1. Componenta hardware

Aplicația este construită avînd la baza microcontroller, acest lucru se datoreaza faptului ca , microcontrollerul poate înlocui cu succes o mare parte din vechile circuite logice TTL, flexibilitatea acestuia ne face să optăm pentru el.

Proiectarea și realizarea sistemelor electronice digitale , utilizănd logica cablată era pe de o parte o operație anevoioasa și pe de altă parte rezultau configurații rigide greu de modificat și reconfigurat.

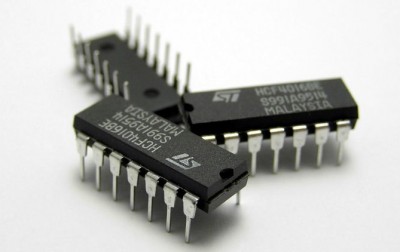


Figura.1.1 Circuite integrate TTL

Un avantaj cert al logicii cablate, era reprezentat de viteza de prelucrare a informației, circuitele logice TTL avănd timpi de propagare prin porțile logice foarte mici de ordinul nanosecundelor.

În figura de mai jos prezentăm un exemplu de montaj realizat cu circuite integrate digitale.

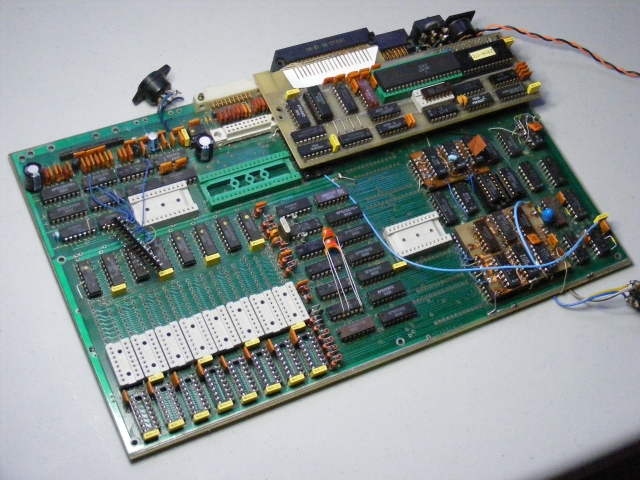


Figura.1.2 Placa de circuit echipată cu C.I. digitale

În prezent , cercetările în domeniul semiconductorilor, al microelectronicii au dus la perfecționarea microprocesoarelor, pe deoparte a scăzut dimensiunea fizică a acestuia, iar pe de altă parte a crescut frecvența de ceas a procesorului care, coraborată cu tehnicile de procesare de tipul pipeline . a făcut posibilă execuția unei instrucțiuni pe ceas.

Astfel. La ora actuală, logica programată nu mai reprezintă un punct sensibil, mai ales pentru aplicațiile în timp real, unde știim cu toții, timpul de reacție al sistemului trebuie sa fie mai mic decăt cel de evoluție al evenimentului considerat.

Evoluția procesoarelor a cunoscut în ultimii ani o adevărată explozie, dacă la începuturi procesorul, era localizat pe mai multe placi de circuite imprimate fiecare realizănd funcții de bază, procesare logică și calcul aritmetic (ALU), adresare memorie externă, gestionare periferice, microcod, etc .. , astazi totul se regăsește încapsulat într-un singur CIP.

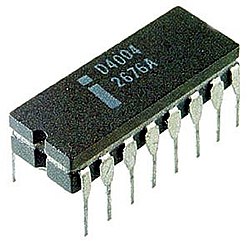


Figura.1.3 Primul procesor on-chip produs de Intel

În figura 1.3 putem vedea primul microprocesor produs de către firma Intel, utilitatea acestuia fiind găsită tocmai la dirjarea circulației prin intersecțiile semaforizate, din S.U.A.

Inițial arhitectura microprocesorului a fost una clasică de tip VonNoimann, ceea ce însemna că memoria program găzduia atît datele căt și instrucțiunile programului.

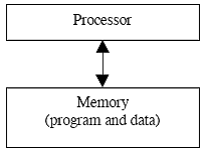


Figura. 1.4 Arhitectura Von Neumann

Cercetătorii de la universitatea Harvard, au pus bazele unei arhitecturi, în care memoria de date era separată de memoria de program, implementănd astfel un paralelism, dar care. Datorită prețului ridicat al hardware-ului la ora aceea , nu a avut succes pe piață, mult timp piața microprocesoarelor fiind dominată de catre arhitectura VonNeumann.

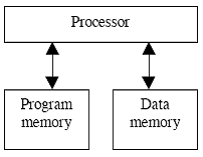


Figura. 1.5 Arhitectura Harvard

La ora actuala, arhitectura Harvard este folosită din ce in ce mai mult , mai ales in domeniul microcontrollere-lor, asigurînd implementarea instrucțiunilor de tip RISC, microcontrolerele dispun astfel de un set redus de instrucțiuni, stabilit de către proiectantul acestuia, considerat arhisuficient pentru rezolvarea oricărui tip de problemă.

Datele și instrucțiunile, provenind din memorii diferite , sunt aduse în unitatea de calcul în acelaș ciclu de ceas, de regulă într-un singur ciclu de ceas, care coraborat cu frecvențe de ordinul sutelor de MHz, face ca timpul de reacție a sistemului să fie comparabil sau poate mai bun decît al circuitelor logice TTL.

1. Microcontrollere
   1. Generalități

Microcontrollerele constituie sisteme de calcul integrate pe un singur cip de siliciu. Acesta conținănd de regulă unitatea de calcul matematic CPU, memoriile pentru date și program, memorii de tip RAM (Random Acces Memory), memorii EEPROM (Electrical Erasable Programable Read Only Memory) de regulă memorii de tip FLASH, o denumire comercială care ne duce cu găndul la viteza de programare care, raportată la vechile circuite EPROM, este comparativa cu durata unui FLASH de bliz.

Găsim integrate circuitele periferice sub forma porturilor de comunicare bidirecțională paralel, in care datele sunt trimise în acelaș timp pe mai multe linii de transmisie,

Porturi seriale, cel mai cunoscut fiind cel care utilizeaza standardul RS-232, în care bitii sunt grupați si trimiși succesiv pe o singură linie de transmisie.

Numarătoare, cu ajutorul căroră se pot contoriza numarul de impulsuri provenite de la diverse surse , interne sau externe.

Temporizatoare, a căror declanșare poate implementa un multitasking in program spre exemlu, condițiile de declanșare putînd fii de asemenea programabile.

Convertoare analog-numerice, ce realizeaza conversia mărimilor electrice analogice în mărimi discrete.

Comparatoare analogice, interfețe dedicate de comunicare. Ethernet, USB, one-wire,JTAG, ISP, SPI.

Deasemenea microcontrollerele moderne pune la dispoziți posibilitatea utilizării unui bootloader, ceea ce face ca aplicația să poată fii încărcată de la distanță utilizănd de exemplu pentru comunicație modemuri și linii telefonice analogice, lucruu cît să poate de util . mai ales în cazul în care sistemul se află montat în zone greu accesibile, cum ar fii poziționarea acestuia la altitudini mari, în locuri izolate făra acces rutier, etc ...

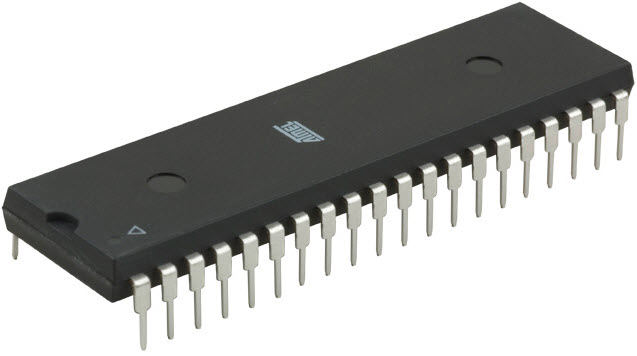


Figura.2.1 Microcontroler produs de Atmel

Fizic, un microcontroller este o componentă electronică, ce se află încapsulată într-o structură de material sintetic cu o rezistență mecanică foarte bună, in vederea evitării deteriorării fizice, conductivitate termică bună, pentru disiparea căldurii și rezistență la acțiunile corozive ale agenților chimici.

În funcție de conectivitatea externă a microcontrollerului, acesta poate fii încapsulat în capsule de tip DIL, PQFP, BGA etc ...

Mai jos putem vedea o serie diferită de capsule , des utilizate în electronică.

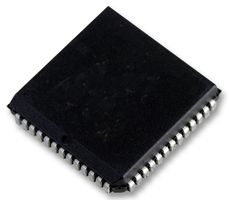
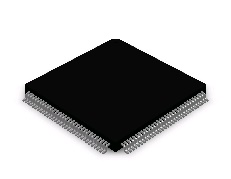
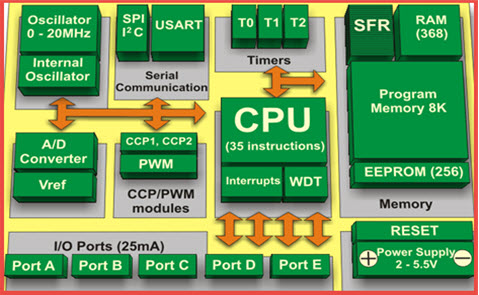


Figura.2.2 Tipuri de capsule , in ordine de la stînga la dreapta, DIL,PQFP,PLCC,BGA.

Evoluția capsulelor este strîns legată, pe deoparte de gradul de integrare a componentelor electronice pe plăcile de circuit imprimat, ur,ărindu-se obținerea unei suprafețe căt mai mici pentru un număr cît mai mare de conexiuni , iar pe de altă parte , de capabilitățile de conectare a integratului.

În anexa acestei lucrări am atașat extras din foaia de catalog privind caracteristicile electrice și mecanice ale microcontrollerului Atmega8535, vezi anexa 2.1

În mare, arhitectura microcontrollerului este exact precum cea ilustrată in figra de mai jos.



Aici se poate vedea, bine evidențiat structurile funcționale, cele mai des utilizate în aplicațiile bazate pe microcontrolere.

Și microcontrollerul utilizat în proiectarea acestei aplicații utilizează aceiasi structură internă, ca cea ilustrată mai sus, în anexa 2.2 vedem extras din foaia de catalog a producătorului, privind structura internă căt și caracteristicile și facilități puse la dispoziție proiectantului.

1. Schema de principiu – proiectarea sistemului.

Încă de la început avem clar scopul acestui proiect și anume proiectarea și implementarea unui sistem automat de comandă a traficului într-o intersecție semaforizată, trafic care înregistrează valori diferite, în perioade distincte ale unei zile.

În consecință, vom avea două tipuri de variabile și anume :

1. Variabile de intrare – număr de vehicule pe fiecare drum
2. Variabile de ieșire – valori numerice afișate pe fiecare drum, stările semafoarelor

ieșiri

Intrări

SISTEM LOGIC

Pornind de la schema abstractă prezentată mai sus, putem contura deja sistemul nostru, și anume caseta notata cu SISTEM LOGIC, reprezintă însuși microcontrollerul, intrările vor fii reprezentate prin potențiometrii liniari , ai căror variație de rezistență se va traduce în variația numărului de vehicole pe minut ce trec prin intersecție, ieșirile vor fii pe doparte afișaje numerice cu ajutorul cărora se informeaza utilizatorul despre traficul curent și semafoarele, reprezentate prin elemente optice vizuale binare, avănd culori distincte pe fiecare drum, respectiv, roșu pentru interzicerea accesului vehicolelor în intersecție și verde pentru permiterea accesului vehicolelor în intersecție.

Metoda de proiectare aleasă este de tipul bottom-up.

Trafic A

M

I

C

R

O

C

O

N

T

R

O

L

L

E

R

Afișaj \_ A

IR

SENZOR

Afișaj \_ B

SemaforA

SemaforB

Trafic B

Figura. 3.1 Schema de principiu

Pentru afișarea traficului au fost alese sisteme de afișare cu 7 segmente , aceste se prezintă precum în figura de mai jos.

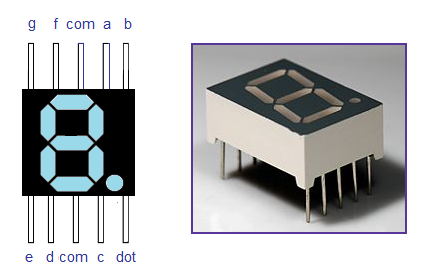


Figura.3.2 Afișaj cu 7 segmente

Pentru afișarea stării semafoarelor am ales LED-uri, pe fiecare sens două leduri avînd culoarea ROȘIE pentru a indica interzicerea trecerii vehicolelor, respectiv VERDE ce indică permiterea trecerii vehicolelor prin intersectia semaforizată.

Sistemul de afișare al numărului de vehicole ce trec prin intersecție este compus din doi digiți ceea ce înseamnă că, pot fi afișate mărimi de ordinul vehicole pe minut, pentru fiecare drum.

Analizănd modul de funcționare a unui digit, am concretizat în tabelul 3.1 logica.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | 1 | 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.3.1 Logica de funcționare a digitului pentru situația anod comun

În acest tabel , în situația utilizării digitului cu conexiune anod comun, ceea ce înseamnă că toți anozii ledurilor din afișaj sunectați înpreună și acesibili din exterior printr-un pin notat cu com, valorile de 0L ce corespund unei tensiuni aflată ăn limita 0 ,, 0.3v aplicate pe un segment va face ca acesta sa lumineze, respectiv valorile de 1L ce corespund unei tensiuni electrice aflate în intervalul 1.8 ... 5 volți, va face ca segmentul respectiv sa nu lumineze.

Conectarea electrică propriuzisă cu microcontrolerul va presupune realizarea unor conexiuni electrice între pinii afișajului și cei ai microcontrolerului precum și dimensionarea unor rezistori pentru limitarea curentului ce va trece prin led-urile afișajului.

Conexiunea se face prin multiplexarea datelor, metodă ce presupune afișarea unor date consecutiv pentru o perioadă scurtă de timp, pentru un anume rang al numarului afișat. În acest mod se reduce numărul de legături fizice necesare, care în situația noastră pentru cei patru digiți ar fii fost 28.

Reglarea numărului de vehicole pe fiecare drum, se realizează pe deoparte cu ajutorul unui senzor de infraroșu, senzor ce va incrementa la fiecare întrerupere a barierei luminoase, numărul de vehicule pe un drum, respectiv potențiometru liniar rotativ.

Acesta practic , este cuplat în sistem ca un divizor de tensiune rezistiv cu raport de divizare reglabil.

Cele două extremități ale potențiometrului sunt legate una la masă și cealaltă la potențialul de 5volți, pe cursor , culegîndu-se o tensiune ce va varia între 0volti și 5 volți în raport cu unghiul de rotire al cursorului potențiometrului.



Figura.3.3 Potențiometru rotativ

Radiația infraroșie este o radiație electromagnetică a cărei frecvență se situează dincolo de spectrul vizibil sub radiația luminoasă ce corespunde culorii roșu.



Receptorul infraroșu este de regulă o fotodiodă, polarizată în sens invers, care este sensibilă la o radiație luminoasă corespunzătoare spectrului infraroșu.

În figura de mai sus observăm un receptor de IR.

Avînd în vedere faptul că logica de comandă a întregului sistem este digitală, microcontrollerul lucrînd doar cu semnale discrete , cu valori bine definite alocate celor două stări binare de 0L și 1L, va trebuii realizată o conversi a semnalului analogic furnizat de către potențiometrii într-o mărime numerică proporțională.

Pentru implementarea semnalizatoarelor rutiere sa optat pentru utilizarea diodelor electroluminiscente de tip LED, avînd la bază mai multe considerente printre care și o ușoară interfațare cu microcontroller-ul , aceasta presupunînd doar o dimensionare a unor rezistori de limitare a curentului prin LED-uri.

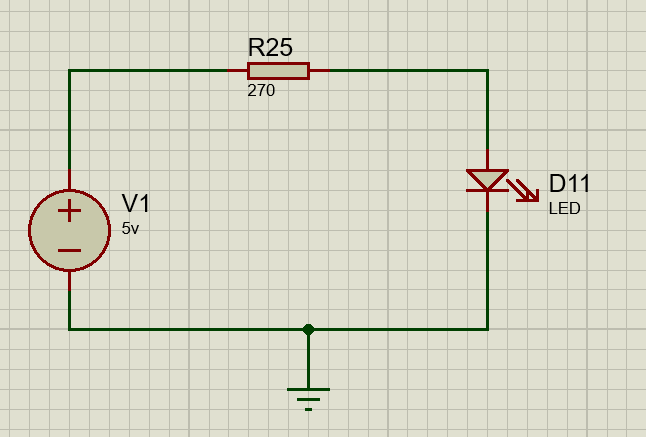


Figura.3.4 Schema electrică de conectare a led-ului

Observănd scema din figura 3.4, putem face analogie între sursa de tensiune , notată V1 și o iesire a portului microcontroller-ului, ieșire care în cazul stării logice de 1L se va afla la un potențial electric de 5volti.

Atunci, cunoscînd , din foaia de catalog , parametrii electrici ai led-ului, tensiune de deschidere, curent , putem dimensiona rezistența de limitare R25, aplicănd teorema lui Ohm pentru un circuit simplu,

Care, după înlocuire numerică, ne va da o valoare a lui R25 de aproximativ 200 ohm

Deasemenea, pentru funcționarea întregului circuit, acesta se va alimenta consecutiv, conform temei proiectului, dintr-o sursă de curent bazată pe panouri solare și o sursă convențională, respectiv o baterie electrică de 9volți.

Schema electrică de principiu se prezintă ca în figura de mai jos.

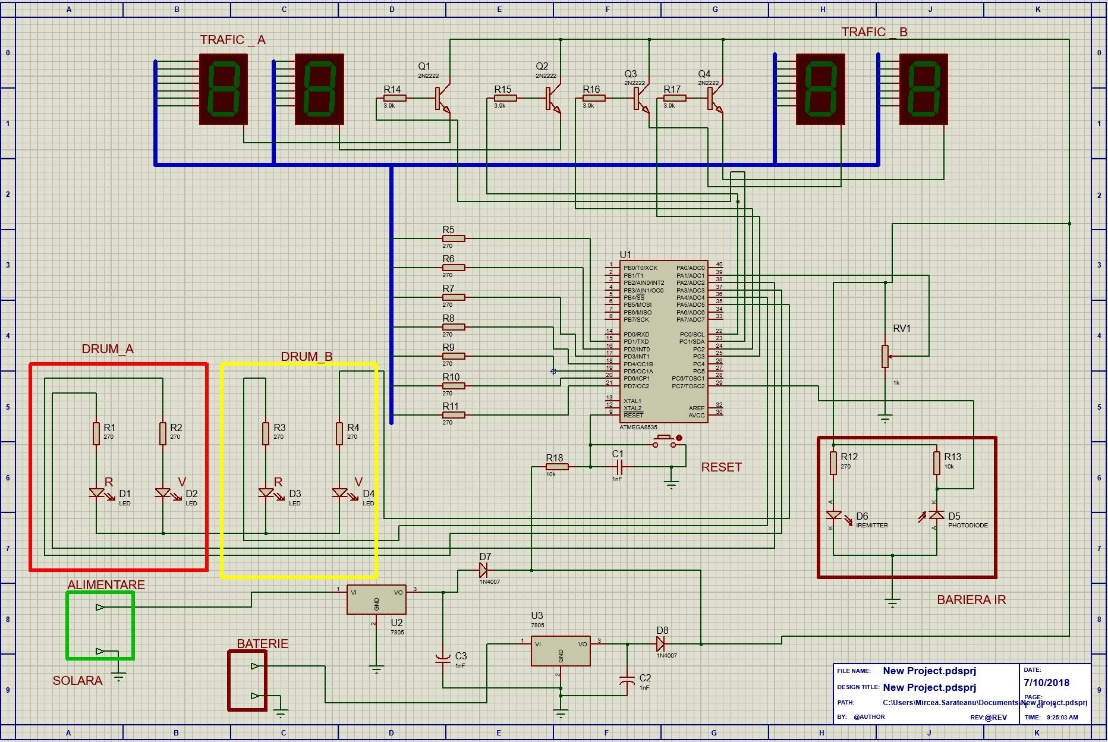


Figura 3.5 Schema electrică de principiu.

În această schemă observăm elementele descrise mai sus, notate cu TRAFIC\_A respectiv TRAFIC\_B regăsim cele două afișaje cu anod comun , a căror multiplexare în timp este asigurată prin tranzistoarele notate Q1, Q2, Q3, Q4.

Conexiunile celor șapte segmente , prin rezistențele de limitare a curentului sau facut , alegănd portul D al microcontrollerului, conform tabelului de mai jos.

|  |  |
| --- | --- |
| **PORTD** | **SEGMENT** |
| PD7 | a |
| PD6 | b |
| PD5 | c |
| PD4 | d |
| PD3 | e |
| PD2 | f |
| PD1 | g |

Tabelul.3.2 Asignarea pinilor pentru subsistemul de afișare

Multiplexarea afișării numerice a traficului se realizează prin comutarea succesivă a anozilor elementelor optice , într-o ordine bine stabilită, începănd de la rangul corespunzător LSB către MSB, în cazul nostru, în baza de numerație 10, de la rangul corespunzător unităților către zeci, comutare realizată cu ajutorul tranzistoarelor bipolare ce lucrează în conexiune colector comun si clasa de amplificare D.

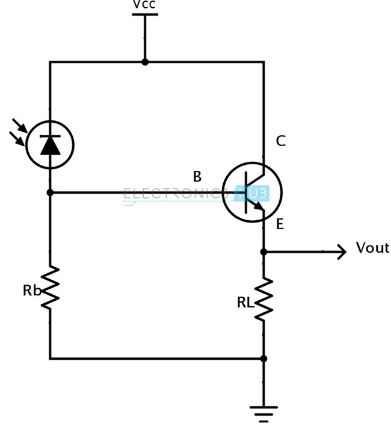
Alocarea portului pentru multiplexare a fost făcută arbitrar, atmega8535, dispunînd de un număr mare de porturi. Totuși a fost ales PORT-C – cu litera C semnificînd comandă.

Tabelul de mai jos ilustrează modul de asignare a pinilor portului C

|  |  |  |
| --- | --- | --- |
| **PORTC** | **RANG** | **AFIȘAJ** |
| PC0 | Unități | TRAFIC\_A |
| PC1 | zeci | TRAFIC\_A |
| PC2 | unități | TRAFIC\_B |
| PC3 | Zeci | TRAFIC\_B |

Tabelul.3.3 Asignarea pinilor pentru multiplexarea afișarii

Implementarea barierei cu radiații infraroșii a fost făcută pornind de la scheme de conexiune indicate de însăși producătorii receptorului de radiație infraroșie, emițătorul fiind de fapt un LED, polarizarea acestuia însemnănd practic dimensionarea unei rezistențe de limitare a curentului electric prin LED, alimentarea lui realizăndu-se din sursa liniară de 5 v.



În figura de mai sus se poate observa modul de conectare al receptorului , amplificarea semnalului fiind asigurată de către tranzistorul repetor pe emitor.

Modul de funcționare este următorul, la apariția unui flux luminos , dioda se deschide polarizănd în sens direct jonctiunea bază-emitor a tranzistorului, acesta se va deschide și el prin colector apărînd un curent ceea ce duce la apariția unei căderi de tensiune pozitive pe rezistența RL.

Conectarea barierie optice la microcontroller sa realizat utilizănd o intrare de întrerupere disponibilă la pinii acestuia, respectiv INT2.

|  |  |
| --- | --- |
| PIN CAPSULA | INTRERUPERE |
| 29 | TOSC2 |

În ceea ce privește intrările analogice, suntem constrînși în alegerea noastră de faptul că microcontrolerul are pini dedicați acestui fapt, la fel cum sunt dedicați și pentru transferul serial de date , conexiuni ISP, intreruperi hardware externe etc ...

Consultînd foaia de catalog, observăm că microcontrollerul pune la dispoziție proiectantului , 8 intrări analogice non-diferențiale, multiplexate concretizate în tabelul de mai jos

|  |  |
| --- | --- |
| **PIN CAPSULĂ** | **INTRARE ADC Nr.** |
| **40** | **ADC0** |
| **39** | **ADC1** |
| **38** | **ADC2** |
| **37** | **ADC3** |
| **36** | **ADC4** |
| **35** | **ADC5** |
| **34** | **ADC6** |
| **33** | **ADC7** |

Tabelul 3.4 Configurația pinilor convertorului ADC

Într-o ordine firească a lucrurilor am ales ca , potențiometrul , notat în schemă cu RV1 și asignat reglării traficului pe drumul A, să fie conectat la ADC0, respectiv pinul 40, iar potențiometul RV2 ce reglează traficul pentru drumul B, să fie conectat pe ADC1, respectiv pinul 39.

Elementele de semnalizare optice a accesului pe fiecare drum, respectiv LED-urile notate în schemă cu D1, D2, D3, D4, au fost conectate, prin rezistorii de limitare a curentului, la portul A, conform tabelului de mai jos

|  |  |  |
| --- | --- | --- |
| **PORTA** | **LED** | **TRAFIC** |
| PA2 | ROȘU | A |
| PA3 | VERDE | A |
| PA4 | ROȘU | B |
| PA5 | VERDE | B |

Tabelul.3.5 Asignarea elementelor de semaforizare

Alimentarea întregului montaj se realizează pe două căi, una ce utilizează energie solară și cealaltă energia electrică generată de o pilă chimică cu t.e.m de 9volți, ambele cuplate cu ajutorul unor stabilizatoare liniare integrate de 5volti, L7805.

Primul stabilizator, notat în schemă cu U2, realizează alimentarea pe ramura de sursă electrică convențională, cel de al doi-lea, notat cu U3, realizează alimentarea cu ajutorul energiei solare, comutarea este făcută automat cu ajutorul diodelor D5, D6, acestea lucrînd astfel.

În cazul în care radiația solară nu este suficientă pentru a asigura buna funcționare a sistemului , dioda D6 va fii închisă iar dioda D5 va fii deschisă asigurînd alimentarea din bateria electrică, situația este existentă de exemplu pe un timp nefavorabil cu cer înnorat sau pe timp de noapte.

La limită, în cazul extrem, cînd și bateria nu poate asigura tensiunea normala, ambele diode fiind închise sistemul se va opri.

1. **Modul de funcționare**

Sistemul de semaforizare inteligentă, realizează o temporizare diferită în funcție de traficul măsurat pe cele două artere de circulație.

Traficul este reglat astfel. Pe drumul principal, prin opturarea radiației infraroșii din contorul de trafic, se realizează incrementarea secvențială , lucru ce poate fi vizualizat pe sistemul de afișare. Aducerea la zero a traficului fiind realizată la acționarea butonului de reset.

Pentru drumul secundar, traficul este reglat în mod continuu, prin acționarea cursorului potențiometrului, în sens orar – pentru creșterea valorii traficului respectiv în sens antiorar pentru scăderea valorii traficului. Identic, ca pentru drumul principal, traficul va fi monitorizat cantitativ cu ajutorul unui sistem de afișare cu doi digiți.

În permanență, valoarea traficului este comparată, iar timpul cît semaforul afișază culoarea verde pe un drum respectiv culoarea roși este dat de numarul de vehicole care este convertit în secunde.

Spre exemplu , dacă pe drumul A există un trafic de 10 vehicole / minut iar pe drumul B un trabic de 50 de vehicole / minut , atunci semaforul va afișa culoarea VERDE pe drumul A timp de 10 secunde, respectiv culoarea VERDE pe drumul B timp de 50 de secunde.

Aceiași logică se aplică și în situașia în care se înregistrează un trafic mai mare pe drumul A față de drumul B.

La egalitate, cănd avem acelaș trafic pe ambele drumuri, timpii de semaforizare vor fi egali pe ambele drumuri și direct proporționali cu valoarea în secunde a numarului de vehicole ce trec pe drumul A.

1. **Logica de programare**

După cum aminteam la început, sistemul foloseste implementarea hardware utilizănd logica programată, ceea ce înseamnă că pentru a putea funcționa este necesară proiectarea si scrierea unui program, ce reprezintă componenta software, nepalpabilă , imaterială, cu ajutorul căruia realizăm operații de configurare a perifericelor (porturi, convertoare analog numerice, temporizatoare), inițializare a anumitor valori pentru diferite variabile, implementarea propriu zisă a algoritmilor, prin funcții .

Pentru o bună structurare a programului , a fost abordat stilul de programare structurat, utilizănd limbajul C.

Teorema de structură stipulează că orice algoritm care are cel puțin o intrare și o ieșire poate fi scris utilizănd urmatoarele structuri :

* Secvența
* Decizia
* Iterația

Secvența reprezintă una sau mai multe instrucțiuni grupate, în vederea realizării unui anume scop.

Ex

PORTB |=(1<<PB0);

PORTB &=~(1<<PB7);

……….

If((PINB&(1<<PB5) == 0){

PORTA |=(1<<PA3);

}

Else {

PORTA &=~(1<<PA3);

}

Decizia, este o modalitate prin care se poate controla evoluția programului prin executarea unui anumit set de instrucțiuni în funcție de rezultatul evaluării unei expresii

Ex. If(expresie ) {instructiune1}

Else {instructiune2}

Iterațiile pot fii de două feluri, cănd se cunosc numărul de pași implementate cu ajutorul buclelor FOR, și cănd nu se cunosc numărul de pași , implementate cu bucle WHILE.

Etapele de realizare a programului presupune declararea variabilelor ce vor fii utilizate .

unsigned int traficA = 0;

unsigned int traficB = 0;

declararea și definirea funcțiilor utilizate

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

{

unsigned int separat[5];

unsigned int unitati, zeci;

unsigned int valoare, count, aux;

valoare = numar1;

count = 0;

aux = 0;

separat[0] = separat[1] = separat[2] = separat[3]= separat[4]= 0;

while(valoare!=0)

{

aux = valoare % 10;

valoare = valoare / 10;

separat[count] = aux;

count ++;

}//END WHILE

......................................

.....................................

}

Declararea unor variabile locale, acolo unde este cazul, variabile care au o vizibilitate restrînsă doar în funcția unde sunt utilizate.

Apelarea funcțiilor în funcția principală main.

Se știe că, pentru a putea funcționa , orice program scris în limbajul C are o funcție main.

Implementarea afișării a fost facută utilizînd o funcție a cărui apel este realizat într-o funcție de deservire a întreruperii alocată pentru TIMER0.

Antetul funcției fiind următorul

afisare\_LCD(unsigned int trafic\_principal,unsigned int trafic\_secundar,unsigned int baleiaj);

functia are trei parametrii de tip întreg fără semn denumiți sugestiv trafic\_principal, trafic\_secundar, al treilea, implementeaza multiplexarea valoarea lui fiind in permanenta comparată și pe baza deciziei se va comuta elementul de afișare potrivit.

void initializare\_ADC()

{

ADMUX = 0;

ADMUX |= (1<<REFS0);//Vezi pagina 211 din datasheet Atmega16

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

}

Initializare\_ADC() este o funcție ce realizează stabilirea parametrilor de funcționare a convertorului analog digital, prin regițtrii specifici acestuia , respectiv ADMUX și ADCSRA.

Se stabilește referința de tensiune ca fiind cea internă, cu valoarea de 5 volți, frecvența de eșantionare fiind rezultatul divizării dintre frecvența de ceas la 128 și se pornește conversia în modul FREE RUNING MODE, ceea ce înseamnă că la sfărșitul fiecărui ciclu de conversie , convertorul va reinițializa alt ciclu de conversie în mod perpetuu.

unsigned int citeste\_ADC(int intrare)

Funcția citeste\_ADC(int intrare), va realiza citirea valorii tensiunii analogice pe una intrările specificate de variabila intrare, respectiv 0 pentru trafic\_A și 1 pentru trafic\_B, ea returnează o valoare numerică pe 10 biți, valoare proporțională cu tensiunea de intrare după formula

Unde Vintrare este tensiunea culeasă de pe cursorul potențiometrului , tensiune care va trebuii să varieze pentru intrare nediferențială între valoarea minimă de 0 volți și Vreferință , în cazul nostru 5 volți.

ISR(TIMER0\_OVF\_vect, ISR\_BLOCK)

Serviciul de tratare a întreruperilor, monitorizează în permanență sursele de întreruperi, iar la apariția întreruperii se va lansa în execuție o funcție ce va trata această întrerupere.

Aici, pentru a realiza în program mai ,ulte fire de execuție ce evoluează independent, am ales utilizarea servicilor de tratare a întreruperilor concretizate prin funcțiile ISR = Interrupt Services Routine.

Temporizatorul, poate fii folosit ca sursă de generare a întreruperii, funcționare lui ciclică determină, generarea unei întreruperi la o anumită perioadă de timp, perioadă stabilită prin depăsirea capacității de numarare a registrului pe 8 sau 16 biți.

Frecvența generării întreruperii se poate calcula cu formula

Unde fclk reprezintă frecvența de ceas, N factorul de divizare, OCRn valoarea încărcată în registrul de deplasare al temporizatorului, în cazul de fața 255.

Afișarea valorilor pentru trafic se face prin apelul funcției afisare\_LCD(trafic\_principal,trafic\_secundar,baleiaj) la fiecare apel al serviciului de tratare a întreruperii alocate pentru temporizatorul TIMER0.

Frecvența de generare a întreruperii a fost determinată în mod experimental pînă la evitarea în totalitate a efectului supărător de pîlpîire a afișajului, fiind o funție care trebuie să se execute cu prioritate, apariția altor întreruperi , putănd deranja modul de funționare a sistemului de afișare, am făcut uz de anumite facilităti și anume de blocarea imediată a sistemului general de intreruperi, prin directiva ISR\_BLOCK.

Cel de al doi-lea TIMER pe 16 biți este utilizat pentru implementarea unui fir de execuție paralel, necesar citirii valorilor tensiunii pe cele două potențiometre.

Situația este mult mai lejeră, viteza de variație a valorilor tensiunilor fiind una lenta, imprimată de mișcarea de rotație a degetelor operatorului pe rotorul potențiomentrului, astfel ca o frecvență scăzută de apel a întreruperii de ordinul hertilor zeci de herți nu crează probleme.

La fiecare aproximativ zecime de secunda este apelat serviciul de tratare a întreruperii prin intermediul funcției ISR .

Astfel

ISR(TIMER1\_OVF\_vect){

trafic2 = citeste\_ADC(1);

trafic\_principal = trafic1;

trafic\_secundar = trafic2/10;

}

ISR(INT2\_vect)

{

Trafic1 = trafic1 + 1;

}

Funcția incrementeză valoarea traficului pe drumul principal la fiecare trecere a unui vehicul prin bariera optică

Algoritmul ce implementează schimbarea stării semafoarelor a fost concretizat prin funcția următoare

void schimba\_semafor(unsigned int faza)

{

if(faza == 0){

PORTA |=(1<<PA3)|(1<<PA4);

PORTA &= ~((1<<PA2)|(1<<PA5));

}

else{

PORTA &= ~((1<<PA3)|(1<<PA4));

PORTA |= (1<<PA2)|(1<<PA5);

}

}

Schimba\_semafor, realizează comutarea diodelor electroluminiscente de tip LED de pe culoarea ROȘU pe VERDE în funcție de valoarea variabilei faza.

Astfel dacă faza = 0 este liber traficul\_A iar dacă faza este 1, este liber trafic\_B

Următoarele două funcții realizează o distribuție uniformă a valorilor de trafic pentru fiecare drum, principal și secundar.

unsigned int calcul\_principal(unsigned int vehicole\_principal){

unsigned int totalizare;

if((vehicole\_principal>=0)&&(vehicole\_principal<10)){totalizare = 10;}

if((vehicole\_principal>=10)&&(vehicole\_principal<20)){totalizare = 20;}

if((vehicole\_principal>=20)&&(vehicole\_principal<30)){totalizare = 30;}

if((vehicole\_principal>=30)&&(vehicole\_principal<40)){totalizare = 40;}

if((vehicole\_principal>=40)&&(vehicole\_principal<50)){totalizare = 50;}

if((vehicole\_principal>=50)&&(vehicole\_principal<60)){totalizare = 60;}

if((vehicole\_principal>=60)&&(vehicole\_principal<70)){totalizare = 70;}

if((vehicole\_principal>=70)&&(vehicole\_principal<80)){totalizare = 80;}

if((vehicole\_principal>=80)&&(vehicole\_principal<90)){totalizare = 90;}

return totalizare;

}

unsigned int calcul\_secundar(unsigned int vehicole\_secundar){

unsigned int totsec;

if((vehicole\_secundar>=0)&&(vehicole\_secundar<10)){totsec = 10;}

if((vehicole\_secundar>=10)&&(vehicole\_secundar<20)){totsec = 20;}

if((vehicole\_secundar>=20)&&(vehicole\_secundar<30)){totsec = 30;}

if((vehicole\_secundar>=30)&&(vehicole\_secundar<40)){totsec = 40;}

if((vehicole\_secundar>=40)&&(vehicole\_secundar<50)){totsec = 50;}

if((vehicole\_secundar>=50)&&(vehicole\_secundar<60)){totsec = 60;}

if((vehicole\_secundar>=60)&&(vehicole\_secundar<70)){totsec = 70;}

if((vehicole\_secundar>=70)&&(vehicole\_secundar<80)){totsec = 80;}

if((vehicole\_secundar>=80)&&(vehicole\_secundar<90)){totsec = 90;}

return totsec;

}

Logica comutării și timpul de semaforizare a fost implementat prin structuri multiple de tip switch , ce stabilesc, într-o buclă eternă, în funcție de cantitatea de trafic o valoare a întărzierilor înainte de a apela funtiile scimba\_semafor.

if(numarvehicule\_principal>numarvehicule\_secundar){

switch(numarvehicule\_principal){

case 10 : \_delay\_ms(10000);break;

case 20 : \_delay\_ms(15000);break;

}

schimba\_semafor(0);

switch(numarvehicule\_secundar){

case 10: \_delay\_ms(10000);break;

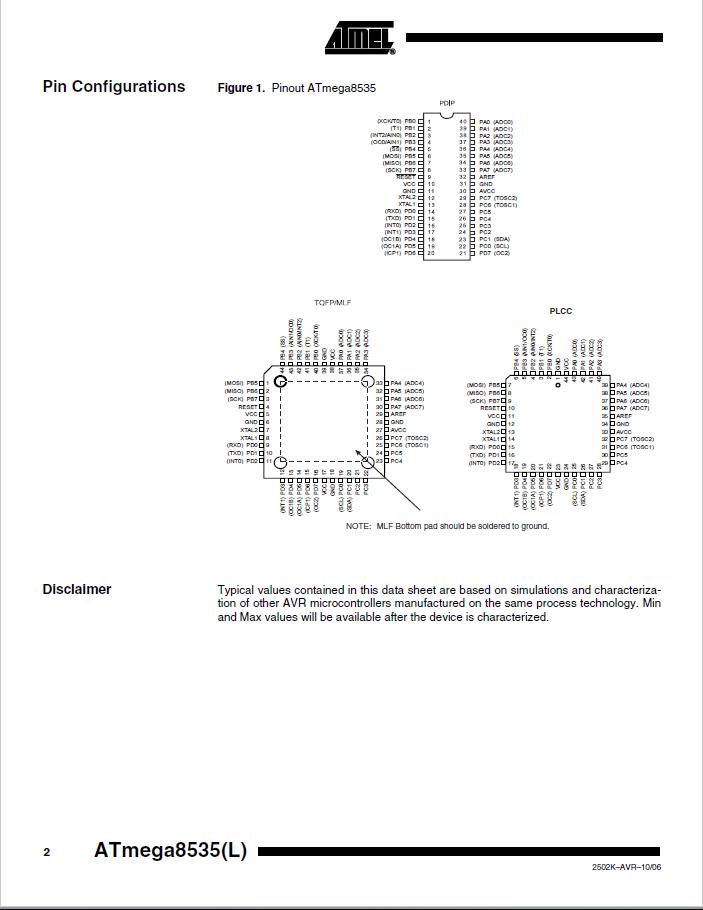
case 20: \_delay\_ms(15000);break;

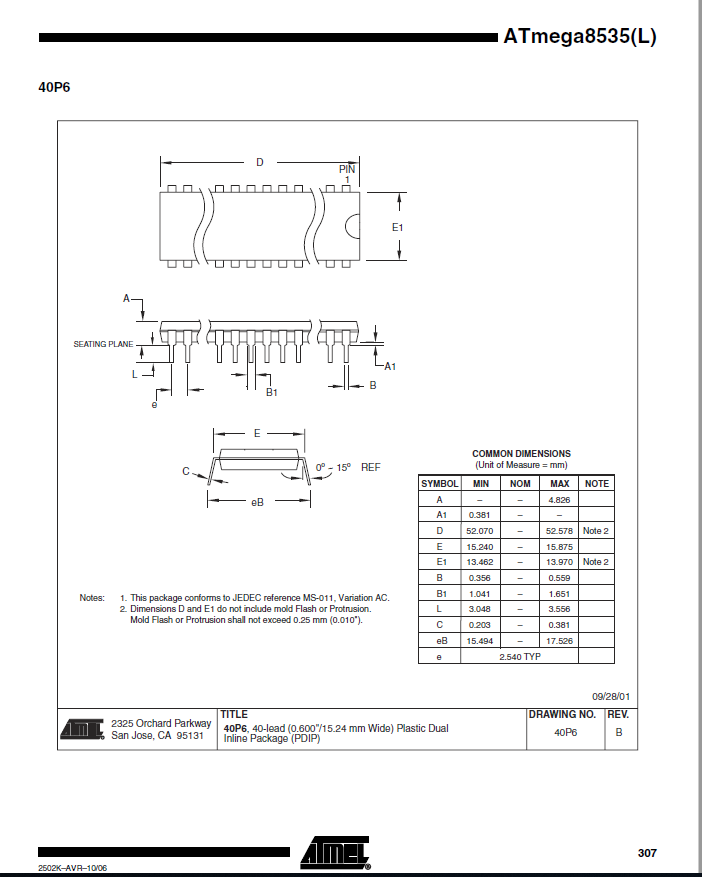
}

schimba\_semafor(0);

Tot aici se vor realiza inițializări privitoare la direcția porturilor, inițializarea temporizatoarelor etc ...

Anexa 2.1





Anexa 2.2

