke pošiljali direktno v MySQL bazo in potem s strežniškim programom na LPT1 vrata, vendar smo se za program »client.c« odločili predvsem zato, da lahko podatke pošiljamo na dva načina, in sicer tako iz glavnega računalnika, kot tudi preko web-a, ki nam je prioriteta.

Programa med sabo komunicirata tako, da si pošiljata pakete (socket-e) z različnimi oznakami, ki jim pravimo komunikacijske kode. Vsaka koda ima drugačen pomen oz. vsaka koda ima za posledico različno akcijo programa. Kako poteka izmenjavanje podatkov med programoma, bomo skušali s pomočjo diagramov pokazati v nadaljevanju.

Kot prvo bomo opisali postopek komunikacije med client.c – server.c programom (slika3.3), kar pomeni, da bomo najprej opisali razne pomembnejše dele programa client.c, kako deluje in kako komunicira s programom server.c, kasneje pa še strežniški program server.c



Slika 3.3 : Komunikacija odjemalec - strežnik

Program **client.c** izvaja dokaj preprost program odjemalca, ki inicializira neblokirajočo TCP/IP povezavo s strežnikom, mu pošlje nekaj nizov znakov (stringov), ki jih strežnik zbira. Pri EOF (zahteva po prekinitvi povezave Ctrl C) prosi odjemalec strežnik, naj mu vrne povezane nize znakov, ki jih nato izpiše in konča delovanje.
V Linux konzoli prevajamo program z ukazom:

gcc –o client client.c

zaganjamo ga pa z ukazom :

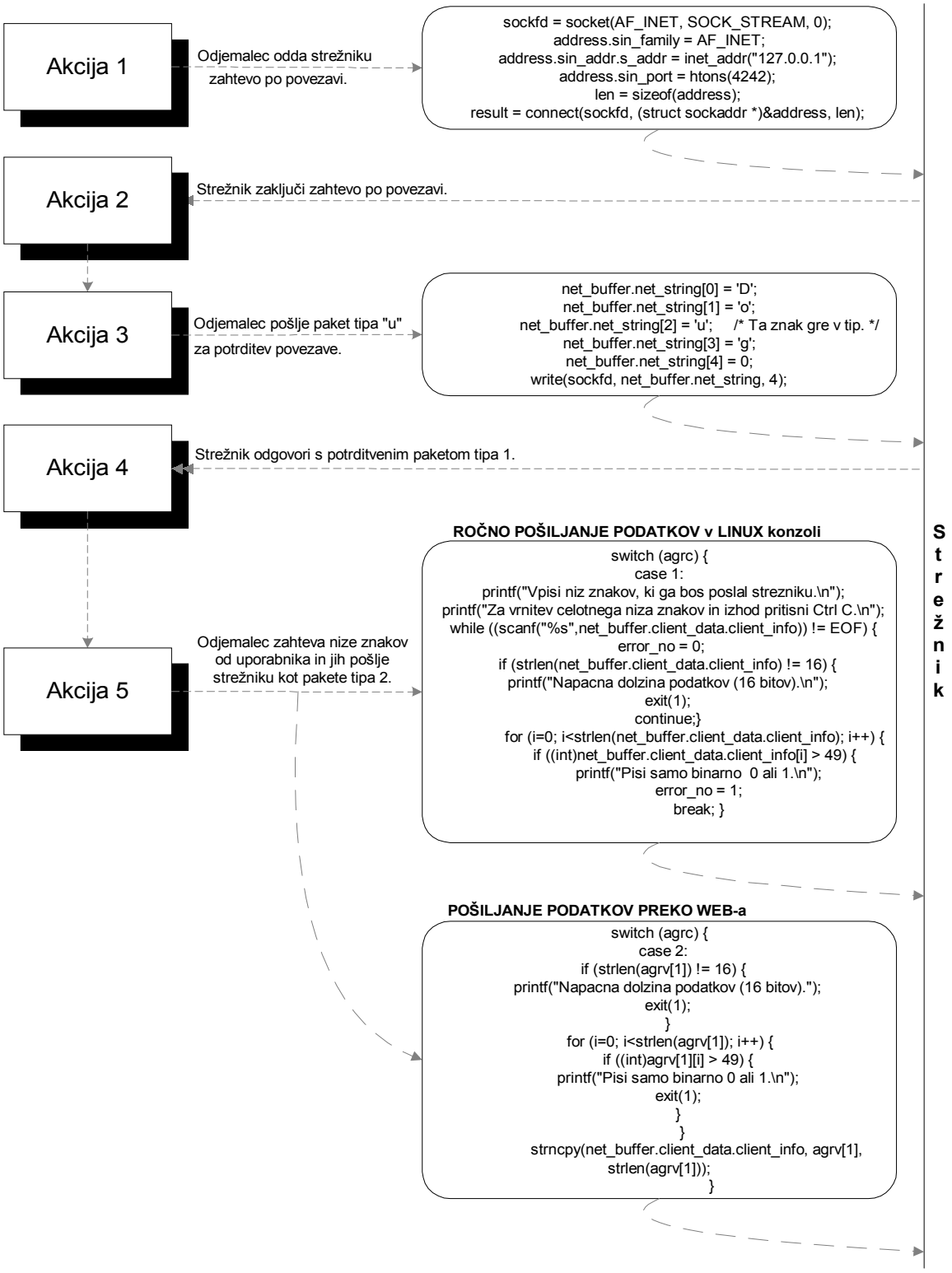
./client

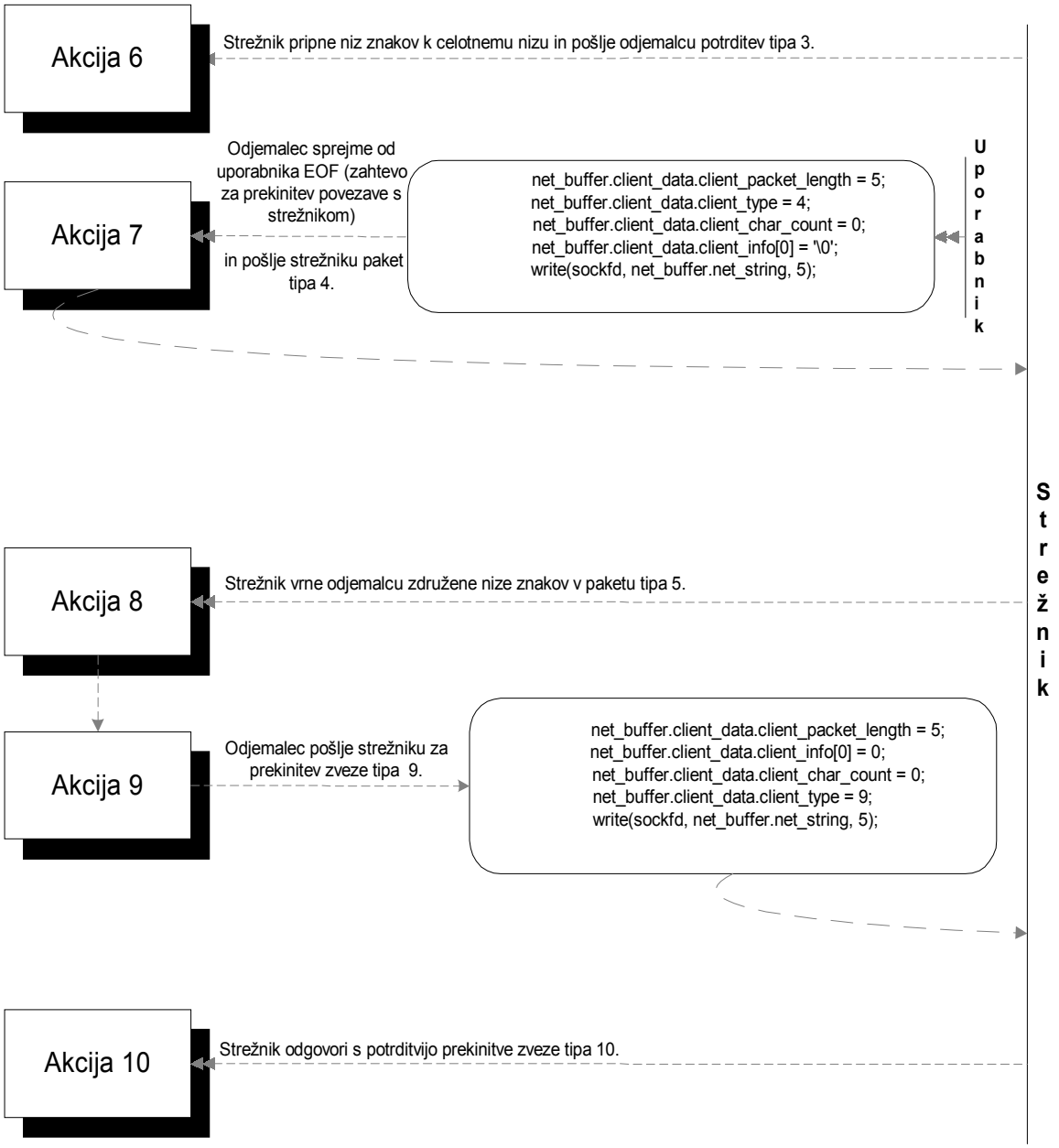
Da bomo v nadaljevanju lažje razumeli diagram delovanja programa »client.c« in »server.c«, moramo najprej razložiti komunikacijske kode, ki jih programa uporabljata kot označbe paketov za izmenjavanje podatkov:

```
/*****************************************************************************************************************/
 *
 *     Komunikacijske kode, uporabljene za komunikacijo Strežnik <-> Odjemalec.

 * Koda                    Funkcija                                                    Smer
 *  ---------------------------------------------------------------------------------------------------------------
 * u   Koda za akcijo.     Zahteva po povezavi.                                       Odjemalec->Strežnik.
 * 1   Koda za odgovor.    Potrditev zahteve po povezavi.                             Strežnik->Odjemalec.
 * 2   Koda za akcijo.     Povezan niz znakov z odjemalca.                            Odjemalec->Strežnik.
 * 3   Koda za odgovor.    Niz znakov za potrditev s strežnika.                       Strežnik->Odjemalec.
 * 4   Koda za akcijo.     Zahteva za povezan niz s strežnika.                        Odjemalec->Strežnik.
 * 5   Koda za odgovor.    Pošiljanje povezanega niza znakov odjemalcu.               Strežnik->Odjemalec.
 * 6
 * 7
 * 8
 * 9   Koda za akcijo.     Odjemalčeva zahteva po prekinitvi zveze s strežnikom.      Odjemalec->Strežnik.
 *10   Koda za odgovor.    Potrditev prekinitve zveze odjemalcu.                      Strežnik->Odjemalec.
 *
/*****************************************************************************************************************/
```

Kode, ki smo jih opisali, uporabljata tako odjemalec kot strežnik, kot oznako paketov. Najpomembnejše v programu »client.c« je del, kjer vnašamo podatke, saj imamo dve možnosti. Prva možnost je, da podatke vpišemo na Linux konzoli, kjer program ročno zaženemo in tudi potem ročno vpišemo podatke, ki jih nameravamo poslati strežniku. Druga možnost, ki pa je najpomembnejša, je da podatke pošiljamo preko web-a. To nam omogoča del, kjer imamo ukaz **switch** ter ukaza **case 1** in **case 2**. Prav tako v tem delu preverjamo dolžino vnesenega niza znakov, ki mora biti dolžine šestnajst in ali se v tem

nizu zakov pojavlja še kaj drugega kot binarne nule (0) ali enice (1). Če kateri od teh pogojev ni izpolnjen, nam program vrne napako na web strani ali Linux konzoli. Sedaj bomo z diagramom poteka opisali, kako odjemalec komunicira s strežnikom (slika 3.4):

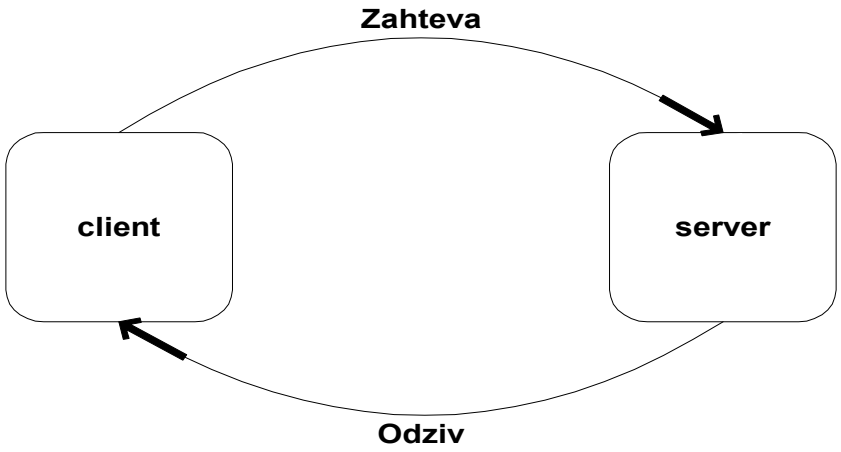




Slika 3.4 : Diagram poteka delovanja odjemalca

Sedaj bomo opisali še razne pomembnejše dele programa **server.c,** kako deluje in kako komunicira s programom **client.c**. Ker je **server.c** relativno dolg program, ga bomo razdelili na več delov. Kot prvo bomo opisali osnovno komunikacijo s programom **client.c,** kasneje pa še kako program zapisuje podatke v MySQL bazo, jih pošilja na LPT1 vrata, ter jih sprejema z LPT1 vrat in le te vpisuje nazaj v bazo.

Kot že samo ime programa »server.c« pove, nam ta strežniški program omogoča akcije na glavnem računalniku in ima to nalogo, da streže odjemalcu glede na njegove zahteve (slika3.5). Preko TCP/IP povezave komunicira z odjemalcem in mu odgovarja s paketi, ki imajo imajo različne oznake, glede na oznako paketov prejetih od odjemalca. Ko enkrat strežnik zaženemo na Linux konzoli, je v pripravljenosti tako dolgo, da začne z njim komunicirati odjemalec oz. da se vzpostavi povezava med programoma. Po vzpostavitvi povezave začne strežnik izvajati akcije, ki mu jih narekuje odjemalec.



Slika 3.5 : Komunikacija odjemalec (client) – strežnik (server)

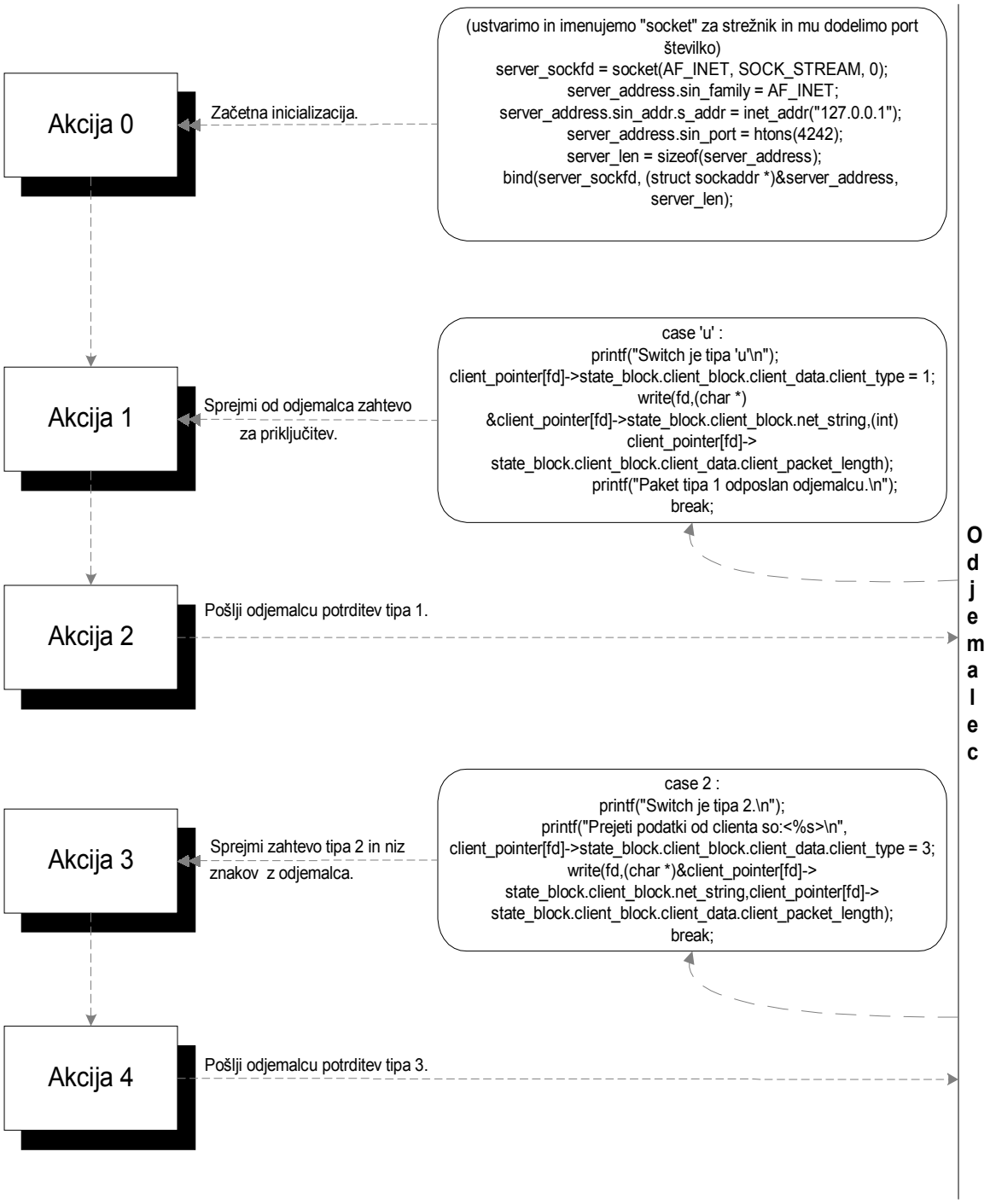
V Linux konzoli prevajamo program »server.c« z ukazom:

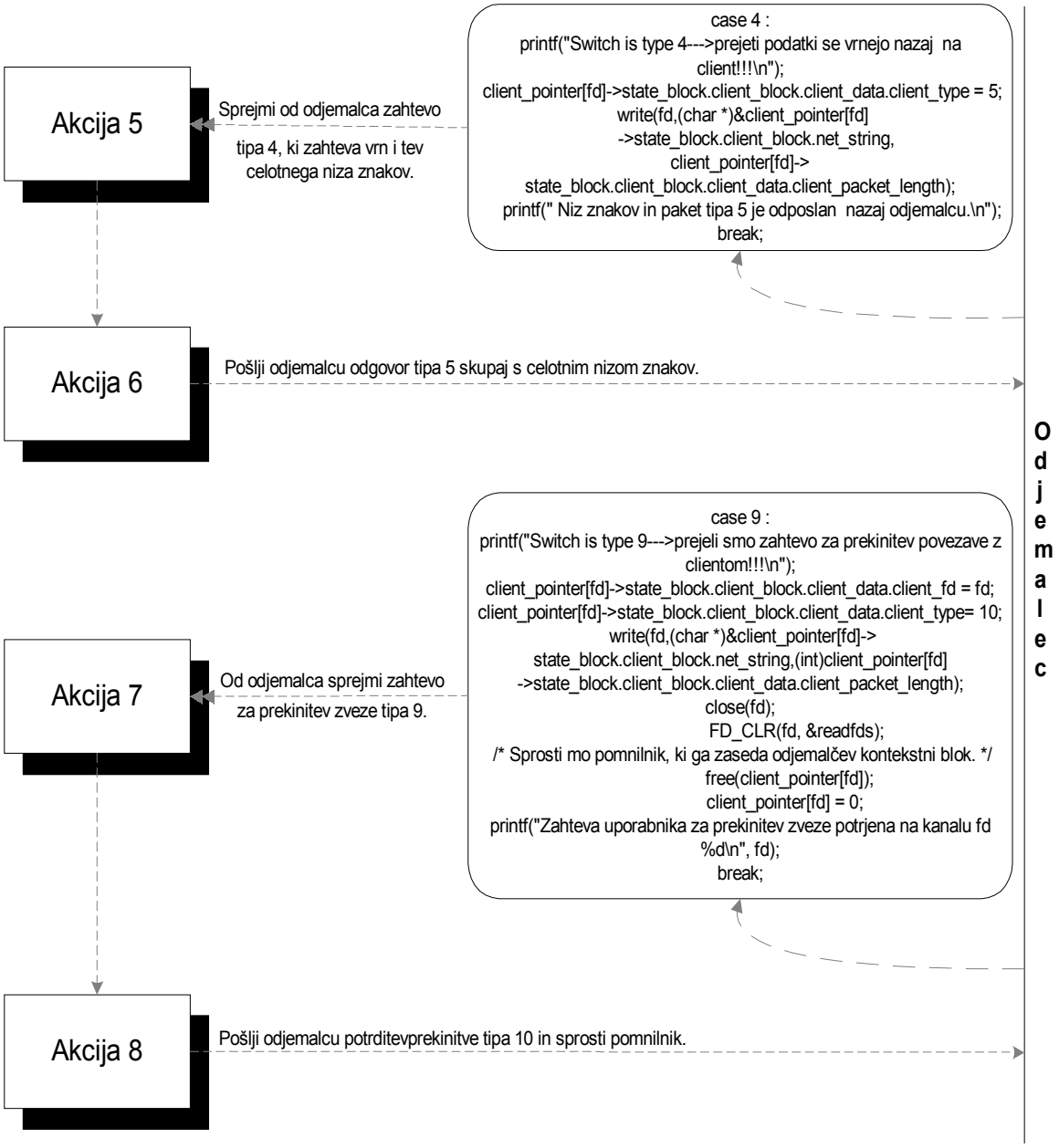
gcc server.c - o server – lmysqlclient –L/usr/lib/mysql

zaganjamo ga pa z ukazom :

./server

Sedaj bomo z diagramom poteka opisali (slika3.6), kako strežnik komunicira z odjemalcem, glede na vhodne oznake (tip) paketov odjemalca:


(ustvarimo in imenujemo "socket" za strežnik in mu dodelimo port številko)
server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = htons(4242);
server_len = sizeof(server_address);
bind(server_sockfd, (struct sockaddr *)&server_address, server_len);
Akcija 0
Začetna inicializacija.
case 'u' :
printf("Switch je tipa 'u'\n");
client_pointer[fd]->state_block.client_block.client_data.client_type = 1;
write(fd,(char *)
&client_pointer[fd]->state_block.client_block.net_string,(int)
client_pointer[fd]->
state_block.client_block.client_data.client_packet_length);
printf("Paket tipa 1 odposlan odjemalcu.\n");
break;
Akcija 1
Sprejmi od odjemalca zahtevo za priključitev.
Akcija 2
Pošlji odjemalcu potrditev tipa 1.
Odjemalec
case 2 :
printf("Switch je tipa 2.\n");
printf("Prejeti podatki od clienta so:<%s>\n",
client_pointer[fd]->state_block.client_block.client_data.client_type = 3;
write(fd,(char *)&client_pointer[fd]->
state_block.client_block.net_string,client_pointer[fd]->
state_block.client_block.client_data.client_packet_length);
break;
Akcija 3
Sprejmi zahtevo tipa 2 in niz znakov z odjemalca.
Akcija 4
Pošlji odjemalcu potrditev tipa 3.



Slika 3.6 : Diagram poteka delovanja strežnika

Na začetku programa, moramo najprej nastaviti razne parametre, ki jih potrebujemo za pravilno delovanje strežnika (ustvarimo in imenujemo »socket-e« za strežnik in mu dodelimo port številko). Potem čakamo na odjemalca in njegove zahteve da nas kontaktira. Ker smo kot »timeout parameter« uporabili »\0«, ne bo prišlo do prekoračitve časa, ki ga ima odjemalec da kontaktira strežnik. V programu imamo »while« zanko, ki ima različne naloge. Najprej ima nalogo, da čaka tako dolgo, dokler ne dobi strežnik zahtevo po povezavi. Ko ugotovimo, da imamo aktivnost, ugotovimo na katerem descriptorju je.

Descriptor uporabljamo v programu zato, da se lahko prijavi na strežnik več uporabnikov. Če je aktivnost na »server_sockfd«, imamo zahtevo za ponovno povezavo oz. novi odjemalec bi se rad prikjučil na strežnik. Če pa ni aktivnosti na strežniku, je aktiven odjemalec, kar pomeni, da je odjemalec nepričakovano prekinil povezavo, zato ga odstranimo iz množice deskriptorjev in sprostimo odjemalčev konteksni blok. Kot smo že omenili, če imamo novo povezavo, lahko začnemo z izmenjavanjem podatkov z odjemalcem. To je sedaj najpomembnejši del programa, kjer strežnik preverja tip bloka (komunikaciijske kode), ki jih prejema od odjemalca. Uporabimo ukaz »**switch**«, ki nam omogoča da preverjamo tip podatkov, prejetih od odjemalca.

Sedaj bomo opisali, kaj strežnik naredi, če dobi določen tip podatkov:

- Ukaz tipa »**case 'u'**« pomeni začetno medsebojno komunikacijo z odjemalcem. Odjemalcu odgovorimo z blokom tipa1, kar pomeni potrditev povezave.

- Ukaz tipa »**case 2**« pomeni, da sprejmemo nov niz podatkov k obstoječemu, če obstajajo. Odjemalcu odgovorimo z blokom tipa 3, kar pomeni da smo potrdili prejem znakov.

- Ukaz tipa »**case 4**« pomeni, da odjemalec zahteva vrnitev celotnega niza znakov nazaj. Odjemalcu odgovorimo z nizom znakov in blokom tipa 5.

- Ukaz tipa »**case 9**« pomeni, da smo od odjemalca sprejeli zahtevo po prekinitvi povezave. Odjemalcu odgovorimo z blokom tipa 10, zapremo kanal s podatki in sprostimo kontekstni blok pomnilnika (zbrišemo spomin v pomnilniku).

Če se na tem mestu pojavi kakršen drug tip podatkov je to napaka, ki jo lahko samo javimo na zaslon. Iz opisanega lahko razberemo, da je komunikacija strežnika z odjemalcem razmeroma enstavna. Najpomembnejši del programa se odvija, ko imamo tip bloka 2, saj takrat sprejmemo podatke od odjemalca, ki nato z njimi operiramo. V našem primeru te podatke zapišemo v MySQL podatkovno bazo. Kako pa to naredimo, pa si bomo pogledali v nadaljevanju.

3.2.1 Vpis prejetih podatkov v MySQL podatkovno bazo

Vpis podatkov v MySQL podatkovno bazo se izvrši, če imamo v programu **server.c** tip bloka 2 (ukaz: **case 2**). Da lahko v samem programu uporabljamo podatkovno bazo, moramo vključiti knjižnico, ki nam omogoča, da program napisan v C kodi lahko operira z MySQL podatkovno bazo. Vključimo tudi »define«, da nam kasneje ni potrebno za vsako pisati gesla, imena baze ... Da pa baza pravilno deluje, ji moramo definirati tudi njene spremenljivke.

```
/* MySQL knjižnica in definicije */
#include <mysql/mysql.h>
#define MYSQL_HOST "localhost"       /* ime baznega serverja */
#define MYSQL_DB "diploma"           /* ime baze */
#define MYSQL_USER "diploma"         /* uporabniško ime za vstop v bazo */
#define MYSQL_PASSWD "diploma"       /* geslo za vstop v bazo  */
#define MYSQL_TABLE "test"           /* ime tabele v bazi  */

 /*MySql spremenljivke*/
 MYSQL mysql;
 MYSQL_RES *rezultat;
 MYSQL_ROW row;
 char q_str[255] = "";
 char q_str1[180] = "";
 char q_str2[2] = "";
 int str_i;
```

Ko smo vse spremenljivke definirali, lahko začnemo pisati kodo za vnos podatkov v bazo. Kot prvo bomo opisali glavne MySQL ukaze, ki smo jih uporabili. Funkcije, ki jih bomo uporabljali, se vse začnejo s predpono **mysql:**

- mysql_init( ) -> inicializacija baze
- mysql_connect( ) -> povezava na bazo
- mysql_select_db( ) -> izbira baze
- mysql_query( ) -> stavek ki ga pošljemo bazi
- mysql_error( ) -> vrne sporočilo o napaki iz baze
- mysql_close( ) -> prekinjanje povezave z bazo

Vsem naštetim funkcijam smo še dodali razne spremenljivke, tako da lahko sedaj napišemo kodo za dostop in vpis podatkov v bazo:

```
case 2 :
                /* printf("Switch je tipa 2.\n"); */
                /* Izpiše vsebino sprejetega vmesnega pomnilnika (bufferja). */
                  printf("Prejeti podatki od clienta so:<%s>\n",
                  client_pointer[fd]->state_block.client_block.client_data.client_info);

                        /*** Insert v MySQL bazo ****/
                        mysql_init(&mysql);
                        if (!mysql_connect(&mysql, MYSQL_HOST, MYSQL_USER, MYSQL_PASSWD)) {
                            printf("Napaka pri povezovanju: .");
                            printf(mysql_error(&mysql));
                            }
                        else {
                            printf("MySQL connect OK!\n");
                              }

                        printf("Pisanje podatkov v bazo...\n");
                        mysql_select_db(&mysql, MYSQL_DB);
                        strcpy(q_str, "INSERT INTO test (Port1, Port2, Port3, Port4, Polje5, Polje6, Polje7, Polje8, Polje9, Polje10,  Polje11,
                        Polje12,  Polje13, Polje14, Polje15, Polje16, datum_vnosa) VALUES(");
                                /* printf("q_str:%s\n",q_str), */
                                strcpy(q_str1, client_pointer[fd]->state_block.client_block.client_data.client_info);
                                for (str_i = 0; str_i<16; str_i++) {
                                        strncpy(q_str2, &q_str1[str_i],1);
                                        strncat(q_str, q_str2,1);
                                        strncpy(q_str2, ", ", 2);
                                        strncat(q_str, q_str2,2);
                                        }
                                        strncpy(q_str2, "DATE_FORMAT(NOW(), '%d-%m-%y %T'))", 34);
                                        strncat(q_str, q_str2,34);
                                        printf("Query:-->%s\n", q_str);
                                        mysql_query(&mysql, q_str);
                                        /*printf("Error: %s", mysql_error(&mysql)); */

                                        /* Zapremo povezavo z MySql bazo */
                                        mysql_close(&mysql);
```

MySQL stavek:

INSERT INTO **test** (Port1, Port2, Port3, Port4, Polje5, Polje6, Polje7, Polje8, Polje9, Polje10, Polje11, Polje12, Polje13, Polje14, Polje15,Polje16,datum_vnosa) VALUES(");

napolnimo s podatki (binarne cifre ki jih vpiše uporabnik), ki jih prejmemo od odjemalca in na koncu, ko se napolni izgleda takole:

INSERT INTO **test** (Port1, Port2, Port3, Port4, Polje5, Polje6, Polje7, Polje8, Polje9, Polje10, Polje11, Polje12, Polje13, Polje14, Polje15,Polje16,datum_vnosa) VALUES(1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,"date format (now(), %d%m%h %T");

Ti podatki se nato zapišejo v tabelo **test** in jih kasneje, ko jih dopolnimo še s povratnimi informacijami iz krmilnika prikažemo na web strani.

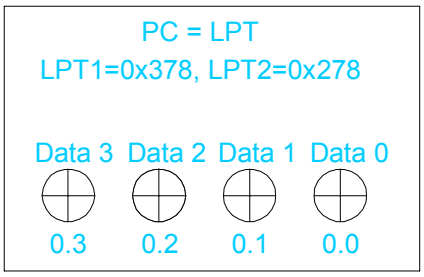
3.2.2 Pošiljanje podatkov na LPT1 vrata in vhod PPK Mitsubishi FX-a

Prav tako, kot pri vpisu podatkov v MySQL podatkovno bazo, iste podatke uporabimo za pošiljanje na LPT1 vrata in s tem na vhod PPK.

Kot prvo moramo definirati naslov lpt1 vrat, ki imajo ponavadi za pošiljanje podatkov naslov 0x378. Ko to storimo, preverimo če imamo dostop oz. na voljo vrata. Nato postavimo vse vrednosti na lpt1 vratih na vrednost nič. Nato lahko začnemo preverjati, če ima kateri od bitov, ki jih bomo poslali vrednost 1. Če jo ima, pošljemo na enega od naslovov

- Data0 (0x01)
- Data1 (0x02)
- Data2 (0x04)
- Data3 (0x08)

binarno enico (slika3.7). Vsi naslovi, kamor pošiljamo podatke so v šestnajstiškem sistemu.



Ime	LPT	Vhod/Izhod
IZH0=1	Data 0 (0.0)	Izhod
IZH1=1	Data 1 (0.1)	Izhod
IZH2=1	Data 2 (0.2)	Izhod
IZH3=1	Data 3 (0.3)	Izhod

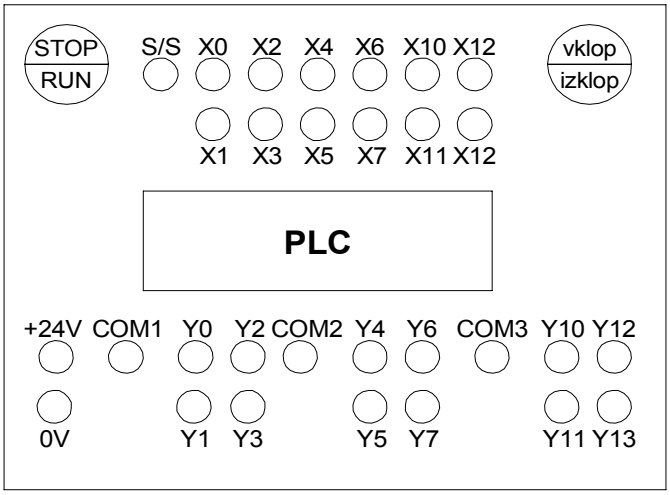
Slika 3.7 : Pošiljanje podatkov na LPT1 vrata

V programski kodi najprej preverimo, če imamo dostop do lpt1 vrat. Nato zbrišemo vse obstoječe vrednosti na vratih oz. jih postavimo na nič (0). Štiri bite, ki bomo poslali na lpt1 vrata, moramo najprej razparsati in preveriti na kakšno vrednost so postavljeni. Vsak bit, ki je postavljen na ena (1), na koncu zložimo skupaj z ostalimi in jih pošljemo na lpt1 vrata.

Kako izgleda programska koda za pošiljanje podatkov na LPT1 vrata si lahko ogledamo tukaj :

```
/* Definicija naslova LPT1 porta */
#define LPT 0x378

/******* Dostop do LPT porta - spremenljivke ******/
char port2[4+1]="";
char port[1+1]="";
int port_i;
int str_j,value,value2;

   /* Preverjanje uporabnika in lpt porta*/
   if (ioperm(LPT,3,1))
                {
                printf("Napaka: Ne morem dostopati do LPT1 %x\n", LPT);
                exit(1);
                }

    /* Vse vrednosti na portu postavimo na nič oz. ga zbrišemo */
           value=0;
           outb(value,LPT+0);

    /**** Pošiljanje podatkov na LPT port *****/
                strncpy(port2, client_pointer[fd]->state_block.client_block.client_data.client_info,4);
                strcat(port2,"\0");
                printf("Poslani štirje biti na port:<%s>\n",port2);

                                                value = 0;              /* Vrednost value postavimo na 0! */

                                                        /* Štiri  bite, ki bomo poslali na lpt 1 port in st tem na vhod PPK-ja, moramo
                                                           najprej razparsati in preveriti njihove vrednosti, da jih lahko na koncu zložimo
                                                           skupaj  in pošljemo na port */

                                                       if (port2[0] == '1')
                                                         {
                                                          strcpy(port,"1");
                                                          printf("port-D0:<%s>\n",port);
                                                          value=0x01;

                                                         } else {strcpy(port,"0");
                                                                printf("port-D0:<%s>\n",port);}

                                                        if (port2[1] == '1')
                                                         {
                                                          strcpy(port,"1");
                                                          printf("port-D1:<%s>\n",port);
                                                          value=value|0x02;

                                                         } else {strcpy(port,"0");
                                                                printf("port-D1:<%s>\n",port);}

                                                        if (port2[2] == '1')
                                                         {
                                                          strcpy(port,"1");
                                                          printf("port-D2:<%s>\n",port);
                                                          value=value|0x04;

                                                         } else {strcpy(port,"0");
                                                                printf("port-D2:<%s>\n",port);}

                                                        if (port2[3] == '1')
                                                         {
                                                          strcpy(port,"1");
                                                          printf("port-D3:<%s>\n",port);
                                                          value=value|0x08;

                                                         } else {strcpy(port,"0");
                                                                printf("port-D3:<%s>\n",port);}

                                                        /*Vse bite z vrednostjo 1 ali 0 pošljemo skupaj zložene na LPT1 port*/
                                                        outb(value, LPT+0);
```

S tem, ko smo poslali podatke na LPT1 vrata, smo jih hkrati tudi na vhod krmilnika (slika3.8). Ko dobimo enkrat podatke na krmilnik, se tam zažene preprost program, ki krmili tekoči trak.



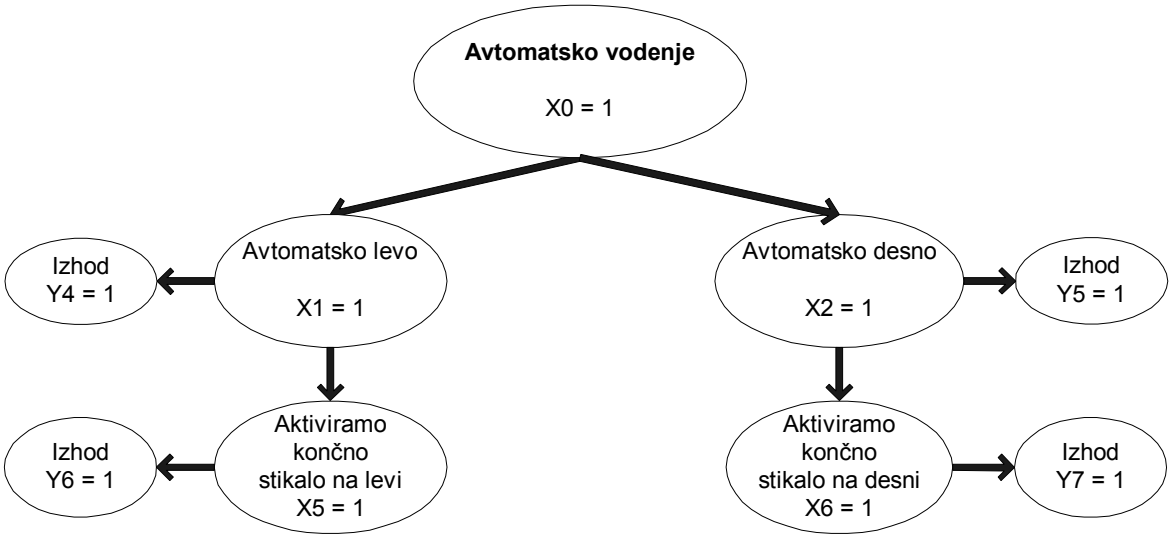
Slika 3.8 : Čelna plošča PPK krmilnika

3.2.3 Izvajanje programa na PPK Mitsubishi FX glede na vhodne podatke

Za Mitsubishijev prosto programljivi krmilnik serije FX, smo morali realizirati krmiljenje sistema tekočega traku. Kot smo že omenili, je krmilnik povezan z osebnim računalnikom preko paralelnih vrat, preko katerih dobi tudi podatke na svoj vhod. Komunikacijska karta ima tri vhode, eden pa je rezervni (VH0, VH1, VH2 in VH3) in štiri izhode (IZH0, IZH1, IZH2, IZH3). Pomen vhodov/izhodov je nasljednji:

Ime	**Vhod/izhod/oznaka**	**Pomen**
VH0	Vhod (X0)	Ročno ali avtomatsko vodenje
VH1	Vhod (X1)	Avtomatsko levo
VH2	Vhod (X2)	Avtomatsko desno
VH3	Vhod (X3)	Rezerviran bit
IZH0	Izhod (Y4)	Tekoči trak se giblje v levo
IZH1	Izhod (Y5)	Tekoči trak se giblje v desno
IZH2	Izhod (Y6)	Tekoči trak – levo končno stikalo
IZH3	Izhod (Y7)	Tekoči trak – desno končno stikalo

Sam program se izvaja na naslednji način. Imamo dve možnosti krmiljenja, in sicer ročno ali vodenje preko osebnega računalnika.
Če imamo avtomatsko vodenje, ga izvedemo na naslednji način:



Slika 3.9 : Prikaz avtomatskega vodenja tekočega traku

Na vhod PPK moramo dobiti za avtomatsko vodenje vhod oblike »1000« v binarnem sistemu. Delovanje programa: - z **X0 = 1** določimo, ali zadevo vozimo ročno ali s PC-ja

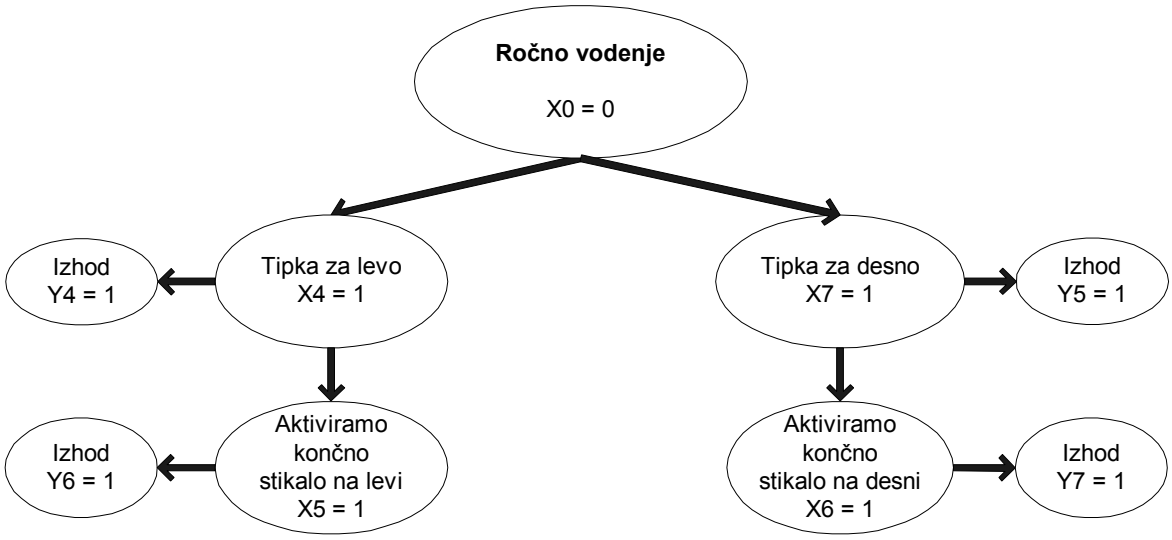
- z **X1 = 1** se tekoči trak vrti levo
- z **X2 = 1** se tekoči trak vrti desno

Torej, če hočemo, da zadevo vodimo s PC-ja in da se trak vrti desno, moramo na vhod PPK-ja poslati dva bita z vrednostjo »1« z lpt1 vrat (0x01, 0x02).

- z **X5 = 1** aktiviramo končno stikalo na levi (senzor)
- z **X6 = 1** aktiviramo končno stikalo na desni (senzor)

Ker podatke pošiljamo tudi nazaj na LPT1 vrata, si moramo pripraviti oz. postaviti izhode na PPK-ju:
- z **Y4 = 1** postavimo izhod ACK (tekoči trak se vrti levo)
- z **Y5 = 1** postavimo izhod PE (tekoči trak se vrti desno)
- z **Y6 = 1** postavimo izhod SELECT (aktiviramo levo končno stikalo)
- z **Y7 = 1** postavimo izhod ERROR (aktiviramo desno končno stikalo)

Torej tekoči trak se začne gibati, ko se na ustreznem vhodu pojavi logična enica in se giblje tako dolgo, dokler ni doseženo ustrezno končno stikalo ali pa ga mi predhodno ustavimo. Sistem lahko vodimo tudi s pritiskom na ustrezne tipke (T4, T5, T6 in T7), kjer pa je vodenje izvedeno tako, da se gibanje izvaja tako dolgo, dokler je pritisnjena tipka. Sedaj lahko še opišemo ročni način vodenja, kot dopolnilo k avtomatskemu vodenju.

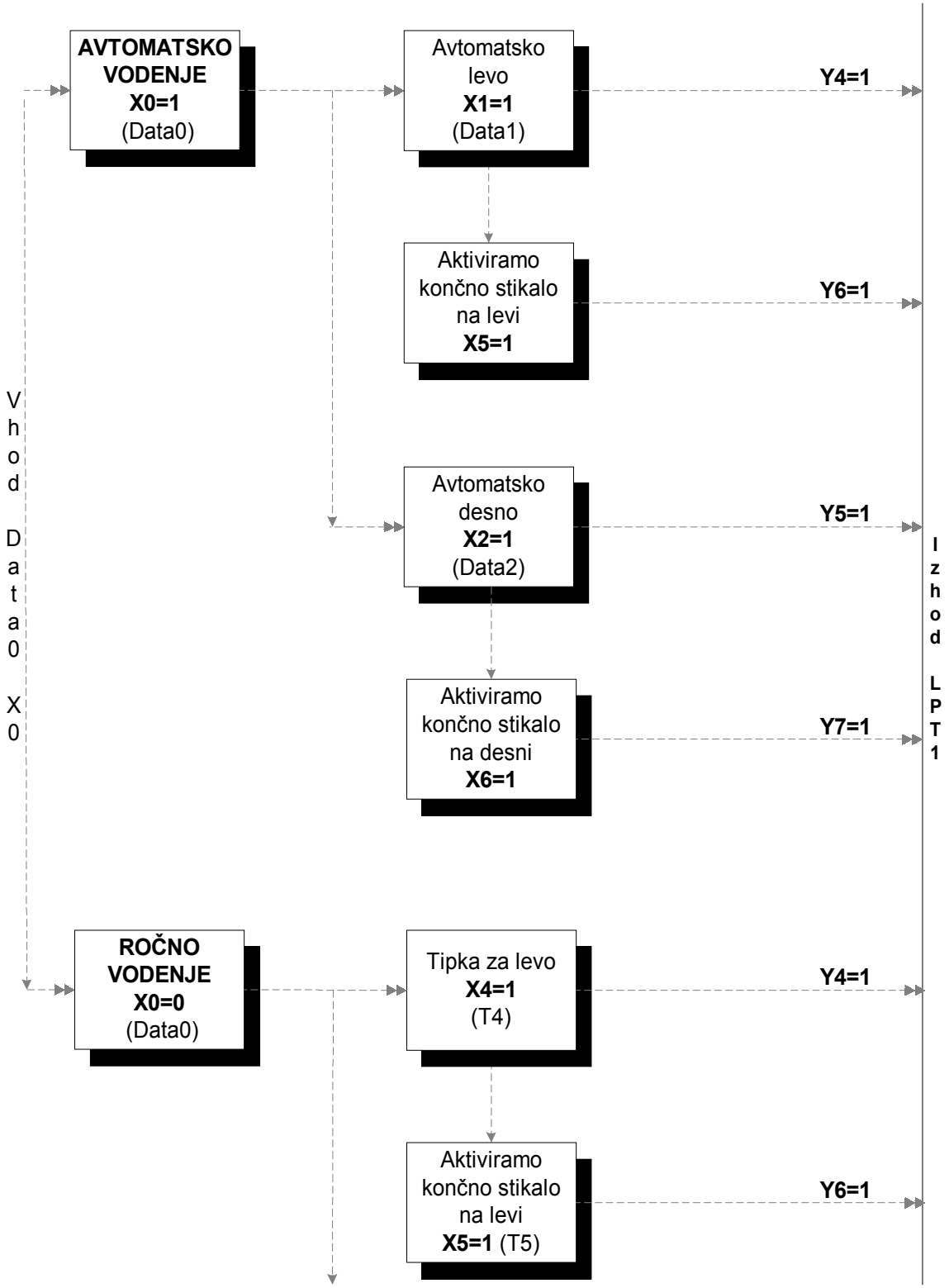


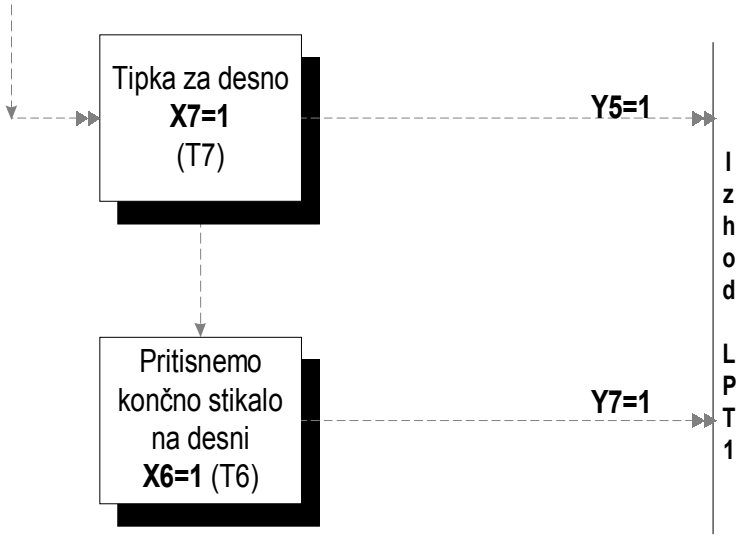
Slika 3.10 : Prikaz ročnega vodenja tekočega traku

Izbiro med ročnim in avtomatskim vodenjem, izbiramo z X0 vhodom. Če imamo X0 vhod postavljen na 1, potem onemogočimo ročno vodenje in komuniciramo s krmilnikom preko paralelnih vrat, v nasprotnem primeru pa imamo ročno vodenje, ki ga izvajamo preko tipk. Tukaj imamo sedaj še tri tipke, ki pa jih uporabljamo zgolj za varno delovanje programa:

- **X10** preprečitev avtomatskega delovanja
- **X11** ponovna omogočitev avtomatskega in ročnega delovanja
- **X12** preprečitev ročnega delovanja

Orodje, ki nam omogoča da napišemo program se imenuje **medoc**, program pa naredimo z metodo, ki se imenuje kontaktna shema (priloga D). Kako se program izvaja, pa bomo pokazali z diagramom poteka (slika 3.11):


AVTOMATSKO VODENJE X0=1 (Data0)
Avtomatsko levo X1=1 (Data1)
Y4=1
Aktiviramo končno stikalo na levi X5=1
Y6=1
Avtomatsko desno X2=1 (Data2)
Y5=1
Aktiviramo končno stikalo na desni X6=1
Y7=1
Vhod Data0 X0
Izhod LPT1
ROČNO VODENJE X0=0 (Data0)
Tipka za levo X4=1 (T4)
Y4=1
Aktiviramo končno stikalo na levi X5=1 (T5)
Y6=1

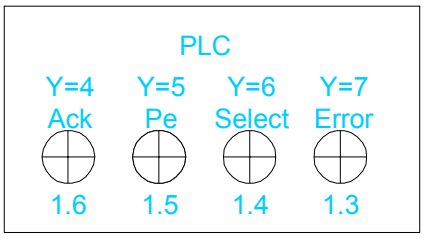


Slika 3.11 : Diagram poteka delovanja programa na PPK

Operandi so:

I	Input (X)	(vhod)	[0..7]
Q	Output (Y)	(izhod)	[0..15]
M	Merker (M)	(vmesni pomnilnik)	[0..31]
T	Timer	časovnik	[0..7]
C	Counter	števec	[0..3]

Povratne informacije (izhodi Y4, Y5, Y6, Y7) o delovanju krmilnika, ki smo si jih pripravili, sedaj pošljemo nazaj na LPT1 vrata in jih vpišemo v MySQL podatkovno bazo (slika 3.12).



VH0=1	Error	Vhod
VH1=1	Select	Vhod
VH2=1	Pe	Vhod
VH3=1	Ack	Vhod

Slika 3.12 : Povratne informacije na LPT1 vrata

3.2.4 Vpis povratnih informacij iz PPK na LPT1 vrata in v MySQL bazo

Vrednosti izhodov (Y4, Y5, Y6, Y7), ki jih dobimo od krmilnika, moramo sedaj poslati nazaj preko LPT1 vrat in zapisati v MySQL bazo. To storimo tako, da v našem glavnem programu **server.c,** kjer imamo ukaz **case 2,** vstavimo kodo za branje z lpt1 vrat. Kot prvo, moramo v program vstaviti zakasnitev (štiri sekunde), da se program na krmilniku izvede do konca in pošlje na izhod vrednosti o delovanju. Ko smo to storili, lahko začnemo preverjati posamezne vhode oz. njihove naslove, kakšno vrednost imajo. To storimo z naslednjo kodo programa:

```
case 2 :
        printf("Povratne informacije iz krmilnika!\n");
        /*Zakasnitev da dobimo pravilne povratne inf. iz krmilnika*/
        sleep(4);
        value2=~(inb(LPT+1))&0x78;

        printf("LPT:%x, Output:%x, Input:%x\n",LPT,value,value2);

        if(value2&0x40){
        error=1;
        printf("ERROR:%d\n",error);}
        if(!(value2&0x40)) {
        error=0;
        printf("ERROR:%d\n",error);}

        if(value2&0x20){
        selec=1;
        printf("SELECT:%d\n",selec);}
        if(!(value2&0x20)) {
        selec=0;
        printf("SELECT:%d\n",selec);}

        if(value2&0x10){
        pe=1;
        printf("PE:%d\n",pe);}
        if(!(value2&0x10)) {
        pe=0;
        printf("PE:%d\n",pe);}

        if(value2&0x08){
        ack=1;
        printf("ACK:%d\n",ack);}
        if(!(value2&0x08)) {
        ack=0;
        printf("ACK:%d\n",ack);}
```

Na lpt1 vratih preverjamovradnosti štirih (4) bitov (d_0, d_1, d_2, d_3). Če imajo vrednost 1, določeno spremenljivko postavimo na vrednost 1, drugače pa ji zapišemo vrednost 0. Glede na določene spremenljivke, lahko spremljamo kaj se dogaja na krmilniku.

Vse štiri izhode, ki jih beremo z lpt1 vrat, moramo sedaj zapisati v MySQL bazo. Zapisati jih moramo na isto mesto, kjer so podatki, ki smo jih poslali na lpt1 vrata. Programsko kodo, ki nam to omogoči, spet dodamo programu **server.c** kjer je ukaz **case 2**. To storimo tako, da iz tabele **test,** kjer imamo shranjene podatke o vnosu, izberemo zadnji **id** (id je samo števec, ki nam oštevilči posamezne vnose)**.** Ta **id** je naš ključ, da lahko sedaj vpišemo povratne informacije v tabelo. Zadnje informacije iz krmilnika, bomo vedno zapisali na isto mesto, kjer je zadnji vnos podatkov v tabelo.

```
case 2 :

/*Iz tabele test potegnemo zadnji vnos(insert),da lahko vpišemo
  povratne informacije iz  krmilnika*/

                strcpy(q_str,"SELECT MAX(id) FROM test;");
                mysql_query(&mysql,q_str);
                rezultat = mysql_use_result(&mysql);
                while (row = mysql_fetch_row(rezultat)) {
                                last_id = atoi(row[0]);
                                }
                                /* printf("Error: %s", mysql_error(&mysql)); */
                                /* printf("last_id:%d\n", last_id); */
                                printf("Query:-->%s\nlast_id:%d\n", q_str,last_id);

                                /* Povratne inf. iz krmilnika vpišemo v tabelo test */
                                sprintf(q_str,"UPDATE test SET Error=%d, Selec=%d, Pe=%d, Ack=%d where id=%d;", error, selec,   pe, ack, last_id);
                                printf("Query:-->%s\n", q_str);
                                mysql_query(&mysql, q_str);
                                /* printf("Error: %s", mysql_error(&mysql)); */

                                /* Zapremo povezavo z MySql bazo */
                                mysql_close(&mysql);
```

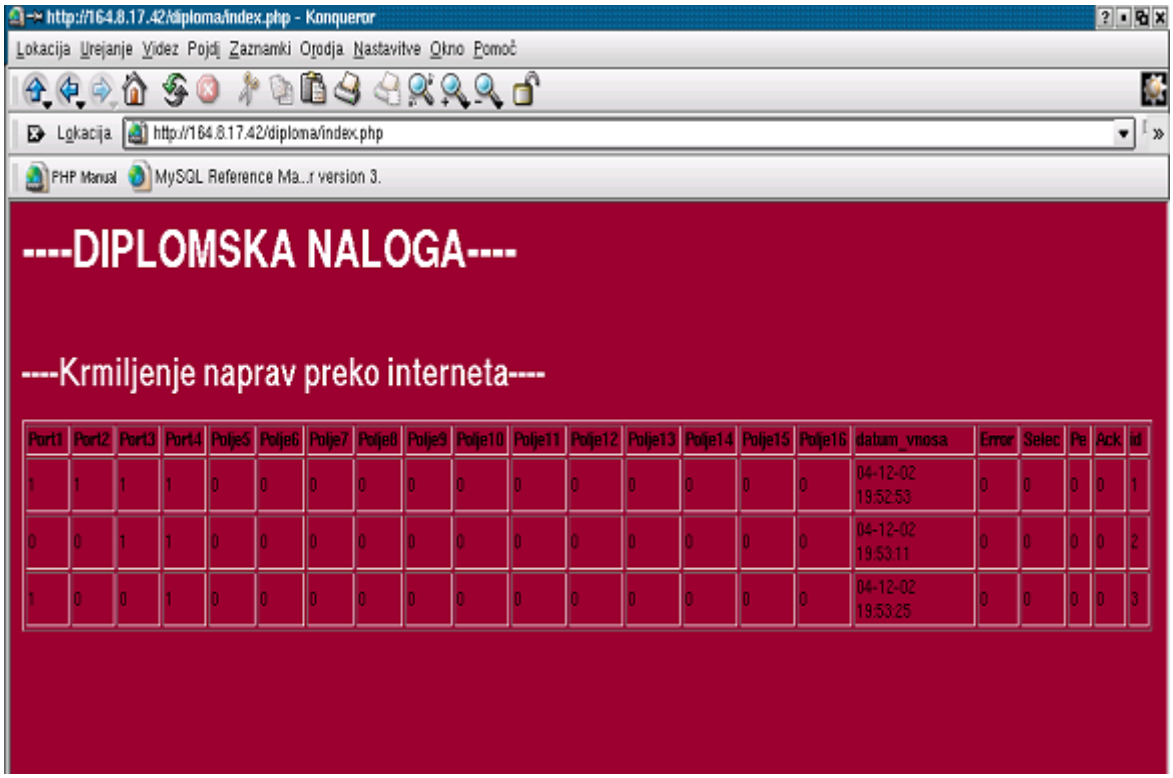
Podatke zapišemo v tabelo tako, da uporabimo MySQL ukaz »UPDATE«, ki nam omogoča, da dopolnimo poslane podatke z web-a s povratnimi informacijami o stanju iz krmilnika v tabeli »test«. Po zapisu povratnih informacij iz krmilnika v MySQL podatkovno bazo, smo končali celotni zapis v bazo. Preostane nam samo še to, da zapisane podatke prikažemo na spletni strani.

Celotno dogajanje, kaj se dogaja s programi, ko se pošljejo podatki iz web-a, lahko opazujemo na Linux konzoli (slika 3.13), kjer nam teče strežniški program:

Slika 3.13 : Prikaz komunikacije strežnik - odjemalec - MySQL - LPT1 - PPK

3.3 Prikaz podatkov na Web strani

Da lahko podatke, ki smo jih poslali na serverski računalnik in povratne podatke poslane iz krmilnika prikazujemo na web strani, smo uporabili Apache spletni strežnik ter PHP skriptni jezik, da smo lahko napisali dinamično spletno stran.



Slika 3.14 : Prikaz podatkov na Web strani

Da lahko prikazujemo podatke, ki smo jih na začetku vpisali na serverski računalnik in kasneje dopolnili s podatki, ki smo jih dobili kot povratno informacijo iz PPK Mitsubishi FX –a, imamo napisan program v PHP skriptnem jeziku. V bistvu je to isti program kot za vnos podatkov, le da imamo dodani **case** ukaz za prikaz rezultatov. Prikazujemo vseh šestnajst bitov ter še štiri bite, kot povratne informacije iz mikrokrmilnika. Dodatne bite imamo zato, da vidimo če se je program na krmilniku pravilno izvedel. Ker vidimo tukaj samo izgled spletne strani za prikaz podatkov, bomo opisali tudi program, s katerim izvedemo to spletno stran.

3.3.1 Program napisan v PHP skriptnem jeziku

Ker je PHP program sestavljen iz dveh delov, in sicer iz dela s katerim vnašamo podatke na glavni računalnik in v podatkovno bazo, ter dela, s katerim beremo podatke iz podatkovne baze in jih potem prikazujemo na spletni strani, bomo tukaj opisali samo del za prikazovanje podatkov. Program se imenuje **index.php** in je shranjen na Apache strežniku in sicer na direktoriju **/var/www/html/diploma.**

/* Tukaj uporabimo html kodo, za sam izgled spletne strani. Na začetku spletne strani se nam izpiše ----DIPLOMSKA NALOGA----, ter malo nižje ----Krmiljenje naprav preko interneta----. Lahko izbiramo velikost črk, ter barvo ozadja. */

```
-------------------------------------------------
<html>
<body bgcolor="#990033" text="#000000" >
<p><font face="Arial,Arial,Helvetica">
<b><font color="#FFFFFF" size="8">----DIPLOMSKA NALOGA----
</font></b></font></p><br></br>
<b><font color="#FFFFFF" size="6">----Krmiljenje naprav preko interneta--
--</font></b></font></p><br></br>
-------------------------------------------------
```

```
/* Sedaj pride na vrsto PHP koda. Uporabimo ukaz switch, saj lahko potem
uporabimo različne case ukaze. V tem delu bomo opisali ukaz case »Show«
za branje podatkov. */

-------------------------------------------------
<?php
      switch ($HTTP_POST_VARS['Action']) {

      /* S tem ukazom »case Show«, se povežemo z MySQL podatkovno bazo.
         Seveda moramo navesti uporabniško ime in geslo baze, da nam
         dovoli dostop. Potem moramo izbrati še pravo bazo(Baza:diploma).
         Ko imamo enkrat izbrano bazo, lahko začnemo brati podatke iz
         tabele test v kateri so shranjeni vsi podatki. Vse te podatke
         moramo nato izpisati in izrisati na web strani!  */

      case "Show":

                  /* Povezava na strežnik */
                  $link = mysql_connect("localhost","diploma","diploma");
                  /* Izberemo bazo */
                  mysql_select_db("diploma");

                  /* Preberemo vse podatke iz tabele 'test' */
                  $query = "SELECT * from test";
                  $result = mysql_query($query);

                  /* Izrišemo tabelo */
                  /* Imena stolpcev */
```

```
                $fields = mysql_list_fields("diploma", "test", $link);
                /* Število stolpcev */
                $num_cols = mysql_num_fields($fields);

                /* Tukaj začnemo polniti tabelo */
                echo "<table border=1>";
                echo "      <tr>"; /* Nova vrstica */
                for ($i=0; $i<$num_cols; $i++) {
                   echo "    <td> <b>" . mysql_field_name($fields,$i)
                   . " </b></td>";
                }
                echo "      </tr>"; /* Konec vrstice */



                /* Izpis podatkov - mysql_fetch_row($result) - Vrne
                FALSE ko pride do konca tabele - Za vsak klic da novo
                vrstico */
                while ($row = mysql_fetch_row($result)) {
                /* $row - so podatki v vrstici (vsi stolpci) */
                      echo "<tr>";
                      for ($i=0; $i<$num_cols; $i++) {
                            echo "<td>".$row[$i] . "</td>";
                                  /* $row[0] = prvi stolpec... */
                      }
                      echo "</tr>";
                }
                echo "</table>"; /* Konec tabele */
                /* Prekinemo povezavo */
                mysql_close($link);
                break;

          /* Seveda če hočemo pošiljati podatke, moramo napisati kako
          mora stran izgledati na prvem mestu pred prikazu podatkov! To
          storimo z ukazom default, ki pa je le en del switch ukaza.
          */

          default:
          /* Izgled strani pred pošiljanjem ... */
          "<form action=index.php method=post>\n" .
          "<input type=hidden name=Action value=Show>\n" .
          "         
                    
                    
                   
           <input type=Submit name=submit value=\"Prikaži podatke\"\n"
          "</form>\n";

                }
?> /* konec php kode */

</body>
</html>
```

S tem delom programa oz. PHP kodo, lahko sedaj prikazujemo podatke na spletni strani.

3.4 Zakjuček

V tem poglavju smo opisali vse programe, ki smo jih uporabili pri izdelavi diplomske naloge. Gre predvsem za opis komunikacije med web-om, MySQL bazo ter PPK Mitsubishi FX-om. Duša te komunikacije je program »server.c«, ki mora biti ves čas v pripravljenosti in čakati, če ga pokliče program »client.c« in mu prenese podatke. Duša komunikacije smo ga imenovali zato, ker se v ta program prenesejo podatki iz web-a in s tem tudi od odjemalca, ti podatki se potem vpišejo v bazo, jih pošljemo na lpt1 vrata, preberemo povratne informacije iz lpt1 vrat in jih vpišemo v bazo. Vse to se zgodi ob zagonu programa »server.c«, seveda če mu pošljemo podatke, ki jih mora obdelati. Vsi programi, ki smo jih uporabili, komunicirajo med sabo in operirajo z enakimi podatki, le da ima vsak program drugačno nalogo kaj narediti s podatki.

4. Sklep

V diplomski nalogi sem opisal izdelavo in delovanje povezave med avtomatiziranim tekočim trakom in podatkovno bazo. Prostoprogramljivi krmilnik tekočega traku sem povezal z osebnim računalnikom in njegovo stanje shranjeval v podatkovno bazo. Dostop do baze in tekočega traku pa sem izvedel preko spletnega brskljalnika. Vsi programi, ki sem jih napisal, imajo osnovno sestavo, kar pomeni da nimajo nobene zaščite pred zlorabami, ki se lahko zgodijo, če krmiliš nek sistem preko interneta. Ker sem bil edini uporabnik tega sistema, vsa ta zaščita ni bila potrebna. Če pa pogledamo malo širše, če bi sistem hoteli razširiti, bi morali dodati kar nekaj zaščite pred zlorabami. Spletna stran za vnos podatkov je tako dostopna vsem in če bi naš sistem dejansko bil priključen na internet, bi lahko kdorkoli spreminjal oz. pošiljal podatke. Zato bi morali na tej strani kot prvo dodati, preden bi se stran odprla okence za uporabniško ime in geslo. Tako bi do spletne strani lahko dostopale samo osebe, ki bi poznale geslo.

Ker sedaj prenašamo šestnajst bitov podatkov, uporabimo pa dejansko štiri za krmiljenje, bi lahko na serverski računalnik priključili tudi druge naprave za krmiljenje. Seveda bi morali spletno stran za vnos podatkov malce spremeniti, prav tako še nekaj drugih programov, vendar bi z relativno majhnimi popravki to kmalu dosegli. Ker smo uporabili MySQL podatkovno bazo, bi lahko razširili zapis podatkov v njo. Sedaj imamo samo eno tabelo v katero zapisujemo podatke. Tu bi lahko skreirali več različnih tabel in jih tudi med sabo povezali. Naprimer lahko bi imeli še tabelo uporabnikov, kamor bi se zapisovali vsi podatki, kdaj je kateri uporabnik operiral s sistemom in kaj je storil. Tukaj je odprtih zelo veliko možnosti, ki bi jih lahko dodali. Ker sem podatkovno bazo sedaj uporabljal sam, imam pravice za uporabo in skrbniški del samo jaz. Če bi bilo več uporabnikov, nam baza omogoča na lahek način da dodelujemo pravice drugim uporabnikom. Kot smo že omenili, bi lahko krmilili več naprav naenkrat. Prirediti bi morali samo program, ki nam pošilja podatke na LPT1 vrata. Cel sistem od vnosa podatkov na spletni strani pa do prihoda podatkov v krmilnik deluje zelo hitro, tako da tukaj nisem imel težav. Seveda temu pripomore pravilno izbrana podpora programju, da delujejo čim bolj optimalno. Zato sem se tudi odločil za že preverjeno povezavo Linux –

Apache strežnik – MySQL baza – PHP skriptni jezik. Kot sem že omenil, je Linux zelo zanesljiv in hiter, prav tako Apache spletni strežnik, MySQL bazo uporablja zelo veliko oseb in je dobro preizkušena, prav tako je zelo veliko spletnih strani napisanih v PHP skriptnem jeziku, tako da so te aplikacije dodobra preizkušene in nam s tem nudijo informacije o njihovi zanesljivosti. Vse te aplikacije se nenehno spreminjajo in izboljšujejo, saj imajo to prednost pred drugimi, da so »open source« in jih lahko spreminja in izboljšuje vsak, ki se s tem ukvarja in hkrati odpravlja stare napake. Najpomembnejše pri vsem tem pa je to, da smo za vse te aplikacije dobili izvorno kodo, ki je brezplača in nam nudi zaneslivo delovanje.

5. VIRI, LITERATURA

[1] http://standards.ieee.org/regauth/posix/

[2] http://www.gnu.org/copyleft/gpl.html

[3] http://www.lugos.si/linux/2.html

[4] http://www.klik.si/bazeinternetsem1.html

[5] http://www.php.net/usage.php

[6] http://www.lugos.si/arhiv/prispevki/sola-php-1.html

[7] http://www.ef.uni-lj.si/predmeti/bp/podip/seminarji/6.doc

[8] Mark Maslakowski, *Teach Yourself MySQL in 21 days*, SAMS, ZDA 2000

[9] http://www.ro.feri.uni-mb.si/predmeti/avt_pro_ob/Welcome.html

[10] http://www.senet.com.au/~cpeacock

[11] http://www.linux.org

[12] http://www.linux.slovenija.net

[13] http://www.apache.org

[14] http://www.mysql.com/doc

[15] http://www.php.net

[16] http://home.primus.com.au/rickard/clientserver/RedHatLinuxServer.html

6. PRILOGE

Priloga A: Razlaga Linux funkcij

Priloga B: Program client.c

Priloga C: Program server.c

Priloga D: Program za PPK

Priloga E: Naslov in življenjepis študenta

Priloga F: Izjava