Mestna občina Celje
Komisija Mladi za Celje

Računalniško voden digitalen merilec porabe električne energije

Raziskovalna naloga

AVTORJI
Alen Verk
Aleš Papič
Blaž Nunčič

MENTOR
mag. Boštjan Resinovič, univ. dipl. inž. rač. inf.

Mestna občina Celje, Mladi za Celje
Celje, 2012/2013

ŠOLSKI CENTER CELJE
Srednja šola za kemijo, elektrotehniko in računalništvo
Pot na Lavo 22
3000 Celje

Računalniško voden digitalen merilec porabe električne energije

Raziskovalna naloga

Avtorji:
Alen Verk, E-4.e
Aleš Papič, E-4.e
Blaž Nunčič, E-4.e

Mentor:
mag. Boštjan Resinovič, univ. dipl. inž. rač. inf.

Mestna občina Celje, Mladi za Celje
Celje, 2012/2013

IZJAVA*

Mentor(-ica), mag. Boštjan Resinovič v skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z

naslovom Računalniško voden digitalen merilec porabe električne energije,

katere avtorji(-ice) so Alen Verk, Aleš Papič, Blaž Nunčič:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo(-ičino) dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na spletnih portalih z navedbo, da je nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 13.3.2013

Žig šole

Šola

Šolski center Celje, Srednja šola za kemijo, elektrotehniko in računalništvo

Podpis mentorja(-ice)

Podpis odgovorne osebe

* Pojasnilo

V skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno **podpisano izjavo mentorja(-ice) in odgovorne osebe šole vezati v izvod za knjižnico**, dovoljenje za objavo avtorja(-ice) fotografskega gradiva, katerega ni avtor(-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

Kazalo vsebine

Kazalo tabel

Kazalo grafikonov

Kazalo slik

Povzetek

V raziskovalni nalogi smo si zadali problem. Želeli smo nadomestiti ročno beleženje porabe električne energije z avtomatskim. Pripraviti smo morali merilec, ki bo beležil porabo, mesto, na katero bomo shranjevali meritve in program s katerim bo uporabnik pregledoval porabo. Raziskava je zajela področji elektrotehnike in računalništva.

Za pripravo raziskovalne naloge smo uporabili anketo z mnenji potencialnih anketirancev. Pred izvedbo ankete smo pripravili analizo problema in zbiranje gradiva. Lotili smo se izdelave merilnika, pripravili strežnik, sprogramirali delovanje merilnika in aplikacij, za konec pa smo izdelek tudi testirali.

V raziskovalni nalogi so predstavljeni rezultati ankete, ki so ponazorjeni tudi z grafi. V nadaljevanju je podrobno predstavljen pripravljen celoten sistem, ki je podkrepljen s teorijo za lažje razumevanje. Za konec smo predstavili še veljavnosti hipotez in predstavili težave, s katerimi smo se srečevali. Dodali smo tudi ideje za nadaljnji razvoj celotnega sistema.

Abstract

With this research work we wanted to replace manual recording of electricity consumption with an automated one. We had to create a meter to record the consumption, find a place to store the recordings and an application enabling a user to check the consumption. With this we ventured into electro technical and computer areas.

We created a short survey to get of our potential users' opinion, but before that we prepared an analysis of our problem and we searched for the existing material. We created the meter, prepared the server, programmed the meter and the applications. In the end we also tested our product.

In the presented research paper we presented the results of the survey, and illustrated them with charts. Following that we described the whole system, combined with the theory for better understanding. In the conclusion we validated our hypotheses and described the problems, we encountered. We also added ideas for future system upgrades.

Ključne besede

- e-Blocks
- ASP.NET
- programiranje
- Flowcode
- Objektno Orientirano Programiranje (OOP)
- uporabniški vmesnik
- merilnik porabe elektrike

Keywords

- e-Blocks
- ASP.NET
- programming
- Flowcode
- Object Oriented Programming (OOP)
- user interface
- electricity consumption meter

Kratice in okrajšave

e-blocks - elektronska prototipna vezja,
PIC - Peripheral Interface Controller,
D-SUB - D-sub miniature (priključek),
USB - Universal Serial Bus,
LCD - Liquid crystal display,
RAM - Random-access memory,
EEPROM - Electrically Erasable Programmable Read-Only Memory,
TCP/IP - Transmission Control Protocol/Internet Protocol,
SD - Secure Digital,
GND - negativno nabit priključek,
Mb - Megabit,
IEEE - Institute of Electrical and Electronics Engineers ,
IPv4 - Internet Protocol version 4,
IPv6 - Internet Protocol version 6,
UDP - User Datagram Protocol,
ARP - Address Resolution Protocol,
DLC - Data Link Control,
MAC - Media Access Control,
I2C - serijski prenos,
SDA - prenos podatkov,
SCK - prenos ure ,
INT - prekinitve,
RJ45 - priključek,
V/I - vhodno/izhodne,
ASCII - American Standard Code for Information Interchange,
IDE - integrated development enviroment,
ARM - Advanced Risc Machine,
AVR - vrsta mikrokrmilnika,
DNS - Domain Name System,
IIS7 - Internet Information Services,
MSSQL - Microsoft SQL server,
SUPB - sistem za upravljanje s podatkovnimi bazami,
ER-Diagram - Entity–relationship diagrams,
AES - Advanced Encryption Standard,
HTTPS - Hypertext Transfer Protocol Secure,
SSL - Secure Sockets Layer,
VB - Visual basic,
CSS - Cascading Style Sheets,
HTML – Hypertext Mark-up Language
OOP – Object orientated programming

1 Uvod

Od dneva, ko smo prvič sedli v srednješolske klopi in do danes, ko smo tik pred zaključkom srednje šole, smo se naučili ogromno novih stvari. Ves čas šolanja smo s profesorji spoznavali različna področja v računalništvu, iz katerih smo črpali smernice za naprej. Vendar za nas to ni bilo povsem dovolj. Poleg začrtanih smernic izobraževalnega programa smo iskali tudi področja, ki se ne navezujejo samo na naše področje kot računalniških tehnikov. Segali smo po elektrotehniki, kajti narediti fizičen izdelek, ki ga je mogoče upravljati s pomočjo računalnika, je bilo za nas neprecenljivo.

Prav zaradi tega smo želeli to dvoje medsebojno povezati; na eni strani vse tisto, kar smo se naučili v šoli, in na drugi strani tisto, kar smo raziskovali ter ustvarjali sami. Zadali smo si cilj, da povežemo znanje iz računalništva z znanjem iz elektrotehnike in tako naredimo uporaben izdelek, ki bi ga bilo mogoče v današnjem času tudi tržiti.

1.1 Predstavitev problema

Naša naloga je bila izdelati sistem, ki bo beležil porabo električne energije in jo prikazoval ob uporabnikovi zahtevi na zaslonu računalnika, tablice, mobilne ali druge naprave, ki se bo lahko povezala na splet. Sistem mora zagotoviti uporabniku rezultate meritev s čim manjšo napako, želimo pa tudi, da bi hkrati lahko beležili meritve za več uporabnikov.

Za izdelavo tega sistema smo potrebovali merilnik in strežnik za shranjevanje meritev. Merilnik hkrati ni smel biti odvisen od strežnika, kar pomeni, da mora imeti mesto za shranjevanje meritev v primeru, da je strežnik neaktiven.

Pripravili smo prototip celotnega sistema in ponazorili njegovo delovanje, ki je bil izdelan ob pomoči potencialnih uporabnikov sistema ter njihovih želja.

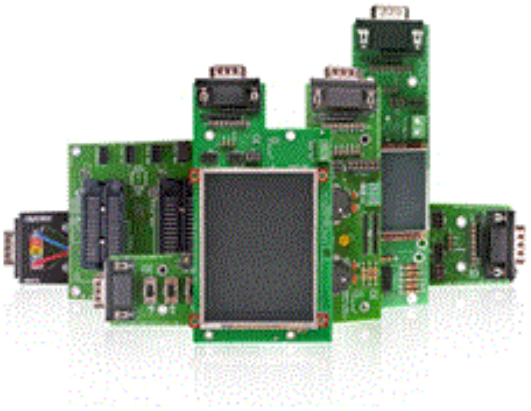
1.2 Hipoteze

- Vsaj 75% anketiranih spremlja porabo električne energije,
- anketirani v večini beležijo porabo,
- merilnik bo vračal rezultate z do +|- 3% napako,
- stroški programske opreme strežnika bodo minimalni,
- uporabnik si bo lahko sam nastavljal izris grafa porabe.

1.3 Raziskovalne metode

Naloga, ki smo si jo zadali, se nam je na samem začetku zdela ozko področna, zahtevna, a vendar zelo uporabna, zaradi česa smo bili toliko bolj zagnani pri iskanju gradiva in idejah za zasnovo sistema. Ko smo našli gradivo, ki bi nam pomagalo, smo pripravili analizo, kjer smo združili in uskladili naše ideje ter predloge.

Kmalu se je začela faza priprave strežnika in merilnika, ki sta ključen del sistema. Za merilnik smo si izbrali elektronska vezja e-blocks in Windows Server 2008. Pripomočke za merilnik nam je priskrbela šola.



Slika 1: e-Blocks, elektronska prototipna vezja

Za tem smo začeli s programiranjem. Pripraviti je bilo potrebno več stvari. Najprej smo potrebovali testno različico merilnika in aplikacijo za sprejemanje meritev na strežniku. Odkrili smo nekaj pomanjkljivosti (spreminjanje nastavitev na merilniku, težave pri povezavi na strežnik, odstopanje med prejetimi podatki in podatki na merilniku). Napake smo odpravili in začeli z novim testiranjem.

Ker so bili ponovni testi uspešni, smo začeli s pripravo podatkovne baze in kasneje tudi s pripravo uporabniškega vmesnika, ki pa nam nista povzročala težav. Za konec smo pripravili še krajšo anketo, s katero smo dopolnili podrobnosti v sistemu in pridobili mnenja potencialnih uporabnikov ter testirali izdelek.

1.4 Pripomočki

1.4.1 Strojna oprema

Pri delu smo uporabili naslednjo strojno opremo:

- e-Blocks (grafični LCD, SD komponenta, TCP/IP komponenta, matrična tipkovnica, tipke)
- čip PIC 18F4550
- tokovni senzor Allegro ACS714 30A

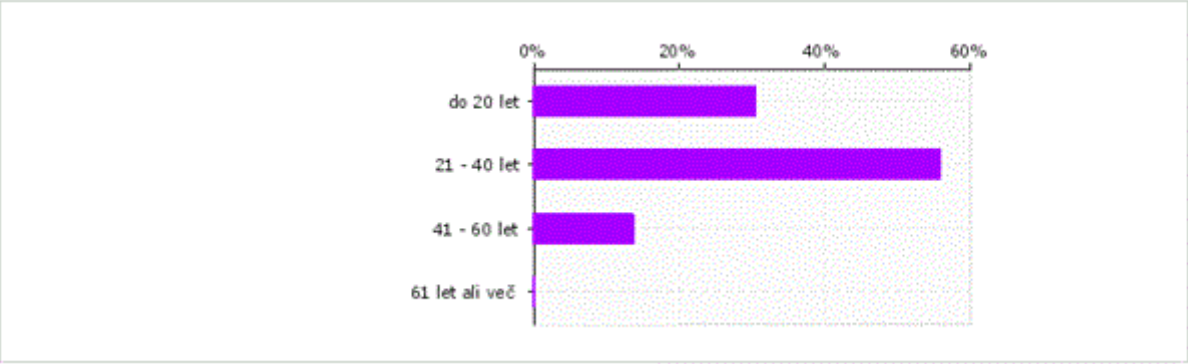
1.4.2 Programska oprema

Glede na kompleksnost raziskovalnega problema ni presenetljivo, da smo pri programiranju morali uporabiti kar nekaj platform in orodij. Ta so bila:

- Flowcode IDE
- Microsoft Windows Server 2008 R2
- MSSQL Server 2008
- Microsoft Visual Studio 2010 & 2012
- Notepad++

2 Rezultati ankete

K sodelovanju v naši anketi smo povabili naključno izbrano populacijo v Sloveniji. S pomočjo oglaševanja naše ankete po različnih forumih in omrežjih, tudi Facebook-u, smo prejeli lep odziv. V anketi je sodelovalo 61 oseb do 61 leta starosti, zaradi česa lahko trdimo, da je anketa realna, saj smo z njo zajeli kar 70% ciljne skupine anketirancev. Ostalih 30% anketiranih so osebe, ki so še odvisne od staršev in še ne poslujejo z denarjem v tej meri, da bi sami plačevali položnice. Za nas niso bili ciljna skupina anketirancev.

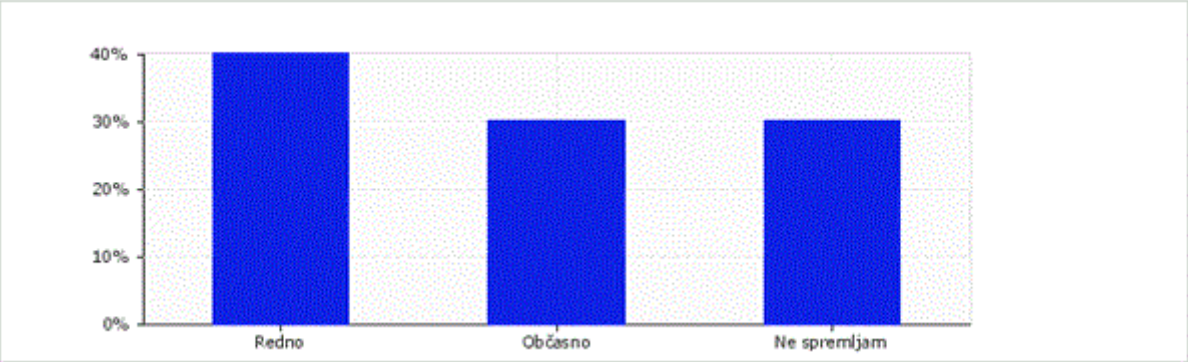


Graf 1: Starost anketiranih, 70% starejših od 21 let

Anketo smo razdelili v dva sklopa. Prvi sklop je bil splošen. V njem nas je zanimalo ali anketirani nadzorujejo porabo električne energije in ali te vrednosti tudi zabeležijo ter analizirajo. V drugem sklopu pa smo se posvetili tehničnim zahtevam sistema, ki jih sami nismo mogli uskladiti in smo se odločili, da nam pri tem pomagajo tudi drugi.

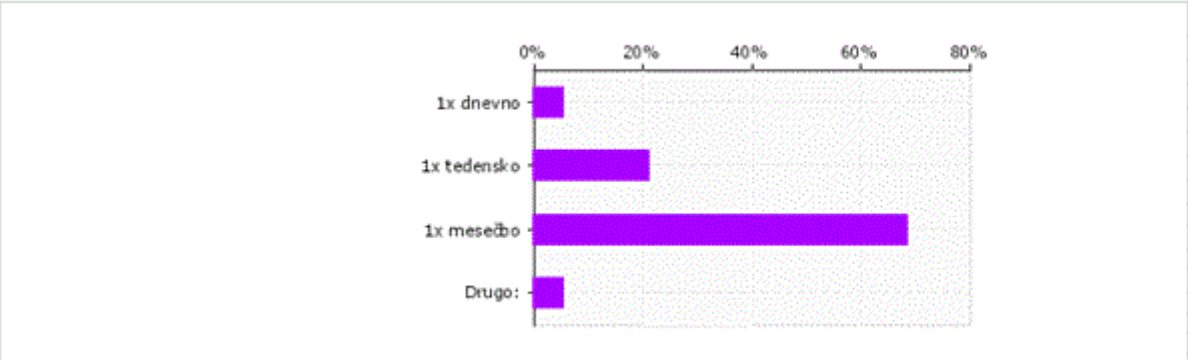
2.1 Splošen sklop vprašanj

V tem delu ankete so anketiranci odgovarjali na vprašanja, ki so povezana z njihovim nadzorom porabe. Odgovori so blizu našim pričakovanjem, saj velika večina anketiranih spremlja porabo, kar je razvidno iz spodnjega grafa (Graf 2).



Graf 2: Spremljanje porabe električne energije

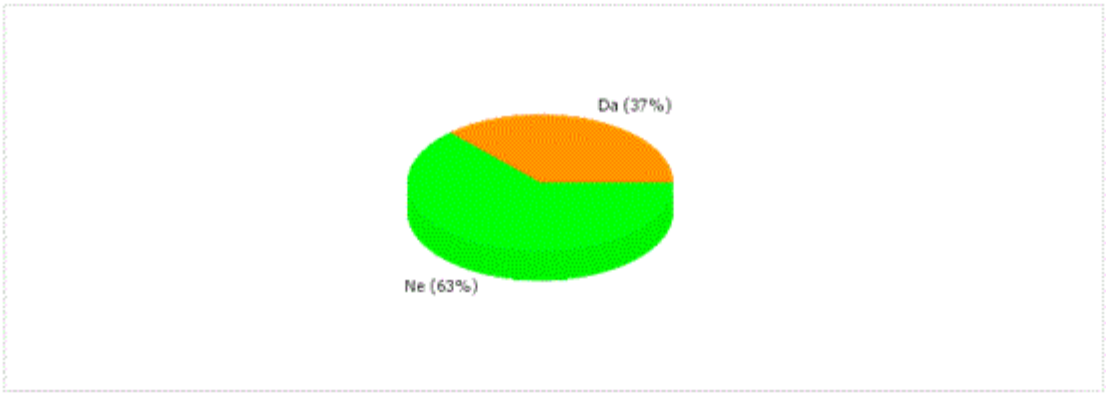
Ugotovili smo, da med vsemi, ki spremljajo porabo, več kot polovica anketiranih ne beleži porabe. Te ugotovitve nas sicer nekoliko presenečajo, saj smo menili drugače. Bilo bi zanimivo izvedeti, kakšni so podrobni razlogi, česar pa nismo spraševali v anketi. Največ vprašanih, ki beležijo porabo, odgovarja, da to počnejo 1x mesečno, veliko manjši odstotek to počne 1x tedensko ali še pogosteje (Graf 3).



Graf 3: Graf prikazuje, kako pogosto vprašani beležijo porabo

Od anketirancev smo želeli izvedeti tudi ali zabeležene meritve analizirajo in kako. Pri tem vprašanju so bila naša pričakovanja nižja, saj vemo, da analiziranje zahteva veliko časa, tempo današnjega načina življenja pa tega ne dopušča. Rezultati so pokazali, da si še nekateri (37%) zmeraj vzamejo čas

in pregledajo podatke (Graf 4). Na vprašanje kako analizirajo porabo pa so vsi vprašani odgovorili, da to počnejo s pomočjo tabel, kjer primerjajo podatke.



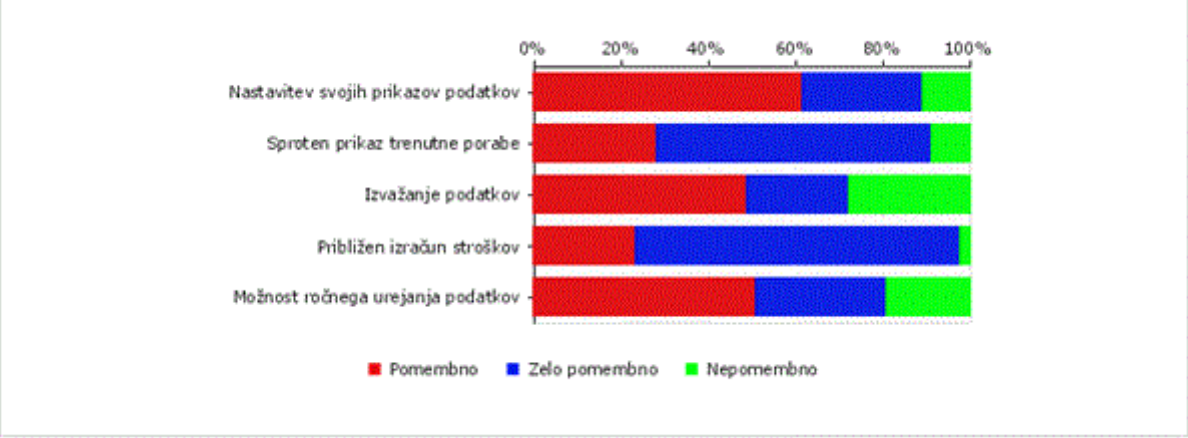
Graf 4: Graf prikazuje, da 37% vprašanih analizira zabeležene meritve

Za konec prvega sklopa vprašalnika nas je še zanimalo ali jim beleženje predstavlja veliko breme ter ali bi bili pripravljeni uporabljati sistem, ki bi to počel za njih. Odgovorili so v 60%, da jim beleženje ne predstavlja velikega bremena, vendar je slabih 90% vseh vprašanih izrazilo željo, da bi uporabljali sistem, ki bi to počel za njih, kar nam daje veliko potrditev, da smo šli z raziskovalno nalogo v pravo smer.

2.2 Tehnični sklop vprašanj

S tem sklopom vprašanj smo želeli izvedeti, kako bi si uporabniki zamislili sistem, ki najbolje ustrezal njim. Ob pomoči teh vprašanj nam je uspelo urediti še manjše podrobnosti.

Najprej smo želeli izvedeti, kako pomembne so določene funkcije za njih. Za vse naštete funkcije, ki smo jih v anketi opredelili, lahko trdimo, da so pomembne za naše uporabnike, saj so vse prejele visoke ocene (Graf 5).



Graf 5: Pomembnost funkcij za uporabnike

Zelo veliko vlogo pri pripravi strežnika sta imeli vprašanji kako goste naj bodo meritve in kako dolgo naj shranjujemo meritve na strežniku. Ti vprašanji sta pomembni zaradi količine podatkov, saj je za gostejše meritve in daljše shranjevanje potrebno zagotoviti mnogo več prostora na strežniku kot sicer. Anketirani so odgovorili, da bi zadostovalo, če bi se meritve shranjevale 1x dnevno, si pa želijo, da bi imeli pregled porabe za nazaj v povprečju 20 mesecev.

Upoštevali smo želje anketiranih in omogočili pregled porabe za 18 mesecev nazaj, gostoto meritev pa vseeno nekoliko zgostili in sicer na vsako uro. Za to smo se odločili, da uporabniku omogočimo pregled tudi tekoče dnevne porabe kar na spletni strani.

Za konec smo anketiranim ponudili prosto pot, da nam podajo še svoje predloge. Prejeli smo pozitivni mnenji o naši raziskovalni nalogi, kar nam je dalo še dodaten zagon pri zaključevanju. Podan je bil tudi predlog, da bi bil sistem cenovno ugoden in dostopen vsem.

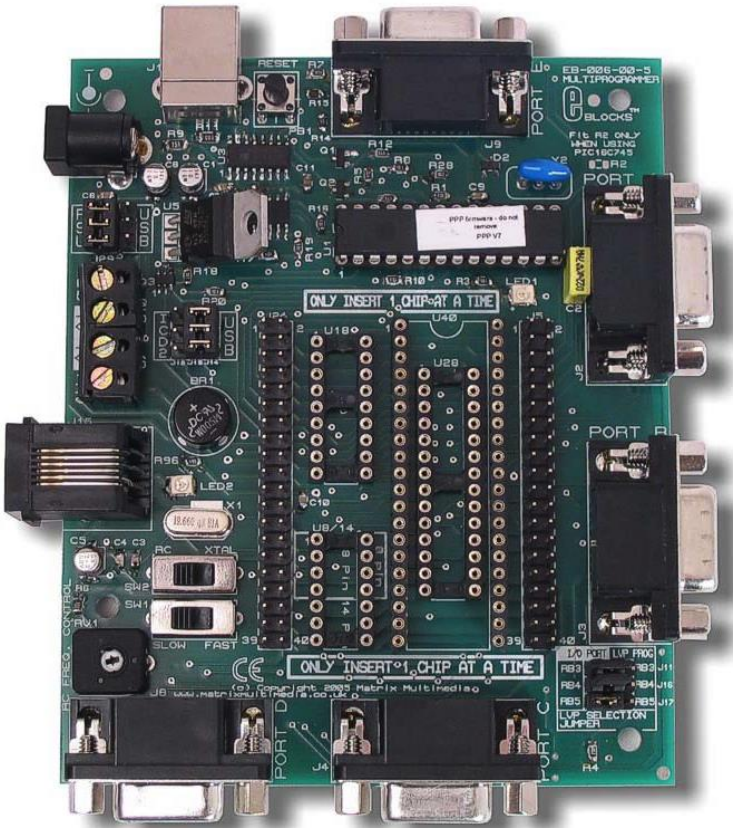
3 Predstavitev sistema

Sistem je sestavljen iz dveh področij. Prvo se navezuje na elektrotehniko, saj vključuje razvoj merilnika in zanj potrebnih senzorjev za izvajanje meritev. Drugo področje je računalništvo, ki smo ga razdelili na dva manjša dela. Ta zajemata pripravo strežnika, ki je potreben za shranjevanje meritev, in pripravo uporabniškega vmesnika, ki bo omogočal uporabniku spremljanje porabe.

3.1 Merilnik

3.1.1 Razvojna plošča EB-006

Razvojno ploščo oz. Multiprogrammer board EB-006 smo uporabili kot osnovno vezje, na katero smo priklopili periferne naprave, ki so ji dale željno funkcionalnost. Podrobni opisi komponent so v sledečih poglavjih. Na plošči je že integriran programator za mikrokontrolerje družine PIC, zato ga ni bilo potrebno izdelovati posebej. Prav tako plošča omogoča izvajanje programa kar na njej. Nanjo lahko priklopimo do 5 komponent preko D-SUB priključkov, lahko pa jih tudi več, če uporabimo »Patch system«. To pomeni, da povežemo komponente s posameznimi kabli iz komponent. Ploščo priklopimo preko USB vrat na računalnik, prav tako pa lahko izbiramo med napajanjem iz USB ali pa med adapterjem. Napetost je regulirana preko L7805CV regulatorja napetosti, vendar se ta že v originalu precej greje, posebej, če je naj priklopljen grafični LCD, zato bi mi dodali hladilno telo.



Slika 2: Razvojna plošča EB-006

3.1.2 Microchip PIC 18F4550

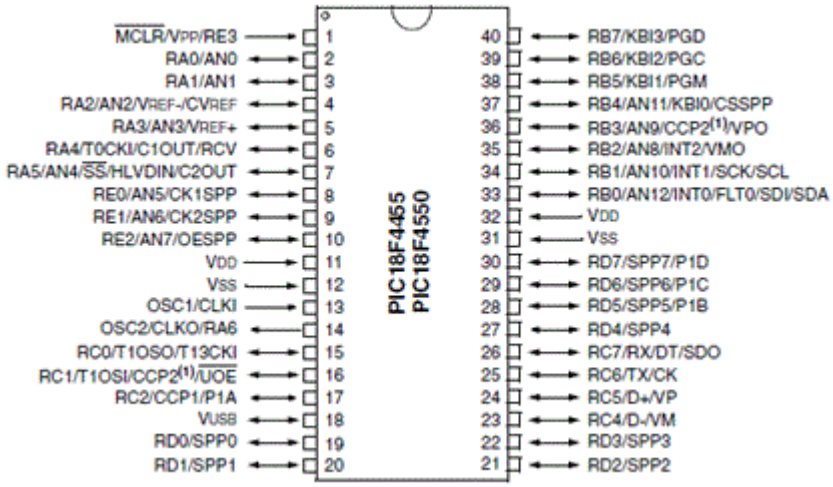
Naš ciljni mikrokrmilnik je PIC 18F4550, ki je ključni člen našega merilca. Preko njega gredo vse komponente in vsa logika. Izbrali smo ga zaradi dovolj velikega pomnilnika RAM, saj je komponenta SD zelo potratna, zato smo bili primorani izbrati 18Fxxxx serijo mikrokrmilnikov.

Mikrokrmilnikove karakteristike so sledeče:

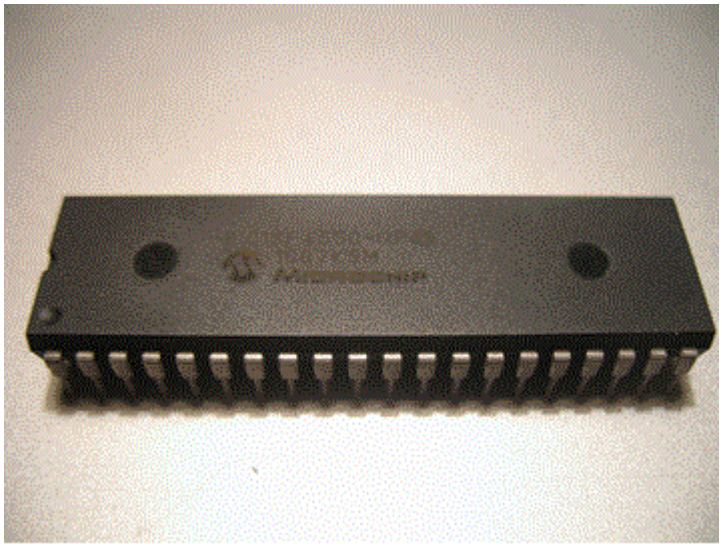
- napajanje od 2,0 V do 5,0 V.
- 32 Kb Flash pomnilnika,
- Kb RAM,
- 256 bajtov EEPROM,
- do 13 10 bitnih A/D pretvornikov,
- možnost serijske in I2C komunikacije,
- časovniki (Timer0-Timer3),
- zunanji oscilator do 48 MHz,
- notranji oscilator od 31 kHz do 8 MHz,
- USB vmesnik 2.0 za neposredno komunikacijo preko vrat USB.

Pregled nožic:

- MCLR – vklop/izklop krmilnika
- VDD – napajanje
- VSS – masa
- OSC1 in OSC2 – priklop kristala preko 22pF kondenzatorjev
- AN0 – AN12 : 10 bitni A/D pretvorniki
- RA0-RA5, RB0-RB7, RC0-RC7/{RC3}, RD0-RD7, RE0-RE3: V/I nožice



Slika 3: Mikroprocesor PIC 18F4550



Slika 4: PDIP model 18F4550

3.1.3 Periferne komponente

Za lažje razvijanje naprave smo uporabili komponente E-blocks, ki so delo podjetja Matrix Multimedia, in pa tudi komponente, ki smo jih naredili sami. Z že narejenimi smo si poenostavili delo, saj so vsi že na svojem tiskanem vezju in se priključijo na razvojno ploščo preko D-SUB priključka, ki ima 9 pinov, ki so uporabljeni kot 8 podatkovnih pinov in maso, odvisno od ciljnega mikrokrmilnika.
Seznam komponent:

- Grafični LCD (E-blocks)
- SD komponenta (E-blocks)
- TCP/IP komponenta (E-blocks)
- Matrična tipkovnica (E-blocks)
- Tipka – vhod v skrbniški meni (samogradnja)
- Senzor električnega toka (samogradnja)

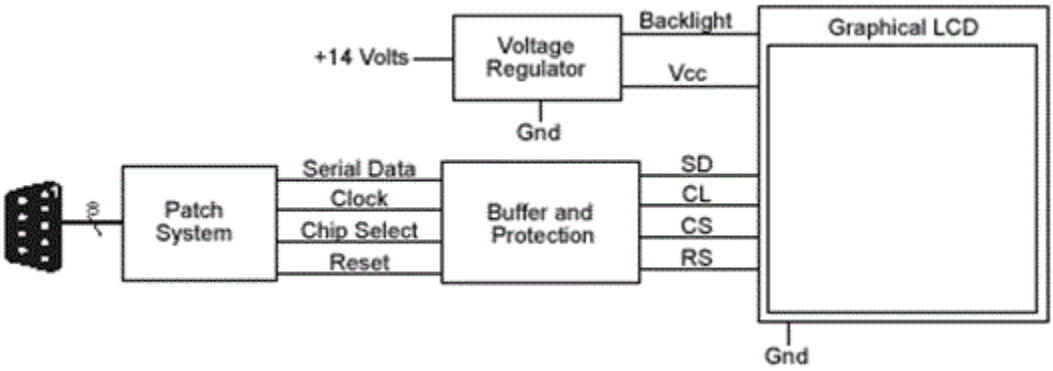
3.1.3.1 Grafični LCD

Uporabili smo grafični LCD (v besedilu gLCD), saj nam navadni LCD ni zadostoval za izpis celotnega administratorskega menija, prav tako pa smo pridobili na preglednosti izvajanja programa, saj je izpisan celoten meni naenkrat in se uporabniku ni potrebno s tipkami pomikati za prikaz točno določene funkcije. Vsekakor pa to ni edini razlog za našo izbiro, Razlog je tudi možnost nadgradnje programske opreme, ki bo prikazovala graf trenutne porabe, a trenutno je to le smernica za nadaljnje razvijanje merilnika.

gLCD ima naslednje karakteristike:

- potrebuje napajanje 14V,
- 130x130 posamezno nastavljivih slikovnih pik,
- zmožen je prikazati do 65535 barv,
- do 16 vrstic besedila, po 21 znakov v posamezni vrstici.

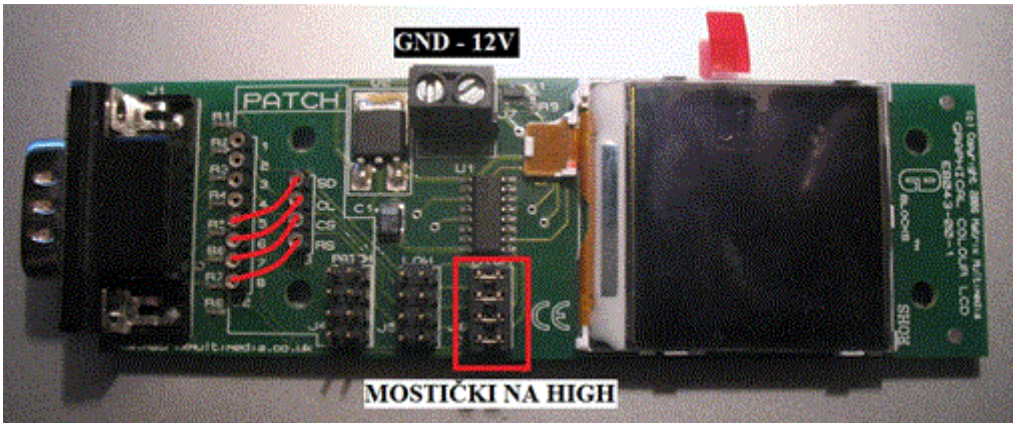
Naslednja shema prikazuje priklop gLCD-ja, ki smo ga priključili na port B. Vsebuje D-SUB priključek, uporabljene pa so samo 4 podatkovne linije, ki jih lahko nastavimo po lastnih potrebah. Mi smo nastavili mostičke na opcijo HIGH (**Napaka! Vira sklicevanja ni bilo mogoče najti.**), kar pomeni da so uporabljene podatkovne linije od bita 4 do bita 7. Tako so biti od 0 do 3 ostali še neuporabljeni.



Slika 5: Shema priklopa GLCD-ja

	Mostički LOW	Mostički HIGH	Ročna nastavitev
SD	Bit 0	Bit 4	Ročno
CL	Bit 1	Bit 5	Ročno
CS	Bit 2	Bit 6	Ročno
RS	Bit 3	Bit 7	Ročno

Tabela 1: Možnost vezave GLCD-ja

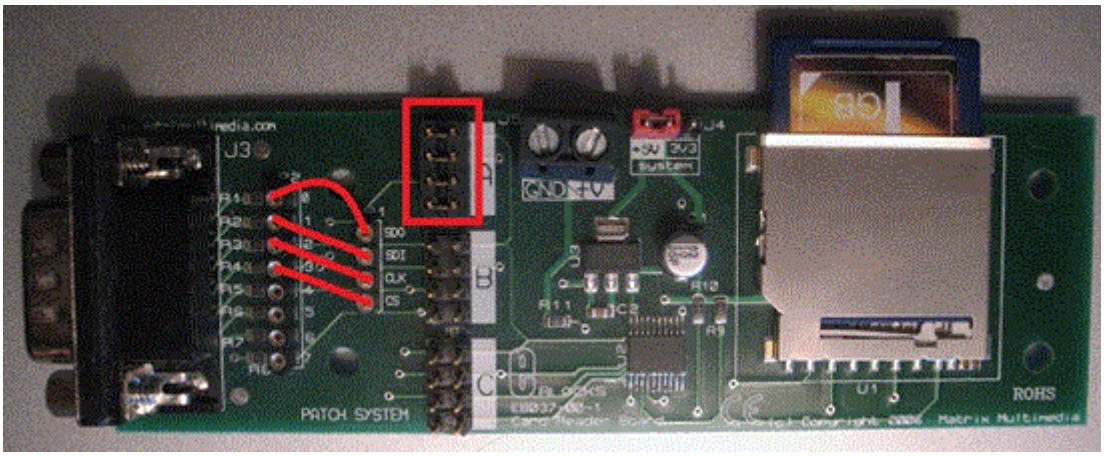


Slika 6: GLCD prikaz mostičkov, napajanja in povezav

3.1.3.2 SD komponenta

Preko le- te shranjujemo meritve na SD kartico v primeru, da strežnik ni dosegljiv. SD kartica mora biti vedno vstavljena, da program sploh deluje, saj le tako zagotovimo karseda zanesljivo delovanje.

Priključena je na port A preko »Low« pinov; tj. izbran sistem A, ki označuje, da je povezana preko pinov RA0-RA3. Tako nam ostaneta prosta še dva pina RA4 in RA5, ki ju trenutno ne potrebujemo. Napajanje pripeljemo preko priključkov za 5V in GND, lahko pa nastavimo mostiček, ki nastavi delovanje kartice v sistemu 5,0 V ali pa 3,3 V.



Slika 7: SD komponenta

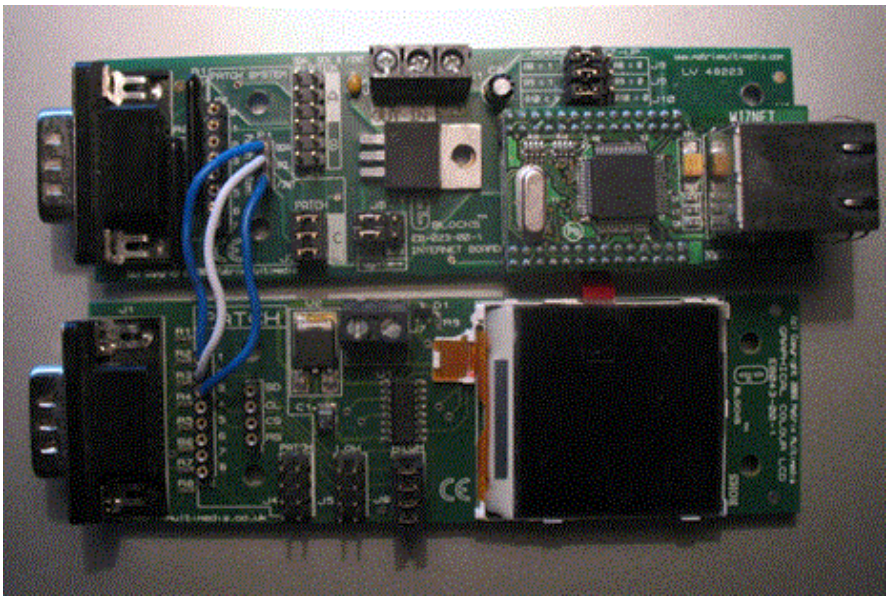
3.1.3.3 TCP/IP komponenta

TCP/IP komponenta skrbi za komunikacijo med merilcem in strežnikom. Kot je vidno s fotografije, je povezan preko kablov iz gLCD-ja. Tako sta obe komponenti priključeni preko porta B. Ker smo povezave nastavili sami, smo spremenili mostičke na »patch« oz. C možnost, ki je namenjena, da se uporabi, kadar uporabnik sam poveže komponento in ne preko D-SUB priključka. Napajanje dobi preko +5 in GND priključka, nastaviti pa je bilo mogoče 5V ali 3,3V obratovalne napetosti.

Karakteristike TCP/IP komponente:

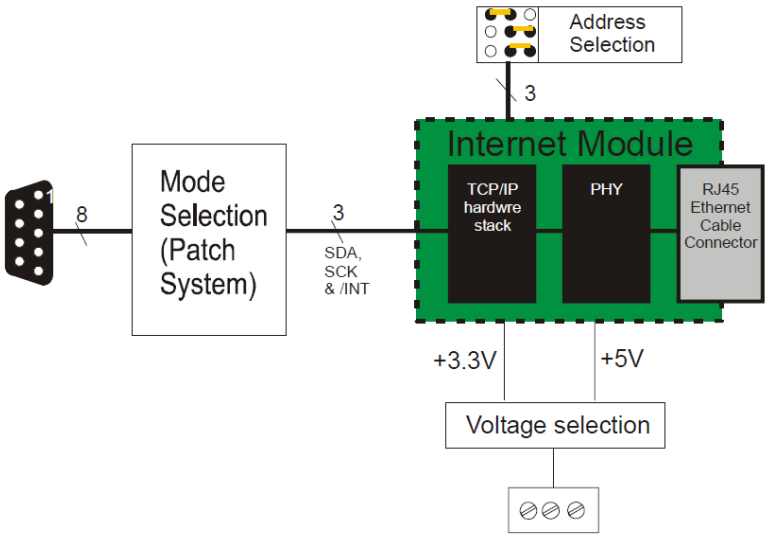
- podpira do 100 Mb prenosa,
- IEEE 802.3/802.3u združljiva,
- napajanje 3,3 V ali 5 V,
- podpira TCP, IP v4, UDP, ARP internetne protokole na fizičnem nivoju,
- podpira DLC in MAC eternetne protokole na fizičnem nivoju,
- z mikrokrmilnikom komunicira preko I2C vodila.

TCP/IP komponenta uporablja na fizičnem nivoju Wiznet Internet Module NM7010A mikroprocesor.



Slika 8: TCP/IP komponenta

Sledeča shema prikazuje vezavo komponente oz. njeno delovanje. Priključimo jo lahko preko D-SUB priključka na EB-006 ali pa preko sistema Patch. Za delovanje komponente so potrebene 3 podatkovne linije SDA-prenos podatkov, SCK-prenos ure in pa INT-prekinitve. Nato lahko izberemo naslovne mostičke, ki smo jih nastavili na 1. Prav tako moramo izbrati pravilno napetost. Na koncu pa je še mikroprocesor za TCP komponento in pa RJ45 priključek za povezavo v Ethternet ali internet omrežje.

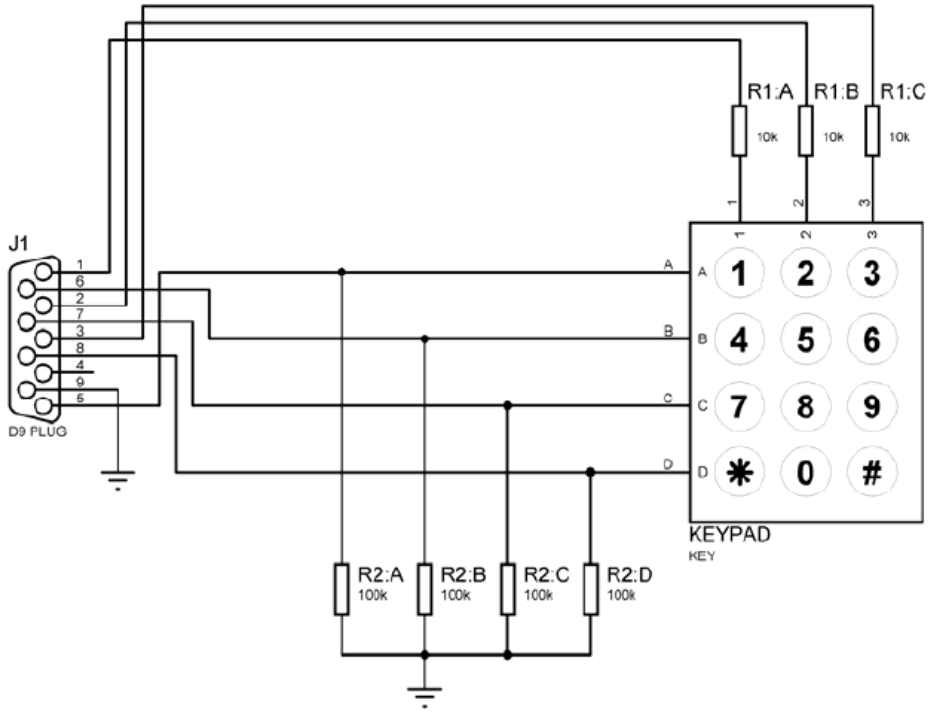


Slika 9: Blok shema TCP/IP komponente

3.1.3.4 Matrična tipkovnica

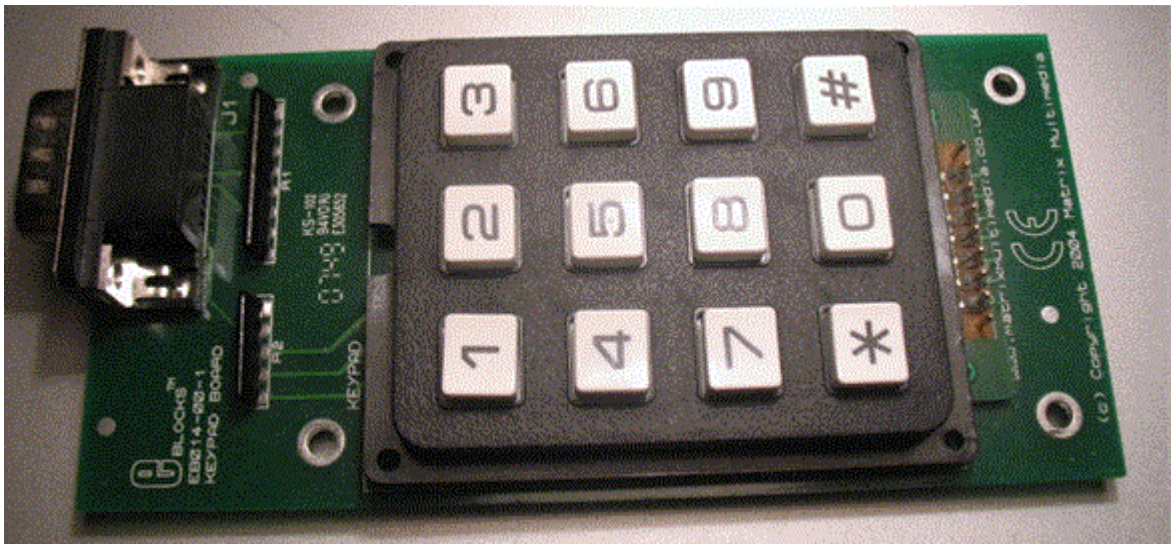
Je medij za vnos podatkov v merilec. Preko tipkovnice administrator vpiše vse nastavitve, ki so potrebne, da merilec pravilno deluje. Prav tako pa smo jo uporabili za možnost nadaljnega razvijanja, ker bo lahko uporabnik nastavljal željeni izpis na zaslonu, kjer se mu bodo izrisovali grafi.

Na spodnji shemi je prikazano delovanje tipkovnice. Preko štirih bitov so kotrolirane vrstice, preko ostalih 3 pa stolpci. Ko pritisnemo tipko, se povezavi med nožicama skleneta, tako lahko mikrokrmilnik zazna spremembo na V/I nožicah in si jo interpretira glede na stanje bita. Če je bit visok oz. so nožice sklenjene, je bila tipka stisnjena in nato temu primerno odreagira, kar je pa odvisno od programa, ki je na mikrokontrolerju.



Slika 10: Shema delovanja matrične tipkovnice

Je zelo priročna komponenta, saj lahko poleg branja številk preberemo tudi ASCII vrednost za točno določeno pritisnjeno tipko. V fazi razvijanja omogoča programerju, da nastavi poljubne vrednosti za specifično tipko. Tako lahko ob pritisku tipke '1' preberemo številčno vrednost 1, ASCII vrednost 49 ali pa programer določi povsem tretjo vrednost npr. 'F'. Če bi zamenjali vrednost tipke, je potrebno fizično tipko prekriti z nalepko, ki označuje novo vrednost tipke.



Slika 11: Matrična tipkovnica

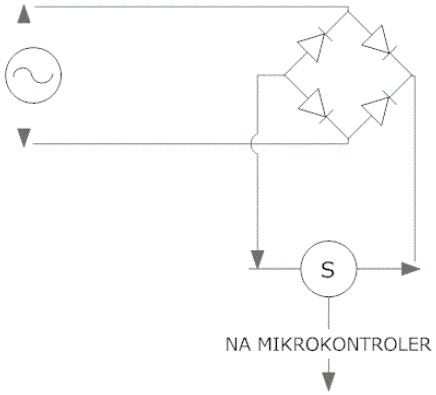
3.1.3.5 Tipka

Za tipko smo uporabili kar klasično tipko, ki skrbi za prehod v administratorski meni. Priključena je preko upora 1K ohm na +5V in na mikrokontroler. Tako se spreminja bit na čipu na 1 ali 0.

3.1.3.6 Senzor električnega toka

Osrednji problem naše aplikacije je vsekakor senzor električnega toka, saj je le-ta potreben za naše meritve porabe električne energije. Za naš projekt bi potrebovali še merilec električne napetosti, a bomo kar predpostavili, da je napetost 230V. Tako bomo lahko dobili moč preko enačbe P=U*I in zanemarili nihanje napetosti na omrežju. S samo enim senzorjem za eno fazo je uporabljen samo en analogni vhod na mikrokrmilniku za en merilec. Tako lahko priključimo do 5 merilnikov na en mikrokrmilnik.

Za naš merilec smo uporabili čip Allegro ACS714 30A, ki je namenjen merjenju enosmernega električnega toka. Tako smo bili primorani izmenično napetost pretvoriti v enosmerno napetost. Slednjo smo pa dobili preko diodnega mosta ti. gretz.



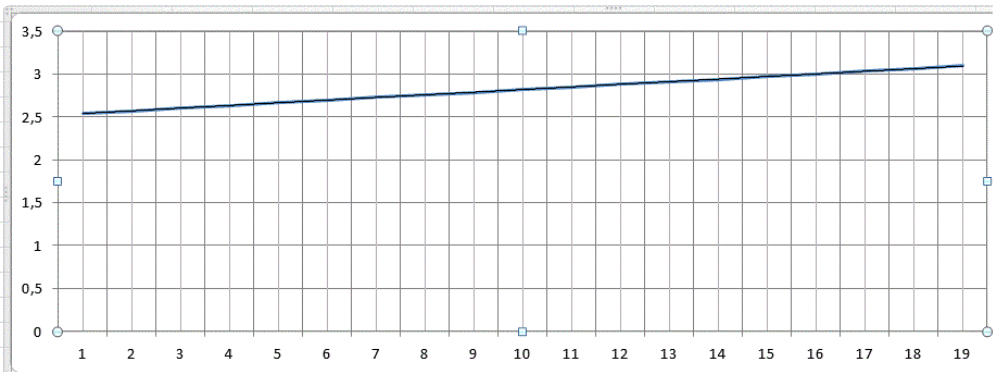
Slika 12: Diodni most ti. GRETZ

Kot je z zgornje sheme razvidno, merimo tok še na izmenični napetosti, ki jo moramo za senzor pretvoriti v enosmerno. Senzor nam vrača napetost glede na tok, kar je vidno na naslednjem grafu in tabeli. Napaka merjenja je do 1,5%, kar je navedeno v navodilih merilnika.

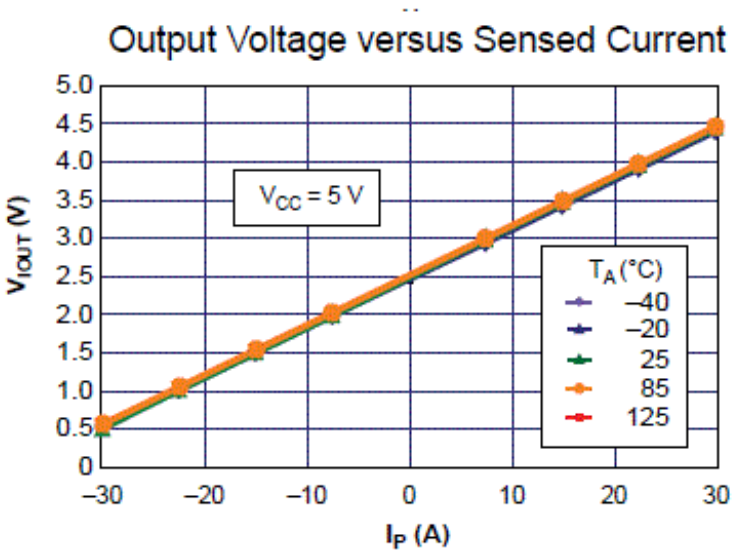
Tako merimo ampere na pretvorjeni enosmerni napetosti, porabniki pa so priključeni še vedno na izmenično napetost. V kolikor bi porabnike priklopili na enosmerno napetost, ti ne bi delovali, razen čistih Ohmskih porabnikov kot je navadna žarnica.

St. meritev	Tok (A)	V(out)
1	0	2,538
2	0,5	2,571
3	1	2,603
4	1,5	2,634
5	2	2,668
6	2,5	2,696
7	3	2,73
8	3,5	2,758
9	4	2,789
10	4,5	2,819
11	5	2,852
12	5,5	2,879
13	6	2,91
14	6,5	2,94
15	7	2,971
16	7,5	3
17	8	3,032
18	8,5	3,06
19	9	3,096

Tabela 2: Meritve izhodne napetosti iz senzorja



Graf 6: Graf izmerjene napetosti na senzorju

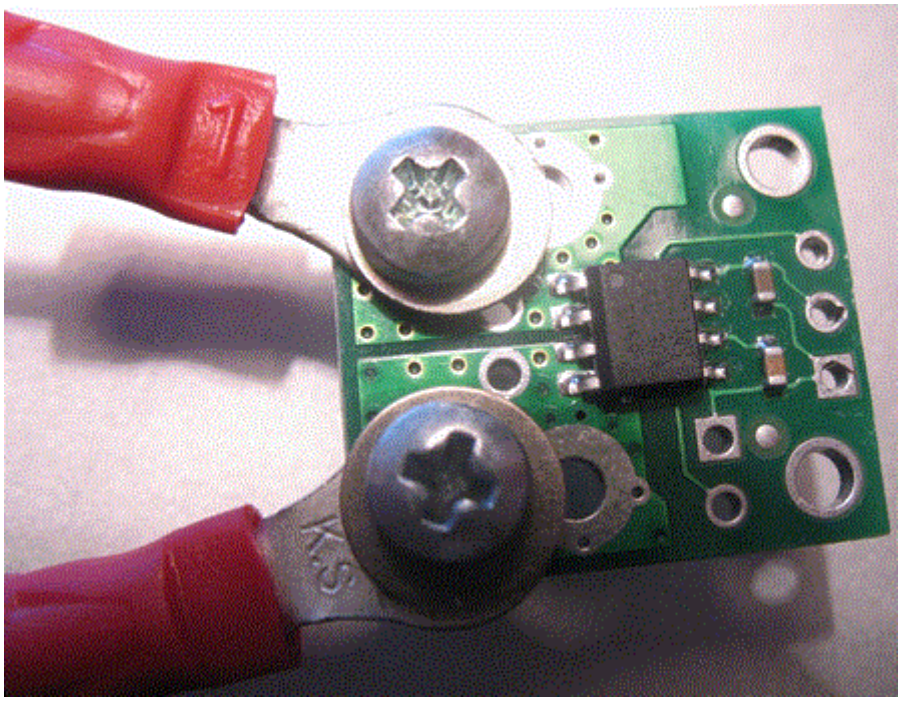


Slika 13: Graf izmerjene napetosti iz navodil Allegro ASC714

Da smo se prepričali v delovanje merilca, smo naredili graf meritev, ki se ujema s tistim iz navodil. Najprej smo senzor razbremenili in kazal je V_{out} 2,538 V, kar je 0A obremenitve. Nato smo senzor obremenjevali s korakom 0,5A. Dobili smo linearne odčitke po 0,033V koraku, kar je pripeljalo do linearnega grafa. Smo pa preverili tudi možnost merjenja negativnega toka, kar bi lahko prišlo v poštev, če bi imeli ta senzor priključen na sončno elektrarno. Tako bi lahko meril in izračunal število shranjene električne energije, ko pa bi se tok obrnil in bi se se akumulatorji začeli prazniti, bi senzor to zaznal in bi zapisal, kar bi prineslo uporabniku pregled nad dejanskim stanjem porabe ob polnitvi in praznjenju akumulatorjev.

V dvomih smo bili še pri pretvarjanju iz izmenične v enosmerno napetost. Izmenična napetost niha s sinusoido, kar pomeni, da je pol periode pozitivno nabit en vodnik, drugo polovico periode pa je drug vodnik s pozitivnim nabojem. Ko pa smo to pretvorili v enosmerno napetost, smo vse pol periode, ki so segale v negativni del, preslikali v pozitivno območje.

Nastalo je sinusno nihanje, ki ima vse pol periode pozitivne, kar je enosmerna napetost. Vendar pa nastopi težava v nihanju, a za njo poskrbi merilec sam, saj je že nanj vgrajen kondenzator, ki služi kot filter, ki gladi vrhe nihanja, tako da nastane skoraj ravna napetost.



Slika 14: Tokovni senzor ACS714 30A

3.1.4 Predstavitev programa

3.1.4.1 Flowcode

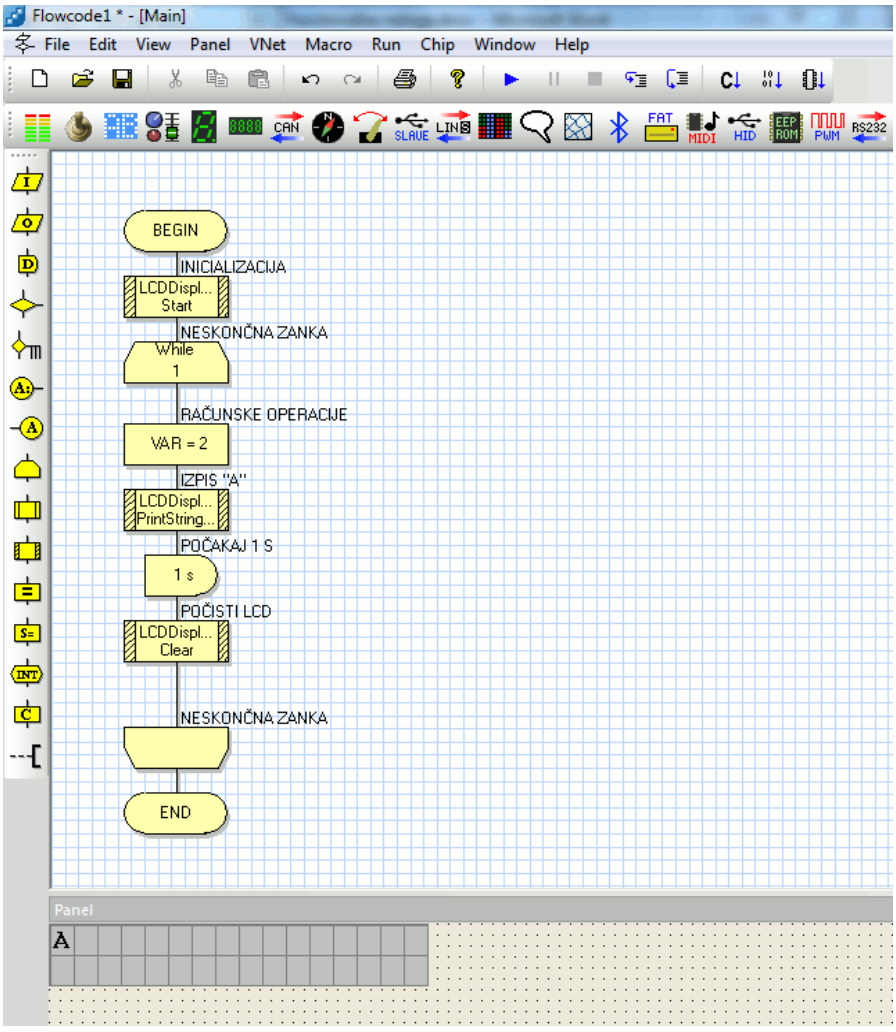
Je razvojno okolje (ang. IDE – integrated development enviroment), v katerem lahko razvijamo aplikacije za mikrokontrolerje in je izdelek podjetja Matrix Multimedia. Je ena izmed zelo naprednih aplikacij, saj nam omogoča tekstovno in grafično programiranje, hkrati pa tudi simuliranje napisanega programa. Podpira programiranje za platformo PIC, ARM in AVR. Za testiranje na strojni opremi pa so že narejene komponente E-Blocks, ki jih uporabljamo tudi mi, mogoče pa je uporabiti Formulo Flowcode in ECIO.

Blokovno ali grafično programiranje je, ko lahko bloke vlečemo na polje in s tem ustvarimo programski cikel. Vsak program se začne z start in konča z end blokoma. Nato dodamo LCD komponento, jo inicializiramo in nato dodamo neskončno zanko. Nato lahko uporabimo že narejene makre za komponente, jim dodamo parametre in že delujejo. Tako testni program izpisuje črko A na LCD s sekundnim razmikom, vmes pa se še LCD počisti, tako da bi lahko izpisali še kaj drugega.

Potem je še tekstovni način, kjer lahko C kodo vstavimo preko blokov v sam program. Lahko si pomagamo z že generirano kodo C, ki je takšna kot za ta testni program. Prisotne je še več dodatne kode, kot so implementacije metod, vendar je prikazana samo glavna koda.

```
void main()
{
    //Initialisation
    ansel = 0;
    cmcon = 0x07;
    //Interrupt initialisation code
    option_reg = 0xC0;
    //INICIALIZACIJA
    //Call Component Macro: LCDDisplay(0)::Start
    FCD_LCDDisplay0_Start();
    //NESKON?NA ZANKA
    //Loop: While 1
    while (1)
    {
    mainendloop: goto mainendloop;
}
```

Slika 15: Generirana C koda za testni program



Slika 16: Primer grafičnega programiranja

3.1.4.2 Glavni program na mikrokontrolerju

Diagram poteka glavnega programa na mikrokontrolerju zgleda nekako tako, kot je prikazano na sliki. Ko se zažene merilec, se najprej nastavijo prekinitvene metode in nastavitve zanj.

```
//VKLOP TIMER0 – t0con

t0con.T0CS=0;  //nastavimo bit na 0, kar izbere interno uro CLKO

t0con.T0SE=1; //nastavimo na 1, kar je povečaj tick ob visok na nizek bit.

t0con = (t0con & 0xF0) | 0x07; //je delilnik za uro 1960800/1024/256=75 prekinitev na sekundo.
```

Vrednost 3 bitov t0con registra		Delilnik
7	*111*	*1:256*
6	*110*	*1:128*
5	*101*	*1:64*
4	*100*	*1:32*
3	*011*	*1:16*
2	*010*	*1:8*
1	*001*	*1:4*
0	*000*	*1:2*

Tabela 3: Nastavitve 3 bitov za delilnik TIMER0

```
intcon.GIE=1; //globalni prekinitveni register, kjer nastavimo bit GIE na 1, kar omogoča prekinitve

intcon.TMR0IE=1; // postavimo bit na 1, ki vklopi TIMER0
```

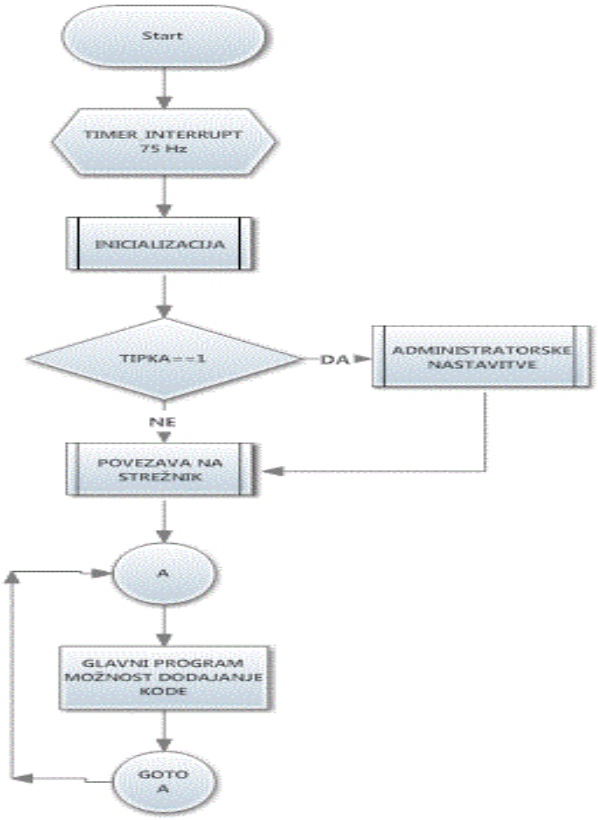
Tako se bo izvedla prekinitev 75 krat na sekundo, ki pokliče makro/metodo TIMER_INTERRUPT, ki preverja če je SD kartica vstavljena in če se je spremenilo stanje tipke za administratorske nastavitve. Prav tako se odšteva ena ura (60 minut) in se ob vsaki polni uri rezultat zapiše na SD kartico, razen če je vzpostavljena povezava na strežnik, nanj. Ustvari se urni arhiv porabe na SD kartici.

V metodi inicializacija vse komponente zaženejo svojo metodo inicializiraj. S tem se pripravijo na začetno vrednost, kajti brez inicializacije bi bile vrednosti lahko zelo nenavadne. Recimo spremenljivka tipa byte bi lahko bila v najboljšem primeru med 0 in 255, v najslabšem pa bi bila vrednost null. Tako bi se ob zahtevi operiranja s spremenljivko program naletel na neznano vrednost in bi se ustavil, vse dokler ga uporabnik ne bi ponovno zagnal. S tem makrom se vse spremenljivke postavijo na ničto ali privzeto vrednost.

V naslednjem koraku preveri, če je bila pritisnjena tipka in v primeru, da je bila, se pokliče makro Administratorske nastavitve, v nasprotnem primeru pa program nadaljuje svoje izvajanje. V nastavitvah se nastavi merilčev ID tj. 8 števil tipa byte. Nastaviti je potrebno še IP merilca in IP strežnika na katerega se povezujemo, za oba pa še porte preko katerih delujeta. Na koncu lahko vse spremembe pregledamo.
Vse nastavitve se shranijo v notranji pomnilnik EEPROM, ki je v mikrokontrolerju.

Naslednji makro je povezava na strežnik, kjer se uporabijo nastavitve iz EEPROM-a ter se poveže merilec na strežnik in preveri, če je ta sploh dosegljiv. Če ni, prikaže obvestilo za ponovno vzpostavitev ali pa shranjevanje samo na kartico. To pride v poštev takrat, ko administrator nastavlja merilec in lahko spremlja potek dogajanja.

Zadnja faza programa merilca je neskončna zanka, katera je nujno obvezna, saj preprečuje, da bi se program končal. V njej se preverja točnost podatkov-meritev in njihov zapis. Prav tako se lahko v tem bloku še dodajo nove funkcije kot je trenuten izpis porabe ali pa izris grafov.



Slika 17: Diagram poteka glavnega programa

3.2 Administracija strežnika

Strežnik je ključen člen med merilnikom in uporabnikom, saj skrbi, da so prejete meritve s strani merilnika pravilno obdelane, shranjene in hkrati zaščitene pred izgubo ali nedovoljenim dostopom. Poleg tega ima strežnik še eno zadolžitev. Na njem teče spletna aplikacija, ki uporabniku prikazuje obdelane podatke, shranjene na samem strežniku.

3.2.1 Kaj je strežnik?

Strežnik, znan tudi pod angleškim izrazom »server«, je večuporabniški računalnik, ki omogoča nadzorovan dostop do njegovih sredstev. Njegova struktura je enaka običajnemu namiznemu računalniku za domačo rabo, le da so ti običajno veliko zmogljivejši. Glede na njihov namen jih delimo v več skupin.

Najbolj znani so aplikacijski strežniki, ki zagotavljajo izvajanje operacij, od katerih prejmemo končen rezultat neke zahteve. Na njih tečejo aplikacije, ki so največkrat spletne strani in jih uporabljamo vsakodnevno s pomočjo spletnih brskalnikov.

Naslednja zelo pomembna skupina strežnikov so datotečni in podatkovni strežniki. Njihova naloga je shranjevanje datotek in podatkov ter omogočanje dostopa do njih. Datoteke, ki so shranjene na teh strežnikih, se odprejo na našem računalniku in jih urejamo lokalno. Podatki se nahajajo v podatkovnih zbirkah, ki jih drugače imenujemo tudi podatkovne baze ali angleško »databases«.

Najpomembnejša skupina strežnikov so prav gotovo domenski strežniki znani tudi pod kratico DNS. Ti omogočajo odjemalcu (npr. spletni brskalnik), da lahko v mreži računalnikov najde tistega, katerega vsebina je zahtevana. Sami odjemalci ne poznajo naslova želenega računalnika, zato jim DNS priskrbi naslov, da lahko pošljejo zahtevo. Brez te vrste strežnikov so vsi ostali neuporabni.

Obstajajo še druge vrste strežnikov kot so tiskalniški in e-poštni. Za potrebe našega sistema pa smo združili aplikacijski in podatkovni strežnik, saj bosta na njem tekli dve ločeni aplikaciji in podatkovna baza.

3.2.2 Programska oprema

Operacijski sistem, ki teče na strežniku je Microsoftov Windows Server 2008 R2. Imeli smo možnost uporabiti Linuxov operacijski sistem, za katerega se kasneje nismo odločili iz preprostega razloga. Aplikaciji, ki bosta tekli na strežniku, zahtevata podporo knjižnic, ki jih na Linuxovem sistemu ne najdemo, zato bi jih bilo potrebno namestiti posebej. To bi načeloma lahko storili brez težav, vendar te knjižnici niso podprte v takšnem obsegu kot na

Microsoftovih sistemih. S tem, da smo uporabili Microsoftov Windows Server, smo se izognili morebitnim težavam ob testiranju aplikacij na strežniku, ki bi lahko nastopile v primeru, da bi katera izmed knjižnic manjkala.

Za gostovanje uporabniškega vmesnika smo uporabili integrirano orodje Microsoftovega operacijskega sistema, IIS7. To je dodatna programska oprema za operacijske sisteme Windows, ki omogočajo preprosto pripravo strežnika za gostovanje spletnih strani in uporabo protokolov. S pomočjo IIS Manager orodja je administracija preprosta, saj ni potrebno pisati dolgih ukazov, ampak z uporabniškim vmesnikom v nekaj klikih pripraviš vse potrebno za spletno aplikacijo.

Za konec je bilo potrebno izbrati še programsko opremo za podatkovno bazo. Odločali smo se med Microsoftovima orodjema MSSQL Server 2008 in Microsoft Accessom. Obe orodji omogočata hitro in učinkoviti pripravo podatkovne baze, razlika je le v tem, da je MSSQL Server 2008 profesionalno orodje za delo s podatkovnimi zbirkami. Odločili smo se, da bomo uporabljali MSSQL Server, saj se bolje obnese pri delu s kompleksnimi zbirkami in ogromnimi količinami podatkov.

3.2.3 Kaj so podatkovne baze?

Skozi zgodovino so naši predhodniki pisali na različne materiale. Prve oblike pisanja je moč najti že iz časa prazgodovine, kjer so pisali v obliki slik na stene jam. Kasneje so začeli pisati na glinene tablice, papirus in pergament, kar je za takratne čase zadostovalo, saj se niso ukvarjali z velikimi količinami podatkov, katere bi bilo potrebno neprestano obdelovati. Nazadnje v 2. stoletju Kitajci iznajdejo papir, ki zamenja vse prednike in je aktualen še danes.

Vse te podlage so danes postale neuporabne v administraciji. Danes ta od nas zahteva maksimalen izkupiček dela. Obdelujemo velike količine podatkov dnevno, kateri vplivajo na delovanje sistema v državi, podjetjih in drugod, kjer do napak ne sme prihajati, zamuda na trgu pa danes pomeni propad v konkurenci. Zato za ta opravila uporabljamo računalnike, saj so ti zmožni sistematično obdelati gore podatkov. Da prihaja do maksimalnih izkoristkov računalnika, je potrebno podatke, ki jih na njem shranjujemo in obdelujemo, pravilno razporediti, združiti in zaščititi, pri čemer nam pomagajo podatkovne baze.

Podatkovne baze so torej zbirka podatkov, v kateri so ti logično povezani med seboj in nadzorovani pri dostopanju s pomočjo SUPB-ja. SUPB je programska oprema, ki deluje kot posrednik med uporabniškimi vmesniki in podatkovno bazo ter deluje kot zaščitnik in nadzornik podatkov katere hrani.

Podatki so v podatkovnih bazah shranjeni v tabelah, ki so sestavljene iz množice stolpcev in ti podatke določajo. Zato, da dosežemo maksimalen izkupiček uporabe podatkovnih baz, se podatki v njih ne ponavljajo, tabele so med seboj povezane, s čimer lahko določimo kateri podatki spadajo skupaj, čeprav so razpršeni glede na kontekst po tabelah v celotni zbirki podatkov.

Podatke iz podatkovnih baz dobimo s pomočjo povezovalnih stavkov ali drugače »poizvedb«. S pomočjo teh stavkov podatke, ki so razpršeni po podatkovni bazi, združimo in jih posredujemo uporabniku. S temi stavki je moč dobiti tudi podatke, ki sicer ne obstajajo v zbirki (Primer 1), uporabniku posredujemo le tiste podatke, ki so pomembni zanj, vsi ostali pa ostanejo njemu nevidni. Za vse te stvari poskrbi administrator podatkovnih baz, kateri ima popoln nadzor nad zbirko in na podlagi analize pripravi vse potrebno za različne skupine uporabnikov.

Primer 1:

V podatkovni bazi hranimo podatke neke osebe, med katerimi so: ime, priimek, naslov, poštna številka, kraj in datum rojstva. Med vsemi podatki, ki se bodo nahajali v tabeli, je le iz datuma rojstva možno dobiti različne podatke, ki pa se ne nahajajo v zbirki. Zakaj ne bi raje vedeli starost, kot pa datum rojstva? V tabeli so podatki sledečih oseb:

	Ime	Priimek	Naslov	Postna_st	Kraj	Datum_roj
1	Cvetka	Kim	Ob Ljubljanici 23	1000	Ljubljana	1967-05-15
2	Bogdan	Lužnar	Tržaška cesta 39	1000	Ljubljana	1954-02-27
3	Matej	Ahlin	Kvedrova cesta 7	1000	Ljubljana	1986-11-09

Slika 18: Podatki oseb v podatkovni bazi - tabela Oseba

Od vseh oseb nas zanima ime, priimek in starost, zato letnico rojstva uporabimo za izračun starosti, s čimer ustvarimo nov podatek, ki se NE nahaja med podatki osebe. Za pridobitev podatkov potrebujemo klasičen poizvedovalen stavek, ki mu dodamo nekaj funkcij za izračun starosti.

```
Select Ime, Priimek, FLOOR(DATEDIFF(DAY, Datum_rojstva, GETDATE()) / 365.25) As 'Starost'
From Oseba
```

*Z ukazom **Select** izberemo ime, priimek in datum rojstva iz tabele oseba, katera je podana z ukazom **From**. Preden uporabnik dobi podatke, se s pomočjo funkcije DATEDIFF izračuna število dni med datumom rojstva in današnjim datumom, ki ga vrne funkcija GETDATE. Število dni se nato deli z 365.25, ki predstavlja število dni v letu in šest ur s katerimi zajamemo tudi*

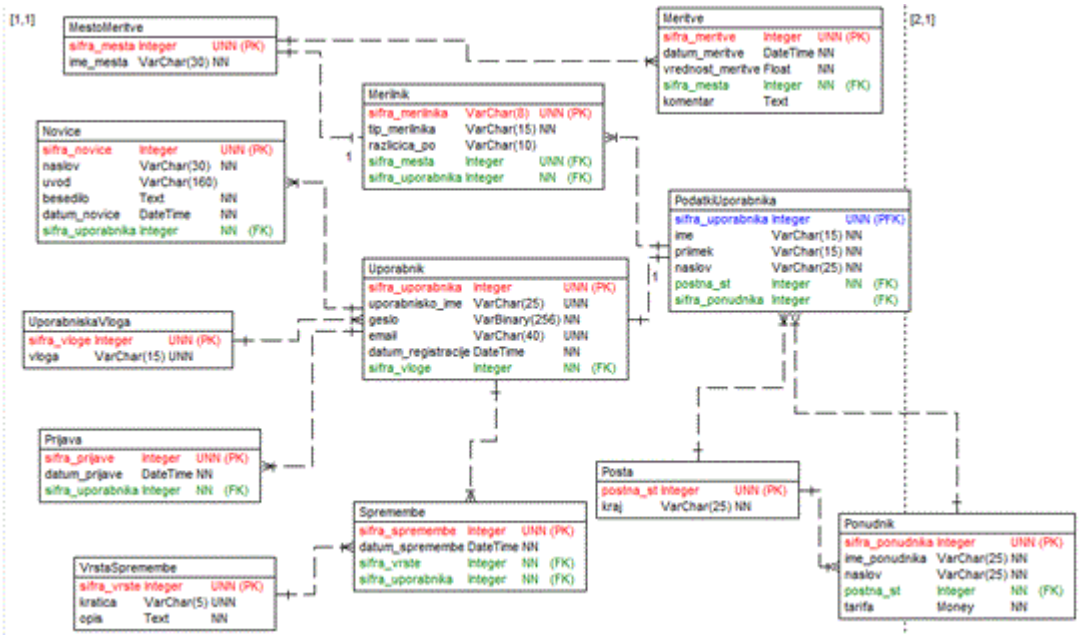
vsa prestopna leta. Rezultat je na koncu zaokrožen navzdol s funkcijo FLOOR. Zaokrožen rezultat je trenutna starost osebe. Dobljen rezultat je naslednji:

	Ime	Priimek	Starost
1	Cvetka	Kim	45
2	Bogdan	Lužnar	59
3	Matej	Ahlin	26

Slika 19: Podatki, ki jih vrne poizvedovalen stavek

3.2.4 Podatkovna baza sistema

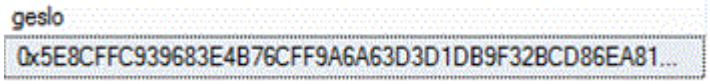
Podatkovna baza je za uporabo sistema sestavljena iz dveh delov, in sicer podatkov o uporabnikih ter njegovih meritvah. Vse skupaj je združeno v eno samo podatkovno bazo, ki beleži tudi uporabnikove zadnje prijave in spremembe na računu. Omogoča mu tudi izbor svojega ponudnika, s čim aktivira možnost približnega izračuna njegovih stroškov za neko obdobje. Ker so v enako podatkovno bazo shranjeni tudi administratorji in moderatorji, se v njej shranjujejo tudi novice, ki jih lahko ti vpisujejo ter objavljajo. Podatkovna baza je strukturirana univerzalno, kar pomeni, da jo je moč uporabiti z različnimi SUPB-ji.



Slika 20: ER-Diagram podatkovne baze

Podatki v podatkovni bazi niso šifrirani, izjema so gesla. Ta so šifrirana z algoritmom AES, ki uporablja fiksno dolžino blokov (128 bit) in 128, 192 ali 256 bitno dolžino ključa. Za šifriranje AES uporablja bloke, ki so razdeljeni na 16 manjših okenc velikosti 8 bitov, katere med sabo zamenjuje v več stopnjah. Velikost ključa določa število zamenjav, ki se lahko zgodijo 10 (128

bit), 12 (192 bit) ali 14 (256 bit) krat. Podatke je mogoče dešifrirat le z enakim ključem ali pa ti ostanejo izgubljeni. Rezultat šifriranja se nikoli ne ponovi.

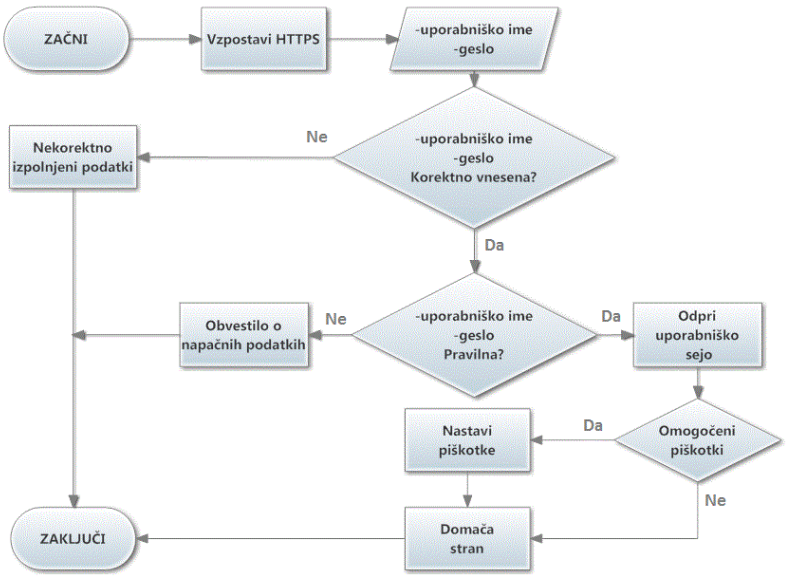


Slika 21: Geslo, ki je shranjeno v podatkovni bazi (AES_256 Encryption)

Dostop do podatkovne baze je omejen še z uporabnikom, ki se mora vpisat ob vsaki vzpostavitvi povezave z bazo. S tem je še nekoliko zaščitena vsebina v podatkovni bazi, saj ima uporabnik možnost le branja, pisanje pa je omogočeno le v posebnih situacijah (npr. ob spreminjanju podatkov, ob vnosu meritev).

3.2.5 Sistem za prijavo

Sistem za prijavo je narejen za identifikacijo uporabnika, saj lahko le ta spremlja svojo porabo električne energije in spreminja svoje podatke. Pred prijavo se najprej vzpostavi HTTPS povezava, ki služi za zaščiten prenos podatkov med uporabnikom in strežnikom. To se izvede simultano ob zahtevi za vstop na stran za prijavo. Uporabnik se prijavi s svojim uporabniškim imenom in geslom, ki si ju je nastavil ob ustvarjanju računa. Ob prijavi se najprej preveri ustreznost vpisanih znakov v prijavna okenca, s čim se zavarujemo pred ukazi, ki bi na nedovoljen način pridobivali podatke iz podatkovne baze. Nato se sproži proces preverjanja uporabniškega imena in gesla. V primeru, da je uporabnik vnesel ustrezne podatke, se zanj odpre seja, če je podal zahtevo »zapomni geslo« se ustvarijo tudi piškotki z njegovimi podatki.



Slika 22: Diagram postopka prijave uporabnika

V primeru, da je uporabnik neaktiven vsaj 15min, se zaključi njegova seja in se pobrišejo vsi podatki. Enako se zgodi v primeru, da se uporabnik izpiše.

3.2.6 Aplikacija za sprejem meritev

Merilnik neprestano meri porabo električne energije, ki se shranjuje na spominsko kartico, vendar za uporabnika tisti podatki niso koristni, saj se ne nahajajo na strežniku. Da se ti shranijo na strežnik, je potrebna aplikacija, ki zna komunicirati z merilnikom preko interneta. Prenos iz merilnika na strežnik se dogaja s pomočjo TCP/IP protokola, ki poveže strežnik z merilnikom preko njunih naslovov v spletu (IP naslovi).

IP naslovi so enolične identifikacijske številke, ki določajo računalnik v spletu. Sestavljen je iz štirih 8 bitnih števil v desetiški obliki, ki skupaj tvorijo 32 bitno unikatno številko. IPv4 naslovi omogočajo več kot 4 milijarde različnih naslovov, ki pa jih danes že primanjkuje, zato so že razvili naslove IPv6. Ti naslovi niso več 32 bitni pač pa so 128 bitni in omogočajo mnogo več različnih identifikacij v omrežju.

192.168.1.100

Slika 23: Primer IPv4 naslova

2021:0AB8:AD10:FE01:0100:1001:0000:0000

Slika 24: Primer IPv6 naslova

Aplikacija omogoča hkratno komunikacijo z več merilci zaradi podpore niti. Niti omogočajo sočasno izvajanje operacij, kar omogoča izvajanje operacij v ozadju, medtem ko sam program ostaja neobremenjen in čaka na uporabnikov ukaz. Tako lahko naša aplikacija hkrati prejema podatke več merilnikov in jih tudi shranjuje.

```
ServerThread _ServerThread = new ServerThread();
Thread _Thread = new Thread(new ThreadStart(_ServerThread.run));
_Thread.Name = "Server - TCPListener";
_Thread.Start();
```

Slika 25: Del kode (C#), ki ustvari novo nit

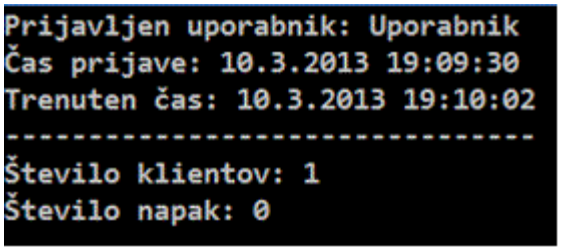
Izsek kode (Slika 21) prikazuje izvajanje niti, ki neprestano čaka na vzpostavitev povezave z novim merilnikom. Ko pride zahteva za novo povezavo metoda AcceptSocket vrne nov

socket. To je končna točka, ki omogoča prenos podatkov preko spleta. Ko se povezava vzpostavi se merilnik najprej predstavi z unikatno številko, ki jo ima shranjeno v eepromu. Če je predstavitev uspela, se ustvari nova nit, ki začne sprejemat podatke in jih vpisovat v podatkovno bazo. V primeru, da identifikacija ne uspe, se povezava zavrže in se čaka na novo zahtevo. Pri vzpostavitvi povezave se lahko zgodi, da pride tudi do nepričakovane težave, katera lahko sesuje program zaradi česar je ta občutljiv del kode zapisan v **try{…}**. Ta preprečuje, da bi se program sesul, v primeru napake pa skoči v predel kode, ki lovi napako (**catch{…}**) in jo zabeleži.

```
while (true)
{
    try
    {
        Socket socket = server.AcceptSocket();

        if (socket != null)
        {
            Merilec m = Identifikacija(socket);

            if (m != null)
            {
                PovezanThread _PovezanThread = new PovezanThread(socket, m);
                Thread _Thread = new Thread(new ThreadStart(_PovezanThread.run));
                _Thread.Name = "Client - TCPReciever";
                _Thread.Start();

                Vrednosti.StClientov++;
            }
        }
    }
    catch (Exception e)
    {
        Log.CustomError(e);
        Vrednosti.StNapak++;
        break;
    }
}
```

Slika 26: Izsek kode (C#) iz aplikacije, ki vzpostavlja povezavo z novimi merilniki

Končna različica aplikacije na namizju prikaže prijavljenega uporabnika na strežniku. Ob uporabniku je moč videti tudi čas prijave v sistem, ki pove, kdaj je bil omogočen sprejem podatkov na strežnik. Aplikacija prikazuje tudi trenuten čas, ki je nastavljen na strežniku, število merilnikov, ki pošiljajo podatke na strežnik in morebitno število napak, ki so se dogodile ob izvajanju aplikacije.



Slika 27: Izpis na aplikaciji

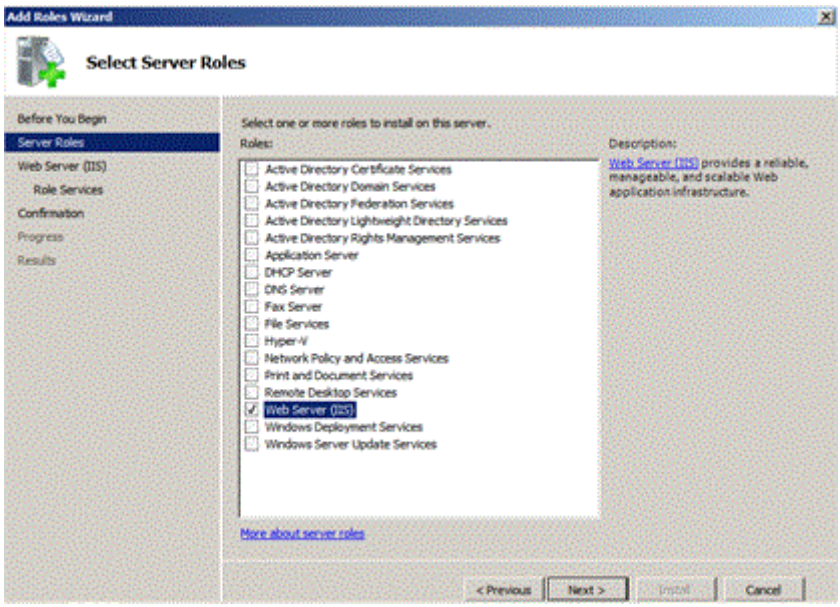
3.2.7 Priprava IIS 7 za uporabniški vmesnik

Zadnji del administracije strežnika je bila priprava okolja za uporabniški vmesnik. Ta zahteva različne nastavitve za varno in učinkovito delo z njim. Nastaviti je bilo potrebno certifikat, avtomatsko vzpostavitev varne povezave, povezavo za delo s podatkovno bazo, knjižnice (.NET Framework), strani za napake in sam uporabniški vmesnik.

3.2.7.1 Postopek priprave okolja

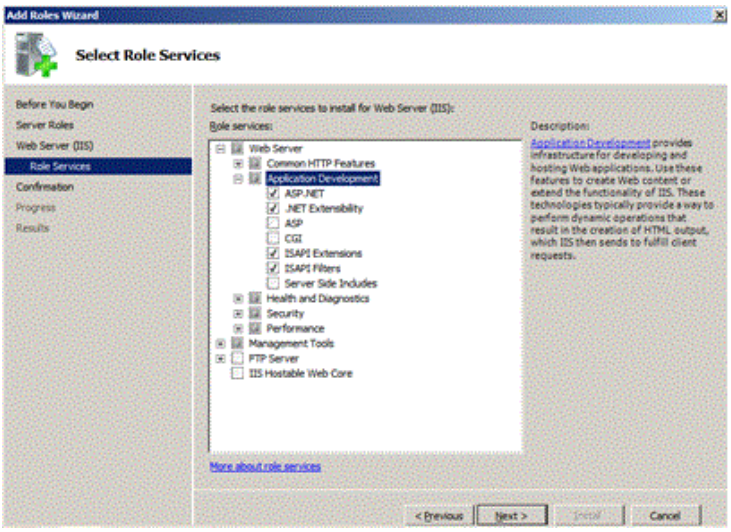
Preden smo lahko nastavili zgoraj naštete nastavitve, je bilo potrebno vključiti možnost spletnega strežnika(nastavitev IIS). Postopek je voden s pomočjo čarovnika, v katerem so natančno obrazložene vse nastavitve, ki jih je mogoče vključiti, tudi za nekoga, ki to počne prvič.

Pripravo spletnega strežnika (IIS) smo opravili s pomočjo orodja »Server Manager«. To orodje vsebuje vse informacije v zvezi z nastavitvami strežnika in omogoča hiter dostop do čarovnikov za spreminjanje nastavitev. Strežniku smo nastavili vlogo IIS (Slika 23), ki omogoča gostovanje spletnih strani.



Slika 28: Čarovnik za nastavitev vlog strežnika (vključena vloga spletnega strežnika - IIS)

V naslednjem koraku je bilo potrebno nastaviti servise spletnemu strežniku. Osnovne funkcije so že bile nastavljene privzeto in njih nismo spreminjali. Dodali smo pomožne servise za .NET aplikacije (Slika 24). Ti servisi omogočajo pretvorbo aplikacije v spletno stran (naš uporabniški vmesnik).

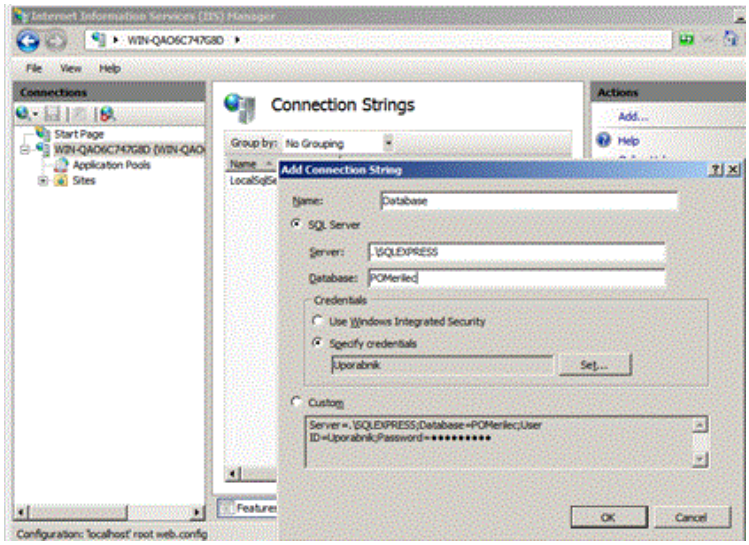


Slika 29: Vključitev servisov za .NET aplikacije

Nato je bilo potrebno počakati, da se naložijo vsi potrebni programi za upravljanje spletnega strežnika. Konfiguracijo spletnega strežnika smo opravili s pomočjo orodja »IIS Manager«. To orodje omogoča v nekaj klikih vključiti nastavitve za spletno stran brez pisanja dolgih ukazov.

Najprej je bilo potrebno nastaviti certifikat, ki bo omogočal predstavitev strežnika odjemalcu, da bosta lahko vzpostavila varno povezavo (HTTPS) preko katere se bodo prenašali podatki. S pomočjo te povezave so podatki šifrirani in zaščiteni pred prestrezanjem.

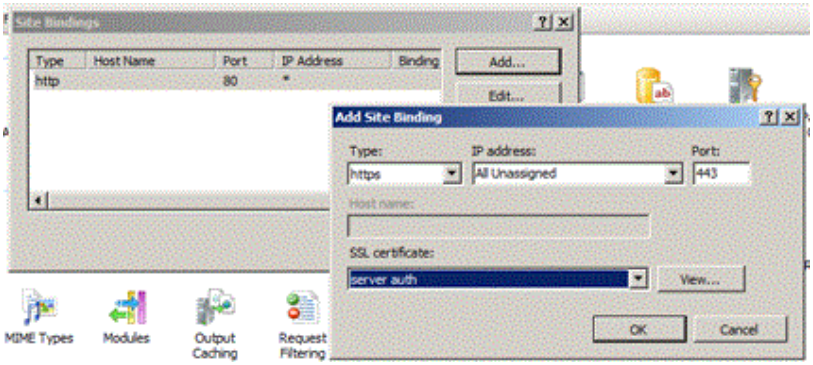
Ko je bil certifikat nastavljen, smo dodali povezavo do podatkovne baze s pomočjo vnosnega okenca (Slika 25). Ta povezava se nato zabeleži v datoteko z nastavitvami in omogoča spletni strani, da jo po potrebi uporabi. S pomočjo te nastavitve ni potrebno vnašati povezave do podatkovne baze v samo kodo spletne strani.



Slika 30: Dodajanje povezave do podatkovne baze

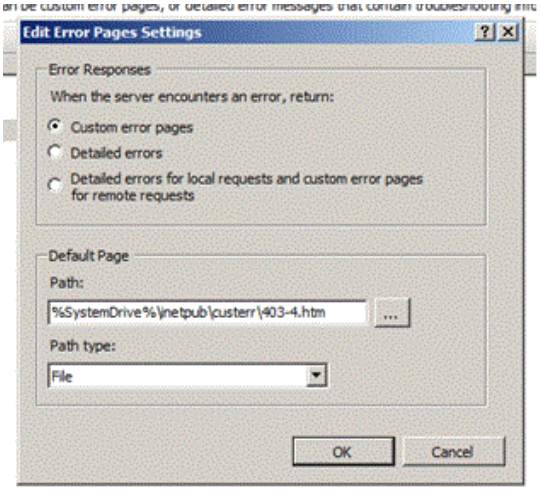
Naslednja stvar, ki smo jo nastavili, so bile knjižnice. S spleta smo prenesli .NET Framework 4.0 s pomočjo katerega je narejen uporabniški vmesnik. Nato smo naložili uporabniški vmesnik na sam strežnik in spremenili še zadnje nastavitve. V IIS Manager smo navedli pot do uporabniškega vmesnika, ga pretvorili v spletno stran in mu nastavili različico knjižnic 4.0 (Framework 4.0).

Na koncu vključili še varno povezavo (Slika 26 in Slika 27) na spletni strani in izklopili podroben izpis napak za spletne strani (Slika 28).



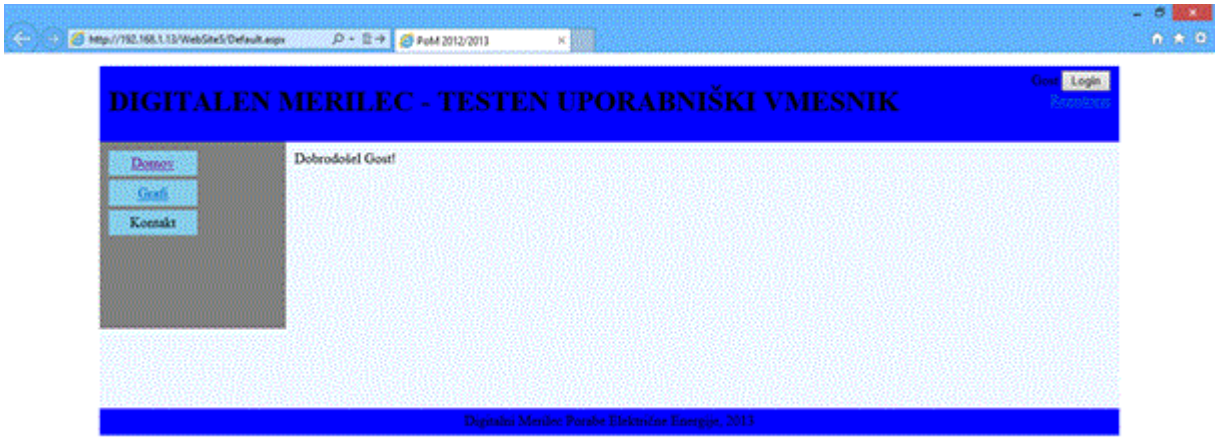
Slika 31: Vključitev HTTPS povezave za spletno stran

Slika 32: Nastavitev zahtevka za varno povezavo (neizpolnjevanje pogojev ne prikaže spletne strani)



Slika 33: Prikaz splošnih napak in ne podrobnih

S tem smo v nekaj minutah nastavili uporabniški vmesnik na strežniku in zaključili z njegovo administracijo. Drugih nastavitev na strežniku nismo spreminjali, saj nam zadoščajo že nastavljene privzete nastavitve ob namestitvi.



Slika 34: Testen uporabniški vmesnik na strežniku

3.3 Uporabniški vmesnik

Za prikazovanje podatkov uporabniku uporabljamo različne stvari, od namiznih aplikacij pa vse tja do spletnih aplikacij. Glavna razlika med tema dvema je ta, da če hočeš podatke prikazati v namizni aplikaciji, jo moraš najprej namestiti na računalnik. Kar je lahko nerodno, saj če želiš te podatke pogledati na drugem računalniku, jo moraš tudi tam namestiti. Glavna prednost spletne aplikacije pa je v tem, da je dostopna od kjerkoli na svetu, saj moramo imeti samo povezavo v svetovni splet in že lahko pregledujemo te podatke. Prikaz podatkov pa mora biti tudi uporabniku zanimiv, da bo z večjim zanimanjem obiskoval to spletno stran in jo tudi priporočal svojim prijateljem, znancem… ,da bodo tudi sami lahko spremljali svojo porabo električne porabe, seveda ob predpostavki, da si bodo to možnost izbrali.

Zaradi te bolj uporabniku lažje zadeve smo se odločili, da naredimo spletno aplikacijo, ki bo prikazovala porabo električne energije v svojem stanovanju oz. hiši.

Za ustvarjanje spletnih aplikacij imamo na voljo kar nekaj orodij, s katerimi si lahko olajšamo izdelavo spletne strani. Predstavnika teh orodij sta:

- Notepad++
- Microsoft Visual Studio

3.3.1 Microsoft Visual Studio

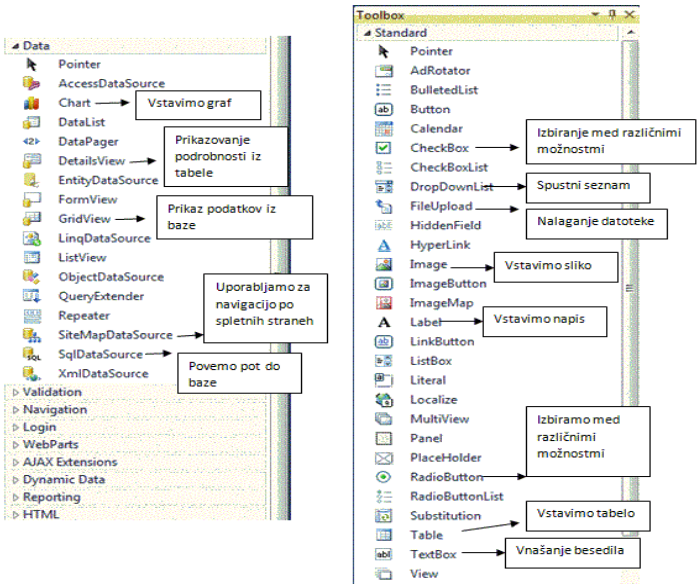
Je razvojno okolje, ki se uporablja za razvoj konzolskih in grafičnih uporabniških vmesnikov. Grafični vmesnik lahko naredimo v Windows Form. Lahko pa naredimo spletne strani, spletne aplikacije ter spletne storitve. Vse te izdelke, ki jih naredimo, lahko uporabljamo na vseh platformah, ki jih podpira Microsoft. Microsoft visual studio je zelo močno orodje, saj lahko v njem naredimo zelo veliko stvari, ker med drugim podpira različne programske jezike kot so C/C++, VB.NET, F# ter C#, prav tako pa podpira tudi XML, HTML, JavaScript in CSS. Pogoj za ustvarjanje je samo znanje enega programskega jezika in že lahko naredimo uporabno aplikacijo ali pa spletno stran v tem razvojnem okolju.

Mi smo se odločili narediti aplikacijo v Microsoft Visual Studiu 2010 Express verziji, katera je na voljo brezplačno na Microsoftovi uradni spletni strani in si jo lahko vsak legalno sname iz interneta. Ta verzija je namenjena izobraževanju, zato je tudi brezplačna. Za Microsoft Visual Studio smo se med drugim odločili tudi zato, ker ga uporabljamo tudi v šoli in ga najbolje znamo uporabljati.

Poznamo pa tudi druge različice Visual Studia kot so

- Visual Studio Professional
- Visual Studio Premium
- Visual Studio Ultimate

Zgoraj naštete verzije so namenjene za podjetja, ki se ukvarjajo z razvojem raznoraznih aplikacij, saj moramo za njihovo uporabo kupiti licenco.



Slika 35: Gradniki, ki jih omogoča Visual Studio za izdelavo aplikacij

3.3.2 ASP.NET

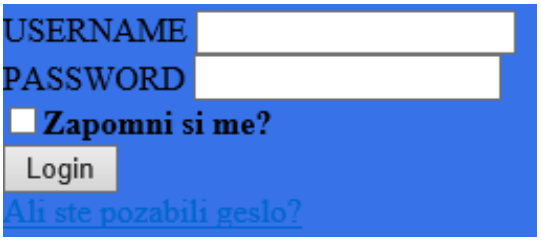
Uporablja se za ustvarjanje dinamičnih spletnih strani. Je produkt Microsoftovih programerjev za ustvarjanje dinamičnih spletnih strani, spletnih aplikacij in spletnih storitev.

Odločili smo se svojo spletno aplikacijo izdelati v Microsoft Visual Studiu 2010, saj nam med drugim omogoča hitrejše dodajanje elementov, ki jih želimo imeti na strani. Te elemente enostavno povlečemo iz menija na katerem se nahajajo in ga postavimo točno na tisto mesto, na katerem želimo, da se vedno pokaže.

V asp.net nam je tudi omogočeno pisanje same kode v C# ali v VB programskem jeziku. Ta se uporablja za vse, kjer želimo dodajati funkcionalnost, česar se ne da napisati z navadnimi HTML jezikom in nam je v veliko pomoč.

```
USERNAME <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<br />
PASSWORD <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>
<br />
<asp:CheckBox ID="CheckBox1" runat="server" Font-Bold="True" Text="Zapomni si me?" />
<br />
<asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Login" />
<br />
<asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/ForgPassword.aspx">Ali ste pozabili geslo?</asp:HyperLink>
<br />
<asp:Label ID="Label1" runat="server" ForeColor="Red"></asp:Label>
```

Slika 36: Izsek kode ASP.NET, ki ustvari gradnike za prijavo



Slika 37: Prijavno okno z gradniki ASP.NET

3.3.3 .NET Framework

.NET Framework je platforma za ustvarjanje aplikacij za Windows, Windows Store, Windows Phone, Windows Server in Windows Azure. Vključena ima programska jezika C# in Visual Basic ter vsebuje knjižnice, ki so skupne obema.

Ker smo delali v Microsoft Visual Studiu 2010, smo imeli privzeto zadnjo verzijo frameworka 4.0, v kateri je nastal uporabniški vmesnik.

3.3.4 Oblikovanje s CSS

Kaskadne stilske podloge, bolj poznane pod kratico CSS, so podloge, predstavljene v slogovnem jeziku, ki poskrbi za lepše oblikovanje samih spletnih strani. S tem brskalniku povemo na kakšen način naj spletno stran pokaže. Določamo lahko razne zadeve kot so barve,velikost,razni odmiki od drugih elementov, lahko pa tudi povemo kaj se naj zgodi, ko uporabnik npr. prekrije povezavo z miško… Z uporabo samostojne datoteke v kateri je CSS pridobimo na tem, da je sama HTML koda bolj pregledna. Prav tako pa tudi zmanjšamo ponavljanje kode, ki brskalniku pove, kako naj prikaže sam element na strani.

Za samo oblikovanje strani smo se odločili za CSS, saj lahko v njem po želji oblikujemo spletno stran. Priporočljivo pa je, da je dizajn spletne strani ločen od same kode, ki jo prebere brskalnik, da ve, kakšne gradnike oz. elemente naj prikaže. To se najbolj pozna v hitrosti prikazovanja same spletne strani, ker se na naš računalnik prenese samo html koda, ne pa tudi CSS, saj se sama koda vključno z asp.net gradniki izvajajo na strežniku in ne na lokalnem računalniku.

```
31  .vsebina
32  {
33      width:auto;
34      min-height:1000px;
35      height:auto;
36      background-color:#3772E8;
37      }
```

Slika 38: Izsek kode (CSS) za oblikovanje spletnih strani

Z min-height povemo kakšna je lahko najmanjša višina strani. Z background-color povemo kakšno barvo ozadja naj prikaže. Barvo lahko napišemo v šestnajstiški obliki (v HEX-a obliki), prikaže nam lahko katero koli barvo, ki jo lahko najdemo na barvni paleti. Samo višino in širino strani smo nastavili na samodejno, v primeru, da ima uporabnik manjšo oz. večjo ločljivost zaslona mu stran prilagodi na njegovo ločljivost ter se s tem sama oblika postavitve strani ne spremeni in je različnim ločljivostim zaslona prikazana enako.

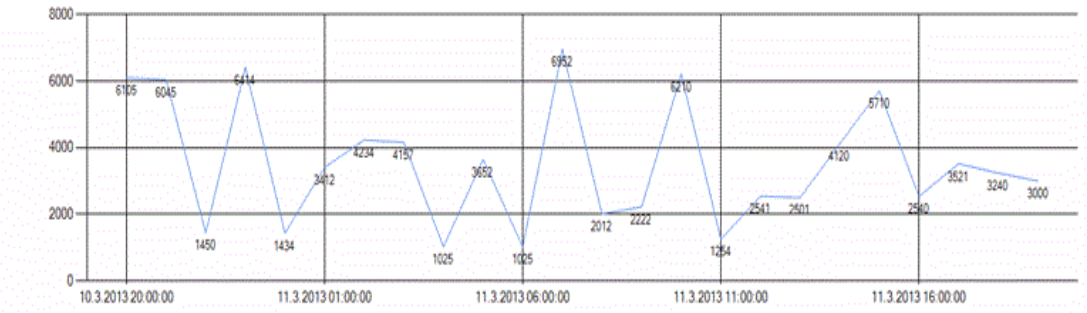
3.3.5 Spletna stran (UV)

Če bo uporabnik želel pregledovati svojo porabo električne energije, se bo moral najprej prijaviti. Ko bo uporabnik prijavljen, bo lahko videl vse od golih številk pa vse do izrisanih grafov, pri katerih je največja prednost, da se uporabniku ni treba preveč ukvarjati s številkami. Videl bo graf porabe električne energije za 24 ur nazaj in za tekoči mesec po dnevih. Lahko si bo sam naredil graf, ki bo prikazoval porabo električne energije med dvema datuma, ki ju bo izbral. Imel bo tudi vpogled v približen strošek za tekoči mesec za svojega

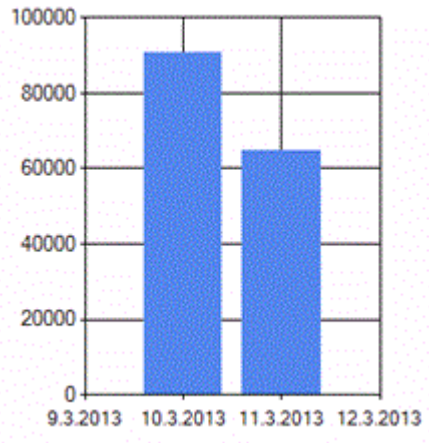
ponudnika. Možnost bo imel tudi spreminjati različne svoje podatke. Lahko si bo tudi podatke prenesel na svoj računalnik v Microsoftov Office Excel. Na možnost bo imel tudi narediti svojo poizvedbo za določen datum, ki si ga bo izbral sam. Če bo uporabnik naletel na kakršno koli težavo, bo lahko vzpostavil kontakt tako, da bo poslal e-pošto.

Skupna poraba(W)	Predvideni stroški
155373	11,0625576

Slika 39: Primer izračuna stroškov za 155kWh



Slika 40: Primer prikaza porabe električne energije za zadnjih 24h



Slika 41: Primer prikaza porabe električne energije za vsak dan v mesecu

```
Chart1.ChartAreas[0].AxisX.LabelStyle.Format = "{0:d/M/yyyy HH:mm:ss}";
```

Ta primer kode nam na grafu, za katerega smo izbrali podatke, prikaže podatke, ki so nastavljeni za x os. V tem primeru DateTime izpiše v obliki, ki je navedena za enačajem. Ta je nastavljena za evropski standard.

Slika 34 prikazuje kodo, ki se izvede ob kliku na gumb Moja poizvedba. Najprej uporabnik s spustnim seznamom izbere dan, mesec ter leto za katerega želi, da se mu prikaže poraba električne energije na sami strani.

```
protected void Button4_Click(object sender, EventArgs e)
{
    SqlConnection sql = null;
    sql = new SqlConnection(sqlConn);
    string s = DropDownList3.SelectedValue + "-" + DropDownList2.SelectedValue + "-" + DropDownList1.SelectedValue + " 00:00:00";
    SqlCommand sqlCommand = new SqlCommand(sqlString, sql);

    sql.Open();
    SqlDataReader reader = sqlCommand.ExecuteReader();
    GridView2.DataSource = reader;
    GridView2.DataBind();
    sql.Close();
}
```

Slika 42: Izsek kode za Moje poizvedbe

4 Razprava

Pred začetkom raziskovanja smo si postavili hipoteze, s pomočjo katerih smo pripravili to raziskovalno nalogo. Te so navedene v uvodu raziskovalne naloge. Hipoteze, ki smo si jih postavili, bazirajo iz vseh področij naše raziskave (anketa, merilnik, strežnik in uporabniški vmesnik).

4.1 Veljavnost hipotez

- **Vsaj 75% anketiranih spremlja porabo električne energije.**

  Po rezultatih spletne ankete, s katero smo spraševali anketirane osebe o njihovem nadziranju porabe električne energije, moramo zgornjo hipotezo ovreči, saj je le 70% anketiranih odgovorilo, da vsaj občasno spremlja porabo električne energije. Menimo, da lahko trdimo, da bi bili ti rezultati mnogo višji in hipoteza potrjena, če bi zajeli več kot le 70% ciljne skupine, za katero je bila anketa pripravljena.

- **Anketirani v večini beležijo porabo.**

  Priznati moramo, da nas rezultati za to hipotezo nekoliko presenečajo, saj med vsemi, ki spremljajo porabo električne energije, več kot 50% vprašanih ne zabeleži porabe. Rezultati so si sicer bili zelo blizu in lahko ponovno trdimo, da bi bili, v primeru zajete večje ciljne skupine anketiranih, rezultati še tesnejši. Tudi to hipotezo smo ovrgli.

- **Merilnik bo vračal rezultate z do +|- 3% napako.**

  Zaradi obsežnosti raziskovalne naloge nam je zmanjkalo časa za podrobnejše testiranje točnosti meritev, zato smo se odločili, da bomo napako izračunali. V navodilih senzorja piše, da ima ta napako pri vračanju rezultatov 1,5%. Ker nimamo senzorja napetosti, smo to nastavili statično, pri čemer nastaja napaka 2,2%. Prišteti moramo tudi napako pri zaokroževanju in upornost žic, kar skupaj prinese 4,7% napako. Žal moramo ovreči tudi to hipotezo.

- **Stroški programske opreme strežnika bodo minimalni.**

  Na strežniku smo uporabili integrirana orodja, ki pridejo skupaj z operacijskim sistemom in brezplačno profesionalno orodje za upravljanje podatkovnih baz. Ocenjujemo, da bi za programsko opremo, ki smo jo uporabili, plačali le licenco operacijskega sistema. Dodaten strošek za strežnik bi nam predstavljala zakup domene in podpis certifikata za varno povezavo. Skupaj ocenjujemo okoli 460€ stroškov za strežnik na leto, kar bi v primeru 100 uporabnikov pokrili z 0,40€ na mesec za uporabnika.

  Z gotovostjo lahko trdimo, da so stroški programske opreme minimalni in hipotezo potrjujemo.

- **Uporabnik si bo lahko sam nastavljal izris grafa porabe.**

  V uporabniškem vmesniku smo sprogramirali možnosti, ki omogočajo, da si uporabnik sam izbere kaj želi, da mu prikazujejo grafi. S tem uporabniku omogočimo, da si sam nevede ustvarja poizvedbe za grafično upodobitev podatkov, ki jih želi spremljati. Hipotezo potrjujemo.

5 Zaključek

Namen naše naloge je bil pripraviti sistem, ki bo sposoben zamenjave ročnega beleženja porabe električne energije z avtomatskim. S to nalogo smo zajeli vsa štiri leta šolanja in v grobem skupaj povezali sklope, ki smo jih v tem času obdelovali. Priznati moramo, da smo se naučili tudi veliko novega, kar ni zanemarljivo. Za uspešno pripravljen praktičen izdelek je bilo ključnega pomena timsko delo, saj je naloga kaj kmalu pokazala, da je izjemno široka. Na trenutke smo bili trije premalo in smo se spraševali ali nam bo uspelo zaključiti raziskavo do postavljenega termina, vendar nam je uspelo in smo z našim izdelkom zelo zadovoljni.

Kljub dobro pripravljeni analizi problema, ki smo jo pripravili takoj na začetku, so skozi pripravo praktičnega izdelka prihajale nove in nove podrobnosti na katere smo bili prisiljeni paziti. Tukaj bi radi izpostavili težavo pri povezavi naše podatkovne baze z Microsoftovo bazo za upravljanje z uporabnikovimi podatki. Njihova podatkovna baza je zaščitena tako močno, da iz nje nismo mogli dobiti ključa, preko katerega bi lahko povezali podatkovni baz. To nam je prineslo veliko težavo, ki smo jo lahko rešili le z lastnim sistemom, ki pa je poleg tega zahteval še lasten sistem za prijavo, registracijo, možnost pozabljenega gesla in še. Ta težava nam je povzročila nekaj skrbi. Nazadnje smo jo vendarle uspešno rešili in če pogledamo iz druge perspektive, smo se zaradi te težave zopet veliko naučili.

Z glavno zasnovo naloge nismo imeli težav. Na začetku smo bili prisiljeni predelati kar nekaj publikacije, saj so bile osnove znanja Flowcoda iz prvega letnika premalo. Nekaj težav nam je povzročal tudi IDE, ki nam na trenutke ni želel prevesti diagrama poteka v C kodo. To je bilo potrebno urejati na roke oz. smo večkrat na novo prevajali tisti del kode, dokler ni bil pravilno preveden.

Priprava strežnika je bila hitra in učinkovita, za kar se gre zahvaliti tekmovanju, ki smo ga imeli v tretjem letniku in smo se morali naučit konfigurirat strežnik. Več dnevno učenje in vaje hitre priprave strežnika so se nam pri raziskovalni nalogi obrestovale, saj se ni bilo potrebno vsega učiti na novo. Potrebovali smo le dobro uro, da smo se po približno polovici leta ponovno vpeljali in postavili testno stran.

Programiranje aplikacij in podatkovne baze nam ni povzročalo nikakršnih težav, saj že dobra tri leta aktivno programiramo in se ukvarjamo z razvojem aplikacij ter podatkovnih baz. Uporabili nismo ničesar novega, kar nismo že prej spoznali pri pouku ali poizkušali doma. Edino kar smo morali storiti, je bilo to znanje samo povezati in pripraviti še en program po vnaprej določenih zahtevah.

Za konec je bilo potrebno le še pripraviti tiskano vezje za tokovni merilec in uro. Ker smo bili prepričani, da bo tokovni senzor deloval in da bomo lahko to pripravili povsem na koncu, smo se odločili, da najprej pripravimo uro na merilniku, ki bo služila za koordiniranje povezav na strežnik. Uro smo pripravili na posebnem čipu, a kaj kmalu ugotovili, da imamo zasedeno povezavo I2C za komunikacijo med glavnim čipom in uro, z komponento za TCP/IP komuniciranje. K sreči nam je uspelo pripraviti proste porte na drugi strani čipa in dodali smo še to podrobnost.

Da ne bi bilo vse tako preprosto, smo po pripravi tokovnega senzorja le tega skurili. Časa za ugotavljanje zakaj se je to zgodilo ni bilo, saj smo bili že blizu oddaje raziskovalne naloge, nam pa najpomembnejši del raziskovalne naloge ni deloval. Bili smo razočarani, a vedeli smo, da poti nazaj ni. Resnično velika zahvala gre tukaj profesorjem elektrotehnike, ki so nam uspeli dobiti identičen senzor

v manj kot štiriindvajsetih urah, sicer bi nanj bili primorani čakati vsaj teden dni in naš trud bi se izjalovil. Da nam ne bi šlo še kaj narobe s tem senzorjem, smo pod njihovim nadzorom in pomočjo le uspeli testirat senzor ter pripraviti testne meritve za dokončanje raziskovalne naloge. Iz te izkušnje smo se naučili, da moramo vedno najprej preverit ključne dele na katerem stoji projekt, da se bomo izognili težavam kot je bila ta in šele nato pripraviti vse ostalo.

Naloga dopušča tudi prostor za nadgrajevanje. Sami menimo, da bi bilo pametno najprej dodelati merilnik, in sicer senzor za napetost. S tem bi dosegli večjo natančnost, ki bi omogočila pot k uspešnemu trženju. Sicer pa kot, upajmo, da nekoč, bodoči programerji vemo, da je vzdrževanje programske kode ključnega pomena in bi bilo pametno dodelati merilnik tako, da bi bilo mogoče posodabljat njegovo programsko opremo kar preko spleta. Aplikacije bilo potrebno še nekoliko dodelati, s čimer bi omogočili prenos strežnika na Linuxov operacijski sistem, kar bi močno znižalo stroške programske opreme in bi sam sistem močno pocenilo. To je le nekaj začetnih idej za nadgradnjo, sicer je mogoče ta sistem nadgraditi še veliko bolj.

Za konec smo pripravili še pisni del raziskovalne naloge pri kateri smo potrdili oz. ovrgli hipoteze. Menimo, da so izpolnile naša predvidevanja, saj so bile tudi ovržene hipoteze zelo blizu potrditvi. Nad končnim izdelkom oz. pripravljenim prototipom smo zadovoljni in se v en glas strinjamo, da smo s to raziskovalno prekosili same sebe in pripravili lepo odskočno desko za bodoče raziskovalce, da našo raziskavo še dopolnijo ter dodelajo, morda pa ta ideja nekoč ugleda tudi luč na trgu. Ne izključujemo pa tudi možnosti, da bomo nekoč sami poskrbeli, da se bo to resnično zgodilo.

6 Viri in literatura

Spletni viri:

Sistem za avtomatsko beleženje porabe električne energije. [splet]. *EnKlikAnketa – najhitreje do podatkov* [11. mar. 2013; 18:10]. Dostopno na spletnem naslovu: https://www.1ka.si/a/23881.

ACS714. [splet]. *ACS714-Datasheet* [11. mar. 2013; 18:23]. Dostopno na spletnem naslovu: http://www.pololu.com/file/download/ACS714-Datasheet.pdf?file_id=0J196 .

SD CARD. [splet]. *Card Reader Board EB037-001* [11. mar. 2013; 18:25]. Dostopno na spletnem naslovu: http://www.matrixmultimedia.com/datasheets/EB037-30-1.pdf.

gLCD. [splet]. *Graphical LCD Board EB043-001* [11. mar. 2013; 18:25]. Dostopno na spletnem naslovu: http://www.matrixmultimedia.com/resources/files/datasheets/EB043-30-1.pdf.

TCP/IP. [splet]. *TCP/IP Board EB023-001* [11. mar. 2013; 18:25]. Dostopno na spletnem naslovu: http://www.matrixmultimedia.com/resources/files/datasheets/EB023-30-1.pdf.

Matrična tipkovnica. [splet]. *Keypad board datasheet EB014-00-1* [11. mar. 2013; 18:28]. Dostopno na spletnem naslovu: http://www.matrixmultimedia.com/datasheets/EB014-30-1.pdf.

Strežnik. [splet]. *Strežniška omrežja* [11. mar. 2013; 19:30]. Dostopno na spletnem naslovu: http://www.egradiva.net/drugo/omrezja/01_omrezja/05_datoteka.html.

Priročniki in učbeniki:

Price, J. *Mastering: C# Database Programming.* 1. Izd. California. SYBEX. 2002. (Zbirka: Mastering).

Price, J. in Gunderloy, M. *Mastering: Visual C# .NET*, 1. Izd. California. SYBEX. 2002. (Zbirka: Mastering).

Mrhar, P. *ASP – Active Server Pages*, 1. Izd. Šempeter pri Gorici. Flamingo Založba d.o.o. 2002.

Zahvala

Zahvalili bi se profesorju mag. Boštjanu Resinoviču, univ. dipl. inž. rač. inf. za podporo in pomoč pri izdelavi raziskovalne naloge, ko je bila ta potrebna. Z njegovo pomočjo nam je uspelo sprotno odpravljati napake in reševati težave, ki so nastajale pri programiranju.

Prav tako se želimo zahvaliti profesorjema Gregorju Kramerju, univ. dipl. inž. el. in Janku Holobarju, inž. el, za svetovanje pri izdelavi tokovnega senzorja.

Hvala tudi profesorju Gorazdu Brezniku, univ. dipl. inž. el., ki nam je bil pripravljen odstopiti prostor, da smo lahko v šolskih prostorih pripravljali raziskovalno nalogo in za ves čas, ki si ga je vzel za pomoč pri razvoju merilnika ter testiranju senzorja. Brez njegove pomoči nam ne bi uspelo pripraviti tokovnega senzorja do roka za oddajo raziskovalnih nalog.

Za konec bi se zahvalili tudi našim staršem, prijateljem, sošolcem in vsem ostalim, ki so nas pri pripravi raziskovalne naloge podpirali in nam kakorkoli pomagali.

Priloge

Anketni listi

Sistem za avtomatsko beleženje porabe električne energije

XSPOL - Spol:

◌ Moški
◌ Ženski

XSTAR2a4 - V katero starostno skupino spadate?

◌ do 20 let
◌ 21 - 40 let
◌ 41 - 60 let
◌ 61 let ali več

Q1 - V kakšni stanovanjski stavbi živite?

◌ enostanovanjska stavba
◌ dvo ali več stanovanjska stavba

Q2 - Ali spremljate porabo električne energije?

◌ Redno
◌ Občasno
◌ Ne spremljam

IF (3) Q2 = [1, 2]
Q3 - Ali popisujete porabo električne energije?

◌ Da
◌ Ne

IF (4) Q3 = [1]
Q4 - Kako pogosto popisujete porabo?

◌ 1x dnevno
◌ 1x tedensko
◌ 1x mesečbo
◌ Drugo:

IF (4) Q3 = [1]
Q5 - Ali rezultate tudi analizirate?

◌ Da
◌ Ne

IF (5) Q5 = [1]
Q6 - Kakšno tehniko pri tem uporabljate?

◌ Grafi
◌ Tabele
◌ Drugo:

IF (6) Q3 = [1] and Q5 = [1]
Q7 - Vam popisovanje in analiziranje predstavlja breme ter ogromno dela?

◌ Da
◌ Ne

Q8 - Bi uporabljali sistem, ki bi to opravljal za Vas?

◌ Da
◌ Ne

IF (7) Q8 = [1]
Q9 - Kako vrednotite pomembnost naštetih funkcij za sistem?

	Nepomembno	Pomembno	Zelo pomembno
Nastavitev svojih prikazov podatkov	◌	◌	◌
Sproten prikaz trenutne porabe	◌	◌	◌
Izvažanje podatkov	◌	◌	◌
Približen izračun stroškov	◌	◌	◌
Možnost ročnega urejanja podatkov	◌	◌	◌

IF (7) Q8 = [1]
Q10 - Kakšna naj bo gostota meritev?

◌ vsako uro
◌ vsakih 12 ur
◌ 1x dnevno
◌ Drugo:

IF (7) Q8 = [1]
Q11 - Omogočanje pregleda porabe za nazaj. Kako dolgo?
Obdobje v mesecih; od 0 do 36 mesecev

IF (7) Q8 = [1]
Q12 - Prosimo Vas, da nam posredujete tudi vaše predloge.

Rezultati ankete

XSPOL	Spol:				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (Moški)	38	61%	61%	61%
	2 (Ženski)	24	39%	39%	100%
Veljavni	Skupaj	62	100%	100%	
		Povprečje	1.4	Std. Odklon	0.5

XSTAR2a4	V katero starostno skupino spadate?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (do 20 let)	18	29%	30%	30%
	2 (21 - 40 let)	33	53%	55%	85%
	3 (41 - 60 let)	9	15%	15%	100%
	4 (61 let ali več)	0	0%	0%	100%
Veljavni	Skupaj	60	97%	100%	
		Povprečje	1.9	Std. Odklon	0.7

Q1	V kakšni stanovanjski stavbi živite?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (enostanovanjska stavba)	19	31%	40%	40%
	2 (dvo ali več stanovanjska stavba)	28	45%	60%	100%
Veljavni	Skupaj	47	76%	100%	
		Povprečje	1.6	Std. Odklon	0.5

Q2	Ali spremljate porabo električne energije?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (Redno)	24	39%	39%	39%
	2 (Občasno)	19	31%	31%	70%
	3 (Ne spremljam)	18	29%	30%	100%
Veljavni	Skupaj	61	98%	100%	
		Povprečje	1.9	Std. Odklon	0.8

Q3	Ali popisujete porabo električne energije?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (Da)	19	31%	44%	44%
	2 (Ne)	24	39%	56%	100%
Veljavni	Skupaj	43	69%	100%	
		Povprečje	1.6	Std. Odklon	0.5

Q4	Kako pogosto popisujete porabo?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (1x dnevno)	1	2%	5%	5%
	2 (1x tedensko)	4	6%	21%	26%
	3 (1x mesečbo)	13	21%	68%	95%
	4 (Drugo:)	1	2%	5%	100%
Veljavni	Skupaj	19	31%	100%	
		Povprečje	2.7	Std. Odklon	0.7

Q4_4_text	Q4 (Drugo:)				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	po potrebi oz. ob prejemu računov	1	2%	100%	100%
Veljavni	Skupaj	1	2%	100%	

Q5	Ali rezultate tudi analizirate?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (Da)	7	11%	37%	37%
	2 (Ne)	12	19%	63%	100%
Veljavni	Skupaj	19	31%	100%	
		Povprečje	1.6	Std. Odklon	0.5

Q6	Kakšno tehniko pri tem uporabljate?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (Grafi)	0	0%	0%	0%
	2 (Tabele)	5	8%	71%	71%
	3 (Drugo:)	2	3%	29%	100%
Veljavni	Skupaj	7	11%	100%	
		Povprečje	2.3	Std. Odklon	0.5

Q6_3_text	Q6 (Drugo:)				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	primerjalno	1	2%	100%	100%
Veljavni	Skupaj	1	2%	100%	

Q 7	Vam popisovanje in analiziranje predstavlja breme ter ogromno dela?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (Da)	3	5%	43%	43%
	2 (Ne)	4	6%	57%	100%
Veljavni	Skupaj	7	11%	100%	

Povprečje	1.6	Std. Odklon	0.5

Q 8	Bi uporabljali sistem, ki bi to opravljal za Vas?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (Da)	53	85%	87%	87%
	2 (Ne)	8	13%	13%	100%
Veljavni	Skupaj	61	98%	100%	

Povprečje	1.1	Std. Odklon	0.3

Q 9	Kako vrednotite pomembnost naštetih funkcij za sistem?								
	Podvprašanja	**Odgovori**							
		Nepomembno	Pomembno	Zelo pomembno	Skupaj				
Q 9a	Nastavitev svojih prikazov podatkov	5 (10%)	30 (63%)	13 (27%)	48 (100%)				
Q 9b	Sproten prikaz trenutne porabe	4 (9%)	14 (30%)	29 (62%)	47 (100%)				
Q 9c	Izvažanje podatkov	13 (27%)	24 (50%)	11 (23%)	48 (100%)				
Q 9d	Približen izračun stroškov	1 (2%)	12 (25%)	35 (73%)	48 (100%)				
Q 9e	Možnost ročnega urejanja podatkov	9 (19%)	25 (52%)	14 (29%)	48 (100%)				

Q10	Kakšna naj bo gostota meritev?				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	1 (vsako uro)	9	15%	19%	19%
	2 (vsakih 12 ur)	5	8%	10%	29%
	3 (1x dnevno)	32	52%	67%	96%
	4 (Drugo:)	2	3%	4%	100%
Veljavni	Skupaj	48	77%	100%	

Povprečje	2.6	Std. Odklon	0.8

Q10_4_text	Q10 (Drugo:)				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	zadnji teden vsako minuto, zadnji mesec vsako uro, za nazaj pa povprečje celega dneva	1	2%	50%	50%
	vsake 3 ure	1	2%	50%	100%
Veljavni	Skupaj	2	3%	100%	

Q11	Omogočanje pregleda porabe za nazaj. Kako dolgo?					
	Veljavno	**Št. enot**	**Povprečje**	**Std. Odklon**	**Minimum**	**Maksimum**
	48	62	20.1	23.37	1	36

Q12	Prosimo Vas, da nam posredujete tudi vaše predloge.				
	Odgovori	**Frekvenca**	**Odstotek**	**Veljavni**	**Kumulativa**
	menim, da je ta zadeva super, ker si lahko sproti preverjaš, koliko porabiš in imaš določen pregled nad tem. želim si, da bi bila ta zadeva tudi dostopna (finančno) vsem ljudem.	1	2%	50%	50%
	zanimam se za tak program in bi ga z veseljem uporabljala	1	2%	50%	100%
Veljavni	Skupaj	2	3%	100%	