1. Microcontrolere
   1. Generalități

Microcontrolerele sunt componente electronice care inglobeaza in ele majoritatea componentelor dintr-un calculator PC, bineinteles toate de mai mici dimensiuni.Astfel acestea au: unitate centrala, porturi, memorii, convertoare analog numerice(o parte din ele), etc.



Figura.1 Microcontrolerul

Fizic, microcontrolerul, ca orice componentă electronica este realizat pe o plăcuță de siliciu care, cu ajutorul radiației ultraviolet, este dopată în mod selectiv cu atomi donori sau acceptori, formîndu-se joncțiuni de tip P-N , astfel realizăndu-se elementele de bază – tranzistoarele, tot aici se realizează traseele de interconectare conform schemei electrice căt și punctele de conexiune cu mediul exterior.

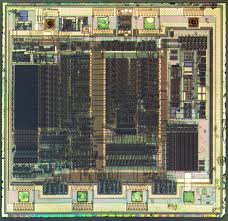


Figura. 1.2 Pastila de siliciu

Încapsularea pastilei de siliciu se realizează , în funcție de numărul de conexiuni electrice, de gradul de disipare termică și nu în ultimul rănd de gradul de integrare, utilizănd diferite tipuri de capsule, pornind de la cele mai vechi, folosite cu precădere pentru circuitele logice MSI și LSI, anume capsulele DIL (Eng. Dual In Line) pînă la capsulele modern BGA(Eng. Ball Grid Array) sau µBGA.

Diferiți producători, concur pe piața componentelor electronice , oferind un spectru larg de microcontrolere ce se disting între ele printr-un spectru larg de facilități cum ar fii porturi seriale și paralele, numărătoare, temporizatoare, convertoare analog-numerice, posibilități de extensie exterioară a capacității interne de stocare,etc …

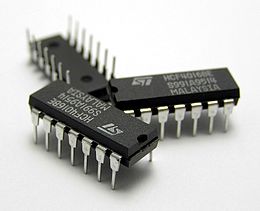


Figura.1.3 Capsule DIL 14 pin

Aceast mod de integrare a componentelor într-un microcontroller face posibilă abordarea unor probleme cu o complexitate ridicată, utilizănd un număr minim de componente electronice externe, crescănd pe deoparte fiabilitatea întregului system, numarul de ore de funcționare.

Arhitectura care stă la baza tuturor microcontrolerelor este cea Harvard, ele avănd un număr redus de instrucțiuni (RISC), majoritatea, executîndu-se într-un singur interval de ceas.

Sistemul, fiind unul cu logică programată, necesită existența și evoluția unui program, în memoria internă a microcontrolerului.

Distingem, în procesul de dezvoltare a unei aplicații bazată pe microcontrolere două etape

1. Etapa aferentă proiectării și implementării hardware – conexiuni electrice, dimensionare elemente electronice passive și active.
2. Etapa aferentă proiectării software – conceperea, scrierea, încărcarea și testarea programului

Astfel, proiectarea cu microcontrolere presupune atît cunoștințe din domeniul electronic dar și cunoștințe din domeniul informatic.

* 1. Familii de microcontrolere

Primul microprocessor integrat a fost conceput, proiectat și realizat de către firma de origine americană Intel, acesta purta numele de I4004, un processor cu o arhitectură pe 4 biți ce lucre la o frecvență de ceas de ordinal 100 de KHz, utilitatea găsită find în aplicații cum ar fi calculatoarele de buzunar sau de birou.

Pentru realizarea unei aplicații , era necesară adăugarea altor circuite externe de memorare, memorii ROM, RAM, circuite de multiplexare și demultiplexare, latch-uri , buffere , etc…

Primul microcontroller a fost realizat tot de către firma Intel, avănd la bază unitatea central 80C51.

Costurile mici, gradul de integrare, au dus la o adevărată explozie a proiectelor pe bază de microcontrolere.

Putem distinge, în funcție de arhitecturile lor , familiile de microcontrolere.

Astfel, funcție de lățimea magistralei de date, respective capacitatea regiștrilor interni

1. Microcontrolere pe 8 biți
2. Microcontrolere pe 16 biti
3. Microcontrolere pe 32 biți

Atmel, produce, bazat pe o arhitectură RISC, o întreagă serie de microcontrolere, avănd facilități și dispunînd de resurse, diferite, special concepute pentru anumite scopuri.

Astfel putem deosebi microcontrolere destinate industriei de automobile, aeronautică, medical, aplicații casnice, comunicații, etc…

Distingem familia de microcontrolere AVR, dezvoltată începînd cu anul 1996.

**Familia de bază AVR**

Caracteristicile definitorii , funcție de reprezentanții seriei dezvoltate de Atmel sunt

* **tinyAVR** 
  + 0.5–16 KB memoria program
  + 6–32-pini la capsulă
  + Un număr limitat de periferice
* **megaAVR**
  + 4–256 KB memorie program
  + 28–100-pini la capsulă
  + Set extins de instrucțiuni, optimizate pentru a fi executate într-un singur impuls de ceas, instrucțiuni speciale pentru manipularea unor zone mari de memorie.
  + Număr mare de periferice
* **XMEGA**
  + 16–384 memorie program
  + 44–64–100-pini la capsulă
  + Facilități extinse cum ar fi DMA, criptografie
* **Microcontrolere orientate pe aplicații** 
  + megaAVRs sunt microcontrolere din familia megaAVR ce dispun de controlere USB, LAN sau LCD.
* **FPSLIC (AVR ce include FPGA)**
  + [FPGA](https://en.wikipedia.org/wiki/Field-programmable_gate_array) cu 5000 pînă la 40000 de porți logice
  + SRAM – memoria de program
  + Frecvența de ceas mare, de pîn la 50 MHz



Figura.1.4 AtxMega128 în capsulă TQFP cu 100 pini

O altă familie de microcontrolere este cea produsă de către firma microchip, sub denumirea comercială PIC (*Peripheral Interface Controller).*

Modelele de început ale PIC aveau memorie numai pentru citire (ROM) sau EPROM programabil pentru stocarea programelor, unele având posibilitatea de a șterge memoria. Toate modelele actuale utilizează memoria flash pentru stocarea programelor, iar modelele mai noi permit PIC să se reprogrameze. Memoria programelor și memoria de date sunt separate. Memoria de date este de 8 biți, de 16 biți și, în cele mai recente modele, este pe 32 de biți. Programul de instrucțiuni variază în număr de biți de familie de PIC, și poate fi de 12, 14, 16 sau 24 de biți lungime. Setul de instrucțiuni variază de asemenea în funcție de model, cu cipuri mai puternice care adaugă instrucțiuni pentru funcțiile de procesare a semnalului digital.

Familiile de microcontrolere PIC

**PIC10 și PIC12**

* Memorie RAM și EPROM avînd capacități reduce de ordinal sutelor de octeți
* Arhitectură RISC pe 8 biți
* Un număr mic de resurse hardware

**PIC16**

* Capacitate sporită a memoriei (16KO)
* Arhitectură RISC pe 8 biți

### PIC17

* Relocarea memoriei registrului acumulator
* Protejarea acesului pentru memoria program (protecție la copier)
* Memorie externă extinsă
* Multiplicator hardware

### PIC18

**PIC24**

Scrierea aplicațiilor poate fi realizată în limbaj de ansamblare, indiferent de tipul microcontrolerului, această metodă este mai apropiată de nivelul fizic, fiind puternic dependent de arhitectura microcontrolerului.

Se obține în acest mod un cod mașină optimizat, ce ocupă în memoria de lucru un spațiu restrîns, asigurănd totodată o bună viteză de execuție a programului.

Metoda prezentată presupune un eford destul de mare din partea programatorului, distrăgînd atenția asupra problemei de rezolvat, programatorul va fii în cea mai mare parte atent la conceperea și alegerea unor instrucțiuni optime în vederea rezolvării problemei date.

La ora actuală, cele mai folosite sunt limbajele de nivel înalt, mult mai appropriate de limbajul natural, aceasta , grație capacitătilor mari de stocare în memoria internă și a frecvențelor ridicate de procesare a microcontrolerelor.

Mediile integrate ce sunt utilizate de către programatori, pun la dispoziția acestora instrumente grafice, intuitive, avansate, pentru funcțiile de editare , depanare, testare, și încărcare a programelor în sistemul tintă.

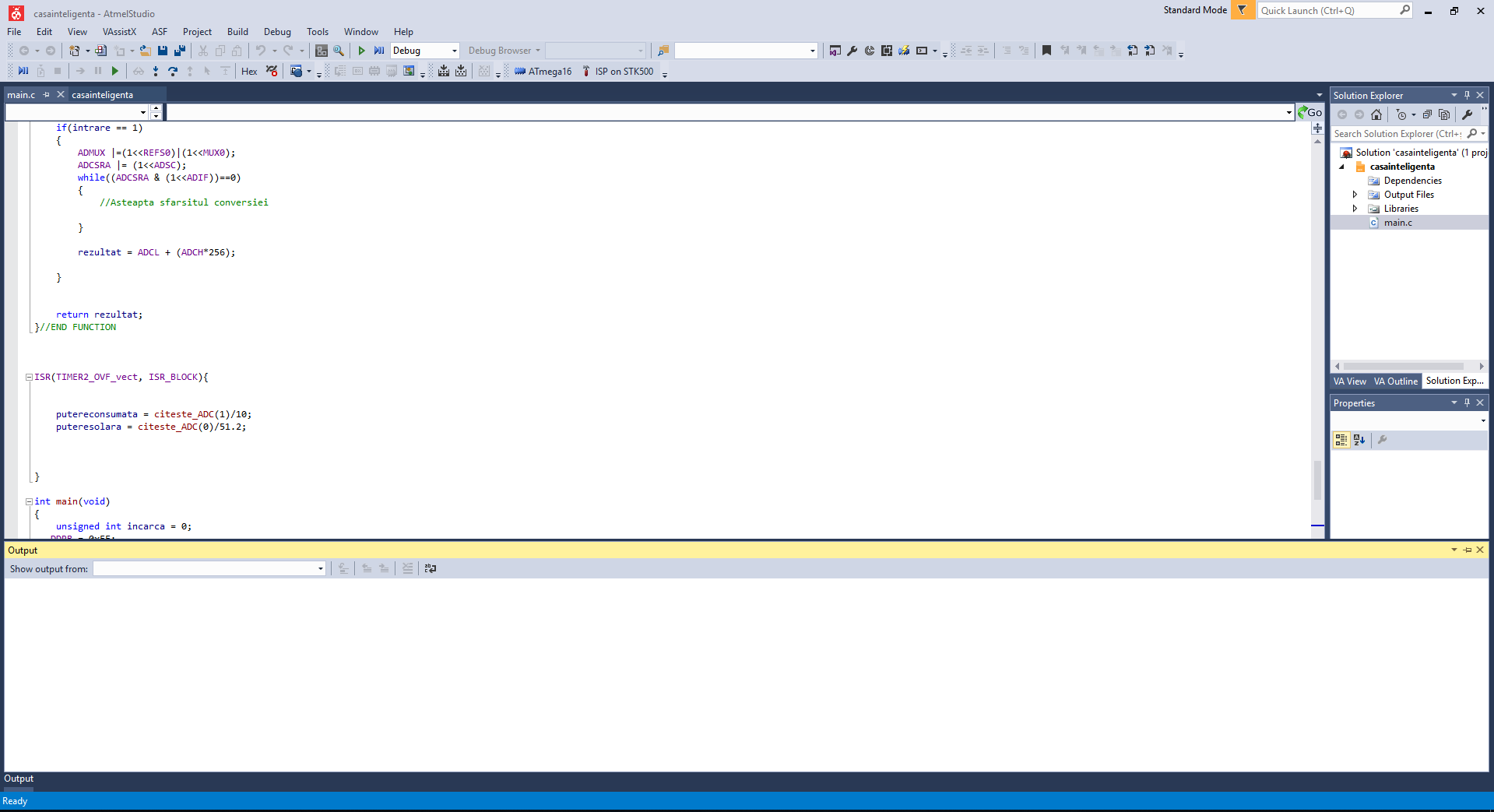


Figura.1.5 Mediu integrat de dezvoltare, Atmel Studio

Atmel studio, ajuns la versiunea 7.0 este un exemplu, a cărei interfață grafică este ilustrată în figura de mai sus. Este un mediu gratuity oferit de către producătorul de microcontrolere – Atmel.

Printer caracteristicile sale, putem enumera, ușurința in exploatare, configurarea facilă pentru diferite tipuri de microcontrolere și plăci de dezvoltare, un editor de cod ce oferă facilități de autocorectare, autosuggest.

Inițierea unei sesiuni de programare se poate face atît la nivel de proiect, caz în care în mod compact este creată o structură de directoare, structură ce conțin toate fișierele resursă necesare elaborării aplicației.

În general, orice aplicație software este dezvoltată etapizat, urmînd o succesiune de pași, standardizați.

1. Punerea problemei
2. Identificarea datelor de intrare și de iesire
3. Conceperea algoritmului – utilizănd diverse limbaje pseudocod
4. Scrierea programului
5. Verificarea programului sintactic și semantic
6. Compilarea programului
7. Testarea programului
8. Întreținerea programului

Atmel studio, facilitează, mai puțin descrierea algoritmului, realizarea tuturor pașilor ce trebuiesc executați în dezvoltarea software-ului.

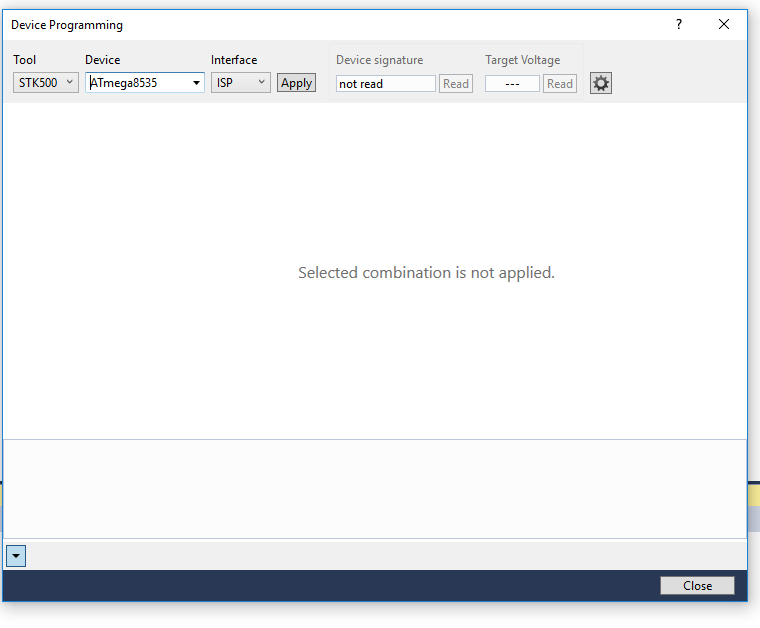


Figura.1.5 Uploadarea codului binary

Programarea sistemului țintă este realizată utilizănd placi de dezvoltare , în cazul nostru sa utilizat o placă realizată de către Atmel -STK500

Această placă oferă un support destul de mare, acomodînd un număr destul de sufficient de microcontrollere începănd de la tinyAVR pîna la megaAVR, posedînd un număr de 8 socluri de extensie.

Găsim aici , pentru testare, dispositive de ieșire, format dintr-un număr de 8 led-uri ce se pot conecta în mod preferential la unul din cele patru porturi paralele prin intermediul unor conectori de tip PINHEAD cu 10 pini dispuși linear pe două rînduri a căte 5 pini.

Regăsim si un număr de 8 pushbuton , ce se pot conecta ca și dispositive de intrare, prin conectori.

Programarea sistemului tintă, este realizată în patru moduri

* High-Voltage Programing
* Parallel High-Voltage Programing
* Serial High-Voltage Programing
* In-Sistem Programing (ISP)

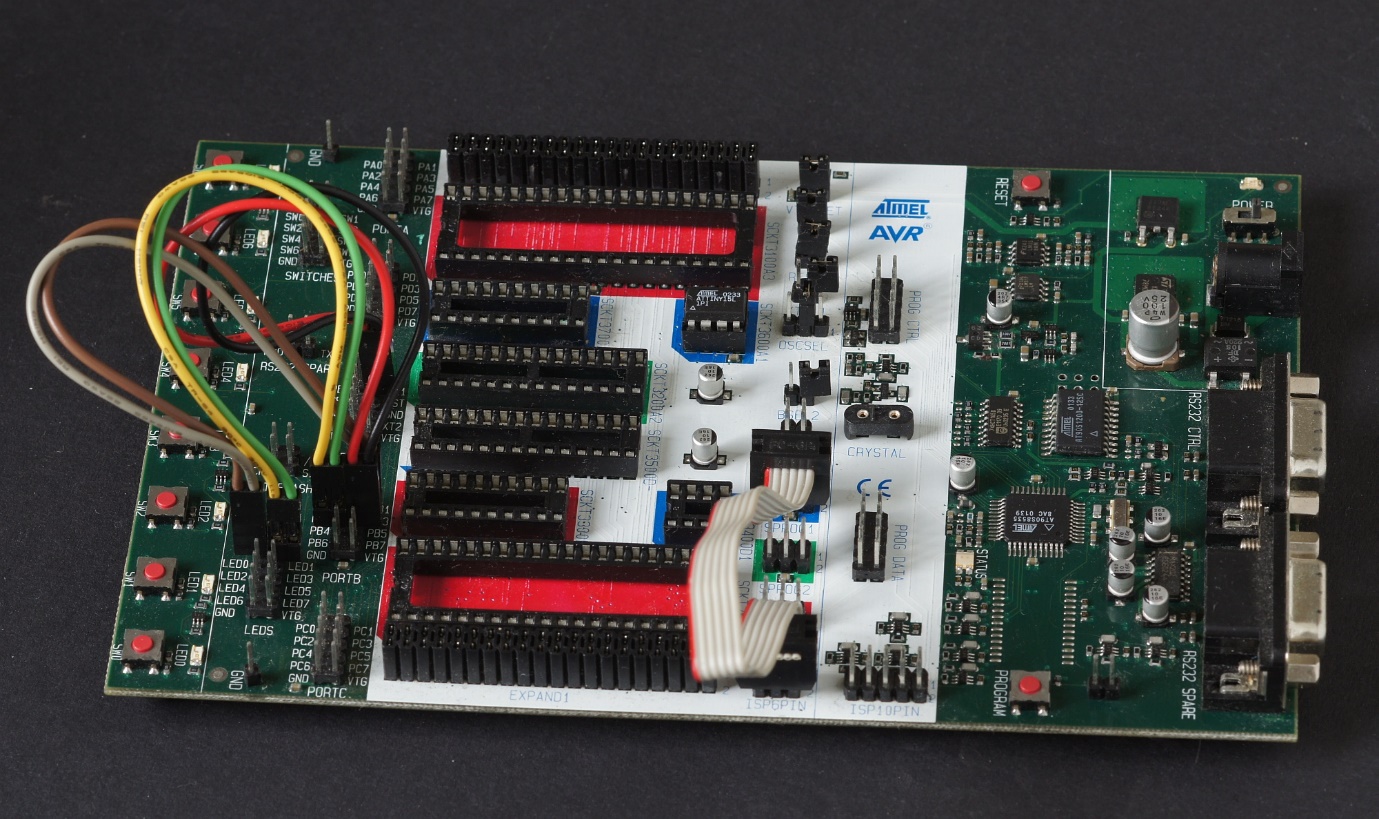


Figura. 1.6 STK500 – Development Board

Pentru realizarea proiectului au fost necesare parcurgerea următorilor pași , în ordinea specificată

* Lansarea aplicației Atmel Studio
* Crearea unui nou proiect
* Alegerea tipului de placă de dezvoltare – STK500
* Alegerea tipului de microcontroller (Atmega16)
* Scrierea codului sursă
* Compilarea codului sursă
* Încărcarea codului executabil în microcontroller
* Testare, eliminarea eventualelor erori.

1. Schema bloc , schema de principiu

În practica inginerească se pot distinge două metode pentru proiectarea unui sistem, metode ce au particularitășile lor.

Prima metodă poartă numele top-down , aici proiectarea sistemului începe prin abstractizarea și eliminarea detaliilor , întregul sistem este vazut ca o cutie neagră ce are un număr de intrări și de ieșiri, acestea determinănd funcția de transfer, funcție care, în funcție de complexitatea sistemului poate fi liniară sau neliniară.

SISTEM

INTRĂRI

IEȘIRI

Figura.2.1 Sistemul considerat ca o cutie neagră

***Abordarea descendentă (top-down)*** constă în a coborî, pe scara piramidei ierarhice până la bază, realizând totodată şi o analiză. Această viziune consideră că un anumit domeniu este compus din părţi corelate între ele şi cu legături cu exteriorul, fiind caracteristică pentru toate sistemele . Adepţii acestei abordări consideră că este mai bine să se creeze şi să se realizeze din start un sistem care să ţină cont de obiectivele planificate, abordată într-o manieră globală, decât să se încerce a se integra *a posteriori* subsisteme informatice independente.[ [www.icbe-ct.com/pacuraru/SIG\_cap2.doc](http://www.icbe-ct.com/pacuraru/SIG_cap2.doc)].

Astfel vom realiza o abordare , recursivă, în care întreg sistemul reprezintă o colecție de subsisteme ce interacționează atît între ele căt și cu mediul înconjurător, schimbănd informație, pînă ajungem la elemente simple constituente. Ca o paralelă, în informatică, acest model de abordare a unei probleme poartă denumirea Divide Et Impera.

NIVELUL DE VĂRF (TOP)

SUBSISTEM N

SUBSISTEM N-1

SUBSISTEM 2

SUBSISTEM 1

Figura.2.2 Proiectarea top-down

O a doua metodă utilizată în proiectare este cea bottom-up, aici se pornește de la sisteme mici, simple cunoscute din punct de vedere al funcționalitășii lor, realizănd cu ajutorul acestora, subansamble mai complexe pînă la atingerea scopului final, propus de problema ce trebuie rezolvată.

Această metodă de abordare a problemei este utilizată cu precădere la sisteme relativ simple , în situația în care cunoaștem foarte bine fiecare structură funcțională de bază.

Problema propusă în lucrarea de față cere găsirea unei soluții de eficientizare a costurilor energiei electrice consumate într-o locință.

Se știe, și vom argumenta, că prețul energiei electrice de pe piața liberă diferă pe intervalele orare dealungul unei zile – 24 de ore, putănd fii distinse diferite intervale orare, de zii și de noapte, unde prețul variază, nepaipunînd ăn calcul orele considerate de vărf, cănd prețul unui kwh este maxim.

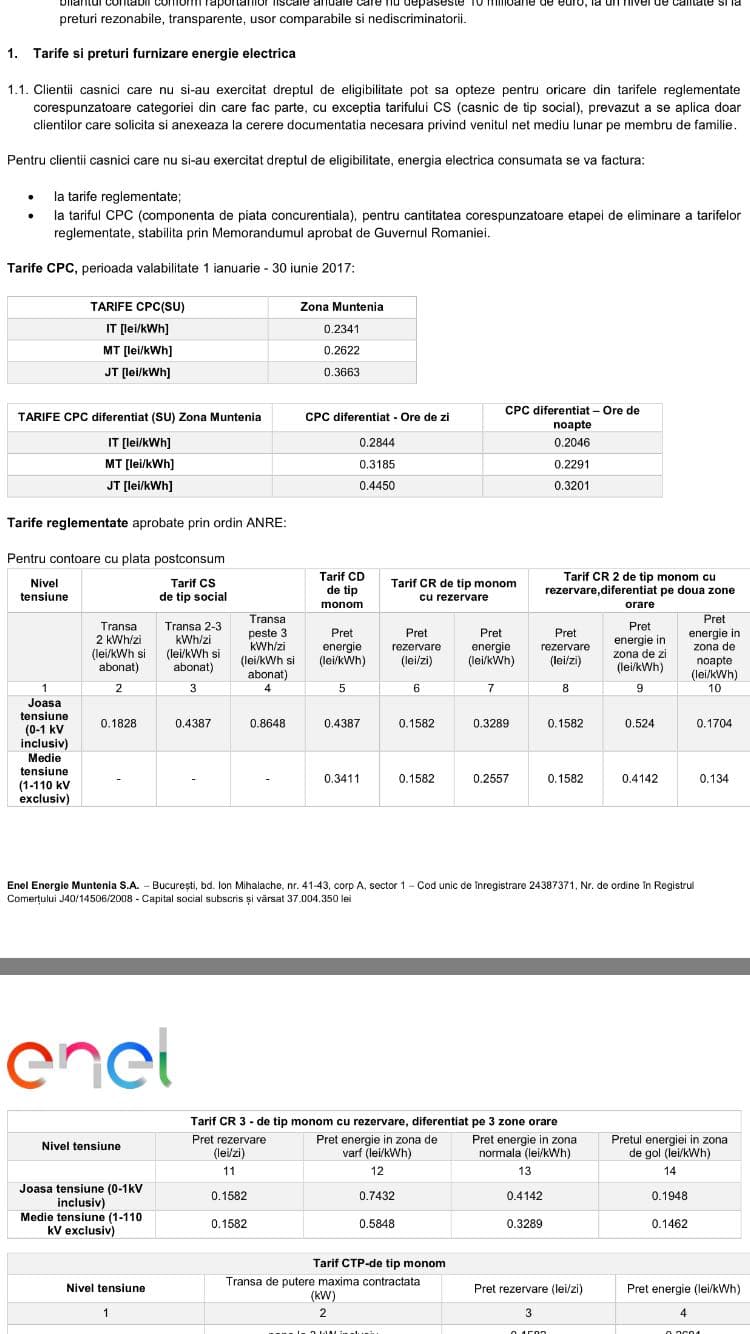


Figura.2.3 Tarife ENEL pentru energia electrică

Observăm, din imaginea de mai sus, că pentru un utilizator casnic conectat la rețeaua de joasă tensiune (JT) există două tarife diferențiate pentru perioada de zi și de noapte.

Astfel pentru perioada de zi prețul unui KWh este de 0.445RON iar pentru perioada de noapte, prețul unui KWh este de 0.320RON.

Saluți propusă în acest caz este de a utiliza pe timp de zi o sursă de energie alternativă, ecologică.

Putem folosii diferite surse alternative, energia eoliană, energia solară, energie termală, etc..

Dintre aceste tipuri cea mai la ăndemănă se dovedește a fii energia solară, fiind necesar un cost de investiție mult mai mic decăt în cazul turbinelor eoliene.

Nivelul de energie solară, în țara noastră se situează la valoarea de aproximativ 1kw pe metru patrat.

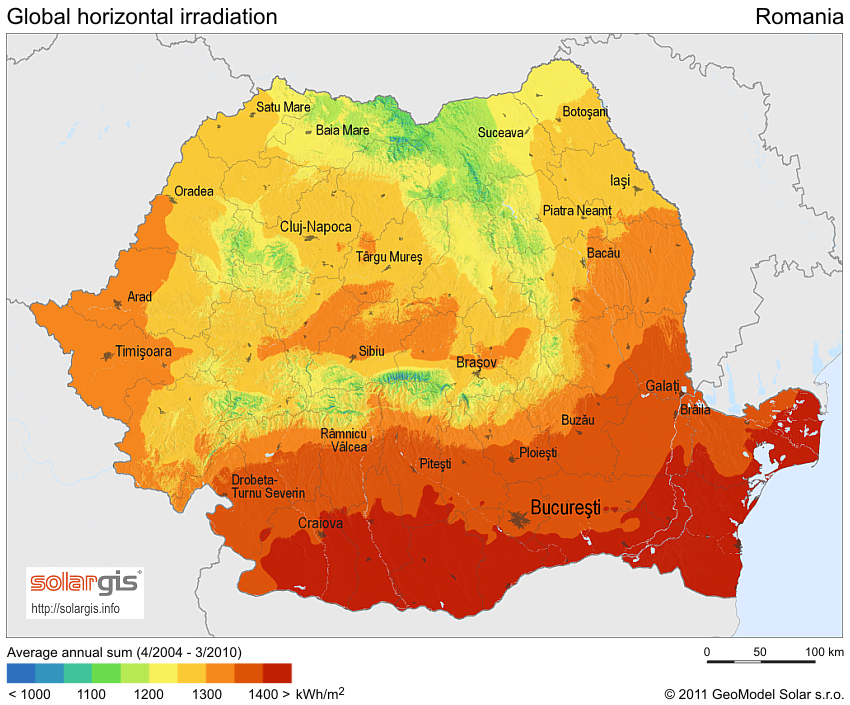


Figura.2.4 Harta radiației solare din Romănia [https://ro.wikipedia.org/wiki/Energie\_solara]

Observăm din harta prezentată în figura de mai sus că, țara noastră se bucură, pe o suprafață destul de vastă , de o cantitate suficientă de energie solară, suficientă, mai ales pe zonele de șes si deal, pentru un necesar zilnic de consum al unei gospodării, fără a implica costuri exagerat de mari.

Comutarea între cele două surse de energie se va face în mod automat prin temporizare, la nivel de prototip se vor scala corespunzător timpii de temporizare pentru a nu influența în mod defavorabil prezentarea proiectului datorită unor perioade de așteptare între comutări prea lungi.

Astfel sa ales ca întreg intervalul de 24 de ore să fie reprezentat în secunde, perioada de zi corespunzănd unei lungimi de 14 secunde iar cea de noapte de 10 secunde.

Pentru a realiza invarianța energiei furnizate la fluctuațiile de intensitate a luminii pe timp de zi (zii cu înorată cu averse de ploaie , lungimea scăzută a unei zile în perioada de iarnă) se va folosii un sistem de stocare a energie electrice convertite de panoul solar, cu ajutorul unui acumulator.

Avînd expuse datele problemei putem contura schema de principiu astfel

Panou solar

fotovoltaic

Acumulator tampon

Sursă de energie convențională

Sistem de comutare

Consumator

Logică de Acționare

Elemente afișare

Figura.2.5 Schema de principiu

1. **Descrierea aplicației**

Sistemul de afișare, realizează interfațarea dintre om (utilizator) și echipament, cu ajutorul lui fiind transmise informații despre

1. Tensiunea furnizată de panoul solar
2. Temporizarea pentru intervalul orar de zii respectiv de noapte
3. Timpul scurs pînă la expirarea intervalului curent
4. Tipul de energie utilizat (convențională sau solară)

Elementele de afișare utilizate sunt de tipul digit cu sapte segmente, acestea asigurînd o bună vizibilitate atît pe tim de zi căt și pe timp de noapte, au un consum de curent relativ mic (80mA) și de asemenea un preț de cost scăzut.



Figura. 3.1 Sistem de afișare compus din 3 digiți

Constructiv, un digit este realizat dintr-o capsulă de plastic ce înglobează un număr de 7 diode electroluminiscente – LED, montate într-o configurație specifică ce amintețte de cifra 8, după cum se poate observa și în figura de mai sus.

Ledurile au fiecare un ghid de unda luminoasă ce posedă caracteristici optice de difuzie foarte diferite pe cele trei axe (X,Y,Z), astfel încăt propagarea luminii se face foarte facil în mod transversal decăt în cel longitudinal.

Led-urile sunt conectate în paralel, fie la catod , conexiunde ce poartă denumirea de catod-comun, fie la anod, conexiune ce poartă denumirea de anod comun.

Anozii, respectiv catozii, pentru fiecare segment, sunt accesibili din exteriorul capsulei, grație unor pini de conexiune.

Digiții sunt produși pe diferite dimensiuni și culori în funcție de necesități, de distanța de amplasare față de operator, erc ...

Deasemenea, fiecare digit mai este prevăzut cu un LED pentru simbolul punctului zecimal, util la reprezentarea numerelor fracționale.

În ceea ce privește gama de culori, este predominant culoarea roșie și verde, mai rar întălnindu-se culori precum galben , albastru , alb.

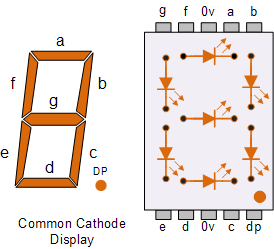


Figura 3.2. Schema electrică de conexiune

Segmentele digitului au denumiri consacrate în literatura tehnică de specialitate , respectiv , a se vedea și figura de mai sus a, b, c, d, e, f, g.

Astfel de exemplu, pentr a afișa cifra unu este necesar sa se activeze segmentele b și c, pentru cifra 8, se vor activa toate cele 7 segmente – a,b,c,d,e,f,g, cifra 4 va avea active segmentele, f,g,b,c , etc ...

Pentru a putea utiliza un afișaj cu 7 segmente, fie se folosesc circuite de decodoare BCD la 7 segmente , un exemplu fiind CD4511 , sau MMC4511, fie, cum este în cazul nostru, folosind un port al microcontrolerului.

Implementerea logicii de afișare se face pe baza următoarelor tabele de adevăr.

Pentru conexiunea catod comun

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **e** | **f** | **g** | **SIMBOL** |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 2 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 3 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 4 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 5 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 7 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |

Tabelul 1. Logica de activare a segmentelor pentru modul cato-comun

În situația în care conexiunea este de tipul anod comun , logica de activare a segmentelor , pentru obținerea unui anumit simbol este cea dată în tabelul 2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | simbol |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 3 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 5 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 6 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 7 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 9 |

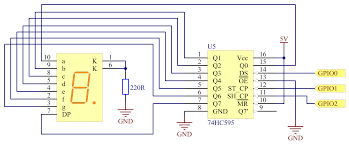
Tabelul 2. Logica negativă

Topologiile de conectare a sistemului de afișare la microcontroler sunt de două feluri.

Prima asigură o conexiune directă, fiecare segment ce apartine unui digit are propriul său circuit electric, fiind conectat la căte un pin din portul microcontrolerului.

Această metodă este foarte neeconomică, necesitănd un număr mare de circuite electrice, care, spre exemplu în cazul nostru, pentru un afișaj de 3 digiți , avem nevoie de 21 de circuite electrice, deci trei porturi din totalul de patru ale microcontroller-ului vor fii ocupate doar de către un sistem de afișare. Situația se prezintă și mai dramatic dacă avem nevoie de mai multe sisteme de afișare.

Schematic, legăturile electrice se realizează conform figurii de mai jos



Aici observăm situația în care se utilizează un driver – decodificator din cod zecimal codificat binar în cod 7 segmente.

Dimensionarea rezistorilor de limitare , ce vor fi amplasati pe fiecare segment, se va realiza prin aplicarea legii lui ohm pentru un circuit simplu, anume

Unde, în cazul nostru = 2.2volți, =0.08A, =5volți

R = 200 ohm

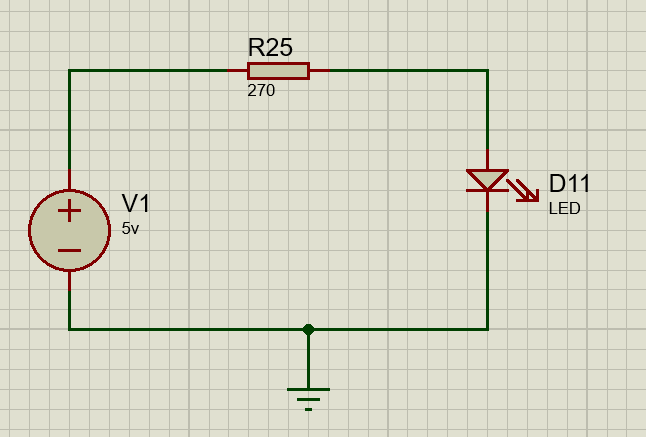


Figura. 3.3 Circuit echivalent de calcul

Conectarea sistemului de afișare , la microcontroller se face conform schemei de mai jos.

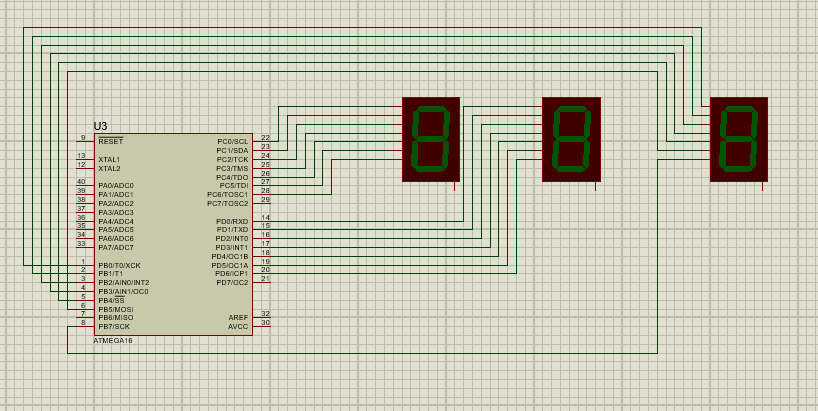


Figura. 3.4. Conectarea digiților la microcontroller

O a doua metodă , are la bază o decoperire veche ce fructifică deficențele simțului nostru vizual, mai exact, faptul ca o imagine persistă o fracțiune de timp pe retina ochiului.

Astfel , succesiuni rapide ale unor imagini distincte, vor crea senzația de imagine compactă, întreagă.

Pe acest principiu se bazează și afișarea cu multiplexare prin diviziune în timp, anume , cifra corespunzătoare unui rang din număr, va fi afișată un interval scurt de timp, urmănd a fi afișată cifra corespunzătoare urmaătorului rang din număr , totul repetăndu-se în mod ciclic.

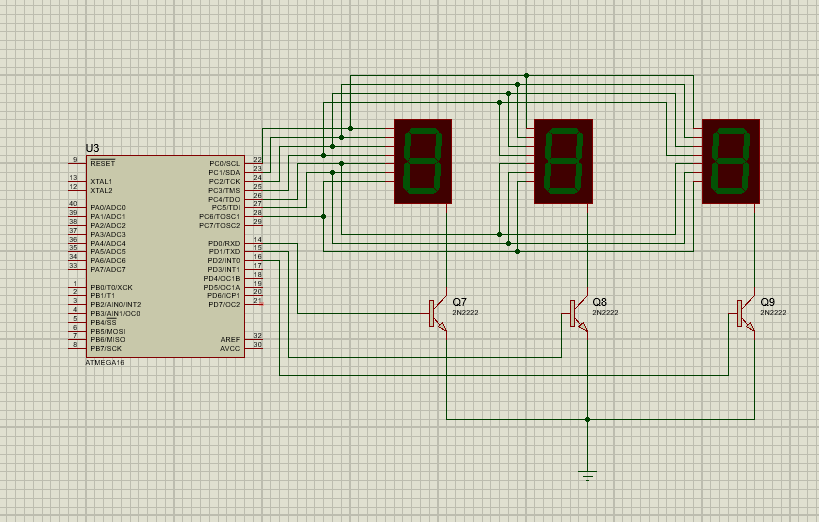


Figura.3.5. Conectarea în mod multiplexat a digiților.

Avantajul acestui tip de conexiune este dat de numărul mic de conexiuni, care, după cum se poate observa din figura de mai sus este , pentru trei digiți, în număr de zece conexiuni.

Legătura este simplă, segmenții fiecărui digit fiind legați în mod paralel, astfel, segmentul A corespunzător digitului unităților este conectat cu segmentul A ce corespunde digitului zecilor și mai apoi este conectat cu segmentul A al digitului pentru simbolizarea sutelor, și așa pînă la epuizarea segmentelor.

Comutarea digiților se face cu ajutorul unor tranzistori bipolari conectați fie în conexiune emitor – comun pentru digiții cu catod – comun, fie în mod colector-comun, pentru digiții cu anod-comun.

Logica de acționare, este reprezentată de însuși microcontroller, acesta coordonează întreg sistemul , pe baza evoluției unui program, conceput special pentru a configura și a controla întregul sistem.

Alegerea tipului de microcontroler se va face ăn funcție de necesarul de resurse utilizate, respectiv, porturi paralele, convertoare analog-numerice, temporizatoare.

Analizănd datele problemei propuse, observăm că avem nevoie de o intrare în convertorul ADC, necesară pentru monitorizarea tensiunii pe panoul fotovoltaic.

Pentu afișarea tensiunii pe panou , cu ajutorul afișajului cu trei digiți , căt și pentru monitorizarea timpului de comutare, avem nevoie de 13 iesiri, 7 pentru segmente si 6 pentru comanda catozilor.

O ieșire pentru comanda comutatorului de schimbare a sursei de alimentare, alternativă sau convenșională

Două ieșiri pentru semnalizarea stării curente a sursei de alimentare

Totalizănd , avem nevoie de un circuit care să aisigure un minim de 16 ieșiri digitale, respectiv o intrare analogică.

Atmega16, este un microcontroler din seria AVR, ce dispune congorm fișei producătorului (Atmel) de un număr de 4 porturi a 8 biți, 8 intrări pentru convertorul analog-numeric, temporizatoare pe 8 respectiv 16 biți.

Aceste resurse fiind suficiente pentru atingerea scopului problemei propuse.În Anexa 1 putem observa schema bloc a microcontrolerului, iar în Anexa2 diagrama conexiunilor la pinii capsulei.

Avănd în vedere faptul că, conform foii de catalog a producătorului , fiecare port poate fi configurat în mod separat și la nivel de pin, atît ca port de intrare căt și ca port de ieșire, realizarea conexiunilor între diversele subsisteme este destul de flexibilă fiind lăsată la latitudinea proiectantului, alegerea fiind una pseudo aleatorie.

Totuși am ales portul PORTD – pentru sistemul de afișare , iar portul PORTC pentru comande catozilor digiților.

Singura constrîngere este pentru intrările convertorului analog-digital, acestea ocupănd cei 8 pini ai portului PORTA, desigur, dacă nu sunt necesare toate intrările analogice, cele rămase pot fii utilizate ca intrari respectiv iesiri digitale la nivel de pin, printr-o configurare software adecvată.

În continuare , vom centraliza într-un tabel atăt porturile căt și dispozitivele periferice conectate, acesta fiind primul pas spre realizarea schemei electrice de principiu.

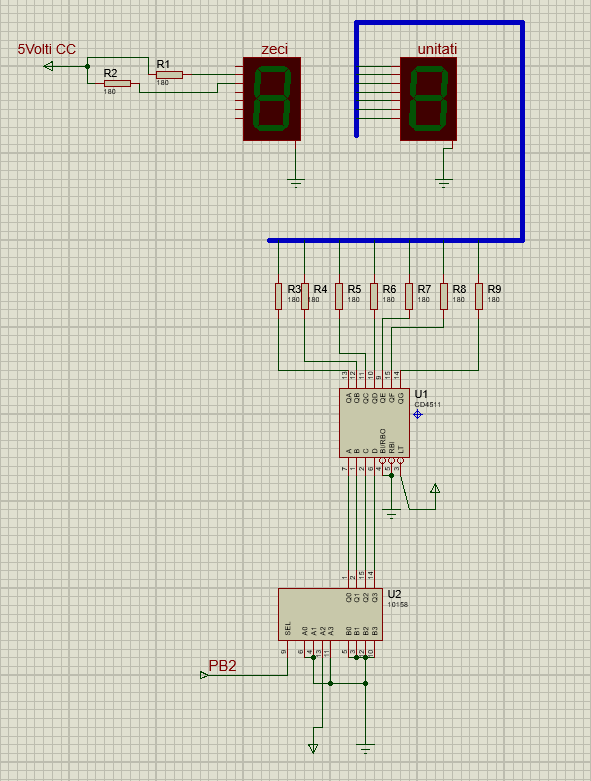
Pentru cele două, sisteme de afișare, conectarea se va realiza astfel

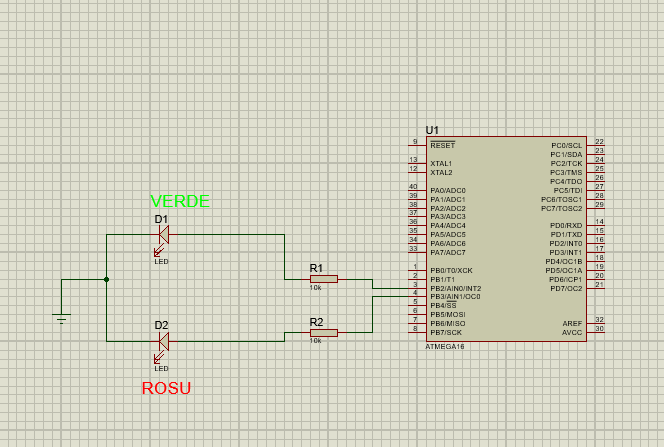
|  |  |
| --- | --- |
| PORTD | SEGMENT |
| PD0 | a |
| PD1 | b |
| PD2 | c |
| PD3 | d |
| PD4 | e |
| PD5 | f |
| PD6 | g |

Pentru realizarea comenzii digiților se vor asigna porturile conform tabelului de mai jos

|  |  |  |
| --- | --- | --- |
| PORTC | Rang digit | afisaj |
| PC0 | unități | Nivel tensiune panou |
| PC1 | zeci | Nivel tensiune panou |
| PC2 | sute | Nivel tensiune panou |
| PC3 | unități | Perioadă temporizare |
| PC4 | zeci | Perioadă temporizare |
| PC5 | sute | Perioadă temporizare |

Pentru afișarea timpului de temporizare atribuit fiecărui interval orar se va utiliza un sistem de doi digiți, conectați în logică cablată și un decodor BCD – 7segmente din seria CD4511, conform schemei electrice de mai jos



Conectarea led-urilor de semnalizare a stării în care se află sistemul se va realiza astfel

Astfel vom asigna, tot în mod arbritrar porturile PB2 respectiv PB3

|  |  |
| --- | --- |
| PORTB | LED |
| PB2 | Verde – sursă convențională |
| PB3 | Roșu – sursă alternativă (solară) |

În situațiile de urgență cînd este necesară deconectarea panoului fotovoltaic, și al acumulatorului, pentru operațiile periodice de întreținere și mentenanță, sa prevăzut o modalitate de comutare în mod forțat, cu ajutorul unui buton electric cu reținere, exclusiv pe sursa electrică convențională.

|  |  |
| --- | --- |
| PORTD | BUTON |
| PD7 | REȚEA |

Singurele elemente ce mai rămîn de conectat sunt, circuitul RC pentru resetarea la punerea sub tensiune a microcontrolerului, sursa de alimentare, sursă care este realizată în modul monolitic, utilizănd un circuit liniar de stabilizare a tensiunii la valoarea nominală de 5 volți, L7805

Elementele de comutare a surselor de alimentare sunt realizate, avănd în vedere perioada mare de comutare, cu relee electrice, comandate cu ajutorul unor tranzistoare bipolare de medie putere

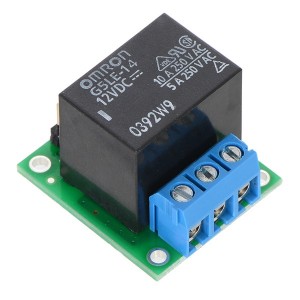


Figura. 3.6 Releul[http://www.hobbymarket.ro/module-atasabile/modul-1-releu-5v]

Releul electric este un element de circuit electromecanic, realizat dintr-un electromagnet și una sau mai multe perechi de contacte, acționate printr-o părghie mecanică.

La trecerea unui curent electric prin bobina electromagnetului, se produce un cîmp magnetic ce va atrace unul din capetele pîrghiei mecanice de acționare a contactelor electrice, acestea modificăndu-si starea.

În funcție de poziția inițială a contactelor electrice , cănd releul nu este anclanșat, distingem relee cu contacte aflate pe poziție normal-deschise (situație în care circuitul electric este întrerupt), respectiv relee cu contacte, normal închise (situație în care circuitul este continuu).

Acționarea releelor, datorită sistemelor mecanice , se face la frecvențe mici de dute pîna la mii de herți, ceea ce putem concretiza ca bobina electromagnetului, raprtată la inductanța sa , functionează în curent continuu.

Astfel la dimensionarea elementelor de circuit, este utilizată valoarea rezistenței electrice a bobinei, valoare specificată și în foile de catalog ale producătorilor.

Tensiunile de anclanșare pot fi de 5, 6, 12, 24 volți.

Tensiunile suportate de contactele electrice pot fii de 110 , 240 volți iar intensitățile curenților ce străbat aceste contacte pot fi de 10 – 100 Amperi

Schema electrică de principiu este următoarea

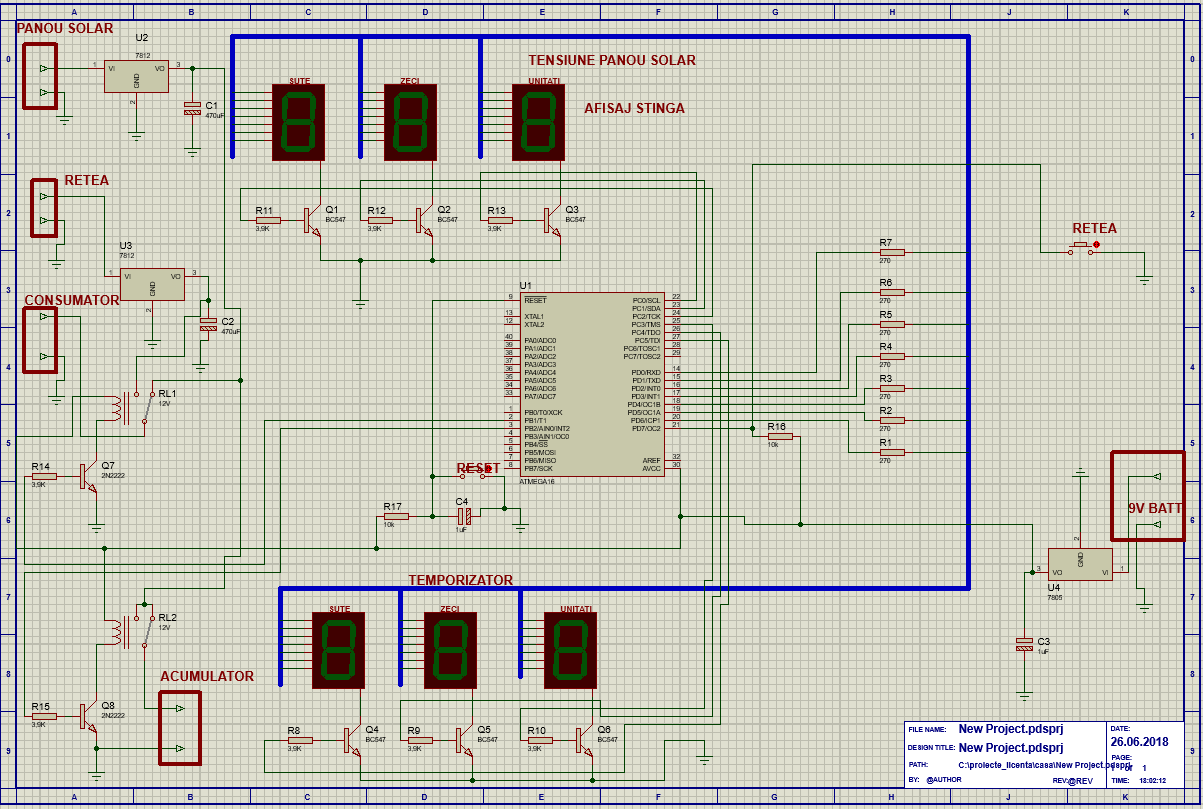


Figura.3.7 Schema electrică de principiu

Consumatorul este reprezentat de un sistem de iluminare de 12 volți, practic un LED ultrabright, ce va fii alimentat în mod automat o cu ajutorul releului RL1 o perioadă de 14 secunde , pe timpul de zi , din sursa solară, respectiv o perioadă de 10 secunde pe timpul de noapte din sursa convențională, în cazul nostru , un alimentator electric stabilizat , cu tensiunea nominală de 12 vcc și un curent maxim de 2 amperi, conectat la planșetă cu ajutorul conectorului RETEA.

Bateria de acumulatori de 12 volți, este un acumulator capsulat de tipul VRLA battery, avănd un curent de 1.2 Amperi.

Încărcarea acestuia se va face din panoul solar, printr-un stabilizator de tensiune de 12 volti de tipul L7812

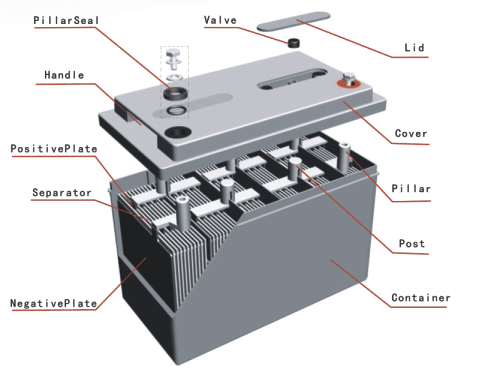
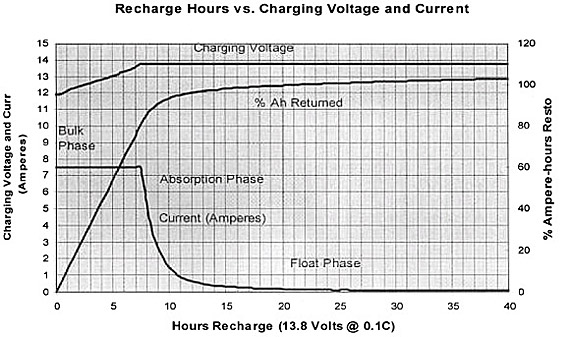
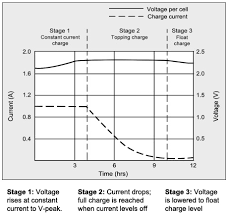


Figura.3.8 VRLA battery – Valve Regulated Lead Acid Battery

Caracteistica de încărcare a acumulatorului este dată de curba de încărcare , ce se poate vedea în figura de mai jos



Observăm în primă fază o crestere liniară a tensiunii în primele 8 ore de încărcare cu un curent constant de ăncărcare , a cărei curbă este paralelă cu axa timpului, urmată de o scădere expnențială a curentului de încărcare pînă la valoarea de 0.[ http://batteryuniversity.com/learn/article/charging\_the\_lead\_acid\_battery]



1. **Modul de funcționare**

Imediat la punerea sistemului sub tensiune, cu ajutorul conectării sursei auxiliare de 9 volti prin cuplarea la conectorul simbolizat pe schema cu 9VBATT, acesta pornește în modul de conectare a consumatorului pe sursa de energie solară, lucru semnalizat prin indicatorul optic LED de culoare roșie.

Afișajul destinat urmăririi timpului de temporizare va începe contorizarea numărului de secunde cît sistemul sta pe energie regenerabila, respectiv 14 secunde.

La expirarea acestui timp, afișajul TEMPORIZARE își va începe de l 0 numărătoarea, contorizănd timpul scurs de această data , cînd sistemul se află pe alimentare cu energie convențională, lucru de asemenea semnalizat cu ajutorul LED-ului de culoare verde.

În permanență se poate monitoriza tensiunea electrică la bornele panoului solar, valoarea acesteia, exprimată în volți fiind afișată pe displayul notat în schema electrică TENSIUNE PANOU SOLAR.

În situația de defectare a sistemului de alimentare cu energie solară, sau în perioadele cănd se realizează mentenanța acestuia, în mod manual și independent de deciza sistemului, se poate trece forțat pe alimentarea din sursa electrică convențională, la simpla apăsare a butonului notat RETEA.

Un al trei-lea display, va afișa alternativ perioada maximă cît va sta sistemul pe durata de zii respectiv pe durata de noapte, respectiv valoarea numerica 14 și 10 secunde.

1. Implementarea softului

Pentru funcționalitate, sistemele ce au logica programată au nevoie pe lînga componenta hardware și o componentă software, aceasta realizănd funcții de bază de configurare , control , implementănd aici , cu ajutorul programării procedurale, funcții ce realizează implementarea algoritmilor de comandă și control.

Avînd în vedere facilitățile puse la dispoziție de către microcontroler, viteza de lucru, capacitatea de memorie, putem spune că dezvoltarea aplicației software se poate realiza utilizănd un limbaj de nivel înalt care, față de limbajul de ansamblare, este mai apropiat de limbajul natural, sintaxa lui , utilizănd expresii formate din cuvinte, deseori rupte din limba engleză.

Librăriile ce sunt puse la dispoziție, acoperă o gamă largă de nevoi, punînd la dispoziție programatorului un set puternic de funcții.

Dezvoltarea aplicației a fost făcută utilizănd metoda spirală

Modelul spirală Modelul spirală este un exemplu bine cunoscut de metodologie a ingineriei programării. Acest model încearcă să rezolve problemele modelului în cascadă, păstrând avantajele acestuia: planificare, faze bine definite, produse intermediare. În locul reprezentării procesului software ca o secvenţă de activităţi, procesul este reprezentat în acest caz sub forma unei spirale. Fiecare buclă a spiralei reprezintă o fază a procesului software. Astfel, bucla cea mai din centru corespunde activităţii de stabilire a fezabilităţii sistemului, următoarea buclă corespunde definirii cerinţelor, următoarea proiectării sistemului şi aşa mai departe.[ <http://andrei.clubcisco.ro/cursuri/f/f-sym/4idp/1_Etapele_dezvoltarii_doc.pdf>]

Funcțiile utilizate în aplicație sunt pe deoparte destinate sistemului de afișare a datelor, inișializare a unor dispozitive hardware, convertoare, temporizatoare, funcții API de tratare a întreruperilor și bineînțeles funcția main.

void afisare\_LCD(unsigned int numar1, unsigned int numar2, unsigned int digit)

Reprezintă funcția ce realizează afișarea datelor numerice pe elementele cu 7 segmente, parametrii ce sunt transmiși funcției sunt chiar valorile ce trebuiesc afișate, respectiv număr1 va trimite valoarea tensiunii la bornele panoului solar, număr2, valoarea timpului scurs cînd sistemul se află în una din stările de comutare, iar digit, numărul curent al digitului care este activat în sistem.

void initializare\_ADC()

{

ADMUX = 0;

ADMUX |= (1<<REFS0);//Vezi pagina 211 din datasheet Atmega16 5V pe AVCC AREF cu capacitate la masa

ADCSRA |= (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);//Enable ADC si stabilire factor divizare frecventa core cu 128

}//END FUNCTION

Funcția de mai sus, realizează operațiunea de inițializare a convertorului analog-digital, cît și pornirea acestuia în modul free running mode.

Ca sursă de referință este utilizată sursa internă stabilizată cu valoarea de 5 volți ceea ce ănseamnă că, dacă aplicăm o tensiune de 5 volți la intrarea ADC, valoarea numerică furnizată de acesta va fi de 1023.

unsigned int citeste\_ADC(int intrare)

{

unsigned int rezultat;

ADMUX = 0;

ADMUX = (1<<REFS0);

if(intrare == 0)

{

ADMUX |=(1<<REFS0);

ADCSRA |= (1<<ADSC);

while((ADCSRA & (1<<ADIF))==0)

{

//Asteapta sfarsitul conversiei

}

rezultat = ADCL + (ADCH\*256);

}

if(intrare == 1)

{

ADMUX |=(1<<REFS0)|(1<<MUX0);

ADCSRA |= (1<<ADSC);

while((ADCSRA & (1<<ADIF))==0)

{

//Asteapta sfarsitul conversiei

}

rezultat = ADCL + (ADCH\*256);

}

return rezultat;

}//END FUNCTION

Această funcție este concepută astfel încăt să întoarcă valoare citita pe unul dintre canalele convertorului ADC

Ea primește ca argument numărul canalului de conversie, o valoare între 0 și 7, și întoarce valoarea rezultatului conversiei pentru canalul selectat.

Penru o mai bună flexibilitate și o implementare mai ușoară fără probleme, sa implementat un mod de programare multithread, firele de execuție paralele, executîndu-se la apariția unor evenimente de tip intrerupere , evenimente generate în principal de depășirea capacității de numărare a numărătoarelor sincrone, atasate temporizatoarelor de 8 sau 16 biți.

Astfel, la apariția evenimentelor mai sus menționate se vor apela în mod automat funcțiile de tip ISR (Interrupt Service Routine) corespunzătoare evenimentului selectat.

Există un fir de execuție separat, pentru sistemul de afișare a datelor implementat astfel

ISR(TIMER0\_OVF\_vect){

afisare\_LCD(puteresolara, tick, baleiaj);

baleiaj = baleiaj + 1;

if(baleiaj == 6){baleiaj = 0;}

}

Un fir de execuție , tot separat, ce realizează citirea portului ADC si scalarea corespunzătoare a valorii citite, pe baza faptului de liniaritate.

ISR(TIMER2\_OVF\_vect, ISR\_BLOCK){

puteresolara = citeste\_ADC(0)/51.2;

}

Unde coeficientul de 51.2 a rezultat dintr-un calcul de calibrare a ADC, cunoscănd faptul că acesta este liniar, ieșirea numerică variază ăn domeniul 0 – 1023, ce corespunde unei intrări analogice ce variază ăntre 0 – 5 Volți, rezultă că pentru a avea un cap de scală a instrumentului de măsură situat la valoarea de 20 de volți trebuie determinată factorul de scalare astfel

= 51,2

Deasemenea sa realizat și dimensionarea unui divizor electric rezistiv, astfel încăt pentru o tensiune maximă furnizată de panoul solar, capătul de scară, să avem la intrarea convertorului digital maximum 5 volți

ISR(TIMER1\_OVF\_vect){

if(tick<=15){if((PIND&(1<<PD7))==0){

tick = 0;

PORTB |=(1<<PB1);//Releu pe conventional

PORTA |=(1<<PA2);

PORTA &=~(1<<PA3);}

tick = tick + 1;

}

else{

tick = 0;

if((PIND&(1<<PD7))==0){

PORTB &=~(1<<PB1);//Releu pe solar

PORTA |=(1<<PA3);

PORTA &=~(1<<PA2);}

else{

if(ciclu == 0){//implementeaza un toggle folosind variabila ciclu

ciclu = 1;

PORTB |=(1<<PB1);//Releu pe conventional

PORTA |=(1<<PA2);

PORTA &=~(1<<PA3);

}

else {

ciclu = 0;

PORTB &=~(1<<PB1);//Aici este pe solar

PORTA |=(1<<PA3);

PORTA &=~(1<<PA2);

}

}

}

}

În această ultimă funcție de tratare a întreruperii generate de către TIMER1, este implementată logica de comutare a releului, RL1 căt și semnalizarea sursei de alimentare utilizate cu ajutorul ledurilor.

Tot aici se realizează și testarea apăsării butonului RETEA cu instrucțiunea de decizie

if((PIND&(1<<PD7))==0)

Funcția main, necesară oricărui program scris în limbajul C, va conține instrucțiuni de configurare a porturilor, instrucțiuni de activare a sistemului global de întreruperi

Căt si configurarea și activarea circuitelor de temporizare, ea arătînd astfel

int main(void)

{

unsigned int incarca = 0;

DDRB = 0xFF;

PORTB = 0;

DDRC = 0xFF;

DDRD = 0xFF;

PORTC = 0;

PORTD = 0;

sei();//Activare sistem intrerupere globala

DDRA |= (1<<PA2)|(1<<PA3);//Leduri pentru semnalizarea sursei de putere utilizate, PA2 - LED ROSU - SOLAR, PA3 - LED VERDE - RETEA//

DDRD &= ~(1<<PD7);//Buton trecere fortata pe energie electrica normala

initializare\_ADC();//Apel funcție prin care se activează convertorul ADC

TIMSK |= (1<<TOIE0)|(1<<TOIE2)|(1<<TOIE1);//Stabilirea sursei de întrerupere aferente timerelor

//TIMER0 , TIMER1, TIMER2

TCCR0 = 0;

//Stabilirea frecvenței de numărare a timerelor prin divizarea frecvenței de ceas a sistemului

//cu un factor de divizare, denumit prescaler

TCCR0 |= (1<<CS01);//TIMER 0 PENTRU AFISAREA PE DISPLAY

//CS02 CS01 CS00 sursa

//0 0 0 fara sursa

//0 0 1 CLK

//0 1 0 CLK/8

//0 1 1 CLK/64

//1 0 0 CLK/256

//1 0 1 CLK/1024

TCCR2 = 0;

TCCR2 |= (1<<CS21)|(1<<CS20)|(1<<CS22);//TIMER 2 PENTRU CITIREA ADC-ului

//CS22 CS21 CS20 sursa

//0 0 0 fara sursa

//0 0 1 CLK

//0 1 0 CLK/8

//0 1 1 CLK/32

//1 0 0 CLK/64

//1 0 1 CLK/128

//1 1 0 CLK/256

//1 1 1 CLK/1024

TCCR1A = 0;

TCCR1B |= (1<<CS11)|(1<<CS10);//1Hz

while (1)

{

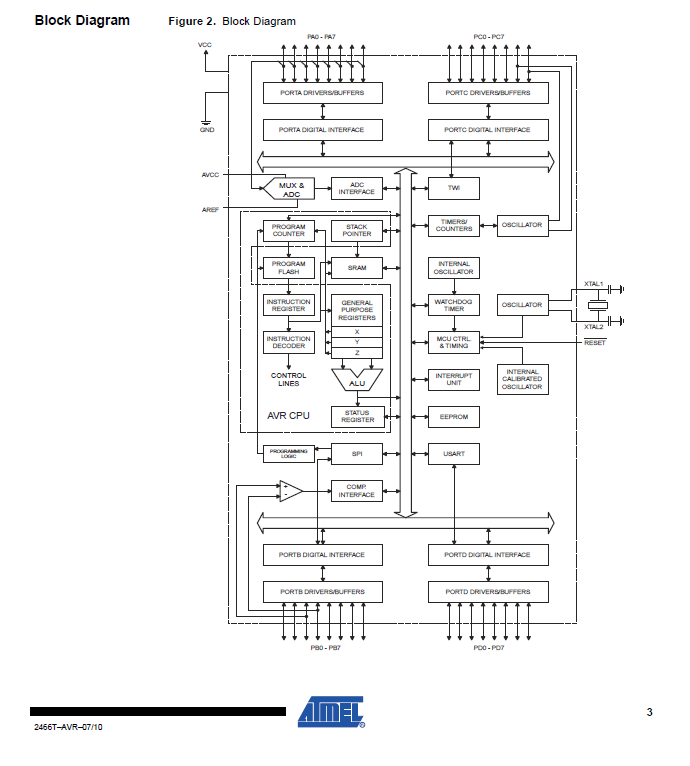
//Aceasta este o buclă infinită ce este necesară pentru ca program counterul sa nu-si //depășească cumva capacitate de adresare, reluînd în acest fel execuția aplicației de la adresa 0

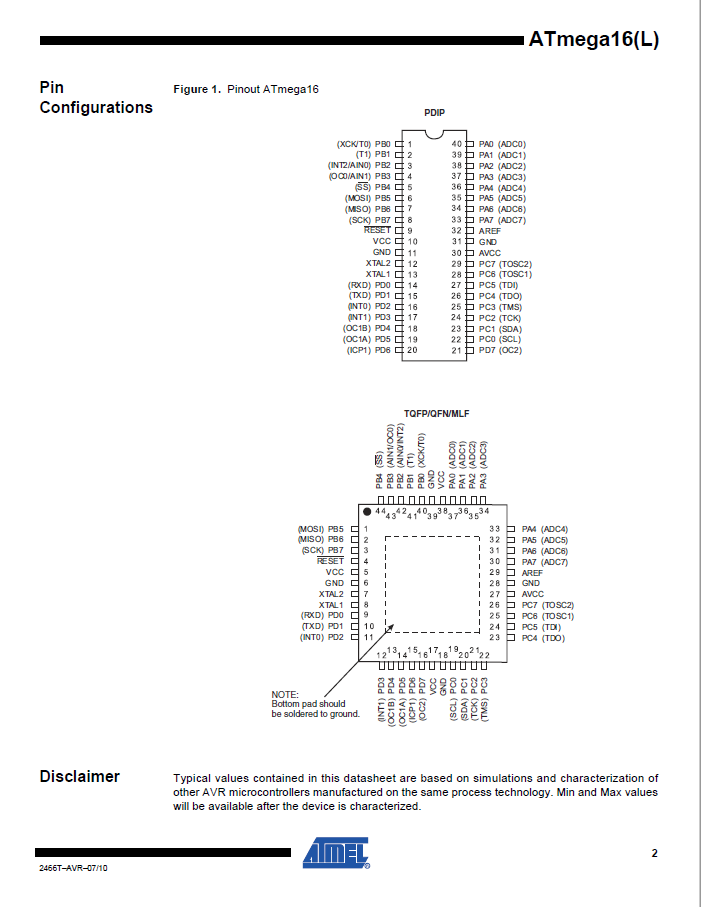
//De regulă și aici se scriu funcții sau instrucțiuni

}

}

Anexa1



Anexa2

Anexa3

