

Universitatea “Politehnica” din Bucureşti Facultatea de Electronică, Telecomunicaţii
şi Tehnologia Informaţiei

***Modul de pornire automatizată a motorului autovehiculului, cu
telecomandă și program implementat***

Lucrare de licență

prezentat ca cerință parțială pentru obținerea titlului de
Inginer în domeniul *Electronică și Telecomunicații*
programul de studii Electronică Aplicată

Conducător științific

Prof. Dr. Ing. Alexandru VASILE

Absolvent

Iuliu Alexandru DINU

Anul

2017

Anexa 1

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Departamentul Electronica Aplicata si Ingineria Informatiei

Aprobat Director de Departament:
Prof. Dr. Ing. Sever Pașca

TEMA PROIECTULUI DE DIPLOMĂ
a studentului Dinu P. Iuliu Alexandru, grupa 442B

1. Titlul temei: Modul de pornire automatizată a motorului autovehiculului, cu telecomandă și program implementat

2. Contribuția practică, originală a studentului va consta în (exclusiv partea de documentare):

Se va concepe un sistem de pornire automatizată a motorului, pornind de la acțiunile care se întreprind și de la fenomenele care au loc în cazul unei porniri obișnuite (din cheie). Se va lua în calcul refacerea distribuției alimentărilor corespunzătoare sistemului cu cheie (pozițiile ACC, IGN, DEM). Sistemul va fi capabil să realizeze pornirea în trei moduri: din buton "START/STOP" – cu mașina nesecurizată, din telecomandă - cu mașina securizată și din meniu de pornire programată – cu mașina securizată, fără intervenția utilizatorului. Pentru implementare vor fi folosite microcontrollere alese convenabil, care să poată acoperi necesitățile proiectului, incluzând: numarul de actuatori pilotați (ieșiri), numarul de senzori, întrerupătoare, butoane (intrari), frecvența de lucru care să poată satisface lucrul în timp real, regimul de temperaturi de funcționare etc. Sistemul va putea fi împărțit în subansamblu în funcție de eventualele constrângeri și pentru a spori flexibilitatea instalării. Se vor realiza simulări pe calculator și se vor realiza și montaje practice. Prezentarea se va putea realiza atât pe machetă, cât și pe autovehicul.

3. Realizarea practică/ proiectul rămân în proprietatea: Student/ETTI

4. Proprietatea intelectuală asupra proiectului aparține: Student/ETTI

5. Locul de desfășurare a activității: La domiciliul studentului/la facultate, asistat de profesor

6. Data eliberării temei: 10.12.2016

CONDUCĂTOR LUCRARE:

Prof. Dr. Ing. Alexandru VASILE

STUDENT:

Iuliu Alexandru DINU

Declarație de onestitate academică

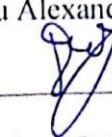
Prin prezenta declar că lucrarea cu titlul “*Modul de pornire automatizată a motorului autovehiculului, cu telecomandă și program implementat*”, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității “Politehnica” din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Electronică și Telecomunicații*, programul de studii *Electronică Aplicată* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățămînt superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 01.07.2017

Iuliu Alexandru DINU



(semnătura în original)

CUPRINS

INTRODUCERE – pagina 8

CAPITOLUL I – FUNDAMENTE TEORETICE – pagina 10

1.1 Definirea microcontrolorelor și familiile importante – pagina 10

1.2 Microcontrolerul ATMEGA8 – pagina 11

1.2.1 Porturile de intrări/ieșiri B, C și D ale lui ATMEGA 8 – pagina 11

1.2.2 Folosirea circuitului de debounce la citirea butoanelor și întrerupătoarelor – pagina 12

1.2.3 Utilizarea timerelor – pagina 13

1.2.4 Utilizarea întreruperilor – pagina 14

1.2.5 Setarea fuse-biților CKSEL pentru alegerea frecvenței și a tipului de ceas – pagina 15

1.2.6 Utilizarea software-ului CVAVR pentru scrierea programelor și încărcarea acestora în MC – pagina 16

1.3 Senzorii fotosensibili – pagina 18

CAPITOLUL II – PREMISELE MONTAJULUI PRACTIC – pagina 22

2.1 Motivația alegării microcontrolerului ATMEGA 8 – pagina 22

2.2 Instrumentar în vederea realizării montajului practic – pagina 23

2.3 Mențiuni în vederea realizării montajului practic – pagina 27

2.4 Descrierea funcțiilor montajului. Implicații practice. – pagina 30

CAPITOLUL III – CONSTRUCȚIA MONTAJULUI – pagina 37

3.1 Necessitatea construcției motorului simulat. Scop și cerințe. – pagina 37

3.2 Realizarea practică a motorului simulat – pagina 38

3.2.1 Etapele realizării – pagina 38

3.2.2 Verificarea programului – pagina 55

3.3 Constructia modulelor propriu-zise – pagina 57

3.4 Interconectarea modulelor și realizarea testelor de integrare – pagina 69

3.5 Descrierea teoretică a celorlale două module – pagina 71

CONCLUZII – pagina 74

ANEXE – pagina 76

BIBLIOGRAFIE – pagina 105

LISTĂ FIGURI

- Fig. 1.1 – Interfață de programare de tipul *AVRISP Mk2* – pagina 16
Fig. 1.2 – Schema de legătură pinii programator – microcontroler – pagina 17
Fig. 1.3 – Captură ecran – opțiunea *Chip Programmer* – pagina 17
Fig. 2.1 – Placă test breadboard – pagina 23
Fig. 2.2 – Cablaj de test – pagina 24
Fig. 2.3 – Cablaj de textolit acoperit cu strat uniform de cupru – pagina 24
Fig. 2.4 - Osciloscop – pagina 25
Fig. 2.5 – Sursă tensiune – pagina 26
Fig. 2.6 - Generator semnal sinusoidal și dreptunghiular cu afișarea frecvenței și reglaj fin – pagina 26
Fig. 2.7 - Transponder – pagina 34
Fig. 3.1 – Cerc cu raze trasate la fiecare 3° – pagina 41
Fig. 3.2 – Delimitarea fantelor – pagina 41
Fig. 3.3 – Șuruburi fixare volanță – pagina 42
Fig. 3.4 – Poziționare, fixare motor și volanță – pagina 43
Fig. 3.5 – Colțar în formă L la 90° – pagina 44
Fig. 3.6 - Reglaj pe înălțime al senzorului – pagina 44
Fig. 3.7 – Schemă testare și funcțiune senzor – pagina 45
Fig. 3.8 – Turății joase – pagina 45
Fig. 3.9 – Turății înalte – pagina 46
Fig. 3.10 – Trigger Schmidt modificat – pagina 47
Fig. 3.11 – Turății mici – pagina 47
Fig. 3.12 – Turății mari – pagina 48
Fig. 3.13 – Placă rapidă de test – pagina 49
Fig. 3.14 – Schemă variator cu PWM modificată – pagina 52
Fig. 3.15 - Modulul de distribuție a alimentărilor – pagina 59
Fig. 3.16 – Schemă montaj plăcuță relee – pagina 59
Fig. 3.17 - Montajului plăcuței cu relee – pagina 60
Fig. 3.18 - Modulul de comanda a starterului – pagina 63
Fig. 3.19 - Modulul de pornire – pagina 67
Fig. 3.20 – Montaj finalizat – pagina 70

LISTĂ TABELE

Tabel 1 – Tabel Pini Simulator Motor – pagina 76

Tabel 2 – Tabel Pini MDA – pagina 82

Tabel 3 – Tabel Pini MCS – pagina 88

Tabel 4 – Tabel Pini MP – pagina 97

Tabel 5 - Tabel Interconexiuni – pagina 104

LISTĂ ACRONIME

ACC – Accessory (trad. Accesorii)

IGN – Ignition (trad. tehnică Contact)

PWM – Pulse Width Modulation (trad. Modularea în durată a impulsului)

AVR – numele unei familii Atmel

CVAVR – Code Vision AVR

ISP – In-System-Programming (numele unei tehnologii de scriere)

MC – Microcontroler

SM – Simulator motor

MP – Modul Pornire

MDA – Modul de Distribuție a Alimentărilor

MT – Modul Telecomandă

MCS – Modul Comandă Starter

PR – Placă Relee

PB – Placă Buton (de Start)

RISC – Reduced Instructions Set Computer (nume arhitectură)

EEPROM – Electrically Erasable Programmable Read-Only Memory

SRAM – Static Random Access Memory

FLASH – Denumirea unui tip de memorie nevolatilă

I2C – Denumirea unei Interfețe Seriale

Introducere

Lucrarea își propune să realizeze un ansamblu de montaje electronice care să implementeze câteva funcții mai deosebite din domeniul automobilelor. Aceste proprietăți sunt întâlnite, la nivelul anului 2017, în special la autoturismele care se încadrează în game de prețuri mai pretențioase, dar au început, treptat, să fie incluse și în gamele mijlocii și inferioare. Această evoluție este accelerată și mai mult de situații în care producătorul, pentru a putea pătrunde pe anumite piețe, trebuie să respecte anumite standarde sau recomandări specifice țărilor respective, în ceea ce privește dotările minime, chiar dacă acest lucru poate fi contradictoriu cu împărțirea dotărilor în funcție de gamă, conform portofoliului producătorului.

Așa se face că, de exemplu, un autoturism din gama low-cost va trebui să includă obligatoriu aer condiționat cu sistem de control automat (cum este cunoscutul Climatronic, de la grupul Volkswagen) dacă se dorește vânzarea acestuia în țările arabe sau în Brazilia, chiar dacă această opțiune, în țările în care nu există astfel de cerințe obligatorii, este oferită doar pentru variantele din gama mijlocie sau înalță. Același lucru se întâmplă și în cazul dotării „volan încălzit electric” pentru autoturismele vândute în anumite zone din Rusia, sau menținerea automată a luminilor de întâlnire (faza scurta) atâtă timp cât motorul este pornit, pentru țările din nordul Europei. Aceste cerințe sunt impuse în general de condițiile climatice și de relief ale respectivelor țări, astfel că unele dotări percepute drept capriciu în condiții naturale moderate, sunt considerate indispensabile în condiții mai extreme. Atunci când discutăm despre o pornire automatizată a motorului nu o putem cataloga drept un moft. Da, cu siguranță, pornirea sau oprirea motorului „fără cheie” nu pare la prima vedere, mai ales pentru necunoscător, un lucru la fel de important ca exemplele de mai sus, însă este așa sau nu? După cum o să fie dezvoltat pe mai departe în lucrare, Start/Stop-ul automatizat aduce beneficii duratei de viață a bateriei, deci unu lucru esențial pentru funcționarea unui automobil.

Revenind la proiectul de față, am avut în vedere implementarea unui modul de pornire automatizată a motorului autovehiculului cu ajutorul unui program aplicat și a unei telecomenzi. Funcțiile pe care dorim să le punem în aplicare în această lucrare sunt următoarele :

- Pornirea/oprirea motorului din buton Start/Stop, fără menținerea apăsării pe durata demarajului;
- Pornirea cu temporizare/oprirea motorului din telecomandă;
- Pornirea programată și cu temporizare a motorului prin intermediul unei interfețe cu display și butoane;
- Gestiona distribuției alimentărilor principale (ACC, Ignition).

Primul capitol „*Fundamente teoretice*” este dedicat, după cum poate fi ușor de identificat din titlu, părții teoretice, unde o să fie dezbatute elementele care m-au ajutat la crearea proiectului propriu-zis. Aici am în vedere o discuție despre ce înseamnă un microcontroler, la modul general, ajungând treptat la tipul care este de folos în această lucrare. De asemenea, o secțiune va fi dedicată senzorilor.

„*Premisele motajului practic*” este cel de-al doilea capitol, parte a lucrării destinată detalierii motivației pentru alegerea microcontrolerului dezbatut în secțiunea anterioară, mențiuni în vederea realizării motajului practic, dar și implicațiile practice ale funcțiilor montajului.

În cel de-al treilea capitol, numit „*Construcția montajului*”, se va regăsi partea cea mai practică, mai exact prezentarea scopului în realizarea motorului simulat, descrierea pe larg a etapelor care au

stat la baza realizării machetei de motor simulat, dar și seria de funcții pe care le îndeplinește montajul. Etapele menționate, 19 la număr, trec prin fiecare stagiu de la replicarea volantei motorului, la testarea microcontrolerului, la proiectarea unor cablaje și până la scrierea codurilor sursă. De asemenea, o secțiune va fi destinată prezentării construcției modulelor propriu-zise, realizarea testelor de integrare, iar în final se vor dezbatе așteptările inițiale și rezultatele.

Concluziile lucrării vor cuprinde analiza rezultatelor și posibilele diferențe față de așteptările inițiale. Tot la concluzii vom întâlni și dificultățile care au apărut pe parcursul fiecărui pas bifat.

Nu în ultimul rând, în cadrul anexelor se vor regăsi toate figurile, de tipul scheme logice, amprenta cablajelor, dar și codurile sursă, tabele de descriere a funcțiilor pinilor și lista completă de interconexiuni din cadrul întregului ansamblu.

Capitolul I – Fundamente teoretice

Capitolul de față este o clarificare pe larg a microcontrolorilor și a familiilor importante, oprindu-ne asupra Atmega 8, MC-ul folosit pentru această lucrare. În cadrul acestui subcapitol se vor regăsi o serie de însăriuri ale aspectelor cele mai importante, precum: porturi intrări/ieșiri, circuitul debounce, timere, intreruperi, fuse-biți CKSEL și utilizarea software-ului Code Vision AVR. Ultimul subcapitol este dedicat sezorilor fotosensibili, axându-ne pe avantajele și dezavantajele acestora.

1.1 Definirea microcontrolorilor și familiile importante

Pentru început, este important de definit și înțeles mai bine funcționarea unui microcontroler, dar și a avantajelor pe care acesta le are, la modul general. „La modul general un controler (“controller” - un termen de origine anglo-saxonă, cu un domeniu de cuprindere foarte larg) este, actualmente, o structură electronică destinată controlului (destul de evident!) unui proces sau, mai general, unei interacțiuni caracteristice cu mediul exterior, fără să fie necesară intervenția operatorului uman. Primele controlere au fost realizate în tehnologii pur analogice, folosind componente electronice discrete și/sau componente electromecanice (de exemplu relee). Cele care fac apel la tehnica numerică modernă au fost realizate inițial pe baza logicii cablate (cu circuite integrate numerice standard SSI și MSI) și a unei electronici analogice uneori complexe, motiv pentru care “străluceau” prin dimensiuni mari, consum energetic pe măsură și, nu de puține ori, o fiabilitate care lăsa de dorit.

Apariția și utilizarea microprocesoarelor de uz general a dus la o reducere consistentă a costurilor, dimensiunilor, consumului și o îmbunătățire a fiabilității” [Microcontrolere, ultima accesare 30.06.2017]. Astfel, ne este artătat faptul că această soluție este una care conduce la scăderea numărului de componente electronice folosite, și implicit a costurilor.

De asemenea, este important de menționat că există mai multe familii ale microcontrolorilor, printre avem 80C186, 80C188 de la Intel, AMD, Z8 de la Zilog, 80C16x de la Infineon, însă cele mai cunoscute fac parte din PIC, de la compania Microchip și AVR de la Atmel.

„Primul microcontroler din această familie (PIC1650) a apărut acum mai bine de 20 de ani pe vremea când firma era proprietatea General Instruments. Este o familie de microcontrolere care, în ultimii ani, a cunoscut o dezvoltare explozivă. Sunt disponibile actualmente sub forma a 6 serii: PIC10, PIC12, PIC14, PIC16, PIC17 și PIC18. În seriile respective există variante cu memorie de program de tip OTP(C) sau FLASH(F). Au fost primele microcontrolere de 8 biți cu arhitectură RISC: PIC16C5x avea un set de doar 33 instrucțiuni (Intel 8048 avea 90). Arhitectura este de tip Harvard și, ca o particularitate, dimensiunea cuvântului pentru program este de 12, 14 sau 16 biți, cuvântul de date fiind tot de 8 biți. Există foarte multe variante pentru cele sase serii, unele din ele fiind caracterizate printr-un număr mic de conexiuni exterioare (pini) și în consecință dimensiuni mici, consum foarte mic, ideea de bază fiind costul redus. Cronologic, ultimul produs al firmei Microchip este seria dsPIC30F, de fapt un procesor numeric de semnal, de 16 biti, cu o periferie specifică optimizată pentru controlul actionărilor electrice (motoare electrice). (...) Un concurent puternic al seriei PIC este familia numită AVR, a firmei ATMEL, familie apărută în ultimii ani, care oferă variante de microcontrolere oarecum asemănătoare ca resurse cu familia PIC, la performanțe similare sau mai bune. Sunt bazate pe o arhitectură diferită, dar unitatea centrală este tot de tip RISC, cu

cuvântul de date de 8 biți. La fel ca la PIC dimensiunea cuvântului de program este mai mare, fiind de 16 biți. Există cel puțin 3 sub familii mari, în ordinea complexității resurselor, acestea fiind: AT Tiny, AT90 și ATMega.” [ibid] ATMega8 este și microcontrolerul pe care îl vom folosi și noi în această lucrare.

1.2 Microcontrolerul ATMEGA8

Enumerăm o parte din specificațiile și performanțele microcontrolerului, revelante pentru aplicația noastră:

- Arhitectura RISC avansată;
- Tensiune de alimentare: 4,5-5,5V;
- Puterea consumată la 4MHz, 25°C: 3,6 mA în stare activă și 0,5uA în stare adormită;
- Intervalul frecvenței de lucru: 0-16MHz;
- Viteza de procesare de până la 16 MIPS (milioane de instrucțiuni pe secundă), la frecvența de 16MHz;
- Memorie de program de tip FLASH, cu dimensiunea de 8 kiloocteți;
- Memorie EEPROM de 512 octeți;
- Memorie SRAM de 1 kiloocet;
- Numărul garantat al ciclurilor de scriere/ștergere: 10000 pentru memoria FLASH, 100000 pentru memoria EEPROM;
- Durata de retenție a datelor: 20 de ani la 85°C, 100 de ani la 25°C
- Două timere de 8 biți cu prescaler și un timer de 16 biți, cu prescaler;
- Interfață serială I2C;
- Oscilator RC intern (pentru frecvențe de până la 8MHz);
- Surse de întrerupere interne și externe;
- 5 moduri de adormire;
- Capsula PDIP cu 28 de pini.

În plus, acesta mai dispune de multe alte funcții și proprietăți, pe care însă nu le abordăm în proiectul de față. Printre acestea enumerăm: 6 canale ADC (convertor analog-numeric), utile pentru citirea tensiunilor sau curentilor și 3 canale PWM, utile pentru generare unei tensiuni variabile prin modulație în durata impulsului. De asemenea, cele 23 de linii de intrare/ieșire programabile se referă la faptul că cei 23 de pini ai porturilor B, C și D pot fi configurați ca linii de ieșire sau intrare binară de uz general, la care se pot emite sau detecta niveluri de tensiune nulă (0V) sau nenulă pozitivă (5V), dar au și diferite funcții alternative asociate, care pot fi folosite setând valorile corespunzătoare an regiștrii de control ai acestora.

1.2.1 Porturile de intrări/ieșiri B, C și D ale lui ATMEGA 8

Prin construcție, MC-ul a fost dotat cu trei porturi: B, C și D. Alte variante din familia ATMEGA disponă și de portul A. Porturile B și D conțin câte 8 pini de intrare/ieșire, numerotați fiecare sub forma PB0...PB7, respectiv PD0...PD7. Portul C este limitat la 7 astfel de pini. În cazul folosirii acestor pini ca linii de intrare sau ieșire de tip binar, vom lucra cu cei trei regiștri ai fiecărui port: DDRx, PORTx, PINx, unde x este numele portului. Aceștia sunt definiți pe 8 biți (respectiv 7 pentru portul C) și sunt descriși la pagina 65 în manualul de utilizare.

- Regiștrii DDRB, DDRC, DDRD („Data Direction Register”) se folosesc pentru setarea sensului de comunicație pentru fiecare pin: 1 pentru ieșire și 0 pentru intrare. Astfel, dacă dorim ca pinul 3 al portului C să fie ieșire, vom seta bitul 3 al registrului DDRC cu valoarea 1.

- Regiștrii PORTB, PORTC, PORTD („Port Data Register”) reprezintă regiștrii de date ai pinilor fiecărui port. În aceștia putem scrie valoarea binară pe care dorim să o avem ca ieșire pentru fiecare pin, unde 1 reprezintă stare înaltă (adică vom măsura 5V pe respectivul pin), iar 0 stare joasă (adică vom avea 0V). Dacă, de exemplu, dorim să avem stare înaltă pe pinii 0, 1 și 2 ai portului B și stare joasă pe restul, vom scrie PORTB=0b00000111 (unde b indică scrierea în binar) sau putem scrie în hexazecimal 0x07. De asemenea, putem referenția direct un anumit pin al unui port, sub forma PORTB.x, unde x este numărul pinului. Astfel, exemplul anterior se poate reformula în opt instrucțiuni separate: PORTB.0=1, PORTB.1, PORTB.2=1, PORTB.3=0, ..., PORTB.7=0. Se folosesc aşadar, pentru generarea ieșirilor.

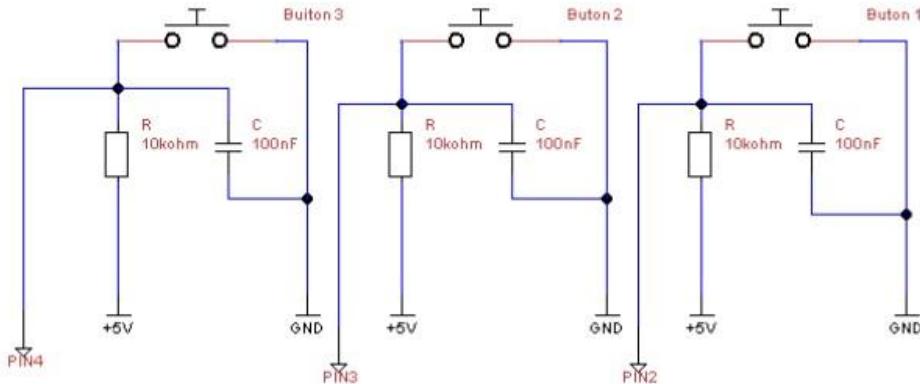
- Regiștrii PINB, PINC, PIND („Port Input Pins Address”) se folosesc pentru citirea prin adresare directă a valorii unui pin. La fel ca la registrul anterior, putem accesa fie întregul registru, citindu-l ca un număr binar pe 8 biti, sau putem citi doar câte un pin. De exemplu, operația $a = PIND$ va atribui valoarea 7 variabilei a , dacă pinii 0, 1 și 2 ai portului D sunt 1, iar restul 0. Dacă, în concordanță cu situația din exemplul anterior, vom scrie $a = PIND.5$, variabila a va primi valoarea 0 întrucât bitul 5 este 0, doar primii trei fiind 1.

În lucrul cu pinii definiți ca intrare va trebui să ținem cont de următoarele considerente: pentru a furniza o stare activă unui anumit pin, va trebui să punem acel pin la masă și astfel să citim valoarea 0, întrucât, în lipsa închiderii la masă, tensiunea pe acel pin are o valoare flotantă (high-Z) sau 5V dacă acesta este *tras la plus* (trad. eng. *pulled-up*) cu o rezistență de pull-up externă. Este recomandată folosirea rezistențelor de pull-up pentru a evita citirile incorecte datorate zgomotului. Această rezistență are valori de ordinul a câțiva kOhm până la câteva zeci de kOhm și leagă pinul respectiv la potentialul Vcc (5V) în lipsa unei scurtcircuitări a acestuia la masă. În momentul scurtcircuitării la masă, potențialul pe acel pin va fi 0V, iar prin rezistența de pull-up va circula un curent foarte mic, datorită valorii mari a acesteia. În general, trebuie să reținem că intrările sunt active în 0, iar ieșirile sunt active în 1.

1.2.2 Folosirea circuitului de debounce la citirea butoanelor și întrerupătoarelor

Datorită naturii mecanice a întrerupătoarelor și a butoanelor, fenomenul de comutare nu este perfect, iar tensiunea pe contactul închizător nu variază sub forma unui impuls treaptă (ca în cazul ideal), ci mai degrabă sub forma unui semnal distorsionat, ce poate avea câteva variații largi într-un timp foarte scurt. Chiar dacă la aprinderea unui bec sau a unui LED, acest lucru nu este sesizabil, atunci când, de exemplu, avem o funcție care contorizează numărul de apăsări ale unui buton, este posibil ca la o singură apăsare, acel număr să se incrementeze de câteva ori, chiar zeci de ori.

„O problemă care apare în cazul utilizării unei elemente de tip buton sau comutator este fenomenul de bounce. Datorită imperfecțiunilor mecanice ale contactelor electrice la închiderea sau la deschiderea elementelor apare un fenomen de oscilație a semnalului care poate conduce la citirea eronată a comenzi (citire multiplă). Pentru a evita această situație se utilizează un condensator (schema de mai jos) pentru a netezi semnalul de trecere dintr-un nivel logic în celălalt.” [Elemente practice de bază în dezvoltarea sistemelor cu microprocesoare integrate, ultima accesare: 01.07.2017]



Sursa: [Elemente practice de bază în dezvoltarea sistemelor cu microprocesoare integrate, ultima accesare: 01.07.2017]

1.2.3 Utilizarea timerelor

MC-ul ATMEGA8 dispune de trei timere, două pe 8 biți și unul pe 16 biți. Un timer este, din punct de vedere logic, un numărător definit pe un anumit număr de biți, a cărui incrementare este comandată de fronturile pozitive ale semnalului de ceas, fie direct, fie prin intermediul unui prescaler, care este un circuit de divizare a frecvenței semnalului de ceas. Mărimea unui prescaler se notează în felul $1:X$ sau X , unde X este factorul de divizare a frecvenței de ceas sau, în alte cuvinte, $X-1$ este numărul de perioade „sărite”. [Manual de utilizare ATmega8, 2011]

„Scopul prescalării este să se crească domeniul de numărare (temporizare ca număr întreg de ciclii de ceas de intrare). Cu cât încercăm să creștem domeniul de numărare al unui temporizator, cu atât vom pierde în precizia cu care măsurăm intervalele de timp. De exemplu, dacă folosim un timer pe 16 biți, iar perioada ceasului este de 1 microsecundă, valoarea maximă ce poate fi măsurată este de 65536 microsecunde = 65,5 milisecunde (2^{16}), care poate fi crescută doar prin prescalarea ceasului sau prin mărirea dimensiunii, în număr de biți a, timerului.” [Microprocesoare și microcontrolere, 2015]

Instrucțiunile specifice lui AMTEGA8 pentru lucrul cu timere

- TIMSK – Registrul de mascare a întreruperilor pentru timere (pe 8 biți). Este folosit pentru activarea și dezactivarea întreruperilor pentru fiecare dintre timere, în funcție de diferite evenimente asociate cu acestea. De exemplu, valoarea 0x04 a acestui registru activează întreruperea la depășirea timerului 1 (TIM1_OVF), iar valoarea 0x00 dezactivează toate întreruperile pentru timere.
- TCCR0, TCCR1B, TCCR1A, TCCR1H, TCCR1L și TCCR2 sunt registrii de control ai celor trei timere (pentru timerul pe 16 biți având la dispoziție diferite configurații). Aceștia sunt folosiți pentru a configura sursa semnalului de ceas pentru respectivul timer (semnal extern de la pinii T0 sau T1, la front crescător sau la front descrescător, ori semnalul de ceas întreg, sau divizat cu 8, cu 64, cu 256 sau cu 1024, sau niciun semnal de ceas – adică timer oprit). De exemplu TCCR1B=0x07 activează timerul 1 cu sursa de la semnal extern pe pinul T1, iar TCCR1B=0x00 oprește timerul.
- TCNT0, TCNT1 și TCNT2 sunt registrii (pe 8, respectiv 16 biți pentru timerul 1) ce conțin valoarea instantanee a fiecărui timer și oferă proiectantului acces direct la citirea/scrierea acestor valori.
- TIFR – registrul de „flag-uri” al întreruperilor timerelor. Este un registru pe 8 biți, ai cărui biți cu numerele 6, 2 și respectiv 0, se setează automat la valoarea 1 în momentul în care intervene o

întrerupere de depăşire (overflow) la timerele 2, 1 și respectiv 0. [Manual de utilizare ATmega8, 2011]

1.2.4 Utilizarea întreruperilor

„În timpul rulării programelor pot apărea unele evenimente neobișnuite (de excepție). Aceste evenimente pot conduce la suspendarea temporară a programului aflat curent în execuție; în acest interval de întrerupere a rulării programului se va executa, de obicei, o rutină de tratare a evenimentelor ce au întrerupt programul. Toate aceste evenimente produse de condiții neobișnuite / neașteptate pentru programul curent în execuție sunt numite la modul general "întreruperi" pentru că ele produc întreruperea programului și devierea / saltul către o rutină specială de tratare a evenimentului "întreruptor". Putem clasifica aceste evenimente «întreruptoare» în:

1. cereri de întrerupere: evenimente generate din exteriorul UCP, asincrone cu programul rulat, care cer tratare. Aceste cereri de întrerupere pot proveni:

- a. de la echipamente periferice care cer servicii de la UCP (cerere de transfer de date, sau informări cu privire la starea perifericului), sau
- b. de la circuite specializate pentru supravegherea funcționării normale a componentelor hardware (eroare de paritate la citirea memoriei, eroare de paritate pe magistrală, cădere iminentă a tensiunii de alimentare etc.);

Evenimentele ce produc acest tip de întrerupere a programelor sunt numite și întreruperi hardware.” [Microprocesoare și microcontrolere, 2015]

În proiectul nostru, dorim să folosim întreruperile generate de intrarea depăşire a timerelor, pentru a incrementa o variabilă ce stochează numărul de depăşiri. Folosind acest număr de depăşiri, avem două situații posibile:

- La măsurarea frecvențelor, unde timerul TIM1 are setat ca sursă semnalul aplicat la intrarea T1 (pe front crescător), noi numărăm de fapt perioadele ale acestui semnal în decursul unei durate exacte. Cum în această durată, în funcție de frecvență semnalului, numărul de perioade poate depăși valoarea maximă a timerului (moment în care timerul repornește din valoarea 0), dorim ca la fiecare depăşire a acestuia să incrementăm variabila ce stochează numărul de depăşiri. Această incrementare este realizată în rutină de servire a întreruperii TIM1_OVF. Așteptarea duratei de măsurare (de exemplu, 1 secundă) se face prin instrucțiunea delay_ms (număr_de_milisecunde), timp în care unitatea centrală de procesare este blocată într-o buclă de așteptare, însă timerul continuă să funcționeze, iar această buclă de așteptare poate lua pauze (înaintea scurgerii timpului) doar prin intervenția unei întreruperi care va lansa rutina de servire a acesteia. După execuția rutinei, bucla este reluată exact de la momentul de timp la care fusese întreruptă. Așadar, după așteptarea duratei, timerul va fi oprit, variabila ce stochează numărul de depăşiri se va înmulții cu valoarea maximă a timerului, apoi se va aduna „restul”, adică valoarea la care a rămas timerul înaintea opririi. Acest rezultat reprezintă numărul total de perioade ce au stimulat timerul și, împărțind-o la perioada de timp a buclei, exprimantă în secunde, obținem nimic altceva decât o mîrime de tip frecvență [Hz].
- La măsurarea duratelor, folosim numărul de depăşiri ale timerului TIM0, fiecare depăşire reprezentând o durată de timp exactă, calculată în funcție de factorul de prescalare folosit și frecvența de ceas la care lucrează MC-ul. Își în acest caz, vom folosi rutina de servire a întreruperilor timerului TIM0, și anume TIM0_OVF.

Registrul lui ATMEGA8 specific lucrului cu întreruperi este MCUCR – Registrul de control al microcontrolerului. Definit pe 8 biți, acest regisztr permite configurarea condițiilor de declanșare a întreruperilor externe INT0 și INT1 (front crescător, front descrescător etc.), prin biții 0-3, precum și configurarea tipurilor de adormire, prin biții 4-6 și respectiv activarea sau dezactivarea funcției de adormire, prin bitul 7.

1.2.5 Setarea fuse-biților CKSEL pentru alegerea frecvenței și a tipului de ceas

Fuse-biții alcătuiesc un regisztr de configurare stocat într-o zonă de memorie distinctă. Ei nu pot fi accesati în cadrul programului asemenea altor registri, ci trebuie citiți și modificați (dacă este cazul) din meniul care se ocupă cu scrierea programului în microcontroler, printr-o opțiune distinctă. Acești fuse-biți se referă la activarea sau dezactivarea anumitor funcționalități ale MC-ului (de exemplu, WDTON – „Watchdog Timer ON”, RSTDISBL – Reset Disable), la configurarea timpului de pornire (biții SUT0 și SUT1 – „Start-Up Time”) și la configurarea tipului de ceas folosit, precum și a frecvenței acestuia (biții CKSEL3...0). Un microcontroler nou, nefolosit, are deja acești biți setați cu niște valori prestabilite și menționate în manualul de utilizare, însă în funcție de necesitățile fiecărui proiectant, aceștia se vor modifica în mod corespunzător.

Ceea ce ne interesează în lucrarea de față este cum trebuie să modificăm biții CKSEL, întrucât vom dori ca modulele proiectului să lucreze la diferite frecvențe, cu siguranță nu doar la valoarea prestabilită de 1MHz. O descriere a semnificației acestor biți este rezumată în manualul de utilizare al ATmega8 la pagina 26.

Table 2. Device Clocking Options Select⁽¹⁾

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

Note: 1. For all fuses "1" means unprogrammed while "0" means programmed

Sursa: [Manual de utilizare ATmega8, 2011]

În cazul folosirii oscilatorului RC intern, avem următoarele variante pentru biții CKSEL3-CKSEL0: 0001 pentru 1 MHz, 0010 pentru 2 MHz, 0011 pentru 4 MHz și 0100 pentru 8 MHz. În cazul folosirii unui oscilator extern cu cuarț, avem două situații diferite:

- cu bitul CKOPT=1, biții CKSEL3...0 pot fi: 1011, pentru frecvențe între 0,4 și 0,9 MHz, 1101, pentru frecvențe în intervalul 1 MHz – 3 MHz, 1111 pentru frecvențe în intervalul 3 MHz – 8 MHz;
- cu bitul CKOPT=0, valorile lui CKSEL3...0 pot fi oricare dintre combinațiile de mai sus, pentru frecvențe cuprinse între 1 și 16 MHz.

Bitul CKOPT realizează selecția dintre două moduri de amplificare a oscilatorului. Cu valoarea 0, acest bit permite operarea la frecvențe mai mari (de până la 16 MHz) și în medii mai zgomotoase electromagnetic. Este de menționat faptul că 0 înseamnă programat (activat), iar 1 înseamnă neprogramat.

Aici este o ocazie bună să adăugăm mențiunea că, în cazul folosirii oscilatorului cu cuarț, producătorul recomandă conectarea a doi condensatori în paralel cu fiecare pin al oscilatorului și masă, valoarea acestuia situându-se în intervalul 12-22pF. [Manual de utilizare ATmega8, 2011]

1.2.6 Utilizarea software-ului CVAVR pentru scrierea programelor și încărcarea acestora în MC

CodeVisionAVR este, după cum se recomandă în informațiile de la secțiunea *Help – About* a programului, un mediu de programare integrat (IDE – Integrated Development Environment), compilator de limbaj C, Generator automat de program și programator de microcontrolere din seria AVR de la Atmel. Penultima mențiune se referă la faptul că acesta poate genera automat anumite secvențe de initializare, asociate diferitelor funcționalități, după ce utilizatorul a selectat acele resurse pe care dorește să le utilizeze, iar această generare se face automat în funcție de tipul de microcontroler ales, dat fiind că programul suportă o gamă largă a familiei AVR. Ultima mențiune se referă la faptul că acest program oferă posibilitatea încărcării programului în microcontroler folosind opțiunea „Chip Programmer”, prin intermediul unei interfețe de programare. În cazul nostru, vom folosi o interfață de programare de tipul „AVRISP Mk2”



Fig. 1.1 – Interfață de programare de tipul *AVRISP Mk2*

Aceasta se conectează la portul USB al PC-ului și la microcontroler, prin intermediul unui conector care este legat la pinii 1, 17, 18, 19 ai microcontrolerului ATMEGA 8 și masă, după schema de mai jos. Semnificația pinilor din conectorul interfeței este redată în figura, care este cu vedere din spate (dinspre fire).

MOSI: 1 (unused): 3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2 :Vcc
RESET: 5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	4 :Ground
SCK: 7	<input checked="" type="checkbox"/>	<input type="checkbox"/>	6 :Ground
MISO: 9	<input checked="" type="checkbox"/>	<input type="checkbox"/>	8 :Ground
			10:Ground

[Sursă: Batsocks.co.uk, ultima accesare: 28.06.2017]

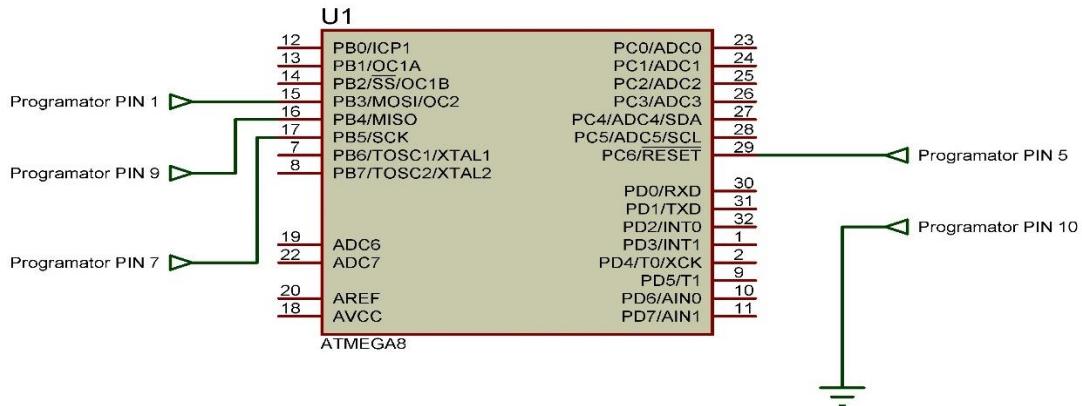


Fig. 1.2 – Schema de legătură pinii programator - microcontroler

Ceea ce dorim în mod deosebit să specificăm, referitor la folosirea acestui program, este felul în care vom face încărcarea programului în MC, întrucât înțelegerea incompletă a acestei proceduri se poate solda cu blocarea iremediabilă a acestuia.

În momentul în care programul a fost compilat și asamblat cu succes, lansăm opțiunea *Chip Programmer* din meniul *Tools*, sau folosind combinația de taste Shift-F4. Va apărea o fereastră ca cea de mai jos.

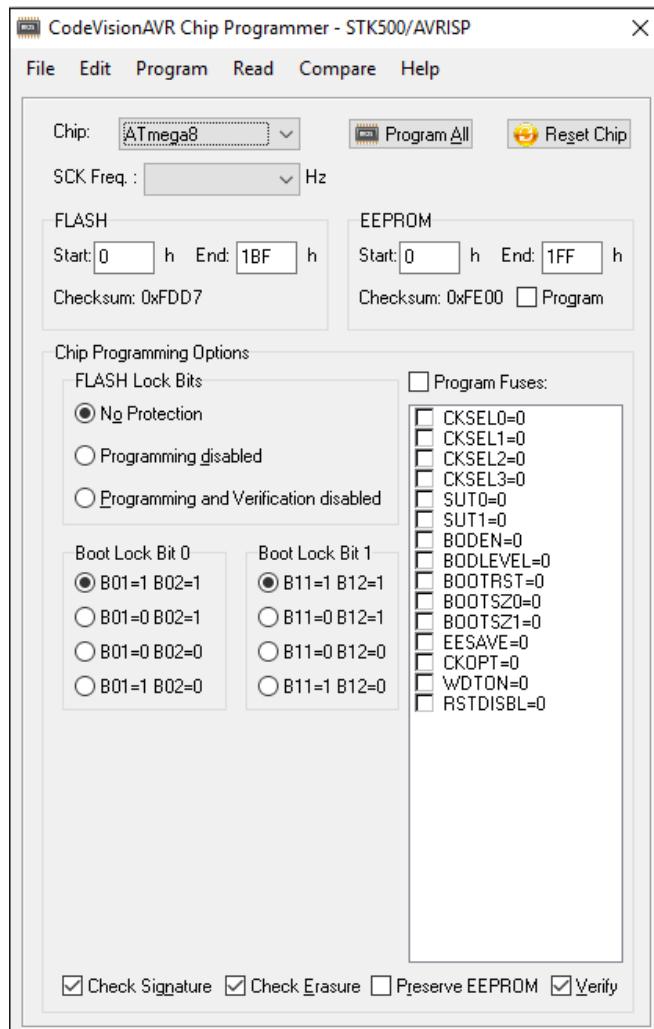


Fig. 1.3 – Captură ecran – opțiunea *Chip Programmer*

La secțiunea *Chip*, deschidem lista și selectăm *ATmega8*. La *SCK Freq* selectăm frecvența de 230400 Hz, aceasta fiind frecvența interfeței seriale a MC-ului prin intermediul căreia se face scrierea programului în memoria FLASH. Vom bifa opțiunea *Program Fuses* doar dacă MC-ul este folosit pentru prima oară și dorim să lucreze la altă frecvență decât cea prestabilită, de 1 MHz cu oscilatorul intern. În acest caz, la această secțiune se vor seta fuse-biții despre care am discutat mai devreme. Vom observa că există o listă a tuturor acestor biți, cu denumirea fiecărui și având atașată expresia = 0. Acest lucru înseamnă că vom bifa doar acei biți pe care dorim să îi setăm cu valoarea 0, iar toți biții care rămân nebifați vor primi automat valoarea 1, indiferent de valorile deținute anterior. Mai departe, este indicat să bifăm opțiunile *Check Signature*, *Check Erasure* și *Verify*, iar restul secțiunilor nemenționate se lasă neschimbate. Vom apăsa apoi butonul *Program All*, care va lansa procedura de încărcare a programului în MC, care ar trebui să dureze în jur de 4-5 secunde. Pentru a evita anumite probleme asociate setării accidentale a unor valori incorecte pentru fuse-biți, începând cu următoarele încărcări de program (pentru același exemplar de microcontroler), se va debifa opțiunea *Program Fuses*. Putem, de asemenea, pentru verificare, să citim valorile acestor biți, tot din fereastra *Chip Programmer*, selectând din meniul *Read* opțiunea *Fuses*.

1.3 Senzori fotosensibili

O clasificare minuțioasă a celor mai întâlnite tipuri de senzori din lumea tehnică poate fi găsită în cartea „Senzori și traductoare” a Dr. Ing. Călinou. Mai exact, în capitolul șase se regăsește definirea senzorului fotosensibil menționat anterior, alături de descrierea fenomenelor fizice ce stau la baza funcționării acestuia.

„[...] Există și efect fotoelectric intern, care constă în generarea unor purtători de sarcină liberi în interiorul unui semiconductor sub acțiunea radiației electomagnetice, schimbându-i astfel proprietățile de conducție electrică. [...] Transformarea radiației luminoase în semnal electric se bazează pe unul din următoarele fenomene: fenomenul de fotoconducție, efectul fotovoltaic și fenomenul fotoemisiv.

Fenomenul de fotoconducție (fotoconductibilitate), sau efect fotoelectric intern, constă în modificarea conductivității electrice a unui semiconductor sub acțiunea radiației electomagnetice (luminii). Ca urmare a absorbției radiației electomagnetice, au loc tranziții electronice, care duc la modificarea conductivității electrice. Excitarea electronilor datorită ciocnirii fotonilor duce la trecerea lor din banda de valență în banda de conducție, crescând atât concentrația electronilor liberi, cât și a golurilor. Saltul electronilor are loc dacă energia fotonilor incidenti este mai mare decât banda interzisă. Există mai multe tipuri de fotoconducție: fotoconducția intrinsecă, fotoconducția de impurități și fotoconducția purtătorilor liberi.” [Senzori și traductoare, 2009]

De asemenea, autorul atinge puncte importante în descrierea sa, ajungând la un alt subiect de interes pentru această lucrare, cum ar fi elementele sensibile la radiații luminoase. „Elementele sensibile care convertesc radiația luminoasă în semnal electric se numesc fotodetectoare, elemente fotosensibile sau fotosenzori. [...] În funcție de modul în care se obține semnalul electric, fotodetectoarele se împart în fotodetectoare de tip generator și fotodetectoare de tip parametric. [...] Fotodetectoarele de tip parametric au nevoie, pentru generarea unui semnal electric la ieșire, de o sursă de tensiune suplimentară. Radiația luminoasă incidentă pe suprafața activă a fotodetectorului modulează un parametru de circuit electric. Cele mai utilizate fotodetectoare sunt fotorezistente, fotodiodele și fototranzistoarele. [...] Fototranzistoarele sunt elemente fotosensibile asemănatoare din punct de vedere funcțional cu tranzistorul clasic. Fototranzistorul are în structura sa o plăcuță

semiconductoare din Ge sau Si, în care sunt realizate regiuni cu conducție de tip „n” și de tip „p”. În funcție de alternația acestor regiuni, fototranzistoarele sunt realizate în două variante, „npn” și „pnp”, jonctiunea bază-colector fiind o fotojonctiune.

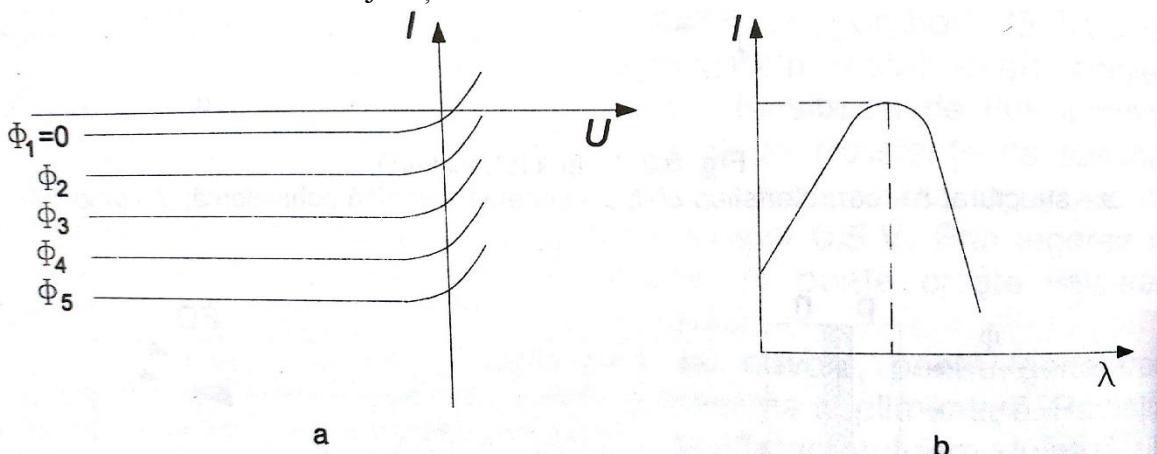


Fig. 6.5. Caracteristicile fotodiodelor:
a – caracteristica curent–tensiune; b – caracteristica de sensibilitate spectrală.

Sursa: [ibid]

Funcționarea fototranzistoarelor este asemănătoare cu funcționarea fotodiodelor, cu deosebirea că sensibilitatea este de 100...500 de ori mai mare, datorită amplificării în curent. [...]

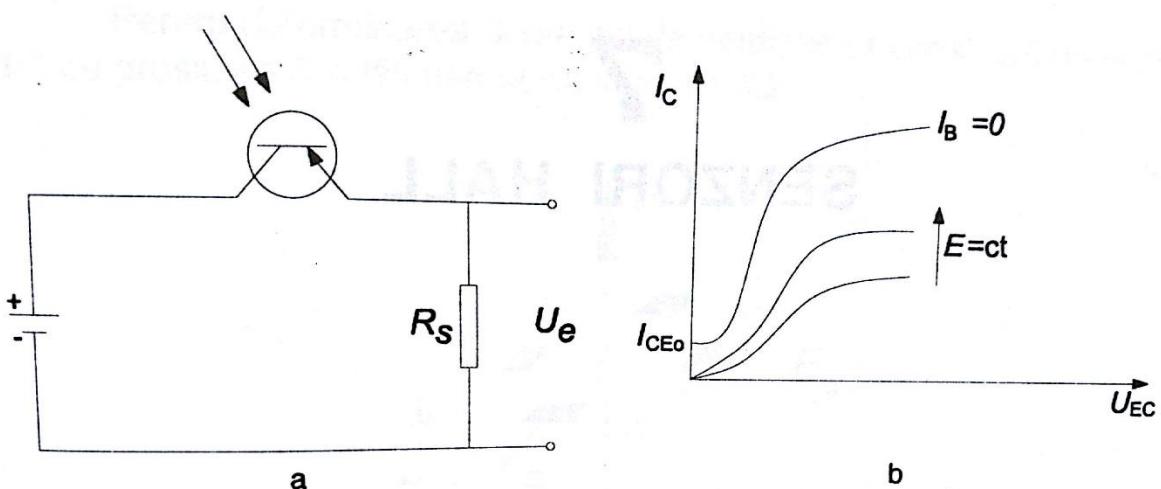


Fig. 6.6. Fototranzistor tip npn:
a – simbolul și modul de conectare; b – caracteristica U – I .

Sursa: [ibid]

Fototranzistoarele sunt utilizate ca fotodetectori în dispozitivele de comandă automată, în realizarea optocuploarelor, în rețelele senzoriale ale roboților industriali etc.” [ibid]

Pornind de la fenomenul fizic, am ajuns la descrierea modului de funcționare al fototranzistorului. În cele ce urmează, vom ajunge la definiția senzorului de turație care folosește fototranzistor. În cel de-al nouălea capitol al lucrării menționate anterior, găsim o secțiune dedicată traductoarelor cu elemente

fotoelectrice. „Traductoarele de turărie cu elemente fotoelectrice folosesc ca elemente sensibile fotoelemente (celule fotoelectrice, fotodiode, fototranzistoare, fotorezistente), care detectează variațiile unui flux de lumină. Traductoarele au în structura lor o sursă de radiații luminoase în spectrul vizibil sau infraroșu, un disc solidar cu arborele a căruia turăie dorim să o măsurăm și elemente fotoelectrice. Pentru întreruperea fluxului luminos, discul este prevăzut cu repere (orificii sau zone transparente și opace). În figura 9.11 este prezentată schema de principiu a unui traductor de turăie cu întreruperea fluxului de lumină. [...] Impulsurile luminoase care ajung la fotoelemente sunt convertite, cu ajutorul unor circuite electronice, în impulsuri de natură unor tensiuni electrice.

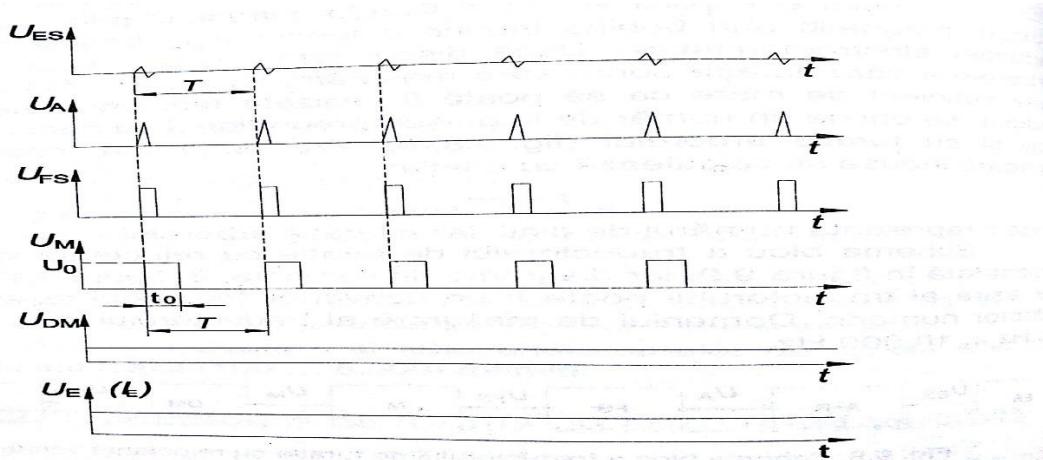


Fig. 9.10. Diagrama de semnale pentru traductorul de turăie cu reluctanță variabilă.

Sursa: [ibid]

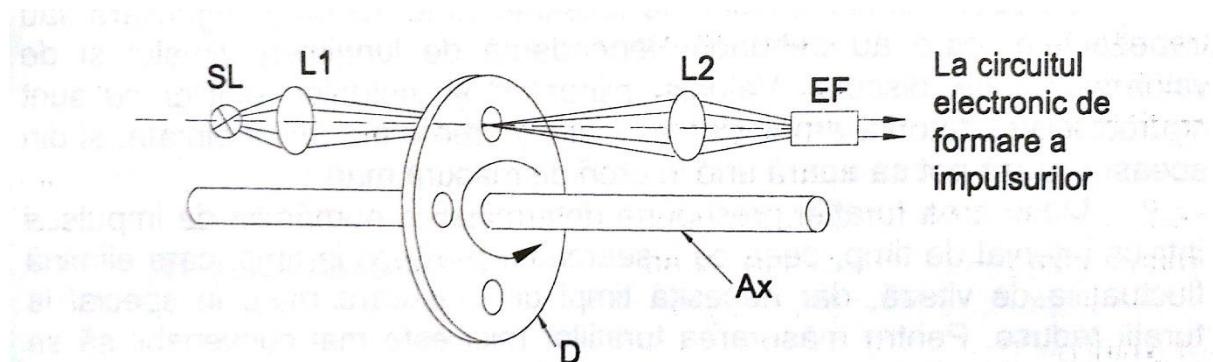


Fig. 9.11. Structura traductorului de turăie cu elemente fotoelectrice cu întreruperea fluxului luminos:

SL – sursă de lumină; L₁, L₂ – lentile; D – disc; EF – elemente fotoelectrice.

Sursa: [ibid]

Domeniul de măsurare al traductoarelor de turăie cu elemente fotoelectrice este 1...107 rotații / minut, în funcție de numărul de repere. Avantajele acestor tipuri de traductoare de turăie sunt: valoarea mică a puterii mecanice consumate (practic cuplul este nul) și posibilitatea utilizării schemelor numerice pentru adaptoarele electronice aferente, fără a fi necesară conversia analog-numerică a unei tensiuni.

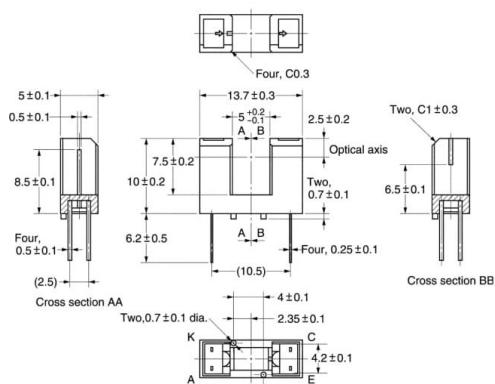
Principalele dezavantaje sunt: sensibilitatea traductorului la şocuri, vibraţii, praf şi lumină puternică.” [ibid] Cu acest citat am descris funcţionalitatea senzorului de turătie ce foloseşte un fototranzistor ca element sensibil. Sursa de lumină, în cazul senzorului folosit în lucrarea de faţă, este o dioda LED cu emisie în spectru infraroşu. Sunt evidenţiate elementele constructive ale lanţului de măsurare, iar în cazul nostru discul cu repere este discul de carton din corpul volantei, care foloseşte şirul de fante de pe marginile sale drept zone transparente, prin care să se realizeze întreruperea periodică a fluxului luminos dintre dioda LED şi fototranzistor.

Se va ține cont și de lista de dezavantaje, printre care amintim vibrațiile mecanice. În cazul nostru, suportul senzorului fiind montat pe aceeași planșă pe care este montat și motorul, va suporta o parte din vibrația produsă de motor și volanta în timpul funcționării, transmițând-o mai departe senzorului propriu-zis, astfel că vibrația reprezintă una dintre limitele constructive care afectează exactitatea procedeului ales pentru măsurarea turătiei. Mai jos se regăsește o fișă tehnică a senzorului de turătie folosită în această lucrare, creată și pusă la dispoziție de firma producătoare a acestuia.

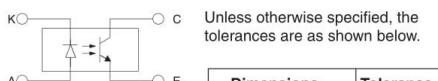
Photomicrosensor (Transmissive) EE-SX1137

Dimensions

Note: All units are in millimeters unless otherwise indicated.



Internal Circuit



Terminal No.	Name
A	Anode
K	Cathode
C	Collector
E	Emitter

Dimensions	Tolerance
3 mm max.	±0.3
3 < mm ≤ 6	±0.375
6 < mm ≤ 10	±0.45
10 < mm ≤ 18	±0.55
18 < mm ≤ 30	±0.65

Features

- General-purpose model with a 5-mm-wide slot.
- PCB mounting type.
- High resolution with a 0.5-mm-wide aperture.

Absolute Maximum Ratings (Ta = 25°C)

Item	Symbol	Rated value
Emitter	Forward current	I _F 50 mA (see note 1)
	Pulse forward current	I _{FP} 1 A (see note 2)
	Reverse voltage	V _R 4 V
Detector	Collector-Emitter voltage	V _{CEO} 30 V
	Emitter-Collector voltage	V _{ECO} ---
	Collector current	I _C 20 mA
	Collector dissipation	P _C 100 mW (see note 1)
Ambient temperature	Operating	T _{opr} -25°C to 85°C
	Storage	T _{stg} -30°C to 100°C
Soldering temperature	T _{sol}	260°C (see note 3)

Note: 1. Refer to the temperature rating chart if the ambient temperature exceeds 25°C.

- The pulse width is 10 µs maximum with a frequency of 100 Hz.
- Complete soldering within 10 seconds.

Ordering Information

Description	Model
Photomicrosensor (transmissive)	EE-SX1137

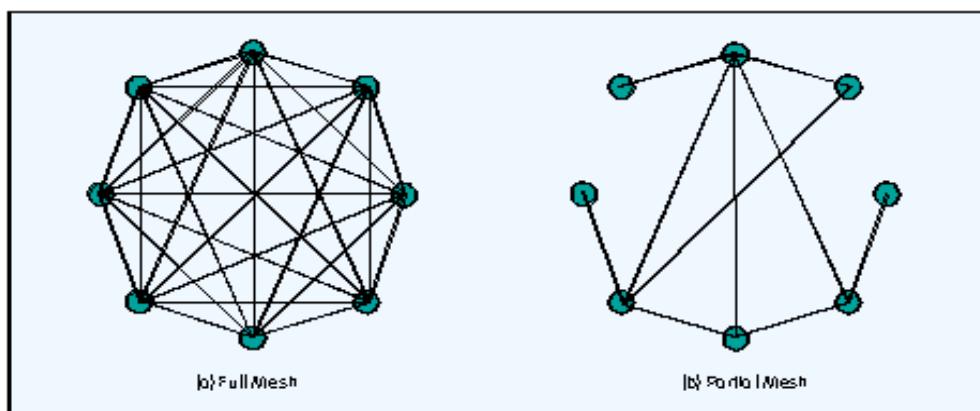
Electrical and Optical Characteristics (Ta = 25°C)

Item	Symbol	Value	Condition
Emitter	Forward voltage	V _F 1.2 V typ., 1.5 V max.	I _F = 30 mA
	Reverse current	I _R 0.01 µA typ., 10 µA max.	V _R = 4 V
	Peak emission wavelength	λ _P 940 nm typ.	I _F = 20 mA
Detector	Light current	I _L 0.5 mA min., 14 mA max.	I _F = 20 mA, V _{CE} = 10 V
	Dark current	I _D 2 nA typ., 200 nA max.	V _{CE} = 10 V, 0 lx
	Leakage current	I _{LEAK} ---	---
	Collector-Emitter saturated voltage	V _{CE(sat)} 0.1 V typ., 0.4 V max.	I _F = 20 mA, I _L = 0.1 mA
	Peak spectral sensitivity wavelength	λ _P 850 nm typ.	V _{CE} = 10 V
Rising time	tr	4 µs typ.	V _{CC} = 5 V, R _L = 100 Ω, I _L = 5 mA
Falling time	tf	4 µs typ.	V _{CC} = 5 V, R _L = 100 Ω, I _L = 5 mA

Capitolul de față poate fi interpretat ca unul intermediar, de legătură între secțiunea precedentă, de teorie, cu cea care va fi desfășurată următor în cel de-al treilea capitol, mai exact partea practică. Aici vom regăsi motivațiile pentru care am ales microcontrolerul ATMEGA 8, tot ce ține de pregătirea instrumentației, dar și mențiunile care mi s-au părut că trebuie amintite. În final, ultimul subcapitol l-am dedicat descrierii funcțiilor montajului, pentru care am dezvoltat implicațiile practice.

2.1 Motivația alegerii microcontrolerului ATMEGA 8

Am dorit realizarea unei arhitecturi modulare a ansamblului, astfel încât să avem o rețea de subansamble specializate pe anumite funcții, interconectate după o topologie de tip *Partial Mesh*. Aceasta face parte dintr-un set de topologii fizice. Varianta de Partial Mesh este folosită atunci când există noduri care nu sunt complet conectate cu celălalte noduri, față de varianta completă, *Full Mesh* [Zte.com.cn, ultima accesare: 20.06.2017]. După cum puteți vedea mai jos, Full Mesh (stânga) și Partial Mesh (dreapta).



▲ Figure 1. Mesh architectures.

Sursa: [ibid]

Am ales această variantă din mai multe motive:

1. Frecvența de lucru relativ redusă a microcontrolerului ATMEGA8, pusă în balanță cu necesitatea unei procesări destul de rapide (îndeosebi pentru măsurarea frecvențelor), este suficient de mare pentru realizarea cu exactitate suficient de bună a unei singure funcții principale, dar insuficiență pentru a acoperi realizarea mai multor task-uri.
2. Numărul de intrări declanșatoare de întreruperi (INTx) redus, este de asemenea suficient în cazul dedicării unei singure funcții principale.
3. Memoria de cod relativ redusă, ar fi insuficientă pentru a acoperi un program echivalent întregului ansamblu.
4. Numărul total de pini de intrare/ieșire ar fi insuficient pentru acoperirea întregului ansamblu.
5. Lipsa suportului pentru o interfață de comunicație rapidă și modernă, precum CAN, ne forțează să alegem o topologie de tip „mesh” în locul uneia de tip „bus”.

6. Lungimea și numărul firelor care realizează interconectarea sunt reduse, având în vedere că toate modulele se vor afla la o distanță destul de mică unele de celelalte, aşadar nici nevoie de flexibilitate în interconectare nu este una atât de mare.
7. Prețul redus al unui microcontroler nu reprezintă un impediment pentru decizia de a construi module separate, fiecare având câte un microcontroler.
8. Caracterul didactic al lucrării ne îndeamnă mai degrabă să alegem o arhitectură modulară, decât una concentrată, de tip „all-in-one”, astfel că putem urmări și înțelege mai ușor funcționalitatea fiecărui modul.

Alegând mai târziu un microcontroler mai performant din acele puncte de vedere care ne constrâng acum, putem „muta” fără probleme întreaga arhitectură într-un singur exemplar de acest fel, reatribuind funcțiile canalelor de intrare/ieșire ale microcontrolerelor separate către canalele unuia singur și economisind totodată canalele care erau destinate comunicării cu alte microcontrolere, astfel că în acest caz, schimbul de informații între diferitele funcții s-ar face la nivel de memorie și, bineînțeles, mult mai rapid. S-a urmărit în același timp să se demonstreze reușita exploatarii unor microcontrolere cât mai simple, în comparație cu ceea ce ne oferă piața la ora actuală, pentru a demonstra cât de puține resurse sunt necesare implementării unor funcții destul de complexe, cel puțin în vizionarea utilizatorului final. Se va evidenția calculul necesarului de resursă computațională pentru proiectul nostru, pentru a justifica decizia de a alege microcontrolerul ATMEGA8.

2.2 Instrumentar în vederea realizării practice

Privind aspectele materiale ale realizării practice, vom enumera instrumentarul minim necesar:

1. Plăci de test “breadboard”, pentru asamblarea rapidă a unor montaje sau porțiuni de montaje aflate în fază incipientă

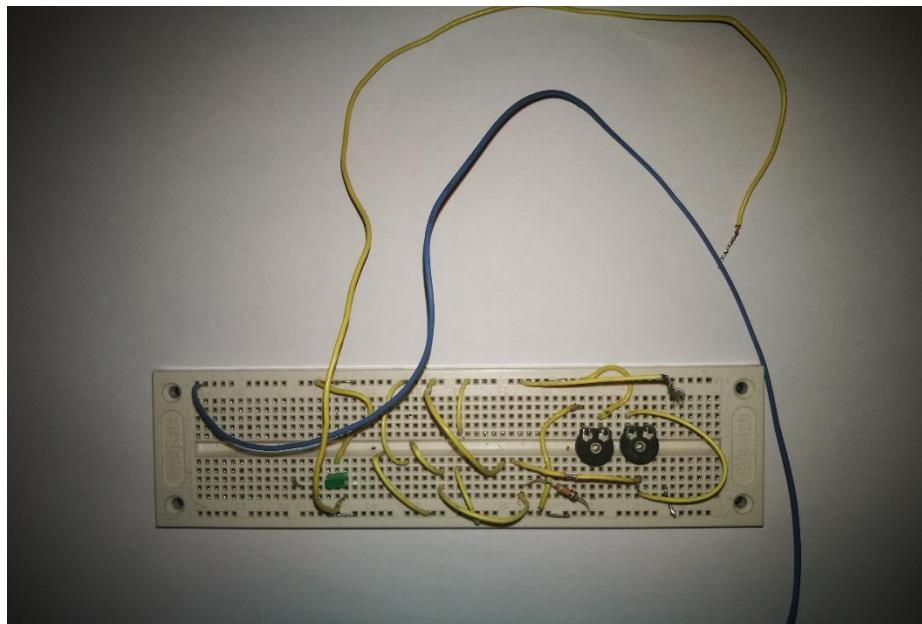


Fig. 2.1 – Placă test *breadboard*

Acestea reprezintă o resursă reutilizabilă, întrucât piesele și firele de legătură se montează și se demontează foarte ușor (prin înfigere/tragere), iar construcția plăcii nu suferă nicio

modificare în urma populării repetitive cu piese. Este totuși o resursă optională, fiindcă se pot folosi cablaje de test.

2. Cablaj de test (cablaj de textolit cu rețea ortogonală de paduri de cupru găurite, gata realizate). Este mai ieftină decât placa „breadboard”, însă popularea cu piese și realizarea legăturilor durează mai mult, presupunând și cositorirea lor. Este indicată pentru realizarea montajelor în faze intermediare, asupra cărora se va reveni de mai multe ori de-a lungul realizării proiectului. Este o resursă parțial reutilizabilă, ținând cont că presupune risipa de cositor la fiecare repopulare, iar după câteva repopulări padurile de cupru încep să se dezlicească de pe placă, făcând neutilizabilă acea poziție.

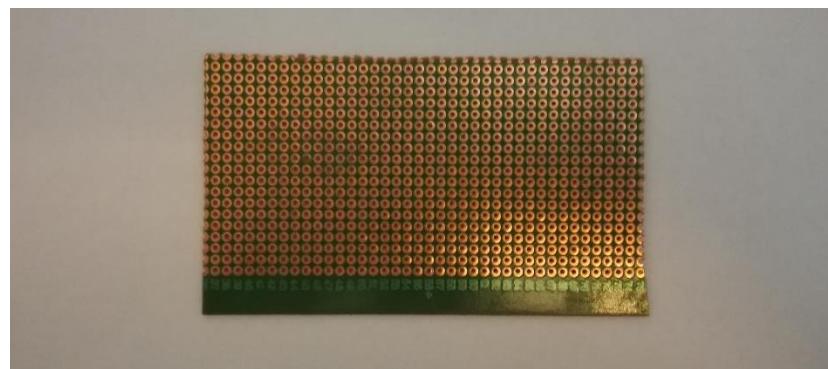


Fig. 2.2 – Cablaj de test

3. Cablaj final, comercializat sub forma unei plăci de textolit acoperite cu un strat uniform de cupru pe o față sau pe ambele fețe (pentru montajele „double sided”). Aceasta presupune desenarea traseelor conform schemei finale și verificate, urmată de un proces de imprimare a circuitului și decapare a zonelor dintre trasee, pentru care există mai multe metode. Este o resursă pe care unii entuziaști o evită, fiind de asemenea optională întrucât se poate folosi tot cablaj de test pentru realizarea variantei finale a montajului.



Fig. 2.3 – Cablaj de textolit acoperit cu strat uniform de cupru

4. Letcon sau stație de lipit, cositor, pastă decapantă/sacâz;

5. Cabluri de diferite secțiuni pentru realizarea conexiunilor;
6. Multimetru digital, de preferință TRUE RMS și cu măsurare de frecvență. Având în vedere că majoritatea semnalelor de interes sunt tensiuni continue, se pot folosi multimetre fără specificație TRUE RMS, însă în acest caz vom avea nevoie de osciloscop pentru vizualizarea semnalelor dreptunghiulare, aferente măsurărilor de turăție a motorului.
7. Osciloscop analogic sau digital, având orice limită de frecvență. Luând în considerare faptul că majoritatea, incluzându-le pe cele mai vechi sau mai ieftine, au o limită de cel puțin 10MHz, iar semnalele noastre sunt dreptunghiulare și au frecvențe maxime de ordinul zecilor și sutelor de kHz (exceptând semnalul de CLOCK – 8 sau 16MHz), orice fel de osciloscop este suficient atât timp cât este perfect funcțional și nu are probleme de sincronizare.



Fig. 2.4 - Osciloscop

8. Sursa de tensiune continuă stabilizată, reglabilă, preferabil cu limitare de curent. Pentru a acoperi intervalul de tensiuni din domeniul auto, este de preferat să atingă valoarea de 14,4 volți. În privința curentului maxim debitat, cu condiția să aibă limitare de curent, cu cât mai mare cu atât mai bine. Un curent maxim de 4-5 A este suficient pentru teste și simulări folosind sarcini de valori moderate.



Fig. 2.5 –Sursă tensiune

- Este binevenit un generator de semnal dreptunghiular ce poate lucra în intervalul [1Hz – 100kHz], dacă dorim să simulăm prezența senzorului de turație motor. În lipsa acestuia, se poate realiza foarte ușor unul, folosind circuitul „555” și având la dispoziție o abundență de scheme, atât pe Internet cât și în cărți de specialitate publicate oricând după anii '70.



Fig. 2.6 - Generator semnal sinusoidal și dreptunghiular cu afișarea frecvenței și reglaj fin

- Scule de uz general: șurubelnițe drepte/cruce, clești, patent, pile, cutter, un obiect ascuțit pentru eventualele curățări ale cablajului, pânză de fierăstrău, ciocan și punctator, bormașina, burghie de dimensiuni mici pentru găurirea cablajului final.
- Materiale de uz general: PFL, placaj sau materiale plastice pentru realizarea unei planșe de lucru (atenție la temperaturile atinse în timpul funcționării !), cutii de plastic sau metalice pentru

realizarea carcaselor de prototip, șuruburi, piulițe, șaipe, colțare pentru asamblări, cabluri de diferite secțiuni, alese după curentul maxim suportat, elemente de conectica (clești « crocodil », borne – banane, mufe mama/tata, papucei, socluri etc.), materiale pentru izolat (bandă izolatoare, varniș termocontractabil), radiatoare pentru elementele de putere (sau materiale metalice pentru realizarea acestora).

După cum se poate observa, necesarul de instrumente de măsură nu este unul foarte pretențios, bucurându-ne de tensiunile joase, de frecvențele reduse ale semnalelor de interes și de faptul că, în mod normal, nu va fi necesar să măsurăm semnalul de CLOCK. De asemenea, necesarul de materiale expus anterior este unul foarte acoperitiv și enumeră elemente de uz general, accesibile atât prin disponibilitatea mare în magazinele de specialitate cât și ca preț.

2.3 Mențini în vederea realizării montajului practic

Pentru realizarea practică a întregului montaj, vom considera că am realizat deja, măcar parțial, următoarele aspecte:

- împărțirea funcțiilor pe module (subansamble);
- realizarea unei scheme-bloc a fiecărui subansamblu, evidențiind pentru fiecare dintre acestea următorii parametri:
 - frecvența de lucru și alegerea tipului de ceas (intern sau extern, cu quart);
 - semnalele de intrare și de ieșire (dinspre/către alte module, precum și dinspre senzori/către actuatori);
 - alegerea unor etichete cât mai sugestive pentru aceste semnale;
- scrierea programului, măcar în pseudocod ori direct în cod C pentru CAVR dacă este posibil, explotând orice resursă a microcontrolerului care ne-ar putea ajuta la o implementare cât mai eficientă a funcțiilor (de exemplu, timer-ul intern, intrările de întrerupere etc.);
- alegerea unor denumiri cât mai sugestive pentru diferențele variabile, constante și nume de funcții din cadrul programului. Se va acorda o atenție deosebită denumirilor stărilor, și se vor folosi aceleași nume de stări în mai multe module, în situația în care acestea implementează împreună un anumit automat finit;
- realizarea schemei electrice, măcar parțială, pentru fiecare modul, pornind de la schema-bloc realizată inițial;
- implementarea schemei pe un cablaj de test sau modul(e) breadboard și pregătirea pentru punerea în funcționare;
- alegerea unor metode de stimulare și testare individuală a fiecărui modul, astfel încât dezvoltarea acestuia să nu depindă de starea dezvoltării celuilalt sau celorlalte module cu care acesta trebuie să interacționeze.

În momentul în care ne-am hotărât asupra temei pe care dorim să o realizăm, am avut în vedere faptul că ar fi convenabil ca montajul să poată fi pus în funcțiune și testat atât sub forma unei machete, cât și prin montarea acestuia pe mașina reală. S-a luat în considerare atât realizarea lucrării pentru susținerea proiectului de licență, cât și utilitatea practica a acestuia. Este evident faptul că, pentru prezentarea lucrării în fața comisiei, va fi necesară realizarea machetei, aceasta variantă fiind aşadar prioritată.

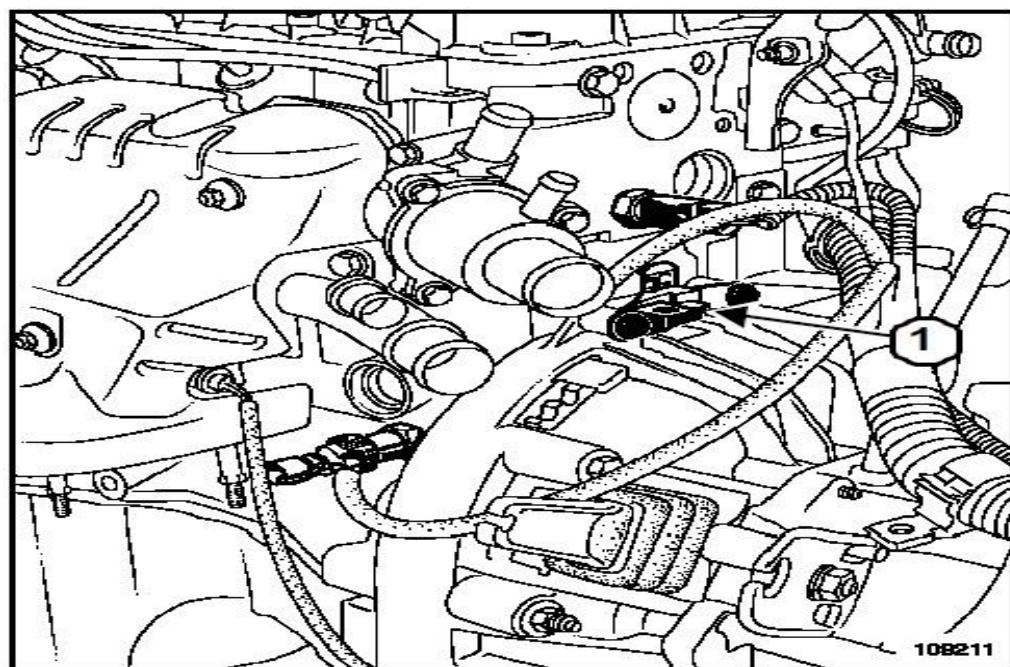
Pentru realizarea machetei, trebuie să ne gândim care anume dintre componentele mașinii trebuie să le simulăm și cum anume. Releele de ACC, IGN și DEM pentru mașină vor fi alese după tensiunea de alimentare a bobinei (12V), și după curentul maxim suportat. În cazul machetei, niște relee oricără de „mici” (cățiva amperi) sunt suficiente pentru a alimenta niște actuatori de mică putere. Demarorul va putea fi modelat foarte simplu printr-un motor electric de mică putere (până la urmă, chiar și un LED sau bec ar fi suficient, important este să îi putem evalua vizual starea).

Senzorul de turăție folosit pe mașină este de tip magnetic, produs de firma Siemens și este exemplificat mai jos.



Sursa: [Blogspot.com, ultima accesare: 20.06.2017]

El este montat pe blocul motor în dreptul volantei, astfel că senzorul „numără” dinții de pe circumferința acesteia, după cum se poate vedea în imaginea mai jos.



Sursa: [Manual service Logan]

O fotografie mai detaliată a volantei este exemplificată mai jos.



Sursa: [Dezmembrarimasini.ro, ultima accesare: 20.06.2017]

Pentru machetă putem folosi, de asemenea, un senzor hall, sau chiar un senzor optic, alcătuit dintr-un LED cu emisie în infraroșii și un fototranzistor, plasate de o parte și de alta a corpului de rotație monitorizat. Corpul de rotație va avea atașat un disc pe periferia căruia se vor decupa una sau mai multe găuri. Ansamblul senzorului optic se va alinia cu șirul de găuri existente pe disc, astfel încât, în timpul rotației discului, succesiunea găurilor să producă un șir de obturări a fluxului optic de la LED la fototranzistor. Ambele feluri de senzori generează un semnal dreptunghiular, mai mult sau mai puțin distorsionat, fiecare perioadă a acestuia semnificând o tranziție. Dacă pe periferia discului este practicată o singură gaură, o tranziție va semnifica o rotație. Dacă sunt practicate 5 găuri, atunci un șir de 5 perioade din semnalul generat va corespunde unei rotații s.a.m.d., informații teoretice pe care le puteți găsi în Capitolul I.

Modulul de închidere centralizată poate fi înlocuit cu un număr de butoane, ținând cont de faptul că acesta are ieșiri diferite pentru fiecare buton de pe telecomandă, fiecare ieșire generând un impuls corespunzător butonului apăsat.

Orice fel de contactori montați pe mașină (contactor de punct mort, contactoare de pedală, contactorul frânei de mâină etc.) pot fi înlocuiți cu intrerupătoare.

Ajungem la componenta esențială a mașinii, pe care dorim, de asemenea, să o modelăm: motorul. Chiar dacă primele gânduri ne duc la lucruri foarte complicate, trebuie să ne amintim care sunt cerințele temei. Astfel, vom realiza că avem nevoie să modelăm doar o anumită parte din comportamentul motorului termic, nicidecum un amalgam de parametri complecși ce țin de regimul termodinamic, forțe, cuplu etc. Astfel, o scurtă descriere a acțiunilor de pornire, menținere în ralant (motor pornit) și oprire, și a stării de motor oprit, observate dintr-un punct de vedere strict mecanic, este:

1. Motorul este oprit. Ce se poate observa din punct de vedere mecanic? Faptul că nu se învârte, deci turația sa este nulă.
2. Motorul este în curs de pornire, acționat de demaror. Ce se poate observa? Motorul capătă o turație, dependentă de mai mulți factori: starea bateriei, starea de ungere a motorului,

temperatura exterioară, starea de uzura a traseului electric de curenți mari dintre baterie și demaror (cât de oxidație sunt contactele, deci cât de mari sunt rezistențele de contact), starea de uzură a demarorului. Această turație se poate menține mai mult sau mai puțin constantă pe durata pornirii. În oricare caz, de la cel mai favorabil la cel mai nefavorabil, va avea mereu o turație vizibil mai mică decât turația de ralanti a motorului pornit, în jurul a 200-400 rpm. De asemenea, durata pornirii poate fi mai lungă sau mai scurtă. O durată foarte scurtă, „la sfert de cheie”, cum se mai spune în jargonul mecanicilor auto, ar fi de circa 200-500 ms.

3. Motorul este pornit și se observă că acesta funcționează la turația normală de ralanti, care este de aproximativ 700 rpm la temperatura normală de lucru (motor cald), sau pana la 1500-1800 în condiții de temperaturi exterioare extrem de mici și motor rece.
4. Motorul se oprește natural, „din cheie”. Se observă că turația acestuia descrește până la 0, într-un timp cu atât mai lung cu cât se află la turații mai mari. Dacă motorul este „omorât”, ca urmare a incapacității de a genera cuplu într-o situație de debraiere bruscă, având cutia de viteze angajată într-o treaptă anume, sau într-o situație în care acesta întâmpină o forță de rezistență care crește brusc, fără a adapta raportul de demultiplicare schimbând în treapta de viteză potrivită, turația acestuia scade treptat până la o valoare la care nu se mai poate menține pornit, după care scade brusc. În oricare dintre aceste situații, finalitatea opririi motorului este o turație nulă.

Pe baza acestor observații, se dorește construirea un motor simulat, care să poată aproxima comportamentul motorului real în situațiile descrise mai sus și căruia să-i putem monitoriza turația.

2.4 Descrierea funcțiilor montajului. Implicații practice.

În acest subcapitol vom face o paralelă între configurația originală a autovehiculului Dacia Solenza și configurația la care se va ajunge în cazul în care acest montaj practic ar fi instalat pe mașină.

1. *Pornirea / oprirea motorului din butonul Start/Stop*, fără menținerea apăsării pe durata demarajului. Odată deschisă din telecomandă, se consideră că mașina a intrat în starea “deblocată și șofer autentificat”. Din acest moment și până la închiderea din telecomandă, este disponibilă funcția de pornire a motorului din buton. Butonul va fi luminat în două culori disponibile – verde și roșu. Cât timp motorul este oprit, lumina butonului va fi roșie și continuă. După acționarea butonului și până când motorul funcționează autonom, adică pe durata acționării demarorului, se va folosi o schemă de iluminare intermedieră care să indice acest lucru, de exemplu un verde clitor, sau un joc de alternanță verde-roșu, având o frecvență aleasă de aşa natură încât, chiar dacă motorul pornește foarte prompt, indicația luminoasă specifică acestei acțiuni să aibă timp să acționeze și să fie observabilă. Odată ce motorul devine autonom și demarorul se oprește, lumina butonului va fi verde și continuă. În cazul în care demarajul a eşuat (se va stabili un timp limită de acționare a demarorului) și nu avem configurația redemarajul automat, secvența de pornire va fi abandonată și butonul va lumina din nou roșu continuu. Dacă este configurația redemarajul automat, se va alege o altă schema de iluminare folosind cele două culori disponibile, pe durata de „odihnire” a demarorului, indicând șoferului că mașina are în intenție să relanseze pornirea motorului în scurt timp.

În denumirea funcției am folosit mențiunea „fără menținerea apăsării pe durata demarajului”, aceasta exprimând următorul lucru: pentru a iniția pornirea motorului, este suficientă o singură apăsare scurtă a butonului, urmând ca programul ce rulează la baza modulului de pornire să gestioneze în continuare

procesul demarajului, lăsând șoferului posibilitatea doar de a-și exprima intenția în privința stării motorului.

În mod similar, pe parcursul funcționării autonome a motorului, o singură apăsare a butonului va duce la oprirea promptă a motorului.

2. *Pornirea cu temporizare/oprirea motorului din telecomandă*. Această funcție este disponibilă atunci când mașina este staționată în apropiere de poziția actuală a utilizatorului mașinii, astfel încât distanța să se încadreze în aria de acțiune a telecomenției. Este utilă în orice situație în care se dorește pornirea și oprirea de la distanță a motorului, menținând totodată mașina încuiată. În general există două motive principale pentru care s-ar dori acest lucru: pentru încărcarea bateriei (pe timpul iernii, dacă bateria este mai veche și/sau dacă mașina este folosită mai rar) sau pentru încălzirea/răcirea în avans a habitaclului mașinii folosind instalația de climatizare sau chiar pentru dezghețarea parțială a lunetei, atunci când este anticipată folosirea mașinii în zecile de minute imediat următoare. Acest lucru ar aduce un plus de confort și chiar economie de timp utilizatorului, acesta nemaifiind nevoie să părăsească locuința mai devreme pentru a realiza aceste acțiuni, mai ales atunci când este foarte frig sau foarte cald afară. Fiind cunoscut faptul că variațiile dese și brusă de temperatură suportate de organism pot afecta serios sănătatea, putem considera că oferim și un plus de prevenție a îmbolnăvirii prin punerea la dispoziție a unei astfel de funcții.

Funcția oferă posibilitatea pornirii motorului din telecomandă și se ocupă de menținerea acestuia pornit pentru o durată fixă, de regula 10 minute (valoarea poate fi modificată din codul programului). După scurgerea acestui timp, motorul va fi oprit automat, iar mașina revine în starea inițială (încuiată, dacă era încuiată, sau desciuiață, în caz contrar). Dacă înainte de expirarea timpului se dorește oprirea motorului, acest lucru se poate efectua prompt, tot prin intermediul telecomenției.

Pentru pornire se va folosi un „cod” prestabilit, alcătuit dintr-o succesiune de apăsări de taste (de exemplu LOCK, LOCK, UNLOCK, LOCK, cu un timp de așteptare de o secundă între apăsări), pentru a evita activarea accidentală a funcției. Pentru oprire se va folosi o singură tastă, de exemplu LOCK.

Trebuie să menționam că acestă formă de confort atrage după sine și un cost suplimentar: consumul de carburant. Iar gestionarea cantității disponibile în rezervor rămâne în grija utilizatorului. În timp ce majoritatea recomandărilor pentru reducerea consumului de carburant și a poluării includ evitarea staționării cu motorul pornit și plecarea către destinație de îndată ce a fost pornit motorul, aceasta funcție se abate de la astfel de recomandări, punând responsabilitatea consumului suplimentar de carburant pe seama utilizatorului.

Această funcție prevede și o normă de siguranță împotriva furtului (presupunând că stă la panda și este dispus să spargă geamul pentru a pătrunde în mașina), astfel că montajul preia ca semnal de intrare și starea pedalei de ambreiaj sau a contactorului de punct mort al cutiei de viteze, oprind imediat motorul la detectarea unei schimbări de stare a acestei intrări.

3. *Pornirea programată și cu temporizare a motorului prin intermediul unei interfețe cu display și butoane*. Această funcție aduce o completare funcției anterioare, în sensul că oferă o interfață pentru programarea uneia sau mai multor porniri viitoare, setând data și ora dorită pentru fiecare. Această abordare ne scutește de necesitatea ca mașina să fie staționată în apropierea utilizatorului. În plus, mai acoperă și situația în care utilizatorul uită să apeleze funcția din telecomandă. Modulul care gestionează această funcție conține un ceas electronic propriu (ore, minute, secunde și data completă), cu baterie de backup pentru a nu necesita o nouă reglare în situația întreruperilor de alimentare. Funcția este disponibilă oricând pentru a programa noi porniri, având drept condiții minime necesare ca mașina să fie deschisă din telecomandă.

Programările nu sunt influențate în niciun fel de apelarea ulterioară a funcției de pornire prin telecomandă în afara timpilor programati. De asemenea, cele două funcții nu pot interfepla. Ele nu pot interveni una peste cealaltă, fiindcă secvența de pornire a motorului ia în calcul turația inițială a acestuia, care dacă nu este nulă, va semnifica faptul că motorul este deja pornit sau în curs de pornire și va conduce la interdicția demarajului. Dacă „ceasul” următoarei programări urmează să „sună” în timpul unei secvențe de pornire prin telecomandă ori în timpul utilizării obișnuite a mașinii, se va marca o demarare programată anulată, pe lista de așteptare trecând direct următoarea programare efectuată, dacă mai există vreuna.

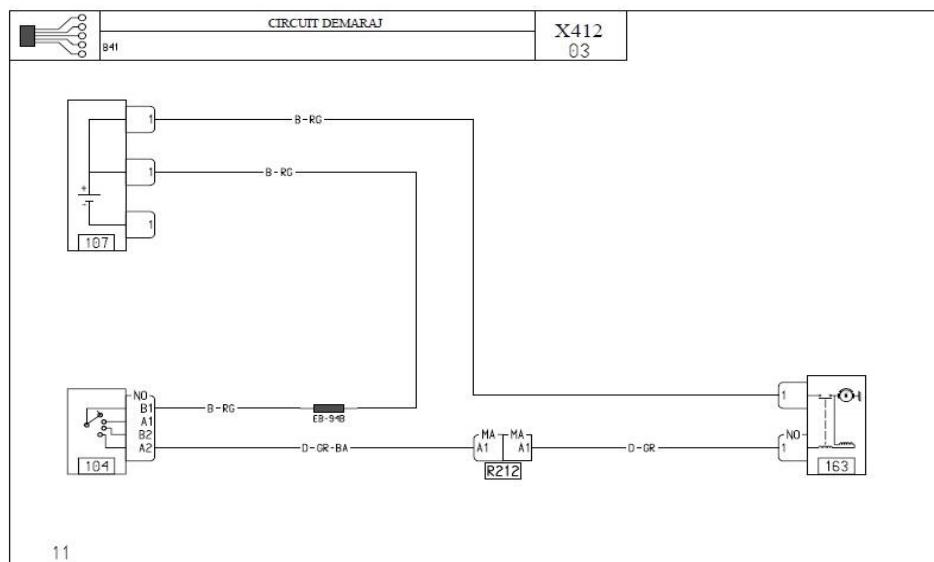
La fel ca în cazul funcției anterioare, sunt prevăzute elemente de securitate antifurt prin definirea unei proceduri care să decoupleze modulul de citire a cheii.

4. Gestionează distribuție alimentărilor principale (ACC, Ignition). Din moment ce dorim implementarea pornirii motorului din buton, trebuie într-un fel să „scăpăm de cheie”.

La orice mașină cu cheie (iar nu cu card), cheia are trei roluri:

- de a deschide orice butuc al mașinii (uși, portbagaj, buson rezervor, torpedo);
- de a comuta între pozițiile butucului central [de vazut cum se cheama pe română]: OFF, ACC, IGN și DEM.
- de a realiza autentificarea șoferului, în vederea autorizării demarajului de către calculatorul de injecție.

Butucul central poartă rolul unui întrerupător multiplu, ale căruia poziții încid succesiv circuitul ACC, circuitul IGN și circuitul DEM.



Sursa: [Manual de reparație pentru echipamentele electrice Dacia Solenza]

Circuitul de alimentare ACC (Accessory) este primul care se închide după comutarea butucului din poziția OFF în poziția ACC. Aceasta se ramifică mai departe într-o serie de alimentări pentru diferite sisteme electrice și electronice ale mașinii, în special cele de confort: sistemul de ventilație (climatizare), radioul și navigația, geamurile electrice etc.

Circuitul de alimentare IGN (Ignition) este al doilea circuit care se închide, după comutarea butucului din poziția ACC în poziția IGN. Acest circuit este, la rândul său, o ramură principală ce alimentează o altă serie de sisteme, precum calculatorul de injecție, sistemul Airbag, semnalizarea, luminile de

stop și de marșarier etc, dar și alte sisteme care trebuie să fie active pe timpul conducerii. Închiderea circuitului IGN nu afectează circuitul ACC, ci îl menține în continuare închis.

Circuitul de alimentare DEM (demaraj) este al treilea circuit care se închide, după comutarea din poziția IGN în poziția DEM. Acest circuit alimentează direct solenoidul demarorului în momentul în care se solicită pornirea motorului. Întrerupătorul aferent acestui circuit este de tip monostabil, întrucât butucul va rămâne în poziția DEM atât timp cât mâna șoferului continuă să exercite un cuplu de răsucire asupra cheii, apoi el revenind în poziția IGN. Acest caracter monostabil este implementat pur mecanic, cu ajutorul unui arc destul de puternic, care exercită un cuplu în sensul DEM → IGN, deci se opune rotirii în sensul DEM. Mai mult decât atât, există un mecanism care, după revenirea din poziția DEM în poziția IGN, blochează direcția IGN-DEM până la următoarea trecere din IGN în ACC și înapoi. Aceasta are rolul de a preveni acționarea demarorului cu motorul pornit, fapt ce poate duce la distrugerea demarorului și chiar la daune mai mari. Butucul mai are și rol de element blocant al volanului, ce acționează la scoaterea cheii din butuc și rotirea ușoară a volanului, în scopul unei măsuri antifurt. Acest mecanism constă într-o limbă metalică groasă careiese din butuc, împinsă de un arc, și intră în primul canel care trece în calea ei la rotirea volanului, dintr-un sir de caneluri identice, frezate de jur împrejur în axul coloanei de direcție.

Din descrierea circuitului electric din butucul central, ne dăm seama că mai ales primele două întrerupătoare (ACC și IGN), dar și al treilea (DEM), sunt niște întrerupătoare de putere, ce trebuie să suporte suma tuturor curentilor consumatorilor arondați celor două circuite în condiția în care toate acestea sunt pornite simultan, respectiv curentul consumat de solenoidul demarorului pe timpul pornirii motorului. Într-adevăr, acestea sunt niște întrerupătoare cu contacte glisante (piste metalice curbate și pini rotativi împinși de arcuri), realizate din cupru și având secțiuni considerabile, calculate conform curentilor maximi ce trebuie suportați. La capătul acestor contacte se află o mufă cu pini bine dimensionați, iar cablurile ce părăsesc mufa pereche conduc către mai multe tablouri cu siguranțe respectiv spre intrarea demarorului, având și aceste secțiuni vizibil mai mari decât cablurile ce părăsesc mai departe tablourile.

Acest rol de întrerupător multiplu poate fi implementat după o logică identică folosind un microcontroler dedicat și relee de mare putere pe post de întrerupătoare. Pentru a scuti șoferul de a comuta între nivelurile IGN-ACC-OFF, programul din microcontroler va implementa o logică de activare selectivă a celor două alimentări în funcție de scenariile de folosire normală a mașinii. Practic, alimentările vor fi activate la cererea diferitelor module în funcție de starea funcțiilor acestora. De exemplu, dacă se cere pornirea motorului, modulul de pornire va cere modulului de gestiune a alimentărilor activarea releeului de IGN, prin intermediul unei ieșiri logice conectate la o intrare logică a acestuia. Cererea de activare a releeului de IGN va fi menținută în continuare pe toată durata funcționării motorului, până la cererea de oprire. Astfel, oprind reeleul IGN, se va „tăia” alimentarea de IGN a calculatorului de injecție (și a altor consumatori dependenți de IGN), oprindu-se astfel motorul. Această logică are loc la varianta autoturismului „cu cheie” (cu pornire din cheie), deci va fi preluată în proiectul de față, pentru a adapta sistemul nostru „fără cheie” la specificațiile electrice și electronice ale unui autoturism „cu cheie”.

Ultimul discutat, dar totuși foarte important, este aspectul funcției de imobilizator electronic, care are rolul de a autentifica șoferul prin intermediul cheii folosite. Această funcție este realizată pe principiul RFID (Radio Frequency IDentification), printr-o tehnologie similară cu cea a cardurilor de acces în clădiri. În cazul de față, „cardul” este cheia, care în interiorul carcasei conține un transponder, iar „cititorul” este o antenă elicoidală aflată în carcasa de plastic care învelește corpul butucului, astfel că distanța dintre antenă și transponderul din carcasa cheii este de circa 5 cm. Transponderul este un

circuit integrat fără pini, produs adesea de către Philips și NXP, care nu are nevoie de alimentare, întrucât este alimentat prin inducția produsă de antenă și apoi transmite un semnal ce conține un cod unic. Semnalul este recepționat de antenă, decodificat de către circuitul care controlează antena (în cazul nostru, Unitatea de Control Habitaclu) și se materializează mai departe într-o comandă de autorizare a demarajului către calculatorul de injecție.

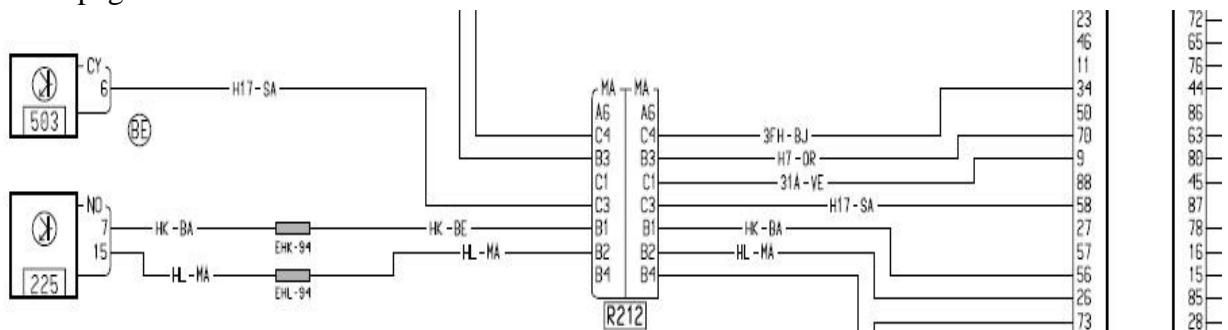


Fig. 2.7 - Transponder

În situația în care transponderul este absent (dacă folosim doar o copie a limbii cheii), cheia se va putea roti în butuc, însă nu se va face autorizarea demarajului. Același lucru se întâmplă și în situația în care transponderul nu este recunoscut (nu a avut loc o procedură de „împerechere” a cheii cu mașina, care se realizează de obicei în service sau în fabrică, sau chiar viceversa, cheia a fost anterior „împerecheată”, iar în urma pierderii sau furtului acesteia, mașina a fost prezentată în service pentru a se aplica procedura de „uitare” a cheii respective – cu ajutorul interfeței de diagnoza auto). Refuzul demarajului în urma neautentificării cheii se manifestă în felul următor: demarorul poate fi acționat, aşadar mașina se comportă ca și cum ar încerca să pornească motorul, însă motorul nu va porni niciodată. Pentru a indica șoferului faptul ca este vorba despre o problema de identificare a cheii și nu o problema a motorului, martorul cu LED specific funcției antidemaraj, din cadrul tabloului de bord, va clipea cu o frecvență tipică (5 Hz). În situația unei autentificări reușite, același led se va aprinde timp de 1-2 secunde apoi se va stinge.

Pentru a putea elimina complet sistemul cu cheie din habitaclu, trebuie să găsim o soluție pentru a înlocui acest mecanism de autentificare, în timp ce, pentru chestiunea distribuției alimentărilor am găsit deja o soluție.

Aici a fost necesară o analiză mai atentă a mecanismului, iar pentru aceasta ne-am concentrat pe semnalul de autorizare transmis printr-un fir de la UCH la calculatorul de injecție, identificând pozițiile acestei interconexiuni în documentația disponibilă în manualul de reparații electrice al Solenza de la pagina 133.



Sursa: [de reparație pentru echipamentele electrice Dacia Solenza]

Făcând o înregistrare a acestui semnal și făcând corelația dintre forma semnalului și cele două acțiuni specifice (autentificare reușită, respectiv eşuată), aflăm că pentru oricâte repetări ale celor două acțiuni, se va obține mereu câte un singur semnal, specific respectivei situații. Dacă, pe durata unei autentificări reușite, am trece acel semnal printr-un convertor analogic-numeric (ADC) cu o rezoluție și o frecvență de eșantionare satisfăcătoare, am memora secvența de valori numerice obținute, pentru ca apoi să o putem trece ori de câte ori dorim printr-un convertor numeric-analogic (DAC), am obține un circuit echivalent pe care l-am putea folosi pentru autorizarea demarajului, înlocuind sistemul RFID cu transponder și antenă. Microcontrolerul ATMEGA8 conține 6 ADC-uri suficient de performante, însă niciun DAC. Folosirea unui DAC extern, chiar dacă este accesibilă ca preț și ca metodă de conectare la ATMEGA8, ar complica schema prea mult pentru scopul didactic al acestei lucrări și, de asemenea, ar încărca suplimentar și codul programului.

Așadar, eliminăm varianta realizării unui circuit imitativ pentru a înlocui sistemul original cu RFID. Varianta eliminării complete a acestui circuit este, de asemenea, imposibilă, întrucât aceasta ar conduce la negenerarea semnalului de autorizare a demarajului către calculatorul de injecție, deci motorul nu ar mai porni niciodată. Ne rămâne, așadar, o ultimă variantă: aceea a mutării și izolării ansamblului antena-transponder, acest ansamblu nemaifiind la îndemâna șoferului. Transponderul va fi scos din cheie, carcasa butucului va fi scoasă și ea de pe butuc, transponderul va fi montat în centrul acesteia (cât mai aproape de centrul bobinei antenei), iar acestea vor fi asamblate împreună și mutate într-un loc cât mai ascuns din spatele bordului, prelungind atât cât este necesar legăturile electrice cu UCH-ul. În această configurație, de fiecare dată când se va activa alimentarea IGN, UCH-ul va transmite semnalul de autentificare reușită către calculatorul de injecție.

Ne punem însă următoarea problemă: dacă permanentizăm starea de autentificare (din punctul de vedere al UCH-ului), nu punem în pericol mașina în față potențialului unui furt? Este suficient ca hoțul să reușească să alimenteze linia IGN și, în cel mai rău, caz, să pornească motorul prin împingerea mașinii. Acest lucru, însă, nu se va întâmpla, întrucât funcția de autentificare a șoferului va fi reatribuită modulului de recepție a telecomenzi și a modulului de gestiune a comenzilor din telecomandă, care va interpreta comanda de descuiere a ușilor drept autentificare reușită și doar în această situație va trimite mai departe un semnal binar modulului de gestiune a alimentărilor. Fără acest semnal, modulul de gestiune a alimentărilor nu va servi cererile de activare a IGN-ului din partea altor module. Cum aceste aspecte nu sunt cunoscute decât de către autorul proiectului și de către cititorii acestei lucrări, punând la socoteală și faptul ca modulele vor fi asamblate și montate într-un loc cât mai sigur și mai ascuns, în spatele bordului, eventualul răufăcător nu va avea timpul necesar pentru a specula aceste acțiuni de „hack-uire” a sistemului nostru. Singura cale naturală de a efectua o autentificare reușită este folosirea telecomenzi. Aceasta emite un semnal radio ce conține un cod unic, recunoscut de modulul de recepție a telecomenzi și materializat în acțiunea unui releu corespondent acțiunii realizate (descuiere ușă, închuire ușă, descuiere portbagaj). Acțiunea de descuiere produce autentificarea, în timp ce acțiunea de închuire produce anularea autentificării.

Ne rămâne de analizat un ultim subiect: blocarea mecanică a volanului, realizată prin intermediul butucului atâtă timp cât cheia este scoasă. La mașinile dotate din fabricație cu sistem „fară cheie”, rolul acesta este realizat de un mecanism electromecanic, în locul clasnicului butuc. Un motor electric de mici dimensiuni, cuplat la un reductor realizat cu o serie de roți dințate, va produce mișcarea de translație a limbii metalice ce blochează coloana de direcție. Eliberarea coloanei este realizată concomitent cu prima activare a alimentării IGN, după descuierea mașinii, iar blocarea

coloanei are loc, în funcție de producător, în diferite situații: fie la scoaterea cardului din cititor, fie abia la încuierea mașinii din card.

Eliminarea butucului ne va lăsa fără aceasta facilitate, în sensul că volanul va rămâne liber și după încuierea mașinii, însă nu afectează imunitatea antidemarajului. O posibilă soluție pentru înlocuirea acestui sistem ar fi modificarea butucului sau realizarea unui ansamblu electromecanic care să se potrivească perfect în locul butucului și să realizeze mișcarea de translație a unei limbi metalice, și integrat din punct de vedere electric și funcțional proiectului nostru. Acest lucru însă presupune mai mult efort în partea domeniului mecanicii decât în partea domeniului electric și electronic, aşadar nu se pretează la lucrarea noastră. Vom considera ca am ținut cont de acest aspect, iar integrarea din punct de vedere electric și electronic ar presupune următoarele: punerea la dispoziție a unei ieșiri de tip binar (un pin de ieșire în plus, la Modulul de pornire - MP) și includerea acestei ieșiri în codul programului, în cadrul funcției care gestionează starea de autentificare.

Capitolul III – Construcția montajului

Cea mai importantă secțiune a acestei lucrări este dezvoltată în paginile care urmează, dat fiindcă este vorba despre partea aplicată, adică proiectul în sine. Pentru a finaliza montajul s-a pus în aplicare construcția motorului simulat, dar și a modulelor propriu-zise. De asemenea, în vederea parcurgerii construcției am luat în calcul interconectarea modulelor și teste de integrare. Mai mult, avem descrierea teoretică a modulului cu telecomandă și a celui cu pornire programată. Ultimul subcapitol este dedicat rezultatelor pe care le-am obținut în această parte practică.

3.1 Necessitatea construcției motorului simulat. Scop și cerințe.

Acesta se va construi sub forma unui modul independent, care nu se numără printre subansamblurile proiectului propriu-zis, ci se prezintă ca un modul auxiliar, ca un instrument de testare, ce ne va ajuta pe parcursul dezvoltării proiectului. El trebuie să fie independent din toate punctele de vedere, pentru a ne permite să fim obiectivi în abordarea funcțiilor pe care le implementăm în proiect. Este independent inclusiv din punct de vedere electric, cu excepția alimentării electrice care va putea fi comună. Dacă alegem eventual soluția alimentării dintre-o sursă separată (neavând nici măcar masă comună între acestea), putem afirma faptul că motorul simulat este chiar izolat galvanic față de proiectul în sine.

Acesta ar trebui să fie punctul de pornire, astfel încât să putem trece la dezvoltarea modulelor din aproape în aproape.

Cum realizăm acestă sarcina? Îi vom dedica motorului un microcontroler separat, pe care îl vom programa astfel încât să interpreteze turația și să comande pornirea și oprirea motorului în funcție de turație.

Resurse necesare din partea microcontrolerului:

- O intrare de timer extern, care va fi generată de senzorul de turație;
- O ieșire digitală, pe 1 bit, pentru acționarea motorului.
- Două sau mai multe ieșiri digitale, pe câte 1 bit fiecare, pentru a alimenta leduri de culori diferite, ce vor indica starea în care se află, din punct de vedere logic, automatul finit.
- O viteză de lucru rezonabilă pentru a putea măsura frecvența unui semnal de ordinul kHz (<10kHz) și a face câteva operații aritmetice la fiecare secundă: 8MHz cu ceasul intern, este suficient.

Celelalte resurse necesare implementării:

- Motorul electric propriu-zis;
- Un etaj de comandă tranzistorizat, care să permită controlul curentului consumat de motor în cazurile cele mai nefavorabile, adică la turația sa maximă, respectiv la creșterile de turație, inclusiv la scoaterea din starea de repaus.
- Un variator de turație cu ieșire în PWM, implementat cu circuitul integrat LM555.

- Doi senzori de mișcare identici, optici sau magnetici, care să poată implementa rolul de senzor de turație.

Logica de funcționare ar trebui să includă următoarele:

- la citirea unei turații mai mari de 50 rpm acționeaza alimentarea motorului și trece în starea „motor în curs de pornire”;
- la citirea unei turații mai mari de 60 rpm trece în starea „motor pornit”;
- la citirea unei turații mai mici de 350 rpm oprește alimentarea motorului și trece în starea „motor oprit”;

La nivel analogic, vom limita pragul inferior al regulatorului, astfel încât turația minimă pe care o putem obține să coincidă cu turația de motor în curs de pornire. Astfel, pentru a aduce automatul în starea „motor oprit”, trebuie să găsim o metodă de a reduce turația sub pragul de oprire. Acest lucru se va putea face:

- mecanic, frânând axul motorului cu mâna – astfel simulam situația de motor calat;
- electric, întrerupând circuitul motorului pentru scurt timp, astfel ca acesta să încetinească din inerție – astfel simulăm situația opririi „din cheie”, care în realitate oprește injecția electronică (și aprinderea).

Un semnal de turație identic (preluat din același senzor de turație sau dintr-unul separat, dar identic ca funcționare) va fi livrat către exterior, mai exact către modulul de pornire din cadrul proiectului propriu-zis.

Știindu-se deja care sunt cerințele, și având o idee destul de bine conturată despre cum ar trebui să arate în final acest motor simulat, am trecut la căutarea și alegerea materialelor electrice și mecanice necesare construcției. Pornim de la premsa că avem deja la dispoziție componentele electronice, acum trebuie să găsim elementele mecanice (și electrice: motorul) prin care vom construi partea mobilă a motorului simulat.

Acestea ar fi:

- motorul electric;
- o piesă rotundă, în formă de disc sau cilindru, pe care să o putem atașa axului motorului, aceasta urmând să joace rolul de volană;
- un material plan, ușor prelucrabil, dar suficient de rigid pentru a susține întregul ansamblu, sub forma unei planșe;
- elemente de prindere: coliere, colțare metalice, fâșii de metal ușor de tăiat, de îndoit și de găurit, șuruburi, șaibe, piulițe, holzșuruburi etc.

3.2 Realizarea practică a motorului simulat

3.2.1 Etapele realizării

1. Am început cu alegerea materialului pentru planșă. Din experiența dobândită în urma construcției unor piese de mobilier, mi-a captat atenția materialul PFL, care poate fi ușor găurit, este plan și destul de rigid, și se poate tăia cu ajutorul unui cutter, într-o manieră asemănatoare tăierii

sticlei. Prezintă un avantaj faptul că zonele colțuroase pot fi ușor pilite, sau chiar tăiate cu foarfeca, pentru a realiza finisaje. De asemenea, una dintr-o fețe este vopsită uniform, astfel că ne oferă și un aspect estetic prin montarea componentelor pe această parte.

2. Alegerea motorului este pasul următor. Acesta trebuie să aibă o tensiune nominală de lucru apropiată de tensiunea noastră de 12 V. Trebuie să ofere un regim de turăție asemănător unui motor termic în 4 timpi multicilindru, adică 700...7000 de rotații pe minut. Variația turăției acestuia se va face modificând tensiunea aplicată între 0 și maxim, adică 12V. Trebuie să prezinte o variație cât mai liniară a turăției în funcție de tensiunea aplicată și să se comporte bine la aplicarea unei tensiuni sub forma PWM. Pentru a compensa eventualele neajunsuri datorate comutației (efectul de *smucire*, turăție instabilă etc.), se va ajuta convenabil frecvența de comutație a variatorului de turăție, astfel încât efectele deranjante să fie minime. Din toate aceste considerații, rezultă că este suficientă folosirea unui motor electric de curent continuu cu magneți permanenti, de uz general, de mică putere. Folosindu-mă de avantajul de a fi păstrat un număr mare de astfel de motoare electrice, recuperate din aparaturi diverse (electrocasnice, scule etc.), în urma căutarilor m-am oprit asupra unui motor având elementele de identificare DA CHANG DRS-365SP-1885, ce provine dintr-un aspirator de mâna pentru autoturisme, alimentat la 12V de la mufă de brichetă.

3. În continuare, trebuie să ne gândim la o modalitate de a prelucra o volană pentru acest motor. Preferabil, aceasta ar fi sub formă unui disc, dintr-un material cât mai prelucrabil, dar rezistent, pentru a nu prezenta riscul de dezintegrare la turății mari. Acesta trebuie să fie subțire cel puțin la margini, pentru a permite prelucrarea danturii.

4. Acum ajungem la etapa care a necesitat un lung sir de decizii, în urma cărora să putem alege tipul de senzor de turăție și să potrivim senzorul ales cu volanta. În cadrul acestei etape, am preferat să definitez întâi problema senzorului, pentru ca mai departe, pe baza avantajelor și dezavantajelor prezentate de acesta, să mă orientez asupra tehniciilor de realizare a volantei.

Din timpul reparării și studierii anumitor aparate, mi-am adus aminte de un tip de senzor de deplasare circulară/senzor de turăție întâlnit destul de des: fotosenzor. Acesta este realizat prin alăturarea unei diode LED cu emisie în infraroșu și a unui fototranzistor, orientate față în față, la o distanță de câțiva milimetri. LED-ul este alimentat permanent, iar lumina acestuia polarizează baza fototranzistorului, menținându-l închis (în stare de conducție). Atunci când un obstacol obturează calea dintre LED și fototranzistor, acesta din urmă nu mai este polarizat, trecând în starea de blocare. Acesta este întâlnit în mouse-urile de PC de generații mai vechi, cu bilă, unde două axe perpendiculare erau rotite de bilă, în funcție de mișcarea făcută, iar la capatul acestor axe există câte o roțiță, având un șir de fante echidistante de-a lungul circumferinței acestora, alcătuind o coroană circulară populată cu fante. Câte un astfel de senzor era montat la capătul fiecărei roțițe, având axul optic centrata pe coroana de fante. Mișările de translație ale mouse-ului erau descompuse pe axa verticală și orizontală, apoi transformate în mișcari de rotație ale roțițelor, iar prin citirea numărului de fante parcuse în dreptul fiecărui senzor, erau reconstituite compoziția verticală și orizontală ale mișcării. Creșterea cerințelor privind rezoluția au condus la creșterea numărului de fante și subțierea acestora, fapt ce a condus la atingerea limitei tehnologice, astfel că mai departe a apărut tehnologia mouse-ului cu reflexie (cu diodă LED sau laser, numit în mod derulant „optic”, fiindcă și cei cu bilă erau, în interior, tot optici). Însă această tehnologie s-a păstrat și pe aceste mouse-uri, pentru roțița de scroll (sus/jos), pentru care nu există cerințe de rezoluție atât de pretențioase.

De asemenea, acest tip de senzori se folosește în continuare și în imprimante, pentru a detecta diverse stări binare (prezență sau lipsă hârtie, detecție de panouri/uși deschise, detecția capului de imprimare în poziția „parking” etc.), dar și pentru măsurări de deplasare de înaltă rezoluție

(deplasarea longitudinală a hârtiei sub acțiunea axului de tragere), datorită faptului că spațiul disponibil într-o imprimantă a permis montarea unor roți de diametre mari ($>10\text{cm}$, față de $1,5\text{-}2\text{cm}$ la mouse-uri) la capetele axelor, permitând astfel practicarea unui număr foarte mare de fante fine de-a lungul circumferinței roții. Inspirat de acest procedeu, am decis folosirea a doi senzori de acest tip, recuperăți dintr-o imprimantă. Deschiderea ferestrei acestora este de 5 mm. Folosind această mărime ca dată de intrare, revin la alegerea materialului pentru realizarea volantei. Pentru a introduce în fereastra de 5 mm un disc, păstrând o distanță de 1mm față de marginea ferestrei pe ambele parti ale discului și acceptând o toleranță de 1mm pentru bătăile axiale, rezultă ca grosimea discului trebuie să fie mai mică de 2mm.

Dorim să simulăm și rezoluția citirii turației în concordanță cu cazul real al motorului E7J / K7J de pe Dacia Solenza. Volanta motorului are o serie de 119 dinți de-a lungul circumferinței sale, detectați cu ajutorul unui senzor magnetic, al 120-lea fiind absent (așa este detectată faza de 0 grade a rotației arborelui cotit). Pentru aceasta ne-am gândit ca va fi necesară practicarea unui număr exact de 119 fante pe circumferința volantei simulate. Aici intervin o serie de constrângeri care m-au obligat să fac multiple încercări în construcția volantei, din cauza proprietății materialelor:

- flexibilitatea și maleabilitatea materialului la o diferite grosimi;
- diametrul discului;
- comportamentul materialului la turații ridicate.

Aceste proprietăți trebuie puse în balanță și alterate unele în funcție de celealte pentru a găsi compromisul cel mai bun în atingerea următoarelor ţinte:

- posibilitatea realizării unor decupaturi la extremitatea discului, în urma cărora fâșiiile obținute să ramână cât mai rigide, să nu poată fi rupte cu ușurință prin atingere accidentală, să nu fie moi, să nu fluture sub acțiunea curenților de aer produși în mișcare;
- discul să nu se deformeze sub propria greutate la ancorarea în plan orizontal;
- diametrul discului să nu se modifice datorită forței centrifuge la turații mari.

La prima încercare, am ales un material plastic care avea deja un diametru convenabil, dar prea casant la încercarea decupării fantelor. Aplicarea unei metode de decupare prin topire cu o lamă încălzită putea fi o soluție, dar ar fi necesitat prea multă muncă de finisare, pentru îndepărtarea plasticului topit din jurul fantelor. În continuare am încercat cu un alt material plastic, foarte ușor de prelucrat, însă era prea flexibil și se deforma sub propria greutate (exagerând spre exemplificare, putem spune că tindea să capete bolta). Acest lucru nu ar fi fost deranjat, ținând cont că după montarea într-o poziție definitivă puteam regla poziția senzorului după forma deformării acestuia. Însă, în mod evident, acesta începea să se îndrepte treptat pe măsura ce creștea turația. Astfel, oricum am fi reglat senzorul, discul l-ar fi lovit fie când avea turație mare, fie când se oprea. O bucată de PFL ar fi fost foarte convenabilă pentru prelucrare, însă materialul este prea gros (3 mm). În cele din urmă, am acordat o șansă unui material de tip carton presat, ce servea drept garnitură de etanșare a capacului unei cutii de produse cosmetice. Acesta avea deja forma circulară dorită și un diametru convenabil (102 mm) și prezenta rigiditate suficientă. Fiind ușor de prelucrat cu ajutorul unui cutter, am decis să continui cu folosirea acestuia.

Pentru trasarea marcajelor în vederea decupării fantelor, am căutat pe Internet o imagine cu un cerc care să aibă trasate, la periferie, toate gradele. Modificând imaginea pe calculator, am trasat 120 de raze, la fiecare 3 grade

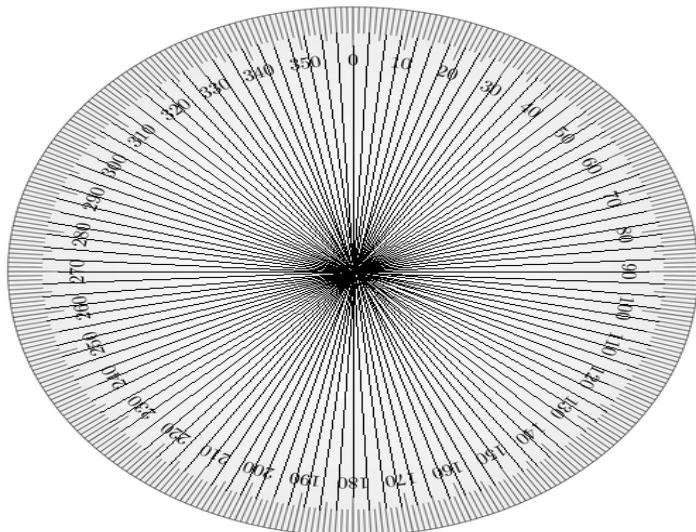


Fig. 3.1 – Cerc cu raze trasate la fiecare 3°

Am imprimat această imagine la o dimensiune mai mare decât dimensiunea discului de carton, am aliniat centrele celor 2 cercuri (centrul discului de carton a fost determinat prin metode geometrice) și le-am asezat pe suprafața biroului, cu discul de carton deasupra. Astfel, la intersecția celor 120 de raze cu periferia discului de carton am trasat semne folosind un marker subțire. Măsurand, apoi, distanța dintre două semne, care a rezultat drept 2,4 mm, am constatat că pot decupa cu lejeritate fante de 0,7 - 1 mm lățime și 2 mm lungime, cu varful unui cutter.

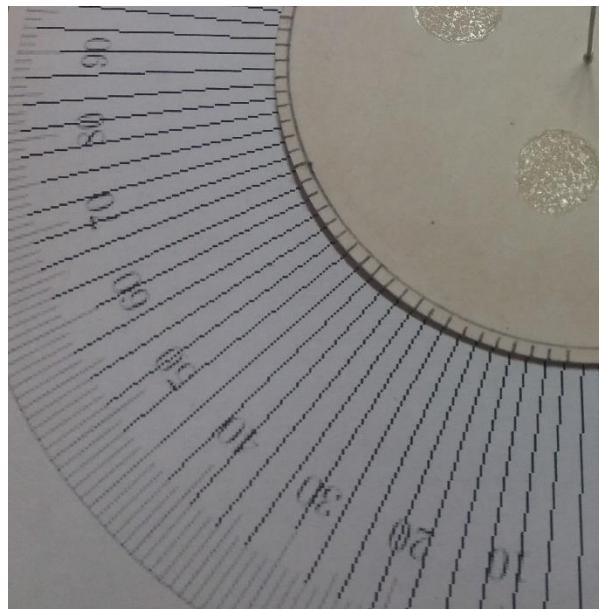


Fig. 3.2 – Delimitarea fantelor

În continuare, am observat că „dinții” rezultați au o ușoară predispoziție la îndoire, fapt pentru care am decis să aplic un strat de lac pe întreaga periferie dințată a discului și să curat, după uscarea completă, excesele de material cu un vârf ascuțit. Predispoziția la îndoire a zonei periferice s-a diminuat semnificativ. În acest stadiu, am obținut un disc suficient de rigid și având o masă redusă, care a fost mult mai ușor de prelucrat decât în cazurile anterioare eșuate. Nu mai este nevoie să menționez că unul dintre cei 120 de dinți a fost eliminat, pentru a obține succesiunea de 119 dinți, ca în cazul volantei motorului real.

Revenind acum la motor, mai este de menționat unul dintre motivele pentru care a fost ales, anume faptul că acesta a rămas echipat cu rotorul din material plastic, rotor din componența ventilatorului centrifugal al aspiratorului din care provine. Pus în funcțiune la turații cât mai mari, s-a observat faptul că acesta prezintă jocuri axiale și radiale extrem de reduse, ceea ce ne permite o marjă de eroare mai mare în operațiunea de centrare a discului pe rotor. De asemenea, forma plată și lipsită de abateri a exteriorului rotorului ne permite o aplicare cât mai bună a discului pe suprafață acestuia. Următoarea etapa a fost centrarea discului pe rotor, care s-a putut face cu ușurință datorită faptului că ambele corpuri aveau deja găurile centrele de rotație, astfel că a fost folosit un ac subțire pentru a le alinia „centru la centru”, apoi au fost practicate găuri prin ambele materiale, cu burghie subțiri, pe baza unor semne trasate deja în mod simetric pe discul de carton. Au fost montate 6 șuruburi mici pentru a prinde discul de rotor, iar poziționarea acestora a fost făcută astfel încât, prin masa și pozițiile lor, să conducă la o deviere cât mai mică a centrului de greutate al întregului ansamblu. Astfel, se dorește ca nivelul vibrațiilor să se mențină minim, la orice regim de turații.

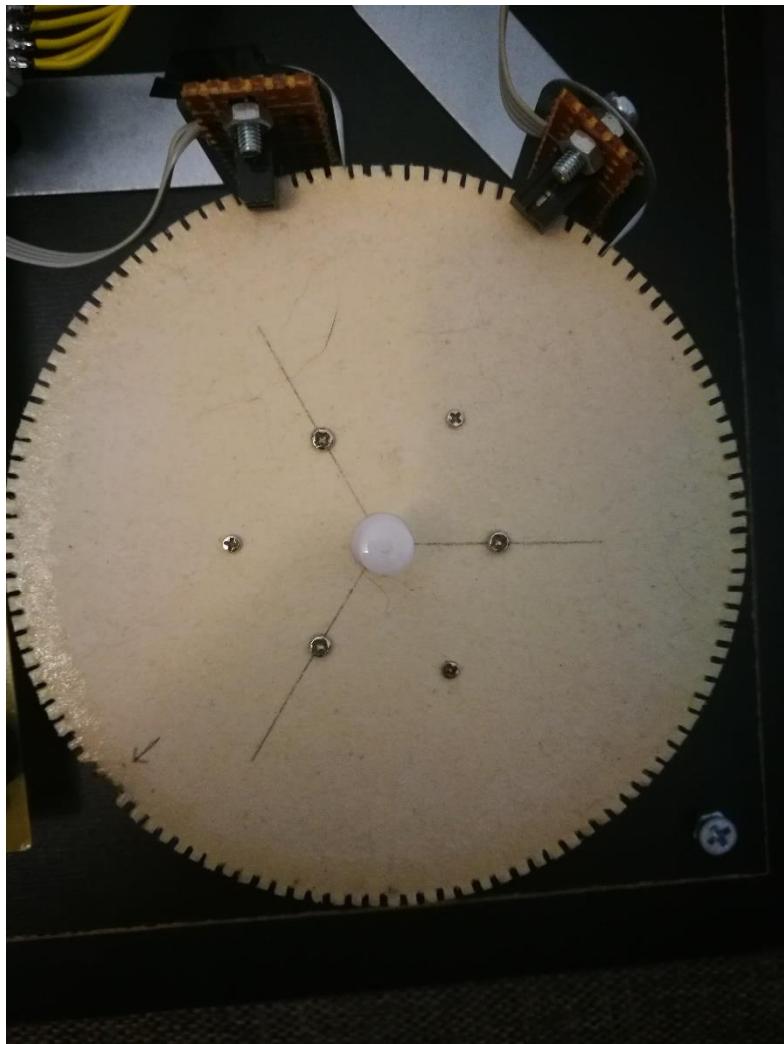


Fig. 3.3 – Șuruburi fixare volantă

Odată obținut ansamblul rotor-disc, avem volanta motorului.

5. Trebuie să găsim o modalitate de a monta pe planșă ansamblul motor-volanta, ținând cont că va trebui montat și senzorul de turație în funcție de poziția fixă a volantei față de planșă. Am luat în calcul 4 posibile variante:

- cu volanta în plan vertical, deasupra planșei;

- cu volanta în plan vertical, la semiînălțime: aici ar fi necesar să decupăm o fântă dreptunghiulară din suprafață planșei, astfel încât volanta să intre în spațiul dintre doi dinți consecutivi, iar motorul să stea în contact cu planșa, în poziție culcată;
- cu volanta în plan orizontal, cu motorul în întregime deasupra planșei;
- cu volanta în plan orizontal, cu motorul la semiînălțime: aici ar fi necesară practicarea unei găuri având diametrul potrivit corpului motorului, astfel încât motorul să intre în planșă până la un anumit procent din lungimea lui.

Este important, în alegerea variantei potrivite, să avem în vedere mai multe aspecte. În primul rând, felul în care vom monta motorul cu volanta ne va afecta felul în care vom monta senzorul. Pe de alta parte, este de dorit ca ansamblul să fie cât mai aproape de planșă, pentru a reduce înălțimea totală a construcției. Astfel, dorim să obținem un montaj cât mai stabil și, în plus, să scădem probabilitatea atingerii accidentale a volantei în timpul dezvoltării și operării proiectului. Am ales, aşadar, ultima variantă din cele enumerate, în care desfășurarea spațială a pieselor este minimă.

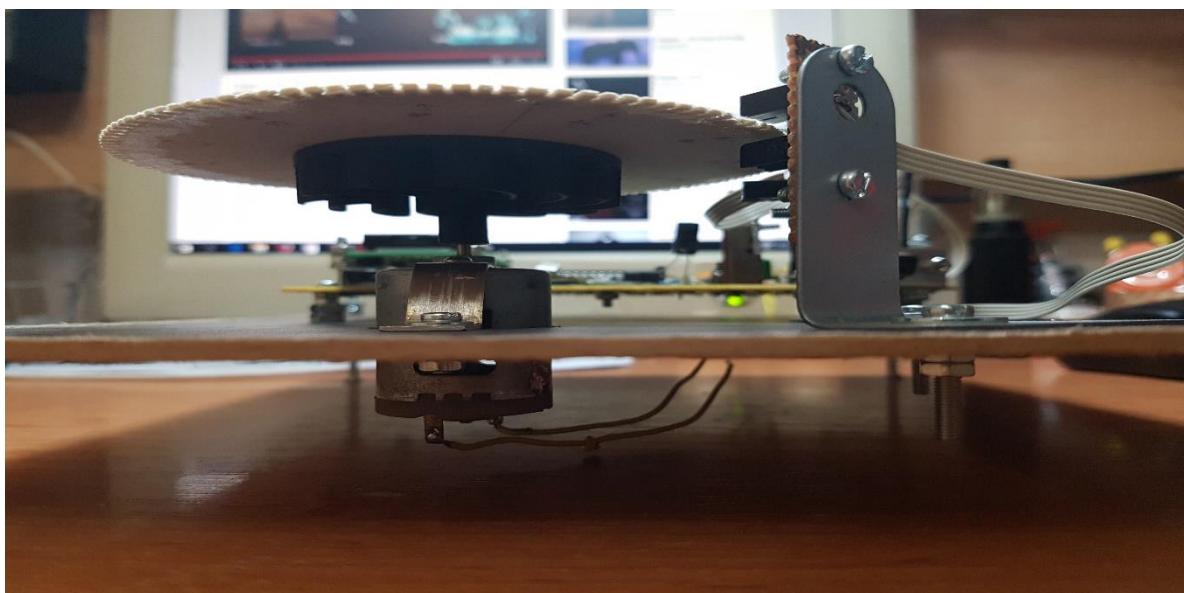


Fig. 3.4 – Poziționare, fixare motor și volantă

6. Ultima etapă care presupune prelucrări mecanice este montarea senzorului și reglarea acestuia pe înălțime și pe adâncime (mai aproape sau mai departe de centrul volantei).

În primul rând, este necesară lipirea senzorului pe o bucată de cablaj de textolit. Acest cablaj va extinde, practic, dimensiunea senzorului cu niște extremități care să poată fi prinse, în suruburi, de piesa de susținere. S-a folosit cablaj de test, întrucât, datorită rețelei de găuri preexistente, a putut fi tăiat mai ușor într-o formă dreptunghiulară, cu dimensiuni convenabile. Au fost practicate găurile din capetele plăcuței obținute anterior, apoi au fost lipite, cu letconul, senzorul și firele de legătură. Găurile din extremități au fost ovalizate intenționat, pentru a permite reglajul pe înălțime. A fost aleasă o piesă convenabilă ca formă pentru realizarea suportului: un colțar plat, în forma literei „L”.



Fig. 3.5– Colțar în formă L la 90°

Acesta a fost îndoit de la mijloc, la un unghi de 90° între bază și pivotul astfel obținut. Folosindu-ne de găurile deja existente ale colțarului, am decis ca prinderea bazei de planșă să se facă într-un singur șurub, strâns suficient de bine și securizat cu două piulițe. Prin mișcarea cu un unghi oarecare în plan orizontal a bazei, pivotul se poate roti în jurul prelungirii axei șurubului, astfel că ne poate permite apropierea și depărtarea, după o traiectorie circulară, a senzorului față de centul volantei. Vom avea nevoie de acest reglaj pentru a centra axul optic al senzorului cu mijlocul coroanei de dinți și fante, astfel încât să fie acoperit eventualul joc radial al volantei.

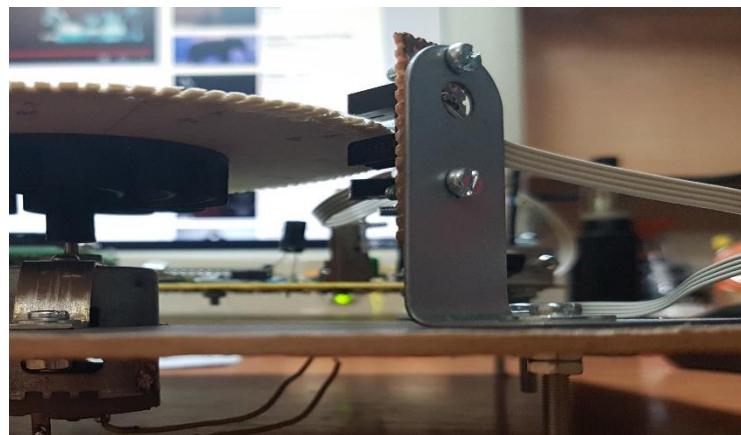


Fig. 3.6- Reglaj pe înălțime al senzorului

Au fost practicate 2 găuri în pivot, pentru montarea plăcuței cu senzorul. După așezarea provizorie, pe placă, a suportului astfel obținut, și apropierea de disc, s-a realizat reglajul pe înălțime și fixarea definitivă a senzorului pe suport. S-a montat, în final, suportul pe planșă, strângând șurubul suficient de tare, dar astfel încât să mai permită o rotire, prin fortare.

7. Este necesară verificarea pe osciloscop a semnalului produs de senzor la învârtirea volantei cu o turăție moderată, în vederea realizării reglajului în adâncime al senzorului. S-a folosit următoarea schemă absolut minimală pentru punerea în funcțiune și testarea senzorului.

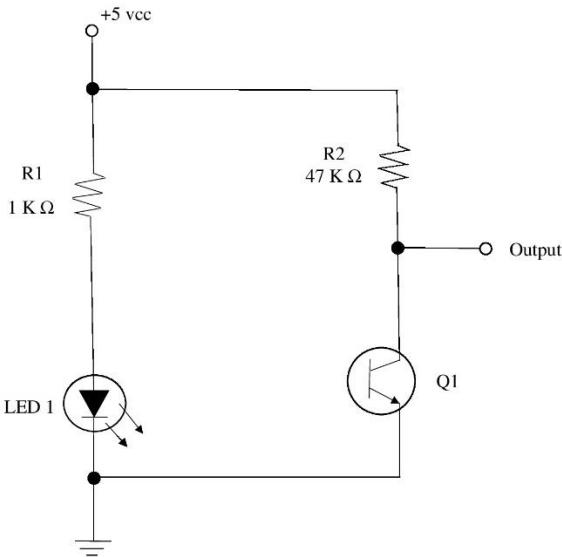


Fig 3.7 – Schemă testare și funcțiune senzor

După cum era ușor de intuit, se obține un semnal trapezoidal, la turații mici. Prin mișcarea mai în față ori mai în spate a senzorului, amplitudinea semnalului observat pe osciloscop variază. Mișcăm senzorul până când obținem o amplitudine maximă și facem un marcaj fin pe suport și pe planșă. Considerăm astfel că reglajul este finalizat și avem un reper de siguranță, în cazul în care stricăm accidental reglajul.

8. În continuare, ne interesează să aflăm cât de corect este semnalul obținut la turații mai mari. Variind progresiv turația, observăm pe osciloscop ca amplitudinea semnalului scade tot mai mult, iar forma semnalului nu mai rămâne la fel de „curată”, începând să semene mai mult cu un semnal de tip triunghiular, și destul de distorsionat.

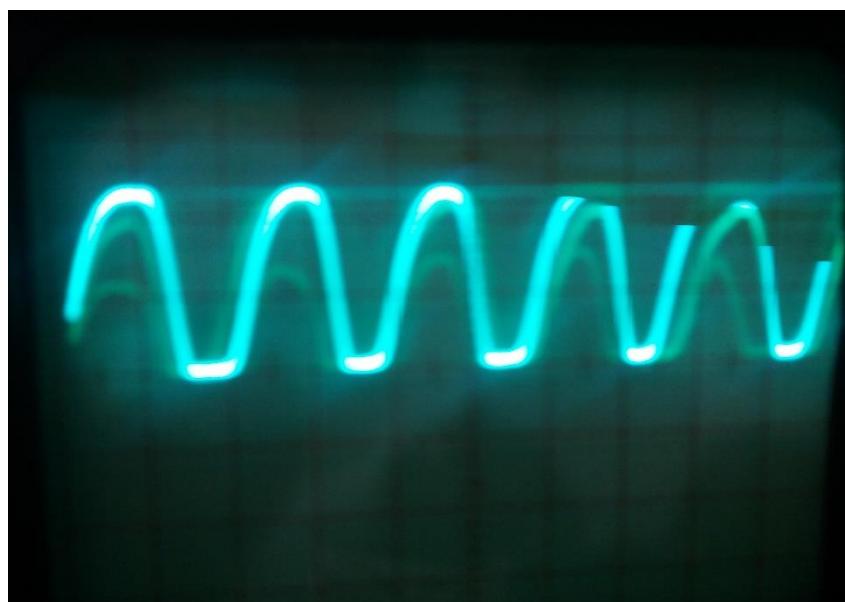


Fig. 3.8 – Turații joase

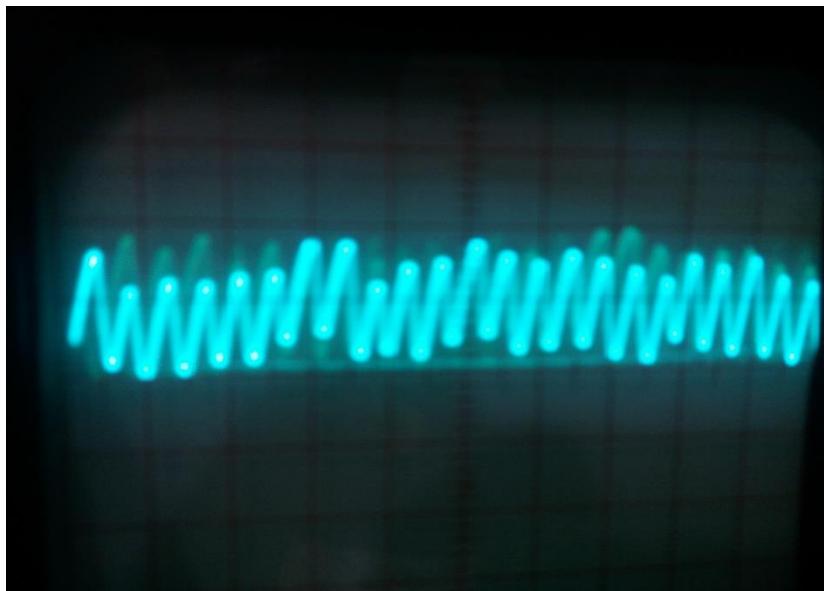
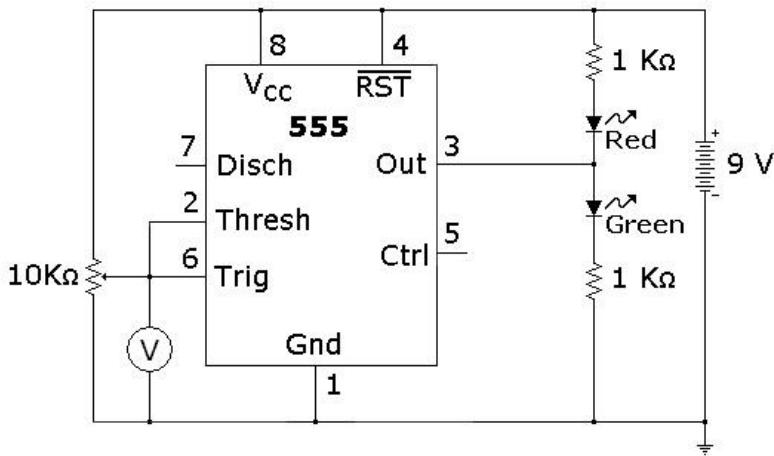


Fig. 3.9 – Turații înalte

În plus, monitorizându-l cu ajutorul unui frecvențmetru, observăm că citirea frecvenței începe să aibă loc în mod eronat. Practic, frecvența măsurată începe să stagneze, apoi să descrească, pe fondul descreșterii amplitudinii. Ne aflăm într-o situație destul de previzibilă, dar pentru care avem o soluție simplă la îndemână, folosirea unui trigger Schmidt, care funcționează ca un comparator cu histerezis. Vom folosi o schemă simplă găsită pe o pagina de internet



Sursa: [Allaboutcircuits.com, ultima accesare: 28.06.2017]

Această schemă folosește circuitul integrat LM555, care însă basculează la două praguri fixe ale semnalului de intrare, de 1/3 și respectiv 2/3 din tensiunea de alimentare a circuitului. Astă înseamnă că, dacă alimentam circuitul LM555 la o tensiune de 6V, acesta va bascula atunci când semnalul de intrare scade la sub 2V, respectiv crește la peste 4V. Măsurând tensiunea medie a semnalului produs de senzor la turații mai mari ale motorului, observăm că aceasta scade până la circa 0,6 V. Acest lucru însemnă că, la o alimentare (minimă) de 5V a circuitului LM555, vom fi în situația în care semnalul nostru va oscila în totalitate sub pragul de 1,66V, care ar fi pragul minim al triggerului la tensiunea de alimentare menționată. Vom completa schema, în acest caz, cu un tranzistor NPN de mică putere în configurație emitor comun, căruia îi vom furniza în bază semnalul senzorului, iar semnalul de pe colector se va furniza trigger-ului.

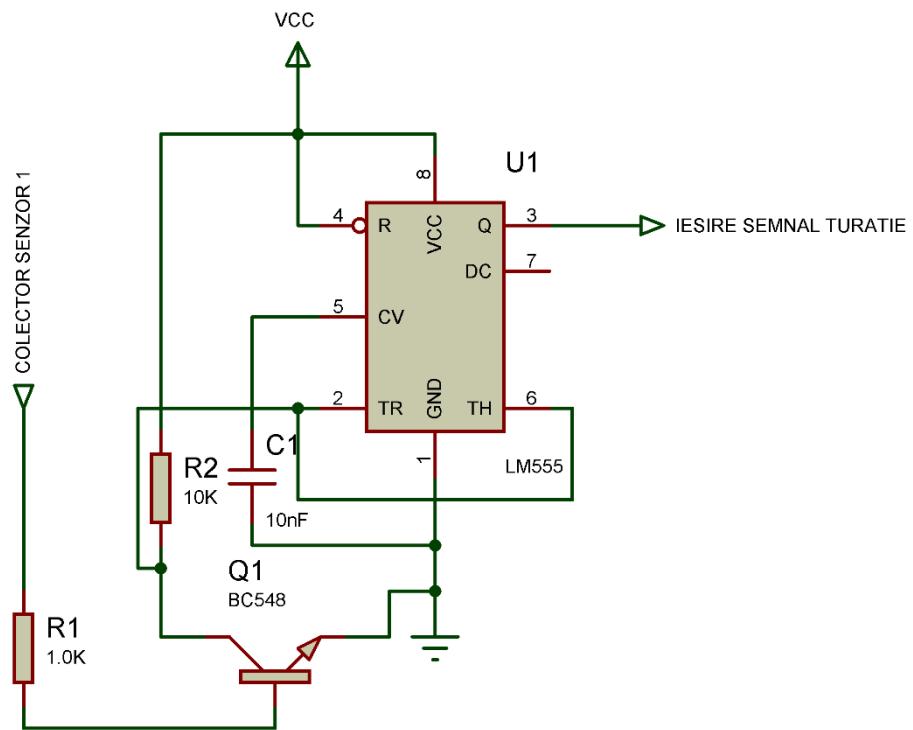


Fig. 3.10 – Trigger Schmidt modificat

Mai departe, vom trece la aplicarea semnalului de turație și vom monitoriza pe osciloscop, în paralel, semnalul furnizat de senzor și semnalul de la ieșirea trigger-ului astfel modificat.

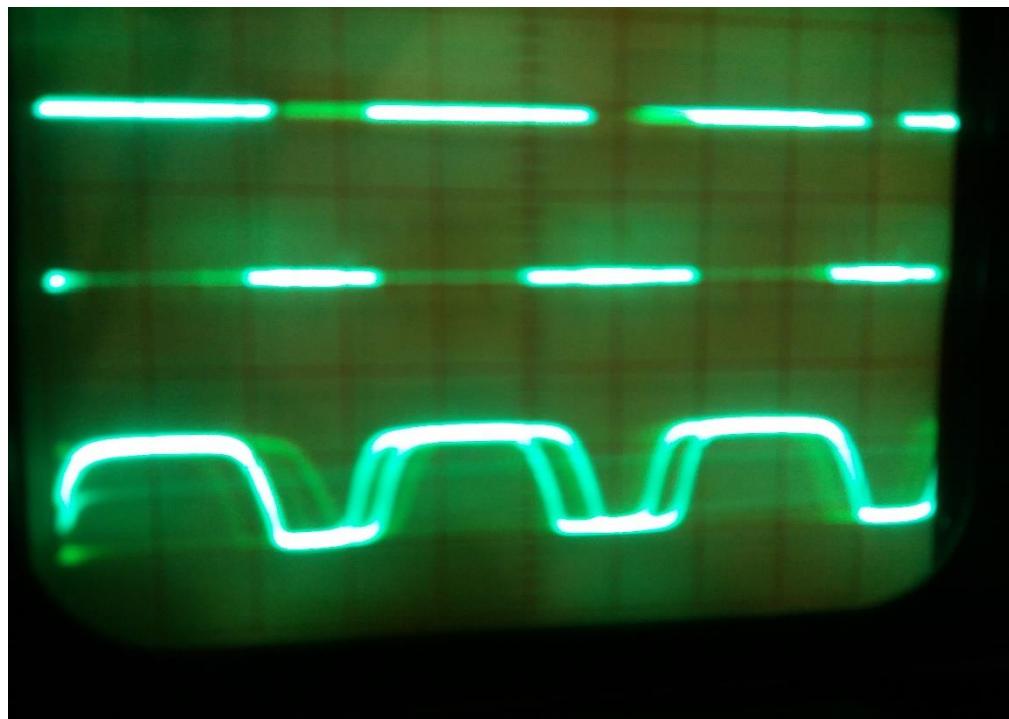


Fig. 3.11 – Turații mici

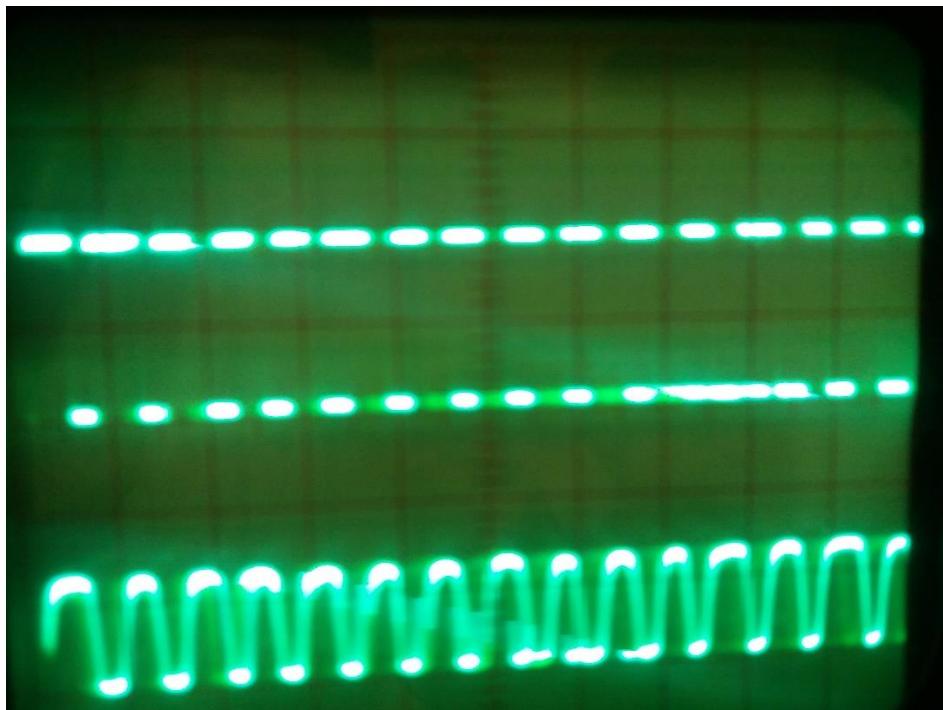


Fig. 3.12 – Turații mari

În partea de jos avem semnalul senzorului, iar în partea de sus apare semnalul prelucrat, de la ieșirea din trigger.

Variem turația motorului până la valori mai mari și verificăm faptul că semnalul prelucrat este semnificativ ameliorat în comparație cu semnalul original. Într-adevăr, acum avem de-a face cu un semnal dreptunghiular, compatibil TTL, ce se pretează mult mai bine aplicării la intrarea în pinul 11 (Timer 1) al microcontrolerului ATMEGA8. Astfel, am obținut un circuit de condiționare a semnalului de turație, care ne permite să trecem la partea digitală a acestui modul.

9. Vom pregăti o „placa rapidă” de încercări pentru microcontroler, dintr-o bucată de placa de test cu găuri, în care să avem spațiul necesar așezării unui soclu DIP 28, a unui regulator de tensiune LM7805 cu un mic radiator, a unui eventual oscilator cu cuarț și în care să externalizăm toți pinii de intrare/ieșiri disponibili la ATMEGA8, prin intermediul unor sârme de cupru un pic mai groase (0,6 mm) scoase prin găurile plăcii în partea frontală, la stânga și la dreapta microcontrolerului, și tăiate la o lungime de circa 5-6 mm. O vom numi în continuare „placă rapidă”, întrucât este rapid de realizat și totodată de reconfigurat. Nepunându-se accent pe partea estetică a acesteia și neîncărcând-o cu nicio componentă, acesta servește strict la alimentarea MCU-ului cu 5V tensiune continuă stabilizată. Toate componentele auxiliare vor fi montate pe alte plăci de test, iar conexiunile cu MCU-ul vor fi făcute prin fir subțiri, prin cositorire pe sârmele de externalizare a pinilor.

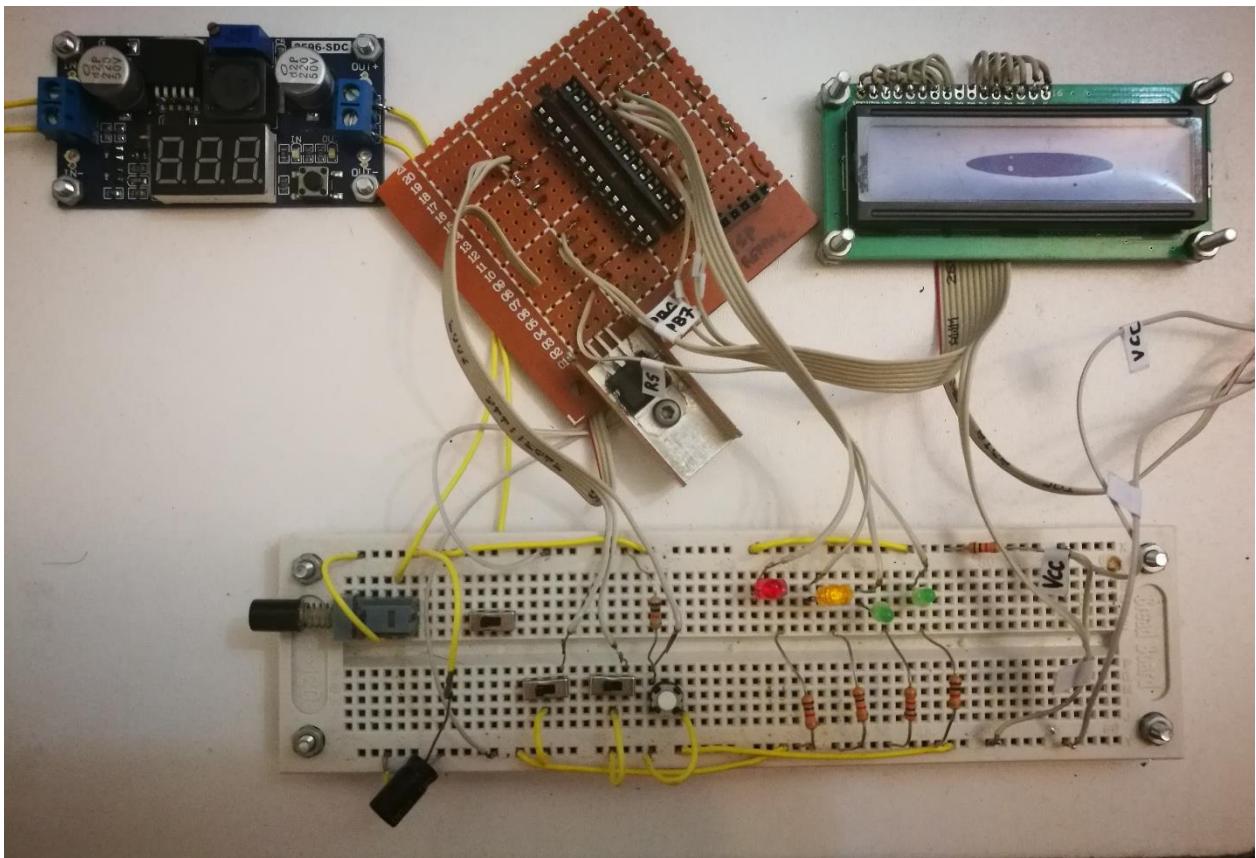


Fig. 3.13 – Placă rapidă de test

10. Vom sări peste etapa scrierii unui program de test pentru MCU, care să verifice funcționalitatea tuturor pinilor de intrare/ieșire, întrucât aceasta etapă a fost deja parcursă în mai multe rânduri înaintea începerii lucrării, pe diverse plăci de test, pentru diferite experimente. De asemenea, considerăm că cititorul acestei lucrări trebuie să aibă deja o experiență minimă în lucrul cu microcontrolerele, iar piesele destinate lucrării sunt presupuse ca fiind deja testate, astfel ca orice disfuncționalitate descoperită în timpul realizării acestei lucrări va indica mai degrabă o eroare de montaj sau de programare, decât un defect de piesă.

În acest caz, vom scrie direct un program de test pentru senzor, pentru a verifica întâi funcționarea de bază a acestuia. Aceasta presupune crearea mai întâi a unei configurații de numărator de fante (și de rotații). Vom folosi un afișor alfanumeric de tip HD44780, cu 2 linii și 16 caractere. Pe prima linie vom afișa numărul total de fante numărate, iar pe a doua linie numărul de rotații efectuate, folosind formula:

$$\text{Nr rotații} = \text{partea întreaga din } [\text{nr fante} / 119].$$

Nu mai expunem aici codul programului respectiv, întrucât acesta va constitui mai târziu o parte din programul final al simulatorului de motor, în funcția `void f_count_revs2()`, și asupra căruia vom reveni.

Pornim MCU-ul, scriem programul în MCU, apoi îl oprim.

Realizăm conexiunile necesare între circuitul de condiționare a senzorului și intrarea T1 (pinul 11) din MCU. Facem un marcat fin cu creionul, pe volanta, în dreptul acelui dintre lipsă (al 120-lea). Locul respectiv este o fanta de aproximativ trei ori mai lată decât celelalte. Aducem această fanta în

dreptul senzorului și pornim MCU-ul. Ambele linii trebuie să indice 0. Mișcând ușor volanta, observam cum MCU-ul începe să contorizeze trecerile fantelor prin senzor. Practicând mișcări fine, efectuăm o rotație completă, oprindu-ne tot în dreptul marcajului de pe volanță. Dacă totul este în regulă, trebuie să citim pe ecran 119 fante și 1 rotație.

- Dacă numărul de fante indicat este mai mic decât 119, este o problemă cu dantura volantei. Este posibil ca una dintre fante să fie obturată cu praf sau resturi de material necurățate după realizarea volantei; în acest caz ea va trece neobservată de senzor. De asemenea, este posibil ca jocul radial al volantei să fie prea mare, iar una sau mai multe fante din zona mai turtită a elipsei descrise în timpul unei rotații să iasă în afara axei optice a senzorului. Lungimea fantelor a fost în mod intenționat aleasă astfel încât, în cazul unor jocuri radiale acceptabile, reglajul de adâncime al senzorului să permită acoperirea acestor erori. Se va reface reglajul!
- Dacă numărul de fante indicat este mai mare decât 119, este posibil ca problema să provină din felul în care a fost realizată mișcarea. Este de la sine înțeles ca senzorul nu tine cont de sensul rotației. În cazul în care, de exemplu, atunci când luăm mâna de pe volanță după finalizarea unei mișcări, o rotim din greșelă câteva grade în sens invers, acele fante care au trecut prin senzor vor fi, de asemenea, numărate. Se va repeta o rotație completă începând din dreptul marcajului, încercând să realizăm mișcări mai fine și să ne oprim de cât mai puține ori până la finalizarea unei rotații complete. Poate fi util ajutorul unei a doua persoane, care să fie atentă dacă nu cumva discul s-a rotit în sens invers, de-a lungul mișcarilor efectuate. Mai există și posibilitatea ca o particulă de praf, mai îngustă decât o fantă, să se afle în interiorul unei fante și să creeze senzorului aparență unui alt dintă, și atunci se va număra în mod eronat o fantă în plus.

După rezolvarea oricărora probleme minore legate de ansamblul volanță-senzor, se va obține cu siguranță o citire corectă. Un număr de 10-20 de rotații complete, pentru a obține un număr de 1190-2380 de fante citite, poate garanta o verificare corectă a senzorului.

11. Scriem un nou program, pe baza programului anterior, în care realizăm măsurarea frecvenței semnalului de turație, numărând fantele timp de o secundă, împărțind apoi rezultatul la 119 și înmulțind cu 60, pentru a ne raporta la minut, obținând astfel mărimea rpm – „revolutions per minute”. Se face apoi afișarea pe display, iar ciclul se repeta la nesfârșit, având astfel turația monitorizată continuu, cu o cadență de aproximativ o secundă (operațiile aritmetice și cele de afișare pe display se execută într-un timp de maxim câteva zeci de milisecunde, astfel că afișarea se va face cu o perioadă < 1050 ms). Relația de legătură este, așadar:

$$RPM = f[Hz] * 60 / 119 \text{ min}^{-1} (2).$$

Dacă volanta ar avea o singură fantă, formula de trecere ar fi $RPM = f[Hz] * 60 \text{ min}^{-1}$.

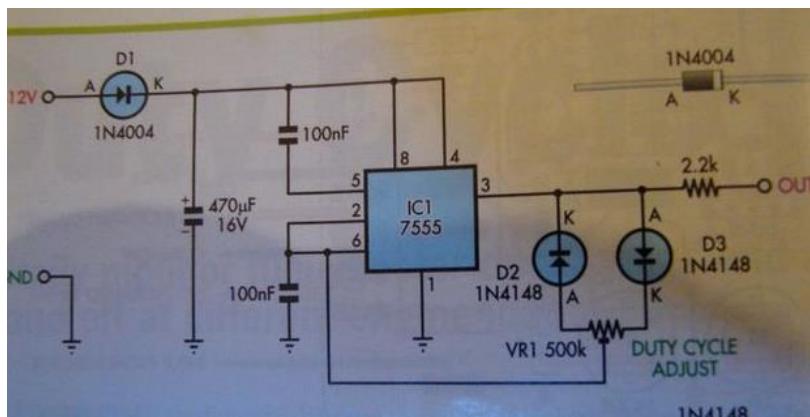
La fel ca în cazul anterior, nu mai expunem programul aici, întrucât funcția de măsurare a turației va constitui fundamentalul programului complet al simulatorului de motor.

Programăm MCU-ul cu noul program. Verificăm acest program, variind turația motorului de la cea mai mică turație stabilă pe care o putem obține, până la circa 4000 de rpm. Măsurăm, cu ajutorul multimetrului, frecvența la ieșirea din circuitul de condiționare al senzorului, pentru câteva valori distincte ale turației. Corelând aceste frecvențe măsurate cu turația afișată de pe display, ajungem la

concluzia că măsurarea se face corect pentru toata plaja de turații. Având aceste rezultate, putem folosi cu încredere acest ansamblu electromecanic acompaniat de un microcontroler ATMEGA 8 pentru a simula un motor termic.

12. Considerând un succes folosirea fotosenzorului în această aplicație, repetăm pașii 5-7 pentru a monta și cel de al doilea senzor pe planșă, pe care îl vom testa împreună cu circuitul de condiționare deja construit. Alimentând diodele LED ale ambilor senzori și rulând programul de la pasul 11, vom folosi un comutator de uz temporar pentru a trece rapid de la senzorul 1 la senzorul 2 oricând dorim, în timpul rulării programului. Atunci când avem o turație stabilă a motorului, prin comutarea de la un senzor la celălalt va trebui să nu sesizam nicio diferență în turația afișată. Facem această probă la câteva turații diferite. Firele acestui senzor le vom conecta direct la mufa de interconexiuni între module, întrucât citirea acestuia se va face Modulul de pornire (MP), acesta rămânând independent de circuitele motorului simulant.

13. Vom realiza un variator de turație cu PWM (Pulse Width Modulation – modularea impulsului în lățime). Până acum am folosit sursa reglabilă pentru controlul motorului, însă dorim alimentarea întregului circuit de la o sursă de tensiune fixă, de 12V, stabilizată. În urma unor căutări repetitive, atât pe Internet cât și în cărți și reviste de specialitate, am găsit foarte multe variante aproape identice ale unui oscilator simplu, realizat cu circuitul LM555. O variantă a unei astfel de scheme o găsiți mai jos.



Sursa: [Instructables.com, ultima accesare: 20.06.2017]

Modificând schema prin adăugarea unui reglaj suplimentar, putem varia atât durata lui t_{ON} , cât și pe cea a lui t_{OFF} , între aceleași intervale, însă păstrând o balanță mai bună între cei doi timpi, deci afectând aproape insensibil frecvența semnalului. Folosind un potențiometru dublu de 10 kOhm și conectând cursoarele celor două potențiometre astfel încât la creșterea valorii una să scadă valoarea celuilalt și viceversa, vom avea ca rezultat scăderea duratei lui t_{ON} simultan cu creșterea duratei lui t_{OFF} și viceversa, suma celor dure păstrându-se aproximativ constantă. Astfel, și frecvența se va menține aproape constantă, iar variația factorului de umplere al semnalului dreptunghiular rezultat se poate face de la 0% până aproape de 100%. Schema este redată mai jos.

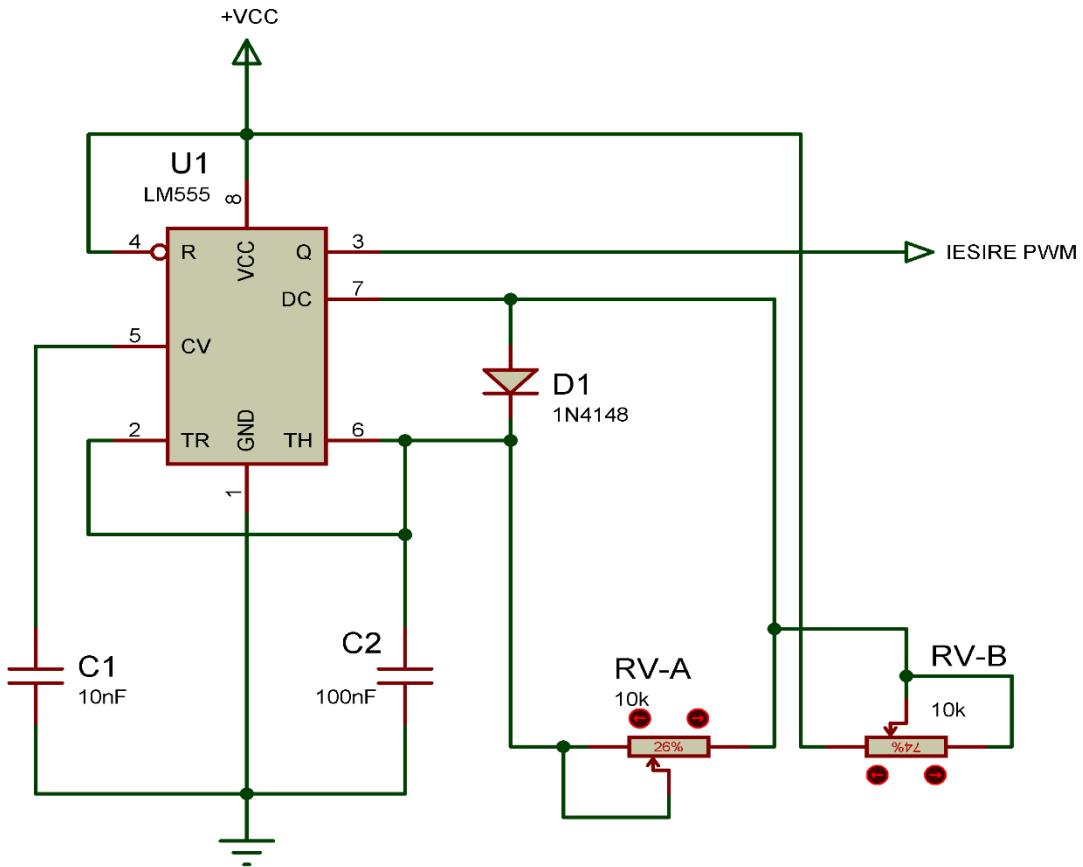


Fig. 3.14 – Schemă variator cu PWM modificată

Semnalul obținut este aplicat în baza unui tranzistor NPN de putere (BD651), cu emitorul conectat la masă, iar cu sarcina (motorul) conectată între colector și Vcc. O diodă redresoare (1N4007) este conectată în antiparalel cu sarcină (cu catodul către Vcc), pentru a realiza protecția tranzistorului împotriva salturilor de tensiune produse de sarcina inductivă în momentul deschiderii circuitului, acestei aplicații fiindu-i atribuită denumirea de *diodă flyback*.

Testând circuitul astfel obținut, alimentat de la o tensiune de 9V, și pornind microcontrolerul programat pe funcția de măsurare a turăției, constatăm că am obținut un circuit capabil să varieze turăția motorului într-o plajă de circa 120-3800 rpm, limita inferioară corespunzând tensiunii PWM minime la care motorul să fie capabil să funcționeze cât de cât uniform și fără întreruperi (fără să mai fie ajutat cu mâna pentru a-și menține rotația).

14. Realizam o schemă finală care să reunească toate circuitele descrise anterior, folosind două tensiuni de alimentare obținute prin folosirea a două regulatoare de tensiune liniare: 9V, dintr-un regulator de tip LM7809, pentru alimentarea motorului și a variatorului cu PWM, și 5V, dintr-un regulator de tip LM7805, pentru alimentarea circuitului de condiționare a semnalului de turăție și pentru alimentarea microcontrolerului.

Aceste două regulatoare vor avea intrările conectate în paralel, alimentarea acestora făcându-se de la o tensiune continuă de aproximativ 12V. Spunem „aproximativ”, întrucât exactitatea acestei valori nu este critică, din moment ce toate modulele sunt alimentate de la regulatoare de tensiune.

Aici este important să menționăm un aspect legat de eficiență energetică datorată folosirii acestor regulatoare: ea este foarte departe de ideal, fiindcă regulatoarele liniare au disipații de căldură ce cresc

semnificativ pe măsură ce crește diferența dintre tensiunea de ieșire și cea de intrare. Însă, în aplicația de față ne concentrăm mai mult pe implementarea unor funcții și pe corectitudinea funcționării acestora, decât pe aspecte constructive (design electric și mecanic, teste de eficiență și de durată, alte considerente specifice producției de serie).

De asemenea, am inclus câteva încrerupătoare care să ne ajute la efectuarea câtorva manevre utile, în timp real. Una dintre acestea ar fi decuplarea alimentării motorului atunci când se dorește asta (cu ajutorul lui SW 2), iar alta să ne permită acționarea permanentă a motorului (SW 1), chiar și atunci când releul de comandă este deschis. Un al treilea încrerupător comută între folosirea unui condensator de 22nF și unul de 100nF (din pinii 6 și 2 ai circuitului 555 pentru regulatorul de turație), selectând astfel frecvența semnalului PWM fie la aproximativ 4,5 kHz fie la 1 kHz.

A fost prevăzut un *push-button* conectat între pinul 1 (Reset) al microcontrolerului și masă, precum și unul între pinul 28 (PC5) și masă, denumit „OPTIUNI”, pentru a realiza funcția de selecție a modurilor de operare ale simulatorului de motor, în cazul în care mai apar idei pe parcursul scrierii programului final. Toate aceste aspecte se pot observa în schema din Anexa [1].

15. Folosind programul de proiectare a cablajelor EAGLE (Easily Applicable Graphical Layout Editor), versiunea 7.6.0 Professional cu licență gratuită pentru scopuri educaționale, efectuăm desenul cablajului. Folosim, în măsura în care este posibil, amprenta pieselor reale ce urmează a fi folosite, sau, dacă acestea nu se regăsesc în librăria inclusă, folosim amprenta altor piese care au aceeași distribuție, spațiere și grosime a pinilor, sau chiar amprente de tip gaură, plasate în locurile corespunzătoare orientându-ne după matricea de linii din fundal, a cărei distanță între linii este și ea configurabilă.

Plecând de la o dimensiune a plăcii care să ne permită montarea display-ului alfanumeric pe lățimea ei (deci vom folosi o orientare de tip portret), alegem o dimensiune de 10x15cm, întrucât avem deja la dispoziție mai multe plăci de acest fel. Urmărим o distribuire uniformă a pieselor pe placă, lăsând mai mult spațiu la dispoziție regulatoarelor (care se prezumă ca vor disipa căldura), și display-ului în partea de sus.

Odată finalizat desenul cablajului, acesta va fi imprimat cu imprimanta laser pe o hârtie de tip fotografie (hârtie mai groasă și lucioasă pe o parte) pentru a aplica „metoda fierului de călcat”. Imaginea cablajului poate fi găsită la Anexa [2].

16. Realizăm imprimarea și decaparea PCB-ului. Metoda amintită mai sus presupune următoarele: desenul cablajului este încadrat într-un chenar, pentru a ne ajuta la alinierea hârtiei pe placa de textolit. Hârtia se aşază cu desenul peste partea cuprătă a plăcii de textolit și, după o verificare vizuală a alinierii, se îndoiește peste partea cealaltă, obținând astfel o împachetare a plăcii în hârtie. Se aşază placa cu partea cuprata în sus, pe un suport termorezistent și se calca cu fierul de călcat setat pe temperatură maximă de funcționare.

O detaliere exactă a procesului și a timpului de călcare este inutilă în acest context, însă ceea ce trebuie să fie clar înțeles este faptul că suprafața trebuie parcursă în mod uniform și de cât mai multe ori, până când desenul începe să devină puțin vizibil, ceea ce înseamnă că porțiunile fără toner din hârtie s-au tasat mai mult decât porțiunile cu toner. După acest proces și după o răcire rezonabilă a plăcii, hârtia se curăță sub un jet continuu de apă caldă, până când pe placa rămân vizibile doar traseele circuitului, fără resturi de hârtie. După uscare, cu ajutorul unui vârf ascuțit, se pot curăța ușor reziduurile de hârtie dintre trasee ce nu au putut fi îndepărtate cu apă. Eventualele încreruperi ale traseelor se vor corecta cu un lac impermeabil după uscare (sau ojă). După o inspecție completă, placa

se introduce într-un recipient în care se va turna un strat de minimum 5mm de clorura ferica (FeCl soluție). Pentru grăbirea procesului de decapare, există două posibilități: agitarea continuă a recipientului sau încălzirea lichidului. În lipsa acestora, decaparea ar trebui să dureze aproximativ o ora. Se va verifica starea de decapare din 10 în 10 minute. Procesul se poate încheia mai devreme dacă se constată o decapare completă, sau se poate prelungi oricât este nevoie dacă decaparea are loc prea lent. După acest proces, cablajul se va spăla și se va lăsa la uscat. După uscare, se va inspecta pentru ultima dată integritatea traseelor, uitându-ne la placă ținuta în dreptul unei surse de lumina mai puternice. Dacă, accidental, traseul a fost alterat (în sensul de întrerupere) în anumite locuri, aceste întreruperi se vor corecta prin orice metodă convenabilă (cositorirea unui fir de la un capăt la altul al zonei întrerupte etc.). În final, se va îndepărta stratul de toner din zonele de lipire a componentelor, prin răzuire, folosind o șurubelnită cu cap drept sau alt obiect ascuțit.

17. Ultima etapă a construcției plăcii constă în realizarea găurilor pentru montarea componentelor, folosind o bormașină și burghie având diametrul cuprins în intervalul [0.6 ; 1.5] mm, și lipirea acestora prin cositorire.

18. După finalizarea montajului de pe placă, se vor realiza conexiunile următoare: alimentarea, senzorul numărul 1 și motorul.

Se va verifica prima dată funcționarea variatorului de turăție și a motorului. Înainte de alimentarea montajului, vom regla potențiometrul RV3 al variatorului la capătul stâng al cursei, corespunzător turăției minime (practic nule) a motorului. Se alimentează circuitul de la o tensiune de 12V c.c. și se verifică cu multimetrul faptul că avem o tensiune de 5V, respectiv 9V la pinul 3 al regulatoarelor 7805, respectiv 7809. Comutam întrerupătorul SW1 în poziția "ON" și acționăm treptat potențiometrul, urmărind pornirea motorului și creșterea corespunzătoare a turăției, până la atingerea turăției maxime. În același timp, verificam vizual ca funcționează LED-ul 4. După primele secunde de funcționare, este indicat să verificam temperatura celor două regulatoare și a tranzistorului Q3. Toate trei au fost prevăzute cu radiator și s-a folosit pasta termoconductoare, iar în cazul unei funcționări corecte, ele nu se vor încălzi în mod sesizabil.

În acest moment întocmim și un tabel al pinilor din mufa DB9, care poate fi găsit în Anexa [3].

19. Se vor scrie, pe rând, programe simple pentru verificarea funcționalității următoarelor componente: reul de acționare a motorului, cele trei leduri de semnalizare a stărilor. Se va verifica afișarea pe display precum și funcționarea butoanelor "Reset" și "Optiuni".
În etapa realizării schemei s-a decis distribuția intrărilor și ieșirilor pe pinii portului C al microcontrolerului.

Distribuția este următoarea:

- PC0 - (ieșire) comanda releu motor;
- PC1 - (ieșire) comanda LED 3;
- PC2 - (ieșire) comanda LED 2;
- PC3 - (ieșire) comanda LED 1;
- PC4 - nefolosit;
- PC5 - (intrare) buton "OPTIUNI";
- PC6 - (atribuit prin construcție) buton "RESET".

20. În continuare, se va concepe un program complet, care să înglobeze o serie de funcționalități care vor fi testate la secțiunea următoare.

Programul astfel obținut va fi găsit în Anexa [4].

3.2.2 Verificarea programului

După încărcarea acestui program în microcontroler, vom face o verificare completă a funcționării acestuia, după un procedeu ca cel descris în continuare. La pornirea alimentării, sau după orice apăsare a butonului "Reset", simulatorul trece în *starea "1 - Motor oprit"*, întrucât aceasta este starea de început. Întrerupătorul SW1 trebuie să fie în poziția "OFF", iar SW2 în poziția "ON". În această stare, LED-ul 1 (roșu) este aprins, motorul este oprit, pe prima linie a afișorului este afișat mesajul „1. MOTOR OPRIT”, iar pe rândul al doilea este afișată turația (0). La o învârtire ușoara a axului volantei, se va putea citi turația curentă, actualizată odată la o secundă.

Învârtind mai rapid axul volantei, astfel încât turația să depășească 50 rpm, simulatorul trece în *starea "2 - Motor în curs de pornire"*. În acel moment se va auzi acționarea releului, iar simultan cu aceasta se va stinge LED-ul roșu și se va aprinde LED-ul Y galben, va fi afișat pe prima linie mesajul „2. MOT. IN PORNIRE” iar pe a doua linie va fi afișată în continuare turația curentă, actualizată la o perioadă de 1s. Dacă potențiometrul a fost lăsat anterior într-o poziție corespunzătoare unei turații de minim 700 rpm, atingerea acestei turații va dura în jur de 2-3 secunde, astfel că simulatorul va petrece în starea 2 un timp suficient de mare pentru a putea urmări toate aceste acțiuni. În cazul în care turația cîtită nu depășește niciodată 600 rpm în parcursul a 6 secunde, simulatorul va trece înapoi în starea 1, realizând următoarele acțiuni: decuplarea releului, stingerea LED-ului galben și aprinderea LED-ului roșu. În cazul depășirii pragului de 600 rpm într-un timp mai scurt de 6 secunde, se trece în starea 3.

Ajuns în *starea "3 - Motor pornit"*, simulatorul va stinge LED-ul galben și va aprinde LED-ul 3 (verde), va menține releul anclansat, va afișa pe prima linie mesajul „3. MOTOR PORNIT”, iar pe linia a doua va continua să afișeze, cu aceeași frecvență, turația curentă. În acest moment putem varia după cum dorim turația cu ajutorul potențiometrului, observând cum valoarea acesteia este permanent actualizată. Din aceasta stare avem trei metode a opri motorul pentru a reveni în starea 1: din potențiometru, din întreupătorul SW1 și mecanic. Prima metodă presupune reglajul treptat al turației spre valori tot mai mici, până la atingerea unui prag mai mic de 350 rpm. Aceasta este o metodă facilă pentru a verifica exactitatea acestui prag, întrucât putem reduce turația în trepte foarte fine. A doua metodă presupune deschiderea circuitului motorului comutând întreupătorul SW1 în starea "OFF", astfel că motorul se va opri din inertie, iar scăderea turației sub pragul de decizie pentru oprire va avea loc destul de repede, neputând surprinde exact momentul în care s-a atins acel prag. Metoda mecanică presupune frânarea volantei strângând axul cu degetele, exercitând o presiune mai mică sau mai mare, deci imprimând motorului o forță de rezistență la învârtire mai mică sau mai mare. Strângând axul suficient de tare de la început, putem obține o oprire aproape instantanee. Cu metoda aceasta putem simula situația reală de "motor omorât", cu metoda anterioară putem simula oprirea normală, "din cheie", iar cu prima metodă, situația panei de carburant la mers constant, angajat în treapta de viteza superioara. Indiferent de metoda aleasă, trecerea din starea 3 în starea 1 se va face

realizând următoarele acțiuni: decuplarea releului, stingerea LED-ului verde și aprinderea LED-ului roșu.

Starea 4 poate fi accesată din stările 1 sau 3, nu și din starea 2. Aflându-ne într-una din aceste stări, apăsam butonul "Optiuni" și îl menținem apăsat, până când cele trei LED-uri 1, 2 și 3 vor clipe simultan, cu o frecvență de 2,5Hz. Timpul necesar de menținere a apăsării este de maximum o secundă, în funcție de momentul în care ne aflăm relativ la timpul de 1s în care microcontrolerul este ocupat cu măsurarea turației. După observarea secvenței de clipire a LED-urilor, se eliberează butonul iar simulatorul trece în starea "4 - Afisare turație". Aici, indiferent din starea din care am plecat (1 sau 3), releul motorului își va păstra starea de acționare anterioară tranzitiei în starea 4. Astfel, dacă am pornit din starea 3, motorul va rămâne acționat chiar și după scăderea turației pana la 0, putând studia regimul de turații joase (de sub 350 rpm) și încercând să vedem care este turația minimă stabilă pe care o putem obține. În aceeași manieră, dacă am pornit din starea 1, motorul va rămâne neacționat din releu, dar îl putem acționa din intrerupătorul SW1, care realizează un "bypass" asupra releului. Această observație este valabilă și pentru stările 5, 6 și 7. În această stare, pe prima linie a afișorului este vizibil mesajul "4. AFİŞ. TURAȚIE", iar pe linia a doua, turația curentă, la fel ca la celelalte stări. De asemenea, cele trei leduri sunt stinse, și vor rămâne așa și pentru stările următoare. La o nouă apăsare, după aceeași regulă, a butonului "Optiuni" putem trece în starea 5.

Starea "5 - Numără rotații efectuate". În această stare, pe prima linie este afișat mesajul "NUMĂRĂ ROTATII", iar pe linia a doua este afișat numărul de rotații complete, adică numărul multiplilor întregi de 119 dinți ce parcurg senzorul de turație. Numărarea începe de la intrarea în această stare. De data aceasta, frecvența de afișare a numărului de rotații pe linia 2 este de 10 Hz. Dacă dorim să folosim un reper vizual pentru a compara cu ușurință numărul efectiv de rotații efectuate cu numărul de rotații afișat pe ecran, vom reveni la starea 1 și, învărtind ușor volanta, vom aduce săgetuța de pe volanta în dreptul senzorului numărul 1. Trecem înapoi în starea 5 și rotim volanta, fie manual, fie pornind motorul din intrerupătorul SW1 și reducând turația astfel încât să putem urmări cu ușurință rotația. Constatăm că, după oricâte rotații efectuate, incrementarea numărului de rotații afișat pe ecran se va face mereu atunci când săgetuța este exact în dreptul senzorului. Putem apoi să creștem turația până la valori mai mari și apoi să o reducem din nou până când putem urmări vizual săgetuța. Ar trebui să ajungem la aceeași concluzie. Reușita acestui experiment ne oferă o garanție a faptului că citirea turației se face fără erori. Apăsăm butonul „Optiuni” pentru a trece mai departe.

Starea „6 – Numără rotații și fracțiuni de rotație” este o extensie a stării anterioare, în sensul că, pe lângă numărul de rotații complete parcuse, este afișată și partea subunitară, ca număr de dinți. Pe prima linie este afișat mesajul „6. ROTATII+DINTI”, iar pe rândul al doilea sunt afișate cele două valori sub forma: „[număr rotații], dinți: [număr dinți]”. Numărul de dinți afișat poate lua valori între 0 și 118. și această stare este utilă pentru a verifica exactitatea măsurării prin intermediul senzorului fotosensibil.

Ultima stare, „7. Afisează grade de rotație”, realizată mai mult din curiozitate, afișează, la fel ca starea anterioară, partea subunitară a rotațiilor, însă sub formă de grade. Pe prima linie este afișat mesajul „7. GRADE ROTATIE”, iar pe linia următoare, numărul de grade, calculat după formula nr dinți * 3°. Având în vedere numărul total de 119 dinți (și o pauză) de pe circumferința de 360°, este evident faptul că rezoluția citirii fazei de rotație a volantei este de 3°. Tot cu această rezoluție este capabil și

calculatorul de injecție al mașinii să citească faza arborelui cotit, pentru a calcula pe baza acestei valori, avansul aprinderii precum și timpii de injecție.

Prin apăsarea butonul „Optiuni” vom reveni la starea 1, astfel ca am obținut o parcurare secvențială a tuturor stărilor și opțiunilor simulatorului. Desigur, dacă dintr-una din stările 4-7 dorim să revenim la starea 1 și parcursul „înainte” al opțiunilor ni se pare prea lung, putem apăsa butonul „Reset”, iar simulatorul se va inițializa în starea 1.

În acest moment, realizarea simulatorului de motor este completă. Vom începe, în sfârșit, realizarea montajului propriu-zis, ce poartă titlul lucrării. Poate am fi tentați să spunem „Câtă muncă pentru nimic!”, însă acest „nimic”, pe lângă satisfacerea unor curiozități privind felul în care poate fi simulat, la modul minimal, un motor, ne va ajuta enorm la testarea modulelor proiectului. Vom vedea că în unele etape va fi indispensabil. Desigur, în condițiile în care dispunem de un atelier foarte dotat (în garaj), având la dispoziție un acumulator auto „bun de sacrificat” cu încercări de demaraj repetate, putem monta și testa modulele direct pe vehicul, folosind componente de putere adecvate (releele, în principal). Acest lucru însă poate fi periculos și anevoie, necesitând montarea și demontarea frecventă a acestora, punerea periodica a acumulatorului la încărcat, asumarea riscului de supraîncălzire a demarorului și acumulatorului, și păstrarea în permanență la îndemâna a unor mijloace de prevenire a riscurilor (stingător, un întrerupător de mare putere montat în serie cu acumulatorul etc.). Realizăm imediat că ne este mult mai ușor să folosim un simulator care să modeleze cu destulă acuratețe tocmai parametrii vizati de proiect.

3.3 Constructia modulelor propriu-zise

Vom privi din nou schema bloc a sistemului de pornire, pe care o găsim în Anexa [5] și vom alege într-o ordine convenabilă, modulele pe care urmează să le realizăm. Aici vom face câteva mențiuni valabile pentru toate modulele: niciunua dintre acestea nu trebuie să îi lipsească un header de 5 pini drept mufa de programare, fiecare trebuie să aibă un regulator de tensiune propriu (în general, LM7805), un LED de indicare a prezenței tensiunii, un condensator de filtrare a alimentării. De asemenea, pentru conectivitate, se va alege varianta cea mai potrivită de mufă, în general DB-9 fiind o alegere potrivită.

Modulul de distribuție a alimentărilor (MDA)

Ne vom concentra doar pe schema bloc a acestui modul și vom stabili o listă cu intrările și ieșirile aferente acestui modul:

Intrări:

- ACC_REQ (din eng. „Accessory Request” – Cerinta ACC). Este un semnal codat pe 1 bit, întrucât are doar două stări – adevărat (1) și fals (0). Aceasta ne este necesar pentru conectarea la Modulul de Telecomanda (MT) care, în momentul descuierii mașinii, comandă pe termen nelimitat alimentarea ACC, urmând să o anuleze la momentul încuierii. Întrucât este primul semnal care intervine la ieșirea din starea de repaus a mașinii, dorim să îl atașam unei intrări de întrerupere externă a microcontrolerului, în eventualitatea în care ne hotărăm să stabilim un protocol de trezire-adormire a acestui modul. Astfel, acesta poate fi semnal de trezire (*wake-up*) pentru modulul de

față. Alegem pinul 2 al portului D (PD2) care are funcția alternativa INT0 (Interrupt 0) prin construcția MC-ului.

- ALIM_REQ („Request” de alimentare). Este un semnal codat pe 2 biți, întrucât are patru stări posibile, după cum se poate vedea în schema logică pe care o vom construi mai târziu. Acesta este recepționat de la Modulul de Pornire (MP), care este suveran asupra stării alimentarilor ACC și IGN. Îl vom atașa pinilor 0 și 1 ai aceluiași port (PD0 și PD1).

Ieșiri:

- ACC_RLY (din eng. „Accessory Relay” – Releu ACC).
- IGN_RLY („Ignition Relay” – Releu IGN).

Acestea două sunt semnale de comandă pentru releele ACC și IGN și sunt, fiecare, codate pe 1 bit, întrucât au câte două valori posibile – pornit (1) și opri (0). Le vom atașa pinilor 0 și 1 ai portului C (PC0 și PC1), deoarece ne-am decis să nu atribuim semnale cu sensuri diferite (intrare – ieșire) pe același port. De asemenea, distribuția oarecum uniformă a pinilor celor două porturi pe către o parte a MC-ului ne ajuta să adoptăm un stil de realizare a conexiunilor în care, pe partea stânga să avem doar intrări, iar pe partea dreaptă doar ieșiri.

Având stabilită configurația electrică a modulului și amintindu-ne convențiile referitoare la nivelurile electrice ale unui semnal de intrare, respectiv de ieșire, atașam semnalelor de intrare un grup de tranzistori NPN de inversare a logicii și o rețea de rezistente de „pull-up”. De asemenea, pentru testarea individuală a modulului, montăm câte un microîntrerupător în paralel cu fiecare semnal de intrare și linia de alimentare de 5 volți.

Pentru ieșiri, ținem cont că va trebui să alimentăm niște relee. Acestea sunt relee de mici dimensiuni și de mică putere (în configurația pentru machetă), de tipul JQC-3F/T73-12VDC (pentru ACC) și GH-1C-12L (pentru IGN).

Acestea sunt proiectate pentru alimentarea solenoidului la o tensiune de 12 volți, deci tensiunea colector-emitor suportată de tranzistorul în blocare va fi egală cu aceasta. De asemenea, măsurând curentul consumat de solenoid la aceasta tensiune, obținem o valoare de 32 mA pentru oricătre relee identice. Astfel, deducem că este suficient un plafon de 100 mA pentru curentul de colector. Tranzistorul BC548 se conformează cu succes acestor cerințe, aşadar îl vom alege ca element de comandă pentru relee.

Nu uităm să includem un „header” de tip mama cu 5 pini drept conector pentru programare, un condensator de filtrare a alimentării de 100 uF / 16V și un LED pentru indicarea prezentei tensiunii de alimentare. Ca element regulator vom folosi clasicul LM7805, însă fără radiator de acum înainte, întrucât toate modulele vor consuma curenți foarte mici, cu mult sub 1A cât garantează regulatorul. Pentru conectivitate folosim o mufă de tip DB-9 mama, iar semnificația pinilor acestora va fi descrisă în tabelul din Anexa [6]. Cele două relee au fost montate pe un cablaj separat (numit „Plăcuță relee” - PR), pentru a evidenția faptul că, pe mașina, este foarte probabil ca montarea acestora să se facă la distanță de locul unde s-ar monta MDA-ul. Schema obținută poate fi consultată la Anexa [7]

În urma proiectării cablajului obținem desenul din Anexa [8], pe care îl imprimăm prin metoda cunoscută. După asamblarea componentelor obținem modulul din figura de mai jos.

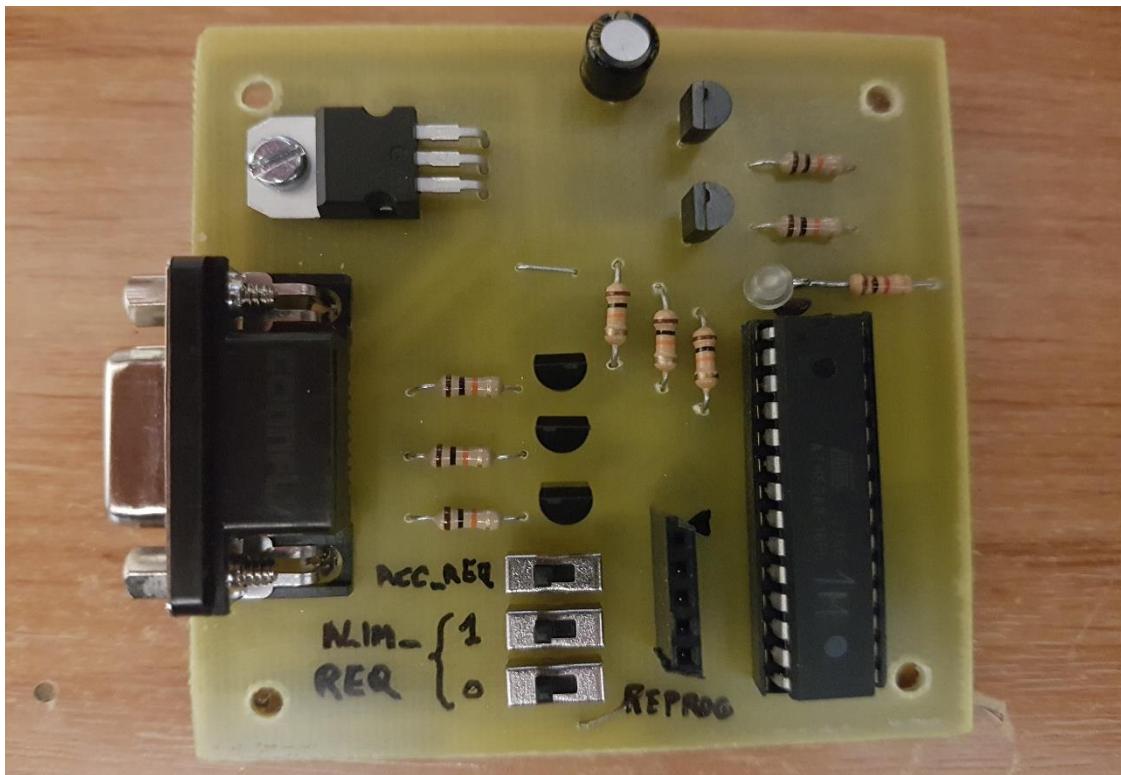


Fig. 3.15 - Modulul de distribuție a alimentărilor

De asemenea, schema și rezultatul montajului plăcuței cu relee se regăsesc mai jos. Menționez că imaginea cu cablajul se regăsește la Anexa [9].

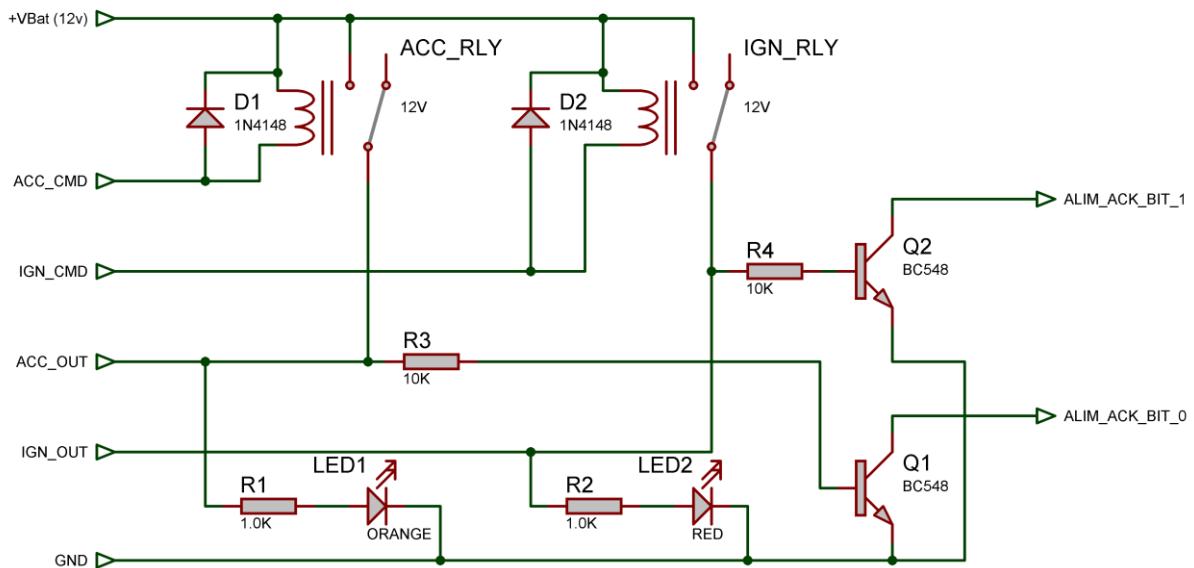


Fig. 3.16 – Schemă montaj plăcuță relee

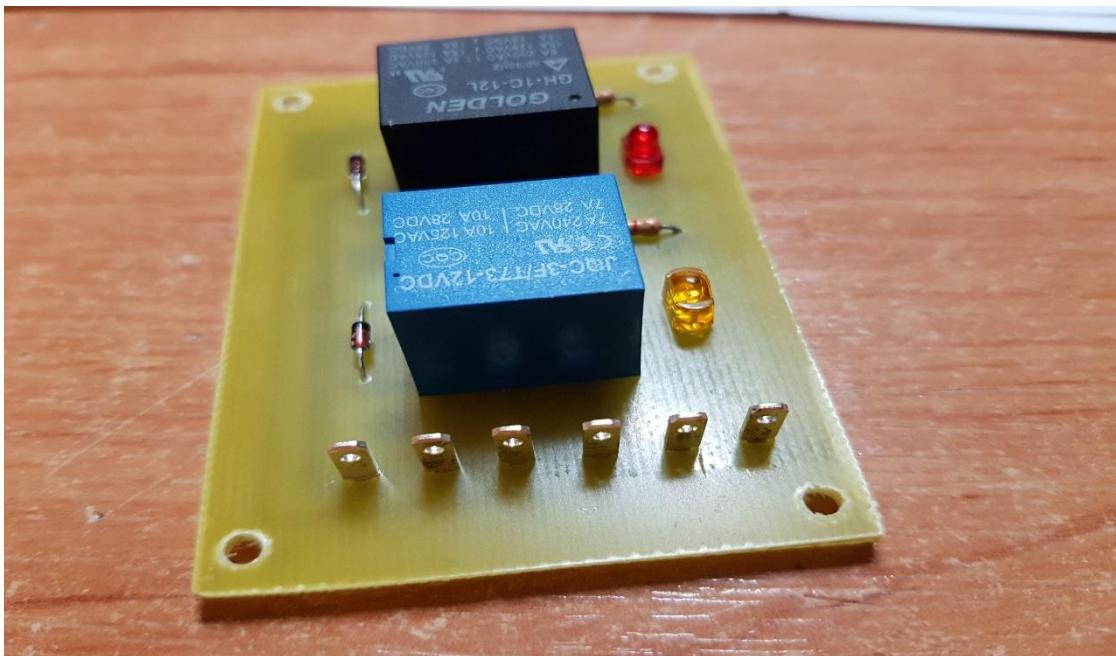


Fig. 3.17 - Montajului plăcuței cu relee

Să concepem acum schema logică de funcționare a MDA-ului, pentru a putea dezvolta mai ușor programul care va rula în acest modul. Dorim să folosim în continuare conceptul mașinii de stări, iar tranziția între stări se va face în funcție de semnalele de intrare. Întrucât nivelurile de alimentare posibile sunt în număr de 4 : ACC și IGN ambele oprite, ACC pornit și IGN oprit, ACC oprit și IGN pornit, ACC și IGN ambele pornite, este convenabil ca stările să corespundă acestor niveluri de alimentare. Semnalul ACC_REQ este prioritat în starea 0, în sensul că menține ACC pornit chiar dacă ALIM_REQ este 0. El însă nu este prioritat și în starea 2, unde ALIM_REQ trebuie să poată cere în mod expres oprirea lui ACC, cerință utilă la pornirea programată, mai ales pe timp de iarnă, când bateria are performanțe limitate datorită temperaturilor scăzute.

Cu aceste mențiuni, am obținut schema logică din Anexa [10]. Scrierea programului este acum mult mai facilă. Nu uităm să ținem cont și aici că, pentru semnalele de intrare, 0 înseamnă adevărat/pornit, iar 1 înseamnă fals/oprit. Vom folosi un artificiu de programare, prin definirea (cu directiva #define) lui ALIM_REQ în felul următor:

```
#define ALIM_REQ 3-((2*PIND.1)+PIND.0),
```

astfel că ALIM_REQ va fi 0 atunci când, de exemplu, PIND.1 și PIND.0 sunt 1 (inactive logic), respectiv 3 atunci când cele două componente sunt simultan 0 (active logic). Programul final se poate regăsi la Anexa [11].

Semnalul de ceas (CLOCK) ales pentru acest modul este de 1MHz (intern), setând „fuse-bitii” CKSEL 3...1 cu valorile „1001”. Am ales această frecvență întrucât tranzițiile între stări nu sunt foarte dese, în timpul unei funcționari obișnuite, și nu există operații complicate care să necesite o viteză de procesare mai mare.

Testam acum modulul MDA conectat cu placuta PR. Acționând întrerupătoarele denumite după semnalele de intrare, putem auzi anclanșarea releelor și totodată putem observa vizual LED-urile montate sub acestea. Dacă acestea sunt în concordanță cu poziția întrerupătoarelor, conform schemei logice, atunci am finalizat cu succes construcția Modulului de Distribuție a Alimentarilor.

Modulul de comanda a starterului (demarorului) (MCS)

Acest modul se concentrează strict pe comanda demarorului și pe monitorizarea parametrilor care condiționează această comandă. Întrucât demarorul este un element de foarte mare putere (consumând în jur de 100-150A, cu un vârf de aproximativ 500A în momentul inițial al acționării) și, în configurația mașinilor de generație veche, nu beneficiază de niciun element de protecție împotriva acționării accidentale în condiții incorecte, acest modul corectează acest neajuns și filtrează condițiile improprii demarajului în momentul cererii de demaraj. Principalele elemente de siguranță necesare unui demaraj corect sunt următoarele:

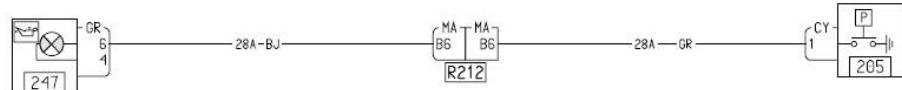
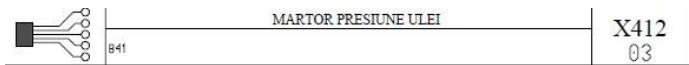
- Turația motorului, care trebuie să fie nulă în momentul pornirii;
- Starea cutiei de viteze sau măcar a pedalei de ambreiaj, din care trebuie să aflăm cu certitudine că transmisia este decuplată în timpul demarajului.
- Presiunea de ulei, care trebuie să ajungă la valoarea normală de funcționare în maxim câteva secunde după pornirea cu succes a motorului.

În plus, pentru a ne asigura suplimentar împotriva unei blocări în starea de demaror acționat, vom folosi două relee pentru comandă fizică. Contactele acestora vor fi inseriate, astfel că, în cazul defectării unui releu în poziția închis, circuitul va fi deschis de celălalt releu. Solenoidele acestora vor fi comandate de două ieșiri separate ale MC-ului. De asemenea, modulul MCS va furniza și un semnal referitor la starea motorului, codat pe 2 biți, având următoarele semnificații posibile: motor oprit (0), motor în curs de pornire (1), motor pornit (2) și solicitare de oprire motor (3). Acest semnal va fi recepționat de Modulul de pornire (MP). Valoarea „3” a semnalului este generată atunci când, după un timp de 5 secunde de la pornirea motorului, presiunea de ulei nu este în parametrii normali, astfel că MP va comanda oprirea motorului printr-o comandă de decuplare a releului IGN (din care este alimentat și calculatorul de injecție).

Pornind de la aceste premise, avem o idee asupra semnalelor de intrare și de ieșire ale MCS.

Intrări:

- START_REQ („Start request” – Solicitare pornire) – este un semnal codat pe un bit, provenit de la MP, având ca valori posibile „adevărat” și „fals”. Acesta va fi atașat pinului PD2, din aceeași rațiune ca la modulul de mai devreme, privind secvența de adormire/trezire.
- TURATIE – semnal provenit de la senzorul de turație numărul 2, prelucrat în același fel ca în cazul Simulatorului de motor. Va fi citit prin pinul PD5.
- PRESIUNE_ULEI – aceasta presiune este preluată sub două valori posibile: presiune corecta și presiune slabă, prin închiderea la masa manocontactului montat pe blocul motor.



Va fi atribuit pinului PD1.

- NEUTRU – provenit de la întrerupătorul de „punct mort cutie de viteze” și întrerupătorul de ambreiaj apăsat, ce va fi conectat în paralel cu primul, pentru a obține funcția logica „SAU”. Este suficient să avem cutia decuplată sau ambreiajul călcat pentru a porni în siguranță motorul. Se va conecta la pinul PD0.

Ieșiri:

- STARTER_REL_1 și STARTER_REL_2 (Relee starter), obținute de la pinii PC0, respectiv PC1.
- ENG_STATUS_BIT_0 și ENG_STATUS_BIT_1 („Engine status”) sunt cei doi biți ai semnalului de stare a motorului, descris mai devreme, și vor fi atașați pinilor PC2 si PC3.

Releele de comandă a demarorului pe care le vom folosi aici sunt de același tip cu cel de la Simulatorul motorului (SM) , NEC/TOKIN EX1-2U1. Vor fi folosiți, aşadar, tot tranzistori de tipul BC548. Semnalul START_REQ nu are tranzistor de inversare a logicii, acesta va fi inclus în modulul MP la ieșirea pinului respectiv. Pentru semnalele NEUTRU și PRES_ULEI vor fi montate microîntrerupătoare pe placă, care prin închidere conectează la masa pinii corespondenți ai MC-ului, obținând stările „cutia la punctul mort” și „presiune de ulei corectă”.

Semnalul TURAȚIE este trecut printr-un circuit de condiționare identic cu cel de la SM. Așadar, vom include două regulatoare de tensiune, un LM7805 pentru microcontroler și un LM7809 pentru circuitul LM555. Pentru conectivitate vom alege, ca de obicei, o mufă de tip DB-9, având corespondența pinilor descrisă în tabelul din Anexa [12]. Desenăm schema conform acestor date, o verificăm și proiectăm cablajul pe baza ei. Schema se găsește la în Anexa [13], iar cablajul la Anexa [14] . După montarea pieselor, obținem rezultatul de la figura de mai jos.

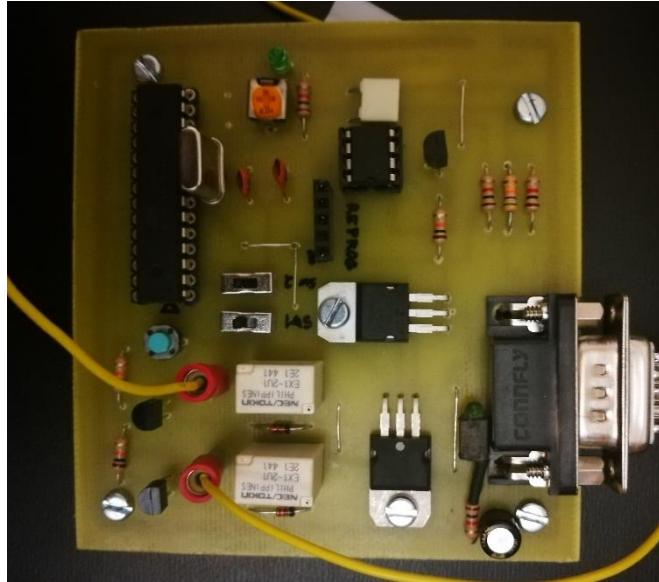


Fig. 3.18 - Modulul de comanda a starterului

Ne gândim la realizarea schemei logice care să descrie funcționarea acestui modul. Continuând pe ideea mașinii de stări, decidem că cea mai bună variantă de definire a stărilor se poate face după starea de funcționare curentă a motorului, cea care se și transmite prin semnalul ENG_STATUS. Avem aşadar patru stări, tranzițiile între acestea făcând-se în principal după variația semnalului START_REQ și fiind condiționate de turatie, presiune de ulei și poziția contactorului de „punct mort”. O schema logică ce modelează foarte bine funcționarea pe care dorim să o obținem se găsește în Anexa [15]. Această schemă este simplificată, în sensul că evidențiază doar tranzițiile posibile între stări, nu și elementele care determină acele tranziții.

Pe baza acestei scheme, putem începe implementarea programului. Vom adauga în continuare niste mentiuni referitoare la acțiunile care au loc în diferitele stări. În fiecare stare se va actualiza semnalul ENG_STATUS corespunzător respectivei stări. De asemenea, am definit trei contoare: incercari_pornire, starting_timer și cicli_motor_pornit. Primul contor numara încercările de pornire, în cazul în care prima sau mai multe porniri succesive esuează. Acest contor este incrementat după fiecare pornire esuată și comparat cu o constantă, NR_MAX_INCERCARI_PORNIRE (având valoarea 4), astfel ca la atingerea valorii maxime, demarajul va fi abandonat. Al doilea contor, având un nume destul de sugestiv în limba engleză, contorizează timpul petrecut de-a lungul unei încercări de demaraj. Acest contor este augmentat cu 200 (milisecunde) după citirea periodică a turatiei, și comparat cu constanta LIMITA_DURATA_PORNIRE (ce are valoarea 4000), pentru a decide că respectiva încercare de demaraj să fie abandonată după 4 secunde de acțiune a demaratorului, urmând că după un timp de „relaxare” de 5 secunde, să fie lansată urmatoarea încercare, în limita numărului maxim de 4 încercări. Funcția de citire a turatiei este identică din punct de vedere procedural cu cea de la SM, cu diferența că de data aceasta timpul de citire este de 200 de milisecunde în loc de o secundă. Acest timp a fost scurtat pentru a reacționa mai rapid la schimbările de turatie și, astfel, pentru a opri suficient de repede demaratorul atunci când turatarea motorului depășește 600 rpm, turatie ce indică motor autonom. Așa se explică durata de 200ms care se adaugă contorului mai devreme menționat. Ultimul dintre aceste contoare, cicli_motor_pornit, reprezintă tot multiplii de 200ms, fiind incrementat în cadrul stării „motor pornit” după fiecare citire a turatiei. El este resetat înainte de intrarea în această stare. Este prevăzut un test ciclic care verifică simultan dacă acest contor este mai mare sau egal cu 10 (deci au trecut 2 secunde) și presiunea de ulei nu are valoarea normală (circuit deschis pe intrarea

respectiva). In caz afirmativ, se trece la ultima stare, „solicita oprire motor”, si se genereaza valoarea 3 pe semnalul ENG_STATUS. In acest moment, modulul de pornire (MP) va sti ca trebuie sa opreasca motorul, iar dupa ce o va face, va trimite MCS-ului drept confirmare un scurt semnal START_REQ (100-200ms) pentru a-l scoate din aceasta stare. Starea de motor pornit va trebui sa cunoasca si o alta cale de iesire, aceea de motor oprit in mod neasteptat (determinat prin citirea turatiei), ce ar corespunde realitatii cu situatia de motor „omorat” (prin neadaptarea cuplului sau prin pana de carburant). Înținând cont de aceste detalii, se redactează un program ca cel Anexa [16].

Am ales pentru acest modul un semnal de ceas de 16MHz, cu oscilator extern cu cristal de cuarț, setând CKSEL 3...1 cu valorile „1111”. Fiind vorba despre acționarea demarorului și luarea unor decizii critice în timp real, ne dorim ca acesta să fie modulul cel mai rapid. Justificăm aici și folosirea celor doi condensatori de 15pF conectați în paralel între fiecare pin al oscilatorului și masă. În manualul tehnic al microcontrolerului ATMEGA8 găsim indicațiile care sugerează adăugarea acestor doi condensatori la folosirea oscilatorului extern.

Pentru testare, conectăm întâi o sarcină în serie cu cele două relee și alimentarea. Mutăm cele două intrerupătoare în poziția „închis”, conectăm senzorul numărul 2 la intrarea în circuitul de condiționare și pregătim un fir legat pe intrarea lui START_REQ. Conectăm multimetrul în configurație de voltmetru la cursorul rezistorului variabil. Acest cursor este legat direct la pinul PD5 deci va trebui să facem un reglaj pentru a obține o tensiune maximă de aproximativ 4...4,4 volți în acest punct. Alimentăm modulul MCS și învărtim ușor volanta pentru a obține nivelul superior al semnalului de turătie prelucrat, apoi facem reglajul menționat mai devreme. Remarcăm că nu a fost nevoie să pornim SM-ul fiindcă senzorul numărul 2 este total independent de acesta. Acum putem testa funcționarea propriu-zisă. Pornim SM-ul și ne asigurăm că acesta este în stare „motor oprit”, intrerupătorul de bypass SW1 deschis, iar cel de separare a motorului de releu, SW2, închis. Firul conectat la START_REQ îl atingem de masă (șurubul de fixare al regulatorului, de exemplu) și îl menținem așa. De îndată ce observăm că sarcina conectată la relee a pornit, învărtim rapid volanta pentru a „porni motorul” și observăm dacă sarcina se decouplează odată cu atingerea unui regim de turătie autonom. Apoi, fără să punem motorul în funcțiune, încercăm să-l lăsăm să realizeze un ciclu complet de încercări de demaraj, până la abandon. Numărăm aceste cicluri și cronometrăm cu aproximativă durata acestora pentru a vedea dacă se conformează cu cele scrise în program. În plus, este util să conectăm temporar 2 LED-uri (cu câte o rezistență de 1kOhm) la semnalele ENG_STATUS_BIT_0 și ENG_STATUS_BIT_1, apoi să repetăm aceste experimente, concentrându-ne acum asupra indicației acestora. Este important ca semnalul ENG_STATUS să fie transmis corect în toate stările, inclusiv în stare 3 pe care o putem obține cu motorul pornit, deschizând intrerupătorul PRES_ULEI după trecerea a două secunde de motor autonom. Aceste aspecte fiind testate, considerăm finalizat și acest modul.

Modulul de pornire (MP)

Acest modul este componenta centrală a montajului, întrucât el decide momentele în care sunt pornite/oposite alimentările ACC și IGN prin comunicarea cu MDA, el citește butonul de pornire aflat în habitaclu și furnizează indicații luminoase către LED-urile ce însoțesc butonul. El comanda pornirea motorului către modulul MCS și tot el îl poate opri, solicitând decuplarea alimentării IGN. De asemenea, el dirijează și pornirea din telecomandă sau pornirea programată prin intermediul unui semnal provenit de la Modulul Telecomenzii (MT). Am ales să implementăm acest modul cu rol de dirijor de funcții întrucât, după cum am menționat și în introducere, dorim o divizare a

funcționalităților montajului, acestea fiind împărțite pe mai multe module mai mici, în locul realizării unuia mai complex.

Jucând rolul unui nod principal în rețea de module astfel concepută, modulul de față va avea cele mai multe semnale de intrare și de ieșire. Privind din nou schema bloc a montajului complet, observăm că MP se va conecta cu MDA, cu MCS, cu MT, cu modulul de citire a cipului cheii și cu ansamblul butonului de start (BS), care va conține butonul propriu-zis și cele două LED-uri. Recapitulând semnalele celorlalte module care se conectează cu MP-ul, întocmim lista de semnale de intrare și ieșire (de data aceasta în ordinea strictă a pinilor microcontrolerului).

Intrări:

- ALIM_ACK_BIT_0 și ALIM_ACK_BIT_1 pe pinii PD0, respectiv PD1. Aceste intrări alcătuiesc semnalul ALIM_ACK („Confirmare alimentare”), codat pe 2 biți, ce provine de la plăcuța cu relee, prin câte un tranzistor de inversare a logicii. Primul bit provine de la ieșirea releului de ACC, iar al doilea de la cea a releului de IGN. Împreună, reconstituie exact un număr cuprins între 0 și 3 ce reprezintă starea de alimentare. Avantajul de a prelua acest semnal de la ieșirea releeelor și nu de la semnalul de comandă furnizat acestora este faptul că în acest fel acoperim timpul de anclansare. Astfel, în scrierea programului pentru MP, atunci când vom anticipa anumite stări ale acestui semnal, vom ști că atunci când condițiile sunt îndeplinite, avem cu siguranță respectivele niveluri de alimentare prezente și în realitate.
- ENG_STATUS_BIT_0 și ENG_STATUS_BIT_1 pe pinii PD2 și PD3. Acestea alcătuiesc semnalul ENG_STATUS („Stare motor”) emis de modulul MCS și care a fost descris în secțiunea respectivă.
- START_BTN pe pinul PD4, se închide prin butonul de START la masă.
- AUTH_INJ_LED pe pinul PD5, este semnalul transmis de calculatorul de injecție și recepționat de tabloul de bord pentru a comanda LED-ul „antidemaraj”. Dacă autentificarea cipului din cheie este reușită, acest semnal provoacă aprinderea acestui LED în mod continuu timp de 3 secunde, apoi stingerea acestuia. Dacă autentificarea este eșuată, atunci semnalul va determina clipirea LED-ului cu o frecvență de 2Hz.

MARTORUL ANTIDEMARAJ

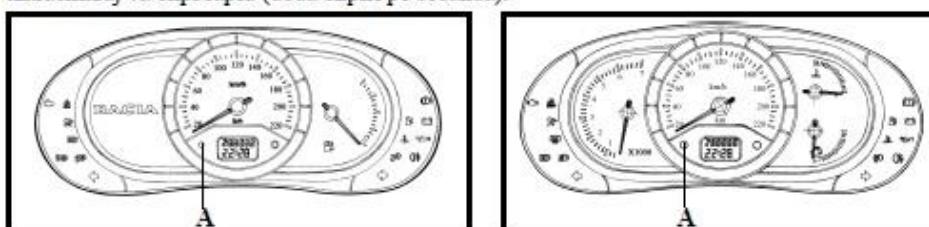
Este situat pe tabloul de bord (poziția A) și are funcțiile următoare:

- semnalează activarea sistemului antidemaraj;
- semnalează cazul de nerecunoaștere a capului cheii;
- semnalează intrarea în modul de lucru “resincronizare” a telecomenzilor, RF cu UCE decodor, pentru automobilele ce folosesc telecomandă RF.

Atunci când contactul este luat și sistemul antidemaraj este activ, fără a exista nici un defect, martorul antidemaraj trebuie să clipească rar (o clipire pe secundă).

După punerea contactului martorul antidemaraj trebuie să se aprindă timp de 3 secunde și apoi să se stingă.

Dacă după punerea contactului sistemul antidemaraj nu funcționează corect, atunci martorul antidemaraj va clipi rapid (două clipiri pe secundă).



Sursa: [Manual de reparație pentru echipamentele electrice Dacia Solenza]

Acest semnal va trebui luat în calcul în sensul de a nu comanda pornirea dacă autentificarea este eșuată.

- REMOTE_IN_BIT_0 și REMOTE_IN_BIT_1, pe pinii PD6, respectiv PD7. Acestea alcătuiesc semnalul REMOTE_IN, codat pe 2 biți, ce provine de la MT și poate avea următoarele valori: „mașină încuiată” (0), „mașină descuiată și autentificare reușită” (1), „pornește motorul” (2), „oprește motorul” (2). Ultimele două valori corespund pornirii/opririi din telecomandă sau pornirii programate.

Ieșiri:

- ALIM_REQ_BIT_0 și ALIM_REQ_BIT_1, pe pinii PB0 și PB1. Alcătuiesc semnalul de solicitare alimentare trimis către MDA.
- START_BTN_LED_RED și START_BTN_LED_GREEN, pe pinii PB2 și PB3. Furnizează informație vizuală utilizatorului privind starea motorului, prin LED-urile roșu și verde aflate în vecinătatea butonului de START.
- STARTING_REQ, pe pinul PB4. Acesta este semnalul de solicitare a demarajului trimis către MCS.
- SEP_CHEIE (Separator cheie), pe pinul PB5. Acest semnal acționează un circuit de întrerupere a alimentării modulului de citire a cheii, pentru a închide circuitul și a face posibilă autentificarea în momentul descuierii mașinii, respectiv pentru a o face imposibilă în lipsa unei descuieri realizate cu telecomandă.
- ACTIVITY, pe pinul PB6. Acest semnal este activ atâtă timp cât motorul se află într-una din următoarele stări: în curs de pornire sau pornit. Fiind furnizat modulului de telecomandă (MT), acesta din urmă nu va da curs cererii de încuiere din telecomandă atâtă timp cât MP raportează în continuare activitate.

În ceea ce privește circuitele auxiliare necesare adaptării logicii semnalelor de intrare, menționăm că următorii pini: PD2, PD3, PD6 și PD7 sunt prevăzuți cu tranzistor de inversare a logicii și rezistențe de pull-up, restul având doar rezistențe de pull-up. De asemenea, pinii PB4 și PB5 sunt cu tranzistori de inversare, deci semnalele de ieșire din acești pini vor fi cu închidere la masă, restul fiind ieșiri directe. Folosind aceste detalii și pe cele generale, valabile pentru toate modulele, vom realiza schema electrică a MP-ului, găsită la Anexa [17], apoi vom desena cablajul, pe care îl găsim în Anexa [18]. Am folosit de această dată două mufe DB-9, întrucât avem nevoie de 17 pini, cu tot cu alimentare. Semnificația pinilor celor două mufe o găsim în Anexa [19]. Ansamblul rezultat poate fi observat mai jos.

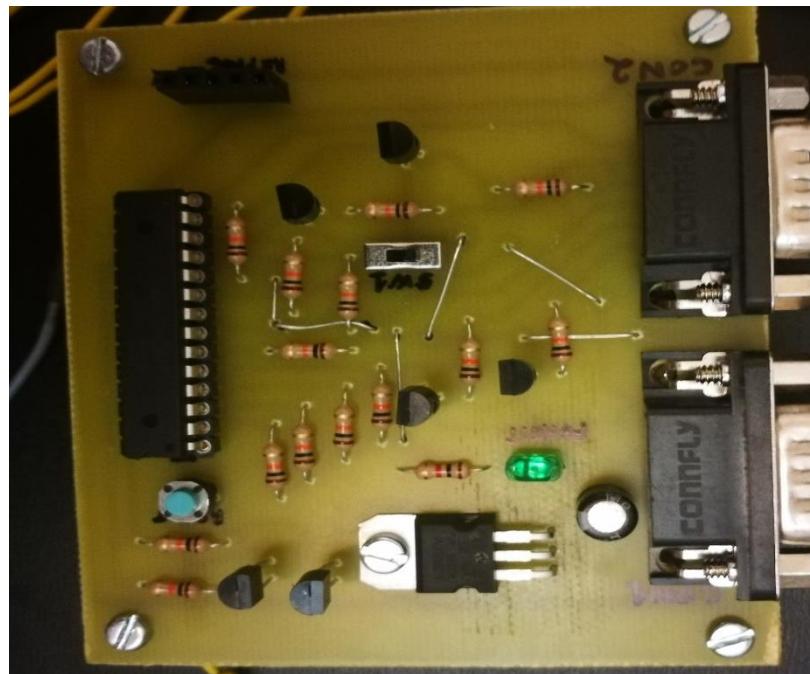


Fig. 3.19 - Modulul de pornire

Vom concepe în continuare o schemă bloc pentru acest modul, gândindu-ne la o serie de stări care să poată modela toate funcționalitățile pe care dorim să le implementăm. O variantă satisfacatoare ar include următoarele stări: CAR_LOCKED (0), STANDBY (1), IGNITION_ONLY (2), ENGINE_STARTING_BY_MANUAL (3), ENGINE_RUNNING_BY_MANUAL (4), ENGINE_STARTING_BY_REMOTE (5) și ENGINE_RUNNING_BY_REMOTE (6). Tranzitiiile între aceste stări vor fi dictate, în principiu, de stările butonului de START, de stările lui ENGINE_STATUS și de stările lui REMOTE_IN, fiind condiționate totodată și de ALIM_ACK, pentru care în anumite părți din program vom insera bucle de așteptare a unui valorii înainte de a avansa. În Anexa [20] vom prezenta o posibilă schemă logică pentru MP, pe baza căreia începem să implementăm programul.

Folosim timer-ul INT0 (pe 8 biți) conectat cu un prescaler de 1:1024 la ceasul intern, și punând în rutina de servire a întreruperilor [TIM0_OVF] o instrucțiune de tipul `i++`, unde „*i*” este o variabilă de tip întreg, obținem 30 de incrementări a variabilei „*i*” în decursul unei secunde. Explicația stă în calculul urmator: $8\text{MHz} : 1024 : 256 = 30.51\text{ Hz}$. Întreruperea TIM0_OVF survine la fiecare depășire (și revenire la valoarea 0) a timer-ului TIM0 ce are 256 de valori posibile. La rândul lui, timer-ul TIM0 se incrementează la fiecare 1024 de perioade ale semnalului de ceas. Astfel, depășirea timer-ului, și deci incrementarea lui „*i*” are loc la fiecare $1024 * 256$ semnale de ceas, deci cu o frecvență de aproximativ 30 Hz la un ceas de 8MHz. Având grija să resetăm valoarea lui „*i*” la momente cheie, ne putem folosi de valoarea acestuia pentru a cronometra diferite evenimente. De exemplu, la apasarea butonului START, ne interesează să cronometrăm cât timp este apăsat. Acest lucru ne ajuta să-i atribuim două funcții diferite în funcție de timpul de apăsare, astfel: pentru o apăsare mai scurtă de 4 secunde, se ia în considerare funcția de pornire motor, iar pentru o durată mai mare, se va face trecerea în starea IGNITION_ONLY (2), care ne asigură pornirea alimentării IGN. Această stare ne este utilă în eventualitatea necesității de a porni mașina prin împingere, dacă bateria se consumă accidental sub pragul energetic necesar acțiunării demarorului. În momentul apăsării butonului, variabila „*i*” este resetată pentru a cronometra apăsarea butonului. În momentul trecerii în starea menționată mai devreme, această variabilă va fi din nou resetată, pentru a constitui un alt contor de timp, necesar

cronometrării unei dure de grație de 3 minute, la expirarea căreia, se va face automat tranziția în starea STANDBY. De asemenea, este posibilă revenirea înaintea celor 3 minute printr-o nouă apăsare lungă a butonului, iar în cazul unei apăsări scurte, se poate lansa și din această stare comanda de demaraj. O altă funcționalitate asociată unei apăsări scurte a butonului este aceea de abandonare a procedurii de demaraj, atunci când suntem în starea ENGINE_STARTING_BY_MANUAL (3), dacă această procedură durează prea mult și ne hotărâm să renunțăm înaintea parcurgerii celor 4 încercări de pornire, sau chiar înaintea scurgerii celor 4 secunde ale primei încercări. Starea de pornire este CAR_LOCKED (0), din care putem ajunge doar în alte două stări, în funcție de semnalul REMOTE_IN, respectiv în STANDBY pentru valoarea 1, sau în ENGINE_STARTING_BY_REMOTE (5) pentru valoarea 2. Din stările 5 sau 6 se poate reveni în starea 0 prin aplicarea valorii 2 semnalului REMOTE_IN. Din starea STANDBY se poate ajunge în starea 0 prin aplicarea valorii 0 semnalului REMOTE_IN. În cazul opririi neașteptate a motorului, întrucât MCS va înștiința MP asupra stării motorului, din starea 4 se va reveni în STANDBY, pe când din starea 6 se va reveni în 0, întrucât aceasta este starea de siguranță, cu cititorul cipului din cheie întrerupt. Având aceste premise la dispoziție, realizăm programul pe care îl regăsim la Anexa [21]

După cum s-a putut observa, când discutăm despre timerul TIM0 am considerat frecvența de 8MHz pentru semnalul de ceas. Într-adevăr, am ales pentru acest modul folosirea oscilatorului intern de 8MHz, setând fuse-biții CKSEL3...1 cu valorile „0100”.

Pentru testarea modulului, este necesar să pregătim plăcuța separată pentru butonul de start (BS), pe care să montăm butonul propriu-zis, cele două LED-uri (roșu și verde), rezistențele de 1kOhm pentru acestea, precum și un condensator de 100nF care va fi montat în paralel cu butonul pentru a realiza funcția de „debounce”, informații care sunt disponibile în Capitolul I. După ce finalizăm această placă, o vom conecta la pinii corespunzători din mufe DB-9 și vom mai pregăti încă două întrerupătoare care să simuleze semnalul REMOTE_IN primit de la MT. Acestea vor fi botezate scurt, „REM0” și „REM1” pentru a le putea eticheta ușor, semnificațiile fiind subînțelese. Conectăm și aceste întrerupătoare la pinii lor corespunzători. O testare completă a acestui modul (în mod independent) este totuși foarte anevoieasă, fiind nevoie să montăm mai multe LED-uri auxiliare (cu rezistențele aferente) la ieșirile ALIM_REQ și STARTING_REQ și să pregătim o serie de întrerupătoare sau butoane pentru a simula răspunsurile așteptate de la celelalte module. Mai mult decât atât, ar trebui în același timp să urmărim activitatea LED-urilor, dar și să acționăm butoanele cu o repeziciune suficient de mare, astfel încât să nu depășim eventualele timeout-uri. Ajungem în acest moment la concluzia că ar fi cazul să trecem la etapa de integrare, adică la interconectarea tuturor modulelor și testarea întregului ansamblu. Întrucât nu mai avem nimic de realizat practic (doar de descris teoretic ultimele două module ramase), și cum interconectarea modulelor era oricum următorul pas de realizat, vom regăsi avantajele de a include testarea ultimului modul în cadrul testării de integrare (testare intersistem).

3.4 Interconectarea modulelor și realizarea testelor de integrare. Rezultate

„Testele de integrare examinează mai multe trăsături și componente ale unui software individual cu scopul de a determina dacă o trăsătură ori o componentă nouă se potrivește bine cu (sau se integreaza în) trăsăturile și componentele deja existente.

Testele de integrare sunt folosite pentru a dovedi că toate trăsăturile și componentele software funcționează bine împreună.” [Testele de integrare ISO 31000, 2009] Acest lucru este valabil și pentru testarea hardware.

În ideea prezentării lucrării într-o formă completă și estetică, vom lua o scurtă pauză de la programul de electronist și vom pregăti o planșă mai mare care să cuprindă tot montajul (inclusiv simulatorul de motor), vom realiza distribuirea în plan a modulelor și le vom monta în șuruburi, sau măcar le vom trasa pozițiile cu un creion în vederea montării ulterioare. Reunim toate tabelele de descriere a semnificației pinilor precum și schema bloc a întregului ansamblu și, cu modulele măcar așezate pe pozițiile stabilite, așezăm și tăiem la dimensiunea corespunzătoare bucăți de fire de la un modul la altul până la parcurgerea completă a tuturor conexiunilor ce trebuie realizate. Aceste fire vor fi apoi lipite între pinii respectivi ai fiecărei mufe, respectând tabelele. Ar fi util să realizăm un tabel de interconexiuni pentru tot ansamblul, pe care să-l putem urmări mai ușor. Acesta se va regăsi sub Anexa [22]. După realizarea acestor interconexiuni, pregătim un alt motor electric de mici dimensiuni care să fie folosit ca sarcină la ieșirea releelor din MCS, simbolizând chiar demarorul. Mai includem și un comutator cu două poziții, prin care să putem face selecția alimentării simulatorului motorului între două posibilități: direct de la alimentarea permanentă, sau prin releul de IGN. Prima varianta ne va fi utilă când vom prezenta separat simulatorul, iar a doua se încadrează în scenariul real de funcționare, în care calculatorul de injecție este alimentat de la IGN.

Înaintea alimentării ansamblului, mai verificăm încă o dată măcar conexiunea alimentărilor din mufe, care are aceeași poziție la toate modulele, respectiv pinul 5 pentru alimentarea cu 12V și pinul 9 pentru masă. De asemenea, butoanele REM0 și REM1 vor trebui setate ambele pe pozițiile „deschis” (pentru a genera un semnal echivalent cu valoarea 0), iar selectorul alimentării simulatorului să fie comutat pe poziția „de la IGN”.

Alimentăm ansamblul, denumit în continuare planșă. Ar trebui ca toate modulele să fie în repaus, inclusiv simulatorul, iar ledurile de indicare a prezentei tensiunii să fie aprinse la toate modulele. Acționând butonul REM0 către poziția „închis”, deci obținând valoarea 1 pentru REMOTE_IN, ar trebui să observăm anclanșarea releului de ACC și aprinderea LED-ului roșu de lângă butonul de START. La apăsarea lungă a acestuia, vom observa că va cupla și releul de IGN. La o nouă apăsare lungă, se va reveni la starea anteroară. Repetăm ultimul pas pentru a intra în starea IGN_ONLY și așteptăm trei minute pentru a constata că va reveni singur în starea STANDBY. Acum, printr-o apăsare scurtă, vom observa din nou anclanșarea releului de IGN și, la scurt timp după aceasta, va porni demarorul și se va aprinde și LED-ul verde de lângă buton. În acest moment trebuie să fim pregătiți să avem reacții rapide. Imediat ce a pornit demarorul, învârtim repede volanta și așteptăm ca motorul să pornească, iar în scurt timp va trebui să observăm că demarorul se oprește. În acest moment LED-ul roșu al butonului trebuie să se stingă și să rămână aprins doar cel verde. La o apasare scurtă a butonului, motorul va fi oprit cu promptitudine, prin decuplarea alimentării IGN, iar iluminarea butonului va comuta imediat din verde în roșu. Facem din nou o acționare lungă a butonului pentru a intra în modul IGN_ONLY, apoi învârtim direct volanta, observând că după pornirea motorului, LED-urile butonului vor comuta din nou direct de la roșu la verde. Acum, cu motorul pornit cu cel puțin 2 secunde în urmă, deschidem întreupătorul PRES_ULEI și observăm din nou cum motorul este oprit cu promptitudine. Nu uităm să reducem întreupătorul în poziția „închis”. Revenim în starea IGN_ONLY și apoi facem o acționare scurtă a butonului, pentru a verifica faptul că motorul poate fi pornit prin buton și din această stare. Cu motorul pornit, facem o oprire neașteptată, fie prințând cu degetele axul volantei și strângându-l pentru a-1 frâna, fie acționând întreupătorul SW 2 al simulatorului în poziția „deschis” (către stânga). Observăm cum modulele MCS și MP sesizează prompt acest eveniment, releul IGN se decuplează, iar lumina butonului va comuta din nou de la cea verde la cea roșie. Reducem întreupătorul SW 2 în poziția inițială. Acum, din starea STANDBY, este momentul să observăm un ciclu complet de demaraje eşuate, apăsând

butonul START și neînvârtind volanta în acest timp. Vom observa că după o serie de 4 acționări ale demarorului a câte 4 secunde, cu pauze de 5 secunde între ele, se va reveni la starea inițială. Repetând experimentul, decidem să învârtim volanta de-a lungul unei încercări de demaraj, oricare din cele 4, la alegere, și vom observa că pornirea motorului va fi luată în calcul de MCS și MP. De asemenea, tot din starea STANDBY, deschidem întrerupătorul NEUTRAL de pe modulul MCS și apăsăm butonul START pentru a încerca o nouă pornire. Releul de IGN va fi cuplat, însă demarorul nu va acționa deloc, iar în scurt timp se va reveni la starea anterioară.

Din starea STANDBY, deschidem întrerupătorul REM0 pentru a obține valoarea REMOTE_IN = 0 și observăm cum releul de ACC este decuplat, iar LED-ul roșu de lângă buton se stinge. Acum închidem întrerupătorul REM1, pentru a obține valoarea 2, și observăm cum se lansează procedura de demaraj automat. Vom verifica și aici o parte din aspectele parcuse la pornirea prin butonul START: demarajul ratat din cauza nepornirii motorului, abandonul demarajului prin acționarea ambelor întrerupătoare REM în poziția închis (pentru a obține valoarea 3), oprirea motorului prin aceeași acționare, eșecul de demaraj datorat întrerupătorului NEUTRAL mutat în poziția „deschis”, oprirea anticipată a motorului datorată mutării întrerupătorului PRES_ULEI în poziția „deschis”, revenirea automată la starea CAR_LOCKED în cazul opririi neașteptate a motorului.

În cazul reușitei tuturor experimentelor expuse mai devreme, se consideră încheiată partea de construcție a modulelor proiectului și revenim la descrierea teoretică a celorlalte două module care au mai rămas de abordat. Mai jos puteți observa niște imagini cu rezultatele montajului finalizat.

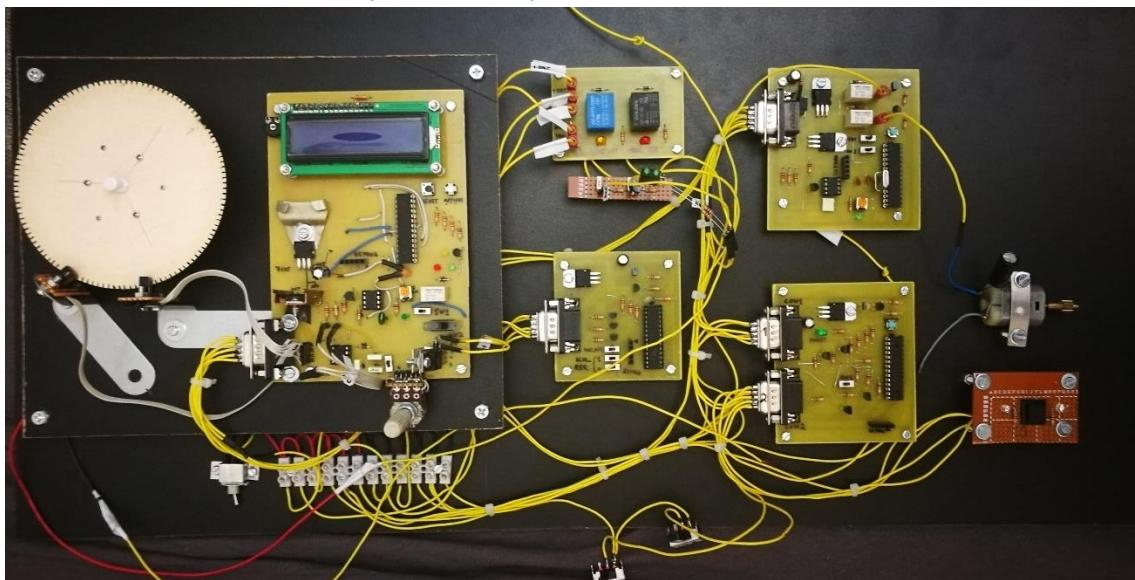


Fig. 3.20 – Montaj finalizat

3.5. Descrierea teoretică a celorlale două module

Modulul de telecomanda (MT)

Acest modul trebuie construit în mod personalizat, în funcție de tipul de închidere centralizată aflat pe mașina și în funcție de tipul sistemului de recepție al telecomenției de care aceasta dispune.

Un sistem de recepție destul de clasic este prevăzut cu un număr de relee egal cu numărul butoanelor de pe telecomandă, iar funcționarea acestui sistem se limitează la recepția și decriptarea

semnalului radio emis de telecomandă și acționarea pentru o durată fixă (uzual, o secundă) a releeului corespunzător butonului care a fost apăsat.

La unele sisteme, toate releele sunt montate cu unul dintre contacte la potențialul Ubat (12V), iar la altele cu unul dintre contacte la masă. În momentul achiziționării unui astfel de sistem, acest lucru trebuie verificat și comparat cu tipul de semnal așteptat de modulul de închidere centralizată a mașinii de la butonul de închidere/deschidere din habitaclu. Și aici situația poate fi de două feluri, fie butonul este legat permanent la potențialul Ubat și modulul așteaptă un front pozitiv pe intrările de închidere/deschidere, fie butonul este legat la masă, iar modulul așteaptă închiderea la masă a intrării respective, pentru a executa o comandă de închidere sau deschidere către actuatorii ușilor. Dacă polaritățile semnalelor celor două sisteme se potrivesc, se poate face legătura directă între ieșirile releelor de închidere și deschidere și intrările modulului de închidere centralizată corespunzătoare acestor acțiuni.

În construcția modulului nostru (MT) trebuie să preluăm de la sistemul de recepție a telecomenției informațiile despre butoanele ce sunt apăsate, citind tensiunea de la ieșirea releelor. Modulul trebuie de asemenea, să citească semnalul ACTIVITY primit de la MP, pentru a ști dacă este cazul sau nu să autorizeze închiderea mașinii. Sistemul de recepție a telecomenției nu va mai fi legat direct la modulul de închidere centralizată. În schimb, semnalele de închidere și deschidere vor fi emise de MT. Astfel, MT va permite funcționarea lui MP în momentul receptiei semnalului de descuiere a mașinii și îl va bloca la receptia unui semnal de încuiere, și doar în condiția în care semnalul ACTIVITY este nul. Comunicarea cu MP se va face, desigur, prin semnalul REMOTE_IN, furnizând valoarea 1 la descuiere și valoarea 0 la încuiere.

Mai departe, pentru a implementa funcția de pornire din telecomandă, trebuie să construim o funcție capabilă să compare ordinea apasării butoanelor de pe telecomandă cu o ordine prestabilită (de exemplu, „închidere-închidere-deschidere-închidere”) și să cronometreze durata dintre aceste apăsări, astfel încât ea să nu fie mai lungă de 3 secunde, de exemplu. Astfel, atunci când MT este în starea „încuiat”, de fiecare dată când receptionează un semnal de încuiere, el va lansa procedura de comparare. Dacă după prima apasare a butonului „încuiere” trece un timp mai lung de 2 secunde fară să se detecteze o nouă apăsare, se consideră timpul expirat, iar apăsarea ca fiind făcută accidental. Dacă, în schimb, după prima apăsare, în cursul de 2 secunde se detectează o nouă apăsare pe „încuiat”, funcția de comparare avansează în vectorul de valori prestabilite, resetează contorul de timp și așteaptă apăsarea în timp util a următorului buton corect. De fiecare dată când timpul expiră sau a fost detectată apăsarea unui buton greșit (conform următoarei valori așteptate), procedura este abandonată și reluată la o nouă apăsare pe „încuiere”. Astfel ne putem stabili un „cod” din apăsări de butoane, pentru a lansa în siguranță procedura de pornire din telecomandă. În cazul codului exemplificat anterior, procedura ar trebui să dureze 1s (timp acționare releu)+1s(pauza)+... = 7 secunde, unde primul timp de 1s reprezintă timpul de acționare a releeului din sistemul de recepție a telecomenției, următorul timp de 1s să fie pauză dintre finalul acționării acestui releu și apăsarea următorului buton (deci pauza de 2 secunde între două apăsări successive), și aşa mai departe până la ultima apasare, dintr-un total de 4. Dacă, apoi, pe parcursul pornirii motorului sau pe parcursul funcționării acestuia, se detectează o apăsare a oricărui buton, modulul MT va trimite valoarea 3 prin semnalul REMOTE_IN și astfel va comanda oprirea motorului (sau încetarea demarajului), apoi va trimite automat valoarea 0, astfel că mașina va rămâne în starea încuiată și securizată. Procedura de pornire din telecomandă va putea fi lansată doar din această stare. Dacă mașina este descuiată, va trebui mai întâi încuiată înainte de a lansa procedura.

În cazul pornirii programate, care va fi gestionată de un alt modul (MPP – Modul de pornire programată), MT va face totuși puncte între acest modul separat și MP, întrucât lansarea procedurii de pornire a motorului se face tot prin semnalul REMOTE_IN. În plus, nu dorim ca MPP să aibă un acces necontrolat asupra motorului, de exemplu, să solicite pornirea atâtă timp cât motorul este deja pornit, fie prin butonul de START fie din telecomandă, sau cât timp mașina este în starea descuiată, deci utilizatorul este prezent în mod întamplător în apropierea unei ore la care era programată o pornire și, aşadar, nu mai este nevoie de aceasta pornire. Astfel, MT va mai avea o intrare pentru un semnal pe un singur bit, TIMER_IN, la recepția căruia va verifica dacă sunt întrunite condițiile pentru o pornire programată și apoi va lansa procedura. La recepția unui nou front pozitiv al semnalului TIMER_IN, MT va comanda oprirea motorului și revenirea la starea de securizare, astfel că gestiunea duratei de menținere a motorului pornit prin această metodă revine în sarcina MPP-ului.

Modulul de pornire programata (MPP)

Acesta este un modul care prezintă și o interfață grafică cu utilizatorul, folosind un afișor alfanumeric identic cu cel de la simulator și câteva butoane având funcționalități de tipul „înainte”, „înapoi”, „OK”, „anulează” etc. Modulul presupune folosirea unui ceas de exactitate ridicată, pentru menținerea în permanență a orei și datei curente. El pune la dispoziție utilizatorului o serie de opțiuni pentru programarea pornirilor, deci în principiu oferă o funcționalitate asemănătoare unui ceas cu alarmă, în cazul nostru alarma fiind pornirea motorului. Opțiunile vor permite setarea datei și orei exacte a unei porniri sau mai multora, dintr-un număr limitat de „alarme” disponibile. De asemenea, se va putea oferi și varianta „pornește zilnic începând cu ora X, de un numar de Y ori, cu pauză de Z ore între porniri”. Va trebui să mai existe și o opțiune prin care să se configureze durata de menținere a motorului pornit. Opțiunile vor fi structurate sub forma unor meniuri, ce pot fi parcuse pe pagini, derulând între aceste meniuri cu butoanele „înainte” și „înapoi”, accesându-le cu tasta „OK” și părăsindu-le cu tasta „anulează”. Structurarea acestor meniuri trebuie făcută astfel încât afișarea lor pe display să fie cât mai intuitivă și lizibilă.

Salvarea datelor introduse de utilizator se va face în memoria EEPROM a microcontrolerului ATMEGA8, despre care găsim informații în manualul acestuia.

Pentru implementarea funcției de ceas, se poate folosi circuitul integrat DS1307, produs de firma MAXIM, care realizează funcția de ceas și calendar într-o formă avansată. El oferă posibilitatea alimentării de la o baterie de 3V printr-un pin separat de alimentarea principală, astfel că va continua să opereze în mod izolat în cazul întreruperii alimentării și nu va pierde data și ora curentă. Acest lucru este extrem de util în cazul deconectării bateriei autoturismului sau în cazul unei pene de baterie. Circuitul DS1307 oferă o interfață de comunicație de tip I2C pentru conectarea la orice microcontroler care are aceasta facilitate, inclusiv ATMEGA8. Conectivitatea se va asigura prin pinii PC4 și PC5, care au funcțiile alternative de SDA (Serial Data) și SCK (Serial Clock).

Singura conexiune a modulului MPP cu celelalte module se face prin semnalul TIMER_IN către MT.

Concluzii

Observând rezultatele obținute la testarea de integrare, consideram ca am atins o rata de succes mai mult decât satisfăcătoare. Urmărind succesiunea tranzițiilor între stările de funcționare ale modulelor, prin prisma unui observator extern, ajungem la concluzia că timpii de reacție ai montajului par mult mai scurți decât ne aşteptam. Ne dam seama că un interval de 500 de milisecunde trece aproape neobservat în contextul în care concentrăm privirea asupra funcționării întregului ansamblu, în timp ce în faza de proiectare ne gândeam de mai multe ori dacă un astfel de interval nu este cumva prea lung, dacă nu încetinește prea mult o anumita procedură, dacă nu există cumva șansa că acesta să conducă la expirarea timpului unei bucle de aşteptare din alt modul etc. Rezultatele noastre, aşadar, sunt foarte apropiate de aşteptările pe care le-am avut în etapa definirii funcționalităților montajului.

In spatele acestor rezultate, însă, s-au aflat câteva momente dificile. Prima astfel de situație a apărut după etapa asamblării simulatorului de motor, atunci când a trebuit să testăm funcționalitatea pe larg. Programul final a fost deja creat și verificat, funcționalitățile parțiale (măsurare frecvență, numărare rotații - din programele de test) au fost testate cu succes înaintea asamblării, când foloseam placă rapidă de test. Si totuși, după asamblare, aveam următoarea problema: MC-ul începea să citească frecvență în mod eronat, afișând chiar și cu motorul oprit turății de ordinul miilor de rpm, la momente aleatorii și destul de frecvențe. După o reverificare temeinică a programului, am constatat că nu aveam nicio eroare în acesta, să că următorul gând a fost faptul că este posibil să avem interferențe cauzate de zgombotul de comutație apărut în apropierea regulatorului PWM, pe linia care leagă ieșirea din regulator și tranzistorul de comanda Q3. Aceasta linie se află totodată destul de aproape de linia care leagă ieșirea din circuitul de condiționare cu pinul T1 al MCU-ului. Am constatat atunci că, în design-ul cablajului am omis niște detalii importante legate de compatibilitatea electromagnetică. Căutând pe Internet apariții ale unor astfel de probleme, am găsit mai multe documente de recomandări în privința design-ului circuitelor electrice, redactate și oferite publicului de către mai mulți producători de microcontrolere și circuite digitale, printre care chiar și Atmel. Era însă prea târziu pentru a modifica design-ul cablajului să ca m-am gândit la alte soluții. Urmărind în timp real, pe osciloscop, semnalul afectat de zgombot, am luat în calcul adăugarea unui condensator în paralel cu intrarea în T1 și masa și prin căruia valoare să se filtreze o parte din spectrul acestui zgombot. Prin tatonări, am găsit convenabilă valoarea de 1nF, iar acesta figurează și pe schema inclusă în lucrare, sub numele de C7. Semnalul a fost ameliorat și comportamentul de după aceasta modificare a fost monitorizat pentru un timp mai îndelungat, iar problema nu a mai apărut niciodată.

Un alt moment care m-a întrerupt din lucru a fost intenția inițială de a lega direct ieșirile de comanda ale MDA către relee (ieșiri active în 0), la intrările ALIM_ACK ale modulului MP. Am realizat apoi că este mai corectă luarea în considerare a stării de alimentare după nivelul tensiunii la ieșirea releeelor și nu după semnalul de comanda, dat fiind că există o mică întârziere de 5...20 de milisecunde între comanda releeului și închiderea completă a contactelor acestuia. Acest lucru a fost constatat în timpul montării modulelor pe planșa de prezentare, aşadar a fost încropit pe loc un mic modul cu 2 tranzistori de inversare a logicii, pe o bucata de placă de test, și legat la ieșirile releeelor, în imediata vecinătate a plăcii cu relee, iar intrările ALIM_ACK ale lui MP s-au legat la ieșirea din aceasta plăcută adițională.

Un ultim moment care nu a fost deloc surprinzător și nici dificil, dar pe care nu l-am luat în calcul inițial, a fost tot o problemă de interferență electromagnetică, provocată de motorul folosit

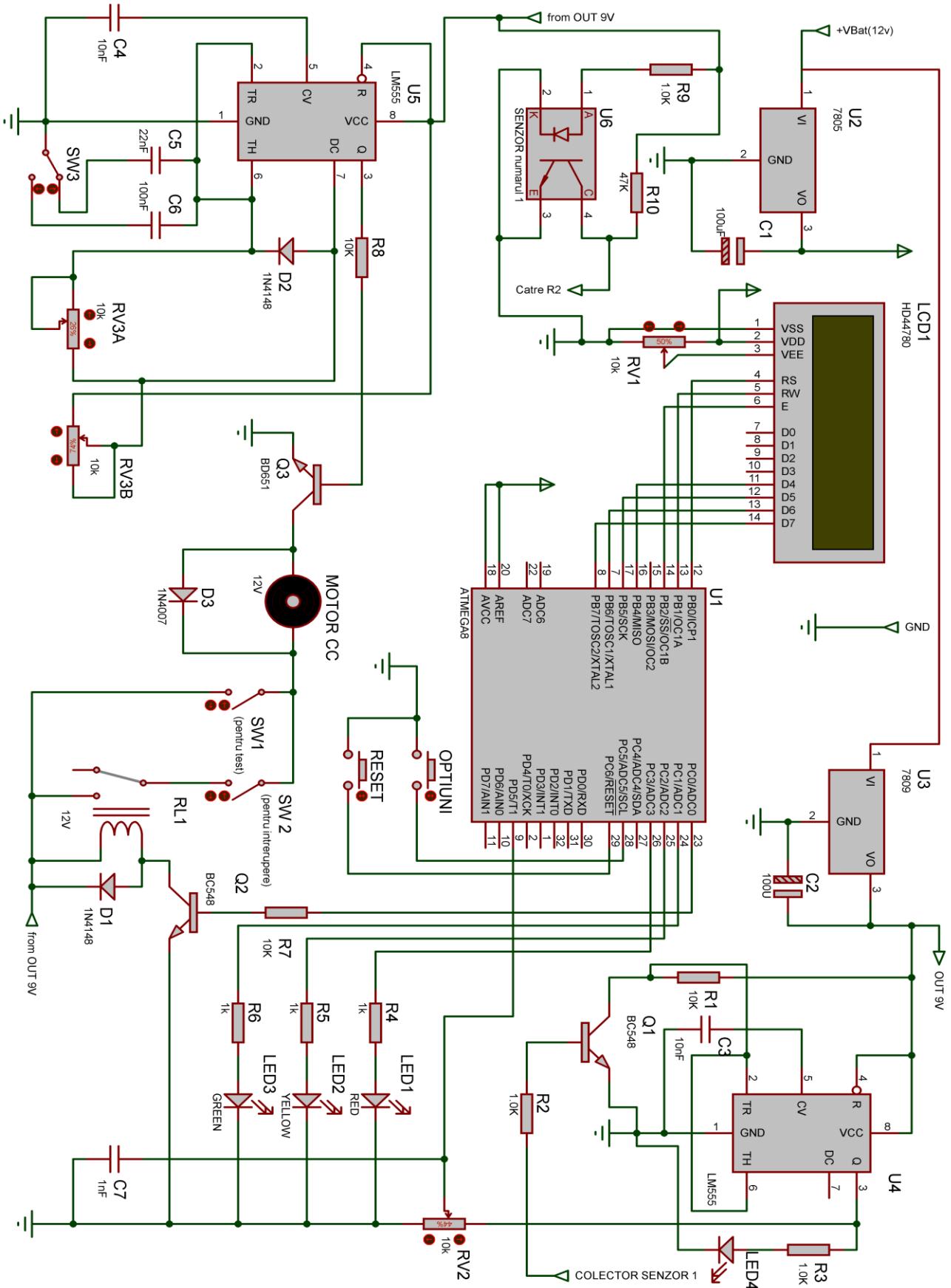
pentru a simula demarorul. Aceasta problema s-a rezolvat foarte prompt, prin adăugarea unui condensator de 100uF in paralel cu bornele motorului. Aceasta valoare ar trebui calculata conform unor formule specifice, dar soluția a fost una de moment.

Acum, după ce am abordat o parte din funcționalitățile microcontrolerului ATMEGA 8, realizam ca mai rămân multe altele de explorat, iar aplicațiile in care pot fi folosite sunt foarte numeroase. Revenind însă la felul in care am structurat montajul nostru, luam in calcul faptul ca o varianta mai evoluata ar fi fost folosirea unei linii de comunicație care sa permită conectarea mai multor module pe aceeași linie, cum ar fi: CAN, USB, LAN etc. In aplicațiile auto, comunicația pe CAN este un standard devenit deja consacrat, fiind introdusa pe scara larga in producția auto pe la începutul anilor 1990.

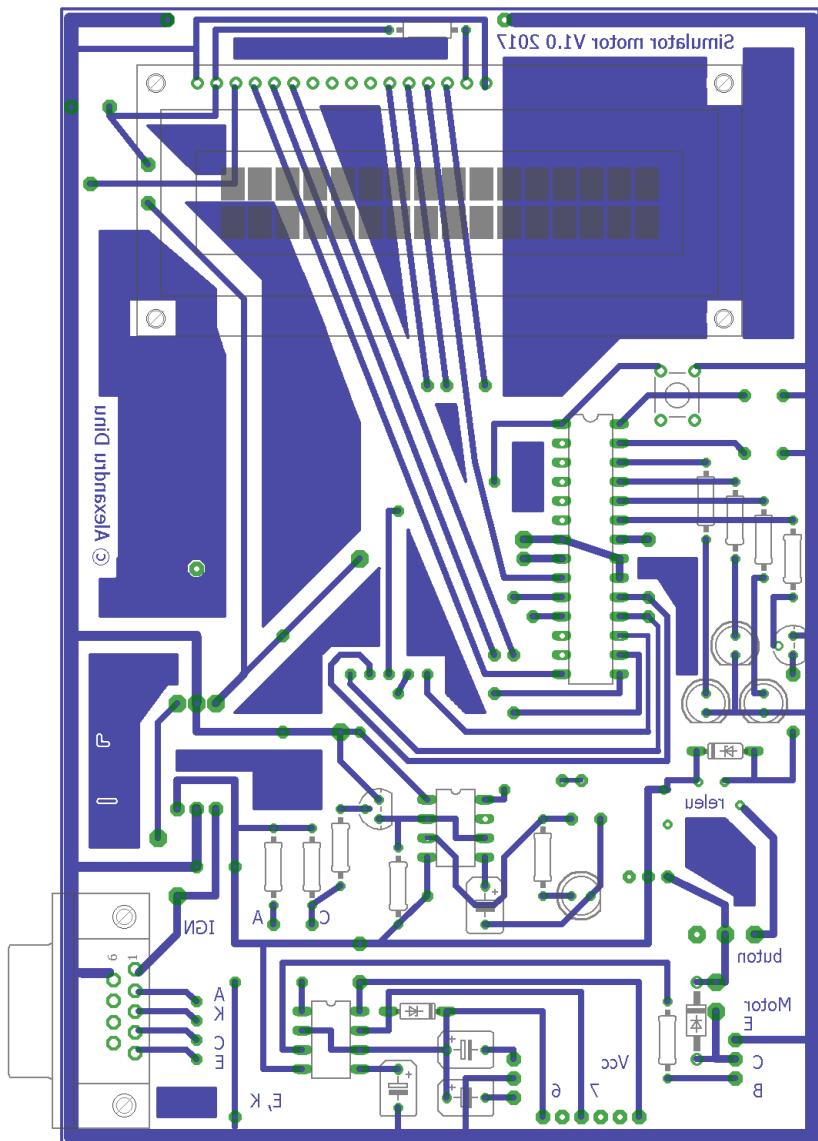
Un astfel de montaj se pretează cu succes la orice tip de mașină care are calculatoarele si modulele electronice conectate intre ele in sistem „filar”, adică folosind cate un fir pentru fiecare informație transmisa. Mașinile care au trecut la interconectarea calculatoarelor prin intermediul liniei CAN nu pot beneficia de un astfel de montaj, întrucât pe parcursul funcționării schimba numeroase informații care trebuie cunoscute si, in mod corespunzător, modificate de modulele montajului conform schimbărilor stărilor. Aceste informații sunt foarte greu de obținut întrucât, in mod normal, trebuie sa rămână secret de serviciu in interiorul firmelor producătoare. Chiar daca informații clasificate despre proiectarea autoturismelor ajung sa se scurgă pe Internet după mai mulți ani de lansarea pe piață, acest lucru nu se întâmplă întotdeauna, iar lucrurile s-au complicat prea mult pentru a mai putea fi „fentate” de orice entuziast al electronicii auto.

Anexe

Anexa [1]



Anexa [2]



Anexa [3] – Tabel Pini Simulator Motor

Pin	Semnificatie	Sensul
1	Emitor senzor 2	iesire
2	Colector senzor 2	iesire
3	Catod senzor 2	iesire
4	Anod senzor 2	iesire
5	+Vcc (12V)	-
6	-	
7	-	
8	-	
9	GND	-

Anexa [4]

```
#include <alcd.h>
#include <mega8.h>
#include <inttypes.h>
#include <delay.h>
#include <stdlib.h>
#include <initializers.h>
#include <functions.h>

// Alphanumeric LCD Module functions
#asm
    .equ __lcd_port=0x1B ;PORTB
#endifasm

#define STOPPED 0;
#define STARTING 1;
#define RUNNING 2;
#define RPM_ONLY 3;
#define COUNT_REVS 4;
#define COUNT_REVS2 5;
#define COUNT_DEGS 6;
#define COUNT_STRIPES 7;

int state,count_sec;

unsigned long int freq; /* to store value of frequency value */
unsigned int i=0,dur; /* i=number of overflows in one second */
/* dur to store the value of TCNT1 register */
char buffer[8];
/* array char to store the frequency value as a string to be displayed on
lcd ( More details ) */

// Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
i++ ; // count the number of overflows in one second
}

int meas_rpm(int state)
{
    TIMSK=0x04; // enable overflow interrupt of timer1
    TCCR1B=0x07; /* start timer1 with external pulses (T1 rising
edge) */
    delay_ms(1000); // wait for one second
    count_sec++;
    TCCR1B=0x00; //stop timer1
    TIMSK=0x00; //disable interrupt
    dur=TCNT1; /* store the number of counts in TCNT1 register */
    freq = 1*(dur + i*65536); /* calculate the frequency as in previous
equation */
    TCNT1=0x0000; /* clear TCNT1 register for the next reading */
    i=0; /* clear number of overflows in one second for the next
reading */
/////////////////////// display ///////////////////////////////
    lcd_clear();
    lcd_gotoxy(0,0);
```

```

switch(state)
{
    case 0: lcd_putsf("1. MOTOR OPRIT"); break;
    case 1: lcd_putsf("2. MOT IN PORNIRE"); break;
    case 2: lcd_putsf("3. MOTOR PORNIT"); break;
    case 3: lcd_putsf("4. AFIS. TURATIE"); break;
}
lcd_gotoxy(0,1);
lcd_putsf("          ");
lcd_gotoxy(0,1);
lcd_putsf("RPM= ");
ltoa(freq*60/119, buffer);
lcd_puts(buffer);
return freq*60/119;
}

void f_stopped()
{// STATE=0 - Engine Stopped State
    PORTC.1=0; PORTC.2=0; PORTC.3=1;
    if(meas_rpm(state)>50)
    {
        state=STARTING;
        count_sec=0;
    }
    if (PIN.C.5==0)
    {   while(PIN.C.5==0) { flashing_leds(); }
        state=RPM_ONLY;
    }
    return;
}

void f_starting()
{// STATE=1 - Engine Starting State
    PORTC.0=1;
    PORTC.1=0; PORTC.2=1; PORTC.3=0;
    if(meas_rpm(state)>600)
    {
        state=RUNNING;
    }
    if(count_sec>6)
    {
        state=STOPPED;
        PORTC.0=0;
        delay_ms(1000);
    }
    return;
}

void f_running()
{// STATE=2 - Engine Running State
    PORTC.1=1; PORTC.2=0; PORTC.3=0;
    if(meas_rpm(state)<350)
    { state=STOPPED;
        PORTC.0=0;
        delay_ms(1000);
    }
    if (PIN.C.5==0)
    {   while(PIN.C.5==0) { flashing_leds(); }
}

```

```

        state=RPM_ONLY;
    }
    return;
}

void f_show_rpm()
{// STATE=3 - showing current RPM
    meas_rpm(state);
    if (PINC.5==0)
    {   while(PINC.5==0) { flashing_leds(); }
        state++;
        TIMSK=0x04; // enable overflow interrupt of timer1
        TCCR1B=0x07; // start timer1 with external pulses (T1 rising
edge)
        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_putsf("5.NUMARA ROTATII");
    }
    return;
}

void f_count_revs()
{// STATE=4 - counting the number of revolutions
    lcd_gotoxy(0,1);
    lcd_putsf("          ");
    lcd_gotoxy(0,1);
    dur=TCNT1;
    freq=dur+i*65536; // use "freq" to store the total number of stripes
    // "i" stores the number of TCNT1 overflows
    itoa(freq/119,buffer);
    lcd_puts(buffer);
    delay_ms(100);
    if (PINC.5==0)
    {   while(PINC.5==0) { flashing_leds(); }
        state++;
        i=0;
        TCNT1=0x0000; /* clear TCNT1 register for the next reading */
        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_putsf("6. ROTATII+DINTI");
    }
}

void f_count_revs2()
{// STATE=5 - counting the number of revolutions + the number of stripes
    lcd_gotoxy(0,1);
    lcd_putsf("          ");
    lcd_gotoxy(0,1);
    dur=TCNT1;
    freq=dur+i*65536; // use "freq" to store the total number of stripes
    // "i" stores the number of TCNT1 overflows
    itoa(freq/119,buffer);
    lcd_puts(buffer);
    lcd_putsf(",dinti: ");
    itoa(freq%119,buffer);
    lcd_puts(buffer);
    delay_ms(100);
    if (PINC.5==0)
}

```

```

    {   while(PINC.5==0) { flashing_leds(); }
        state++;
        i=0;
        TCNT1=0x0000; /* clear TCNT1 register for the next reading */
        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_putsf("7. GRADE ROTATIE");
    }
}

void f_count_degs()
{// STATE=6 - counting the number of degrees of rotation (sum+current)
    lcd_gotoxy(0,1);
    lcd_putsf("          ");
    lcd_gotoxy(0,1);
    dur=TCNT1;
    freq=dur+i*65536; // use "freq" to store the total number of stripes
    // "i" stores the number of TCNT1 overflows
    itoa((freq%119)*3,buffer);
    lcd_puts(buffer);
    lcd_putchar(223);
    delay_ms(100);
    if (PINC.5==0)
    {   while(PINC.5==0) { flashing_leds(); }
        state=STOPPED;
        i=0;
        TCCR1B=0x00; //stop timer1
        TIMSK=0x00; //disable interrupt
        TCNT1=0x0000; /* clear TCNT1 register for the next reading */
    }
}
}

void main(void)
{
//Initialize timers, clock etc.
Initialize_components();

// LCD module initialization
lcd_init(16);

// PORTC - pins 0-4 are outputs, the rest are inputs
DDRC=0b00011111;

// Set Global Interrupt Flag
#asm("sei")

//Initialize starting point
state=STOPPED;
count_sec=0;

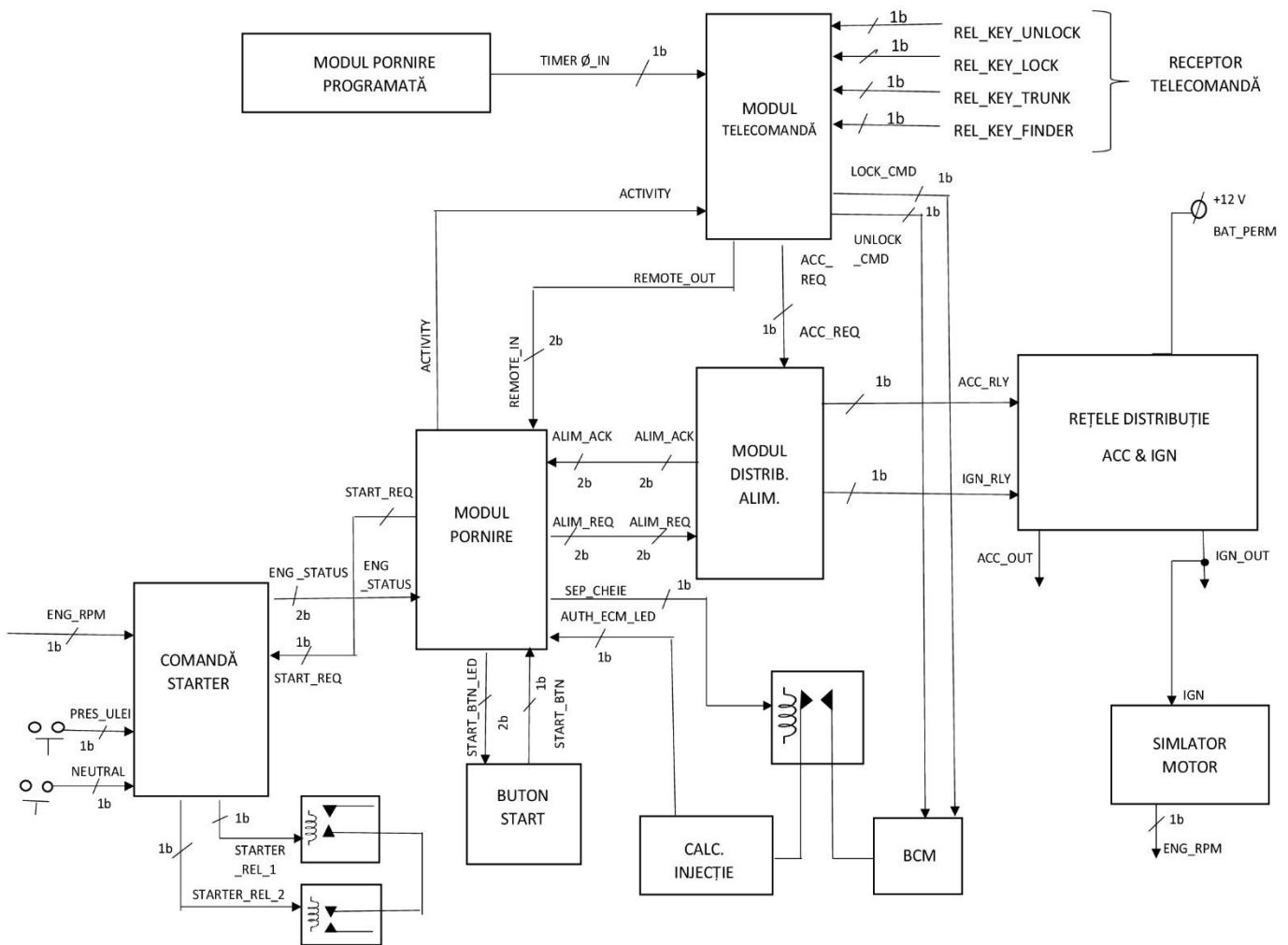
lcd_clear();

while(1)
{
    switch(state)
    {
        case 0:
            f_stopped(); break;

```

```
    case 1:  
        f_starting(); break;  
    case 2:  
        f_running(); break;  
    case 3:  
        f_show_rpm(); break;  
    case 4:  
        f_count_revs(); break;  
    case 5:  
        f_count_revs2(); break;  
    case 6:  
        f_count_degs(); break;  
    default:  
        lcd_clear();  
        lcd_gotoxy(0,0);  
        lcd_putsf("MASINA DE STARI");  
        lcd_gotoxy(0,1);  
        lcd_putsf("STARE ERONATA:");  
        itoa(state,buffer);  
        lcd_puts(buffer);  
        break;  
    }  
}  
}
```

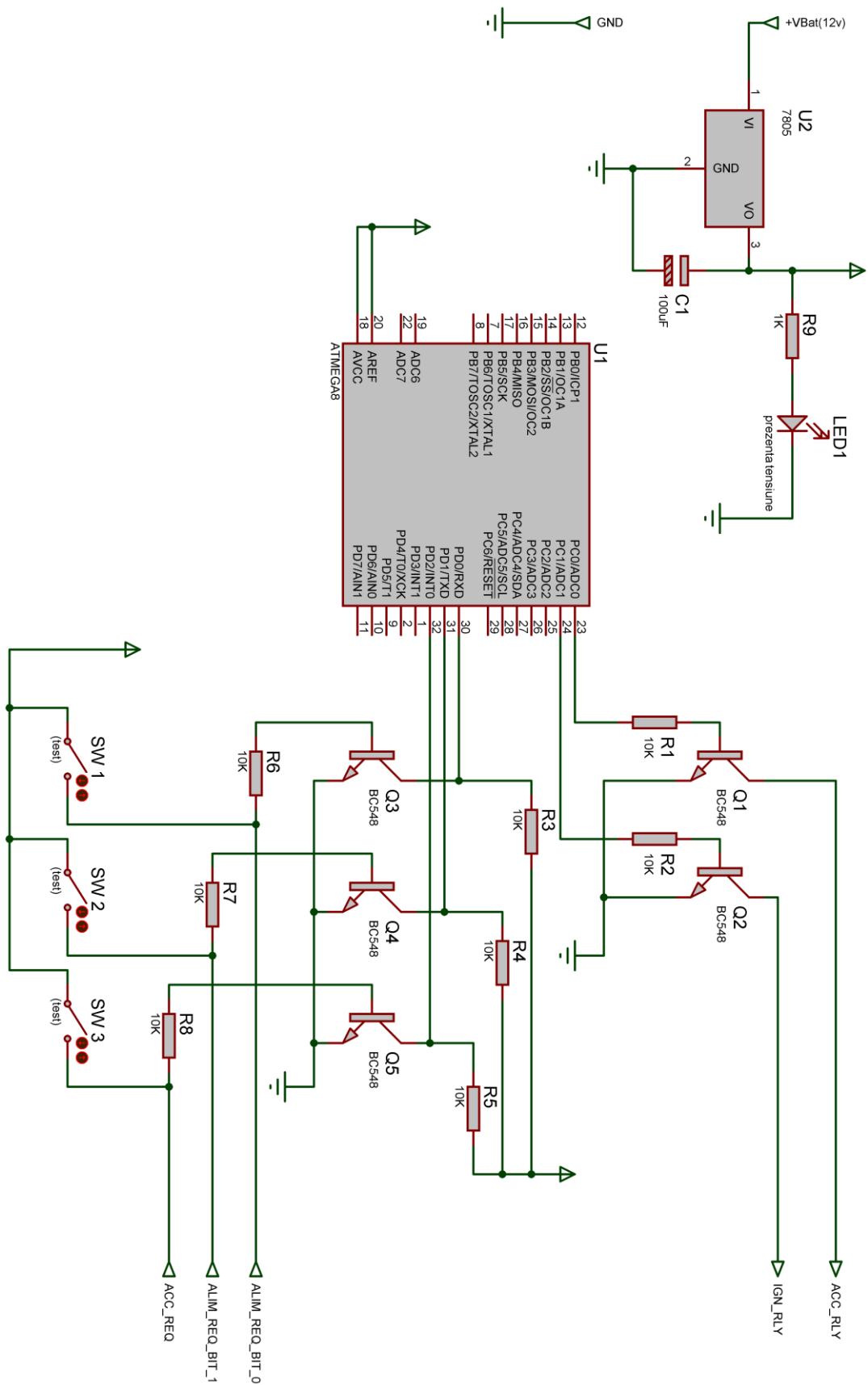
Anexa [5]



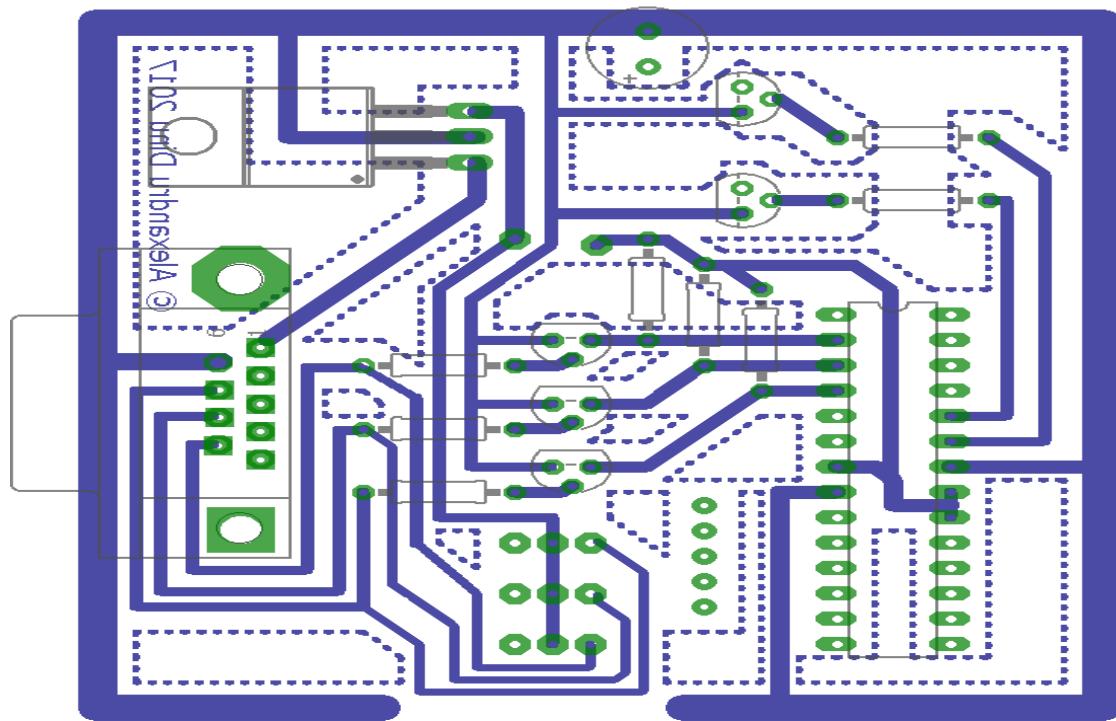
Anexa [6] – Tabel Pini MDA

Pin	Semnificație	Sensul
1	ACC_RLY	iesire
2	IGN_RLY	iesire
3	-	
4	-	
5	+Vcc (12V)	-
6	ALIM_REQ_BIT_0	intrare
7	ALIM_REQ_BIT_1	intrare
8	ACC_REQ	intrare
9	GND	-

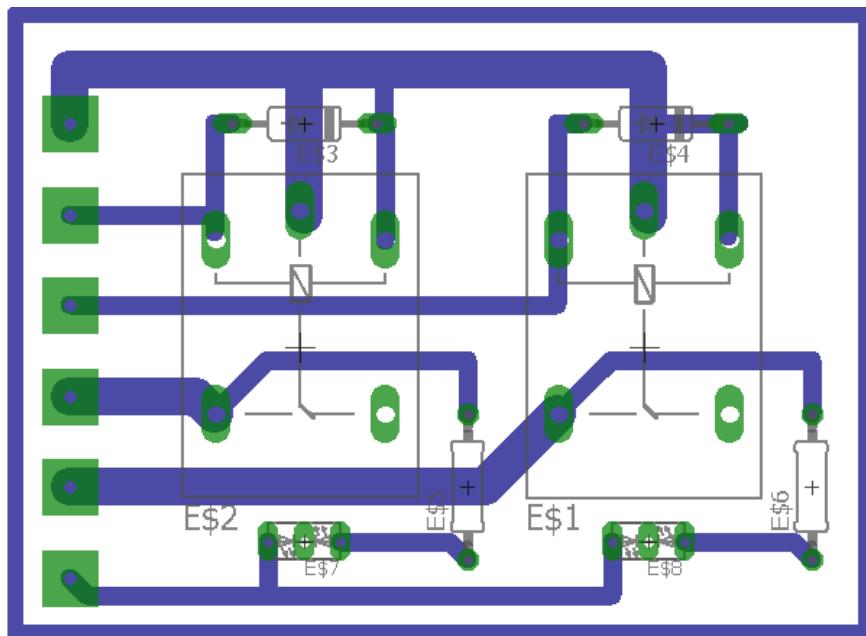
Anexa [7]



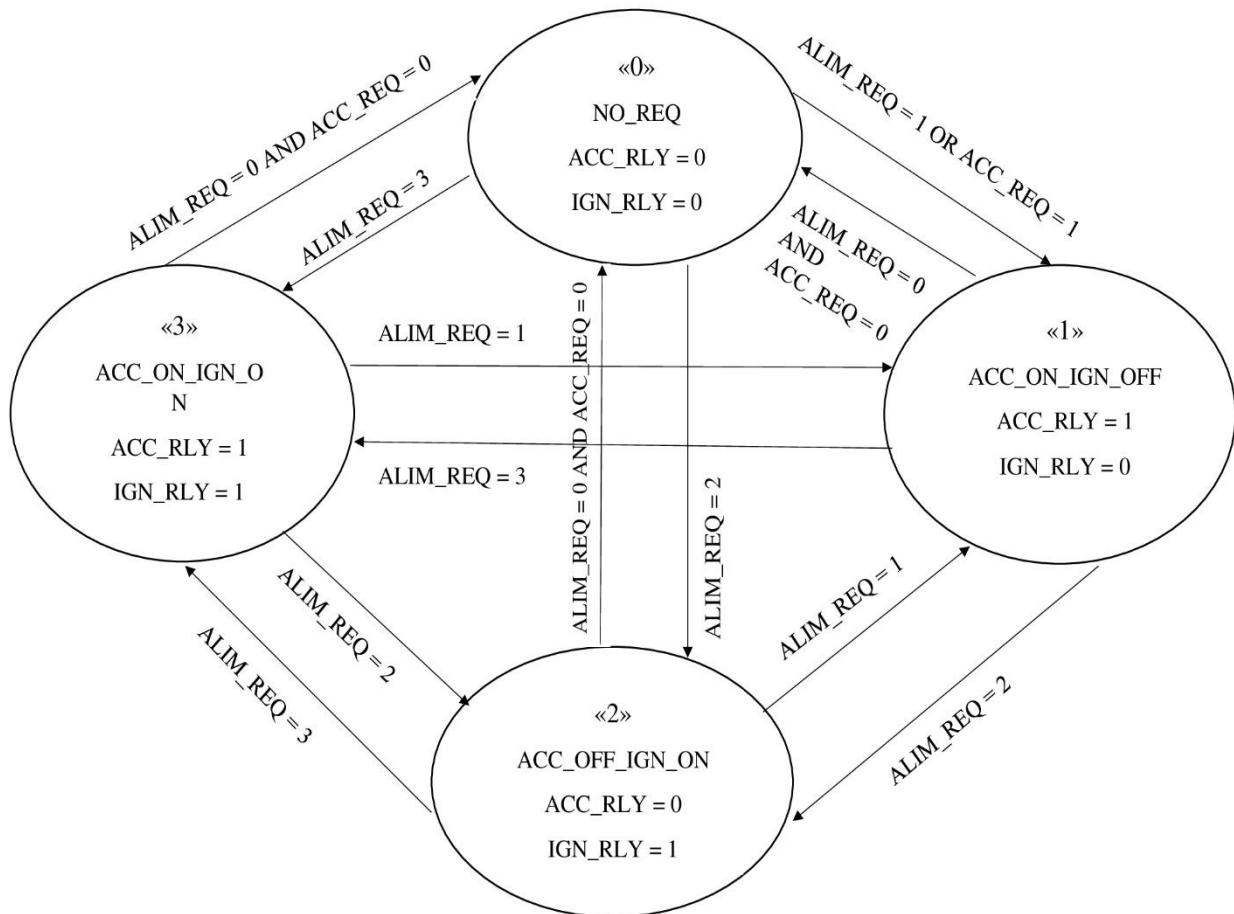
Anexa [8]



Anexa [9]



Anexa [10]



Anexa [11]

```

#include <iostream>
#include <mega8.h>
#include <inttypes.h>
#include <delay.h>
#include <stdlib.h>
#include <initializers.h>
//#include <functions.h>

//definition of the states
#define NO_REQ 0
#define ACC_ON_IGN_OFF 1
#define ACC_OFF_IGN_ON 2
#define ACC_ON_IGN_ON 3

//definition of the input/output pins
#define ACC_RLY PORTC.0
#define IGN_RLY PORTC.1
#define ALIM_REQ 3-(2*PIND.1)+PIND.0
  
```

```

#define ACC_REQ 1-PIND.2

// Declare your global variables here
int state;

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
}

void f_NO_REQ() // state 0
{
    ACC_RLY=0; IGN_RLY=0;
    delay_ms(25); //200ms
    switch(ALIM_REQ)
    {
        case 1: state=ACC_ON_IGN_OFF; return;
        case 2: state=ACC_OFF_IGN_ON; return;
        case 3: state=ACC_ON_IGN_ON; return;
        default:
            if(ACC_REQ==1)
            { state=ACC_ON_IGN_OFF; return; }
            else
            { state=NO_REQ; return; }
    }
}

void f_ACC_ON_IGN_OFF() // state 1
{
    ACC_RLY=1; IGN_RLY=0;
    delay_ms(25); //200ms
    switch(ALIM_REQ)
    {
        case 0:
            if(ACC_REQ==1)
            { state=ACC_ON_IGN_OFF; return; }
            else
            { state=NO_REQ; return; }
        case 2: state=ACC_OFF_IGN_ON; return;
        case 3: state=ACC_ON_IGN_ON; return;
        default: state=ACC_ON_IGN_OFF; return;
    }
}

void f_ACC_OFF_IGN_ON() // state 2
{
    ACC_RLY=0; IGN_RLY=1;
    delay_ms(25); //200ms
    switch(ALIM_REQ)
    {
        case 0:
            if(ACC_REQ==1)
            { state=ACC_ON_IGN_OFF; return; }
            else
            { state=NO_REQ; return; }
        case 1: state=ACC_ON_IGN_OFF; return;
        case 3: state=ACC_ON_IGN_ON; return;
        default: state=ACC_OFF_IGN_ON; return;
    }
}

```

```

    }

}

void f_ACC_ON_IGN_ON() // state 3
{
    ACC_RLY=1; IGN_RLY=1;
    delay_ms(25); //200ms
    switch(ALIM_REQ)
    {
        case 0:
        if(ACC_REQ==1)
        { state=ACC_ON_IGN_OFF; return; }
        else
        { state=NO_REQ; return; }
        case 1: state=ACC_ON_IGN_OFF; return;
        case 2: state=ACC_OFF_IGN_ON; return;
        default: state=ACC_ON_IGN_ON; return;
    }
}

void main(void)
{
Initializers();
DDRC=0x03;
DDRD=0x00;

// Globally enable interrupts
//#asm("sei")
state=0;

while (1)
    switch(state)
    {
        case NO_REQ:
        f_NO_REQ(); break;
        case ACC_ON_IGN_OFF:
        f_ACC_ON_IGN_OFF(); break;
        case ACC_OFF_IGN_ON:
        f_ACC_OFF_IGN_ON(); break;
        case ACC_ON_IGN_ON:
        f_ACC_ON_IGN_ON(); break;

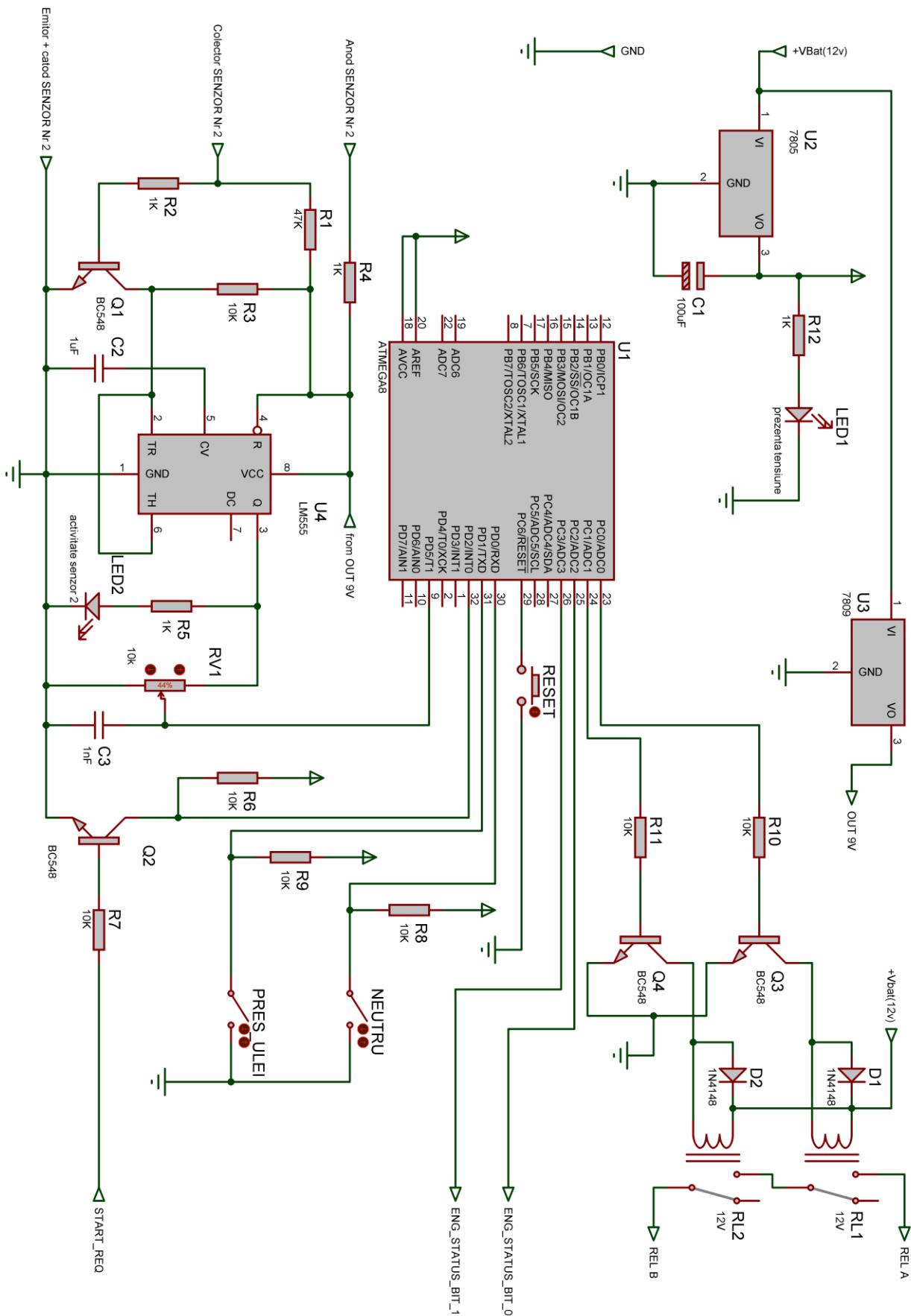
        default:
        break;
    }
}

```

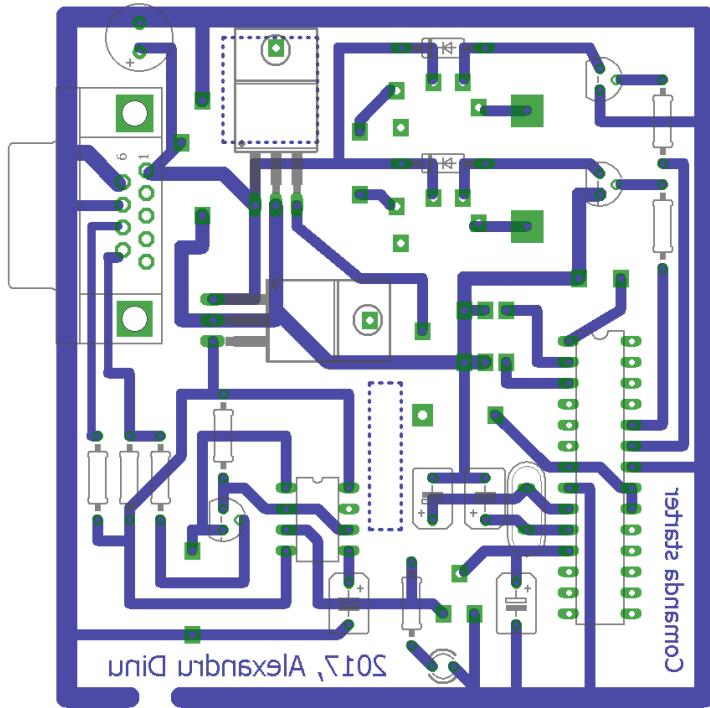
Anexa [12] – Tabel pini MCS

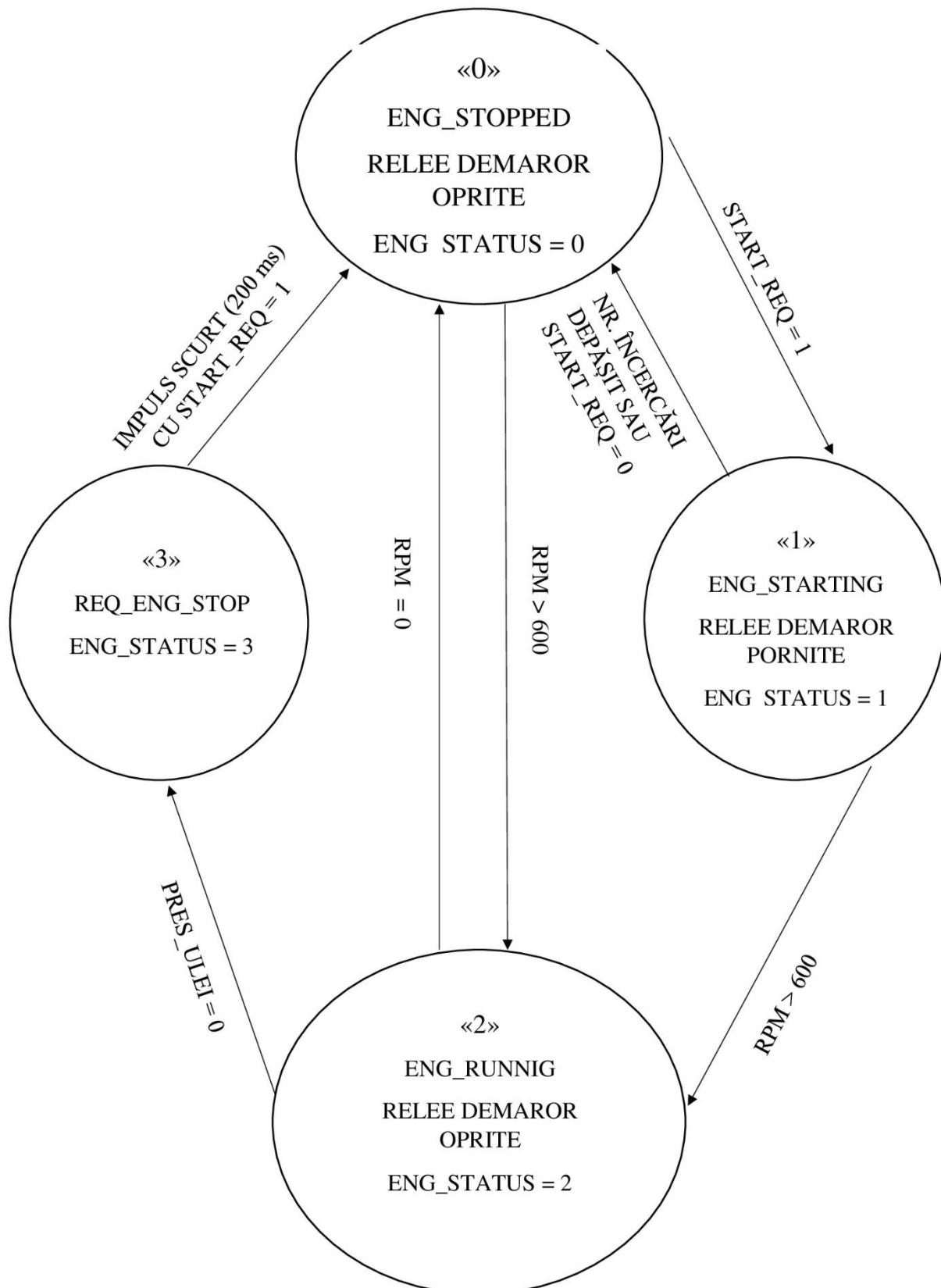
Pin	Semnificatie	Sensul
1	ENGINE_STATUS_BIT_0	iesire
2	ENGINE_STATUS_BIT_1	iesire
3	STARTING_REQ	intrare
4	-	
5	+Vcc (12V)	-
6	Colector senzor 2	intrare
7	Anod senzor 2	intrare
8	Emitor+catod senzor 2	intrare
9	GND	-

Anexa [13]



Anexa [14]





Anexa [16]

```
#include <io.h>
#include <mega8.h>
#include <inttypes.h>
#include <delay.h>
#include <stdlib.h>
#include <Initializers.h>

// definition of machine states
#define ENG_STOPPED 0
#define ENG_STARTING 1
#define ENG_RUNNING 2
#define REQ_ENG_STOP 3

// definition of input pins
#define NEUTRAL 1-PIND.0
#define PRES_ULEI 1-PIND.1
#define START_REQ 1-PIND.2

// definition of output pins
#define DEM_RLY_1 PORTC.0
#define DEM_RLY_2 PORTC.1
#define ENG_STATUS_BIT_0 PORTC.2
#define ENG_STATUS_BIT_1 PORTC.3

// definition of time constants
#define LIMITA_DURATA_PORNIRE 4000 // miliseconds
#define NR_MAX_INCERCARI_PORNIRE 4

// Declare your global variables here
int state, starting_timer, incercari_pornire, auto_restart,
cicli_motor_pornit;

unsigned long int freq; /* to store value of frequency value */
unsigned int i=0,dur; /* i=number of overflows in one second */

// Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
    i++ ; // count the number of overflows in one second
}

int get_freq()
{
    TIMSK=0x04; // enable overflow interrupt of timer1
    TCCR1B=0x07; /* start timer1 with external pulses (T1 rising edge)
*/
    delay_ms(200); // wait for one second
    TCCR1B=0x00; //stop timer1
    TIMSK=0x00; //disable interrupt
    dur=TCNT1; /* store the number of counts in TCNT1 register */
    freq = 5*(dur + i*65536); /* calculate the frequency as in previous
equation */
    TCNT1=0x0000; /* clear TCNT1 register for the next reading */
}
```

```

        i=0;      /* clear number of overflows in one second for the next
reading */
        return freq*60/119;
    }

void f_ENG_STOPPED()
{
    delay_ms(1000);
    DEM_RLY_1=0; DEM_RLY_2=0; // demaror oprit
    ENG_STATUS_BIT_0=0; ENG_STATUS_BIT_1=0; // trimite stare motor (0)
    incercari_pornire=0;
    starting_timer=0;
    cicli_motor_pornit=0;
    if(get_freq()>=600) // daca motorul a fost pornit prin impingere
    {
        state=ENG_RUNNING;
        return;
    }
    if(START_REQ==1) // daca avem comanda de pornire
    {
        if(get_freq()==0 && NEUTRAL==1)
        {
            state=ENG_STARTING;
            starting_timer=0;
            return;
        }
    }
    return;
}

void f_ENG_STARTING()
{
    ENG_STATUS_BIT_0=1; ENG_STATUS_BIT_1=0; // trimite stare motor (1)
    if(START_REQ==1 && starting_timer<=LIMITA_DURATA_PORNIRE)
    {
        DEM_RLY_1=1; DEM_RLY_2=1; // demaror pornit
        starting_timer+=200;
        if(get_freq()>=600) // daca motorul este autonom
        {
            DEM_RLY_1=0; DEM_RLY_2=0; // demaror oprit
            state=ENG_RUNNING;
            return;
        }
    }
    if(START_REQ==1 && starting_timer>LIMITA_DURATA_PORNIRE)
    {
        DEM_RLY_1=0; DEM_RLY_2=0; // demaror oprit
        if(auto_restart==1 && incercari_pornire <
NR_MAX_INCERCARI_PORNIRE)
        {
            delay_ms(5000); // relaxare demaror
            incercari_pornire++;
            starting_timer=0;
        }
        else // daca numarul maxim de redemaraje a fost atins SAU nu avem
redemaraj automat
        {
            incercari_pornire=0; starting_timer=0;

```

```

        state=ENG_STOPPED;
        ENG_STATUS_BIT_0=0; ENG_STATUS_BIT_1=0;
        delay_ms(500);
        return;
    }
}
if(START_REQ==0)
{
    incercari_pornire=0; starting_timer=0;
    state=ENG_STOPPED; return;
}
return;
}

void f_ENG_RUNNING()
{
    ENG_STATUS_BIT_0=0; ENG_STATUS_BIT_1=1; // trimite stare motor (2)
    cicli_motor_pornit++;
    if(cicli_motor_pornit>10000) cicli_motor_pornit=10;
    if(get_freq()==0)
    {
        state=ENG_STOPPED; return;
    }
    if(cicli_motor_pornit>=10 && PRES_ULEI==0)
    {
        state=REQ_ENG_STOP; return;
    }
    return;
}

void f_REQ_ENG_STOP()
{
    ENG_STATUS_BIT_0=1; ENG_STATUS_BIT_1=1; // trimite stare motor (3)
    if(START_REQ==1)
    {
        delay_ms(500);
        state=ENG_STOPPED; return;
    }
    return;
}

void main(void)
{
// Declare your local variables here

Initializers();
state=ENG_STOPPED;
auto_restart=1;

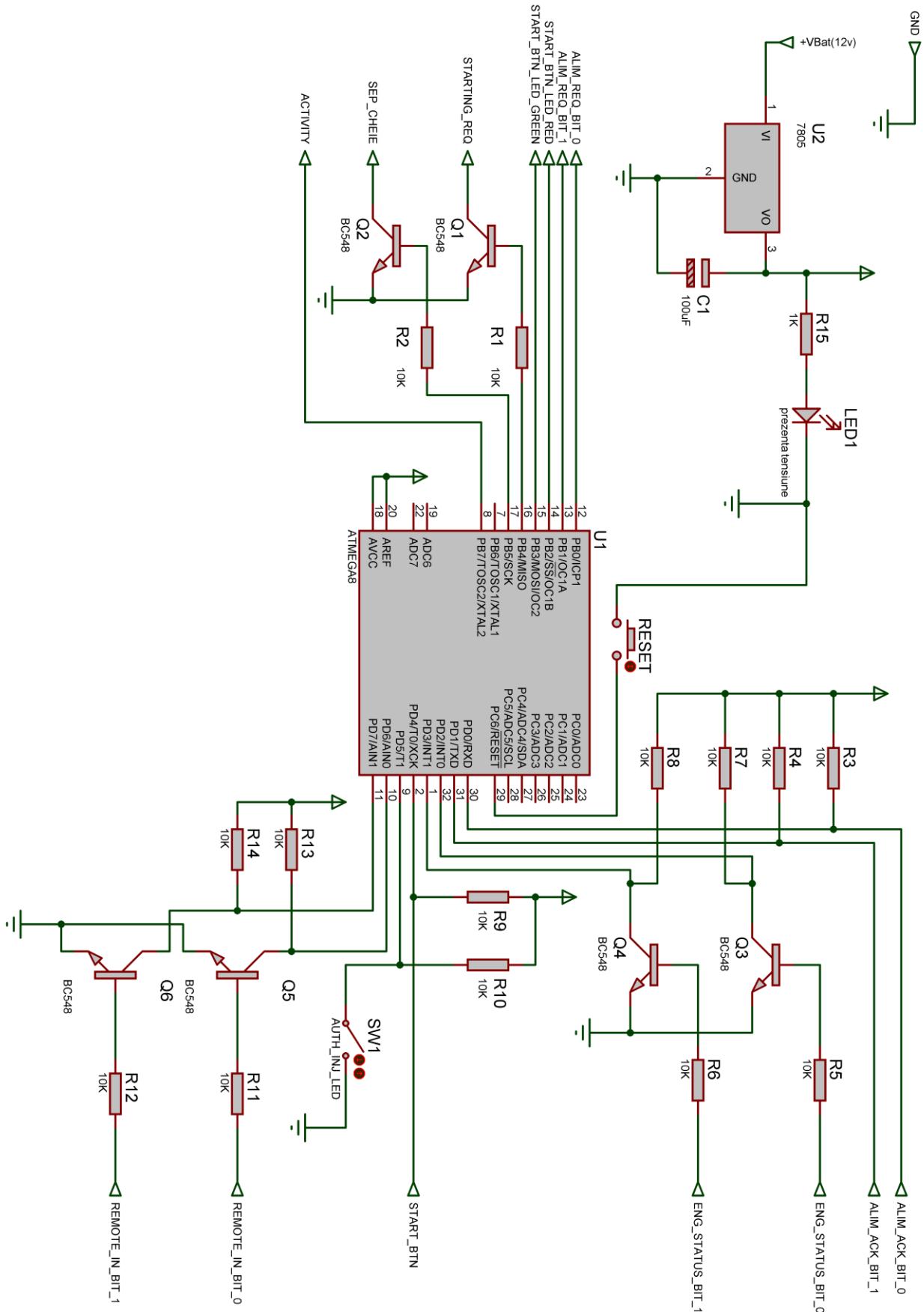
while (1)
{
    switch(state)
    {
        case ENG_STOPPED:
            f_ENG_STOPPED(); break;
        case ENG_STARTING:
            f_ENG_STARTING(); break;
        case ENG_RUNNING:

```

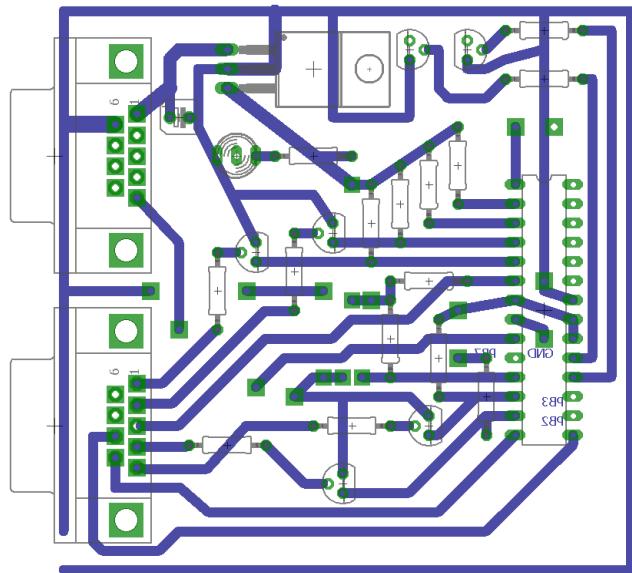
```
f_ENG_RUNNING(); break;
case REQ_ENG_STOP:
f_REQ_ENG_STOP(); break;
default:
f_ENG_STOPPED(); break;
}
}

}
```

Anexa [17]



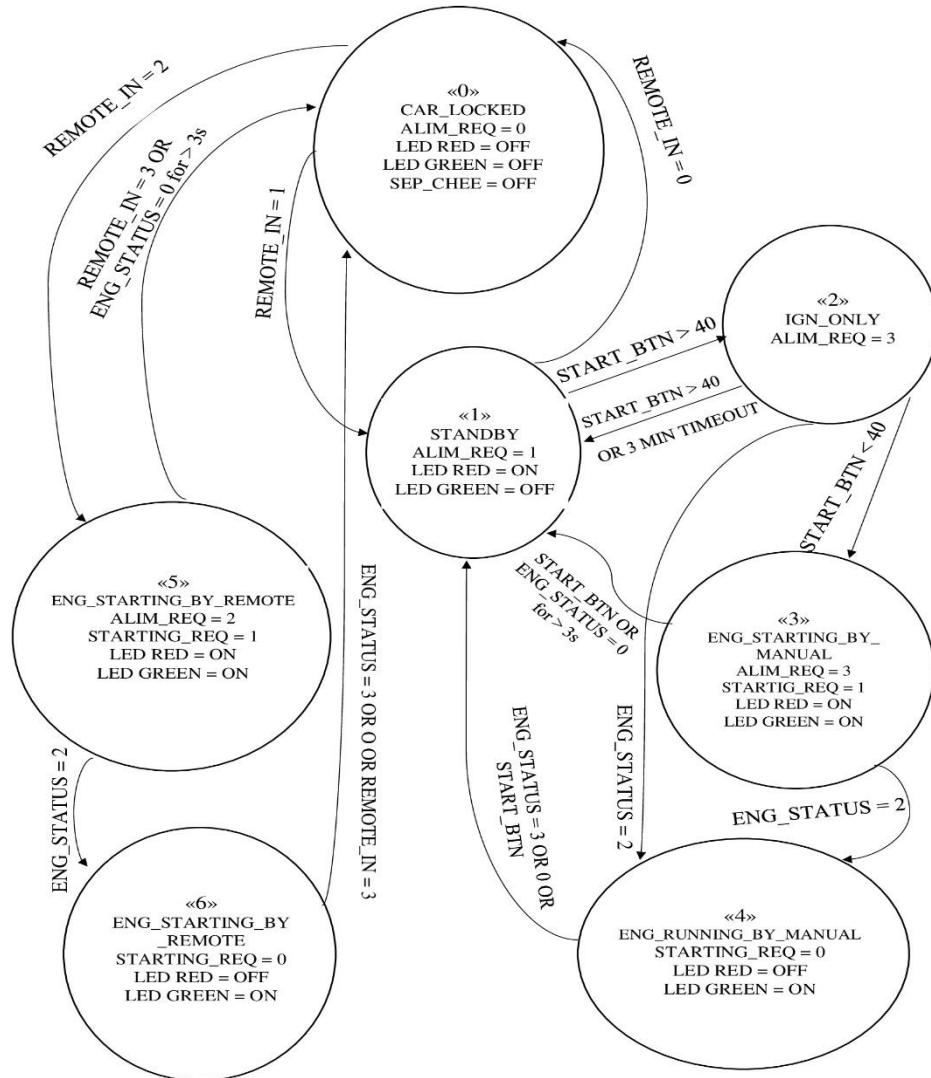
Anexa [18]



Anexa [19] – Tabel Pini MP

Pin mufa 1	Semnificatie	Sensul
1	ACTIVITY	iesire
2	ALIM_ACK_BIT_0	intrare
3	ALIM_ACK_BIT_1	intrare
4	-	-
5	+Vcc (12V)	-
6	SEP_CHEIE	iesire
7	STARTING_REQ	iesire
8	-	-
9	GND	-
Pin mufa 2	Semnificatie	Sensul
1	REMOTE_IN_BIT_0	intrare
2	REMOTE_IN_BIT_1	intrare
3	START_BTN	intrare
4	ENG_STATUS_BIT_0	intrare
5	ENG_STATUS_BIT_1	intrare
6	ALIM_REQ_BIT_0	iesire
7	ALIM_REQ_BIT_1	iesire
8	START_BTN_LED_RED	iesire
9	START_BTN_LED_GREEN	iesire

Anexa [20]



Anexa [21]

```

#include <iio.h>
#include <mega8.h>
#include <inttypes.h>
#include <delay.h>
#include <stdlib.h>
#include <initializers.h>

// definition of output pins
#define ALIM_REQ_BIT_0 PORTB.0
#define ALIM_REQ_BIT_1 PORTB.1
#define START_BTN_LED_RED PORTB.2
#define START_BTN_LED_GREEN PORTB.3
#define STARTING_REQ PORTB.4
#define SEP_CHEIE PORTB.5
#define ACTIVITY PORTB.6

```

```

// definition of input pins
#define ALIM_ACK_BIT_0 1-PIND.0
#define ALIM_ACK_BIT_1 1-PIND.1
#define ENG_STATUS_BIT_0 1-PIND.2
#define ENG_STATUS_BIT_1 1-PIND.3
#define START_BTN 1-PIND.4
#define AUTH_INJ_LED 1-PIND.5
#define REMOTE_IN_BIT_0 1-PIND.6
#define REMOTE_IN_BIT_1 1-PIND.7
#define REMOTE_IN 3-((2*PIND.7)+PIND.6)
#define ALIM_ACK 3-((2*PIND.1)+PIND.0)
#define ENGINE_STATUS 3-((2*PIND.3)+PIND.2)

// definition of time constants
#define TIMP_IGN_FORTAT 120 // = 4 seconds

// definition of states
#define CAR_LOCKED 0
#define STANDBY 1
#define IGNITION_ONLY 2
#define ENG_STARTING_BY_MANUAL 3
#define ENG_RUNNING_BY_MANUAL 4
#define ENG_STARTING_BY_REMOTE 5
#define ENG_RUNNING_BY_REMOTE 6

// declaration of global variables
int i=0;
int state;

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
i++; // 30 increments per second
}

void f_CAR_LOCKED()
{
    ALIM_REQ_BIT_0=0;
    ALIM_REQ_BIT_1=0; // no request
    ACTIVITY=0;
    START_BTN_LED_RED=0;
    START_BTN_LED_GREEN=0;
    SEP_CHEIE=0;
    delay_ms(1000);
    if(REMOTE_IN==1)
    { state=STANDBY; SEP_CHEIE=1; return; }
    if(REMOTE_IN==2)
    { state=ENG_STARTING_BY_REMOTE;
        START_BTN_LED_GREEN=1;
        START_BTN_LED_RED=1; SEP_CHEIE=1; i=0; return; }
    return;
}

void f_STANDBY()
{
    START_BTN_LED_GREEN=0;

```

```

START_BTN_LED_RED=1;
ALIM_REQ_BIT_0=1;
ALIM_REQ_BIT_1=0; // ACC request
while(ALIM_ACK!=1) {} // wait for ACC to be ON

if(START_BTN==1)
{
    i=0;
    while(START_BTN==1 && i<TIMP_IGN_FORTAT+1) {}
    if(i<TIMP_IGN_FORTAT)
    {
        state=ENG_STARTING_BY_MANUAL; i=0;
        return;
    }
    ALIM_REQ_BIT_0=1;
    ALIM_REQ_BIT_1=1; // ACC & IGN request
    while(START_BTN==1) {} // wait for button to be depressed
    state=IGNITION_ONLY; i=0;
    return;
}

if(REMOTE_IN==0)
{ state=CAR_LOCKED; SEP_CHEIE=0; return; }
return;
}

void f_IGNITION_ONLY()
{
    while(ALIM_ACK!=3) {} // wait for ACC and IGN to be ON

    if(i>5400) // 3 minutes passed
    {
        state=STANDBY;
        return;
    }

    if(START_BTN==1)
    {
        i=0;
        while(START_BTN==1 && i<TIMP_IGN_FORTAT+1) {}
        if(i<TIMP_IGN_FORTAT)
        {
            state=ENG_STARTING_BY_MANUAL; i=0;
            return;
        }
        ALIM_REQ_BIT_0=1;
        ALIM_REQ_BIT_1=0; // ACC request
        while(START_BTN==1) {} // wait for button to be depressed
        state=STANDBY; return;
    }

    if(ENGINE_STATUS==2)
    { state=ENG_RUNNING_BY_MANUAL; return; }
    return;
}

void f_ENG_STARTING_BY_MANUAL()
{
}

```

```

START_BTN_LED_GREEN=1;
ALIM_REQ_BIT_1=1; // IGN request
while(ALIM_ACK!=3) {} // wait for ACC and IGN to be ON

if(ENGINE_STATUS==2)
{ STARTING_REQ=0; state=ENG_RUNNING_BY_MANUAL; return; }

if(ENGINE_STATUS==0 && i<90) // below 3 seconds, engine not started
yet
{
    STARTING_REQ=1;
    while(ENGINE_STATUS==0 && i<90) {} // wait for engine status to
become 'STARTING'
}

if(ENGINE_STATUS==0 && i>90) // above 3 seconds, engine starting
failed
{
    STARTING_REQ=0;
    state=STANDBY; return;
}

if(START_BTN==1)
{
    while(START_BTN==1) {} // wait for button to be depressed
    STARTING_REQ=0;
    while(ENGINE_STATUS==1) {} // wait for engine status to become
'STOPPED'
    state=STANDBY;
    return;
}
return;
}

void f_ENG_RUNNING_BY_MANUAL()
{
    START_BTN_LED_GREEN=1;
    START_BTN_LED_RED=0;

    if(ENGINE_STATUS==0) // daca moare motorul
    { state=STANDBY;
        START_BTN_LED_GREEN=0; return; }

    if(ENGINE_STATUS==3) // request engine stop
    {
        ALIM_REQ_BIT_1=0; // cut IGN
        delay_ms(2000);
        STARTING_REQ=1;
        delay_ms(200);
        STARTING_REQ=0;
        delay_ms(500);
        state=STANDBY;
        START_BTN_LED_GREEN=0;
        return;
    }

    if(START_BTN==1)
    {
        while(START_BTN==1) {} // wait for button to be depressed
    }
}

```

```

        ALIM_REQ_BIT_1=0; // cut IGN
        while(ENGINE_STATUS==2) {} // wait for engine status to become
'STOPPED'
        state=STANDBY;
        START_BTN_LED_GREEN=0;
        return;
    }
    return;
}
void f_ENG_STARTING_BY_REMOTE()
{
    ALIM_REQ_BIT_1=1;
    ALIM_REQ_BIT_0=0; // IGN ON & CUT ACC request
    while(ALIM_ACK!=2) {} // wait for ACC to be OFF and IGN to be ON

    if(ENGINE_STATUS==2)
    { STARTING_REQ=0; state=ENG_RUNNING_BY_REMOTE; return; }

    if(ENGINE_STATUS==0 && i<90) // below 3 seconds, engine not started
yet
    {
        STARTING_REQ=1;
        while(ENGINE_STATUS==0) {} // wait for engine status to become
'STARTING'
    }

    if(ENGINE_STATUS==0 && i>90) // above 3 seconds, engine starting
failed
    {
        STARTING_REQ=0;
        state=CAR_LOCKED; return;
    }

    if(REMOTE_IN==3)
    {
        ALIM_REQ_BIT_1=0; // cut IGN
        STARTING_REQ=0;
        while(ENGINE_STATUS==1) {} // wait for engine status to become
'STOPPED'
        state=CAR_LOCKED;
        return;
    }
    return;
}

void f_ENG_RUNNING_BY_REMOTE()
{
    START_BTN_LED_RED=0;
    ALIM_REQ_BIT_0=1; // put back ACC request

    if(ENGINE_STATUS==0) // daca moare motorul
    { state=CAR_LOCKED;
        START_BTN_LED_GREEN=0; return; }

    if(ENGINE_STATUS==3) // request engine stop
    {
        ALIM_REQ_BIT_1=0; // cut IGN
        delay_ms(2000);

```

```

STARTING_REQ=1;
delay_ms(200);
STARTING_REQ=0;
delay_ms(500);
state=CAR_LOCKED;
START_BTN_LED_GREEN=0;
return;
}

if (REMOTE_IN==3)
{
    ALIM_REQ_BIT_1=0; // cut IGN
    while(ENGINE_STATUS==2) {} // wait for engine status to become
'STOPPED'
    state=CAR_LOCKED;
    START_BTN_LED_GREEN=0;
    return;
}

//if(conditions to abort engine running while car secured)
// abort engine running
return;
}

void main(void)
{
Initializers();

DDRB=0xFF;
DDRD=0x00;
// Declare your local variables here

// Globally enable interrupts
#asm("sei")

state=0;
STARTING_REQ=0;
ACTIVITY=0;

while (1)
{
    switch(state)
    {
        case CAR_LOCKED:
            f_CAR_LOCKED(); break;
        case STANDBY:
            f_STANDBY(); break;
        case IGNITION_ONLY:
            f_IGNITION_ONLY(); break;
        case ENG_STARTING_BY_MANUAL:
            f_ENG_STARTING_BY_MANUAL(); break;
        case ENG_RUNNING_BY_MANUAL:
            f_ENG_RUNNING_BY_MANUAL(); break;
        case ENG_STARTING_BY_REMOTE:
            f_ENG_STARTING_BY_REMOTE(); break;
        case ENG_RUNNING_BY_REMOTE:
            f_ENG_RUNNING_BY_REMOTE(); break;
        default:
}

```

```

f_CAR_LOCKED(); break;
}

}

```

Anexa [22] – Tabel Interconexiuni

Modul	Pin/mufa	Modul	Pin/mufa
SM	1	MCS	8
SM	2	MCS	6
SM	3	MCS	8
SM	4	MCS	7
MDA	6	MP	6/mufa 2
MDA	7	MP	7/mufa 2
MDA	1	Placa relee	"ACC_CMD"
MDA	2	Placa relee	"IGN_CMD"
Placa relee	"ALIM_ACK_BIT_0"	MP	2/mufa 1
Placa relee	"ALIM_ACK_BIT_1"	MP	3/mufa 1
Placa relee	"IGN_OUT"	SW selector simulator	unul dintre contacte
MCS	1	MP	4/mufa 2
MCS	2	MP	5/mufa 2
Placuta buton start	"START_BTN"	MP	3/mufa 2
Placuta buton start	"START_BTN_LED_RED"	MP	8/mufa 2
Placuta buton start	"START_BTN_LED_GREEN"	MP	9/mufa 2
MCS	3	MP	7/mufa 1
Toate modulele	5 (5/mufa 1) -> +Vbat (12v)		
Toate modulele	9 (9/mufa 1) -> GND		

Bibliografie

1. Călinoiu, C., *Senzori și traductoare*, vol. I, Editura Tehnică, București, 2009.
2. Romanca, M., Microprocesoare și microcontrolerele, Editura Universității Transilvania, Brașov, 2015.
3. 2486Z-AVR-02/11; *Manual de utilizare ATmega8 – ATmega8 Datasheet*, Atmel, 2011
http://www.atmel.com/images/atmel-2486-8-bit-avr-microcontroller-atmega8_1_datasheet.pdf, ultima accesare: 30.06.2017.
4. Definiții certificare ISO 31000, Testele de integrare, 2009, <http://dexcert.ro/iso-31000/testele-de-integrare.html>, ultima accesare: 30.06.2017.
5. *Elemente practice de bază în dezvoltarea sistemelor cu microprocesoare integrate*,
https://www.robofun.ro/docs/00ac800e-b0bb-4763-8a6d-2e80be1a5764_lucrare5.pdf, ultima accesare: 01.07.2017.
6. *Microcontrolere*,
<http://web.ulbsibiu.ro/laurean.bogdan/html/Microcontrolere%20introducere.pdf>, ultima accesare: 30.06.2017.
7. *Batsocks.co.uk*, http://www.batsocks.co.uk/img/info_isp/ISP%2010%20way%20pinout.png, ultima accesare: 28.06.2017.
8. EE-SX1137, Omron Electronic Components LLC,
[https://www.components.omron.com/components/web/pdflib.nsf/0/A1F899F383D0FFEC85257201007DD5E9/\\$file/EE_SX1137_1010.pdf](https://www.components.omron.com/components/web/pdflib.nsf/0/A1F899F383D0FFEC85257201007DD5E9/$file/EE_SX1137_1010.pdf), ultima accesare: 28.06.2017.
9. Allaboutcircuits.com, <https://www.allaboutcircuits.com/textbook/experiments/chpt-8/555-schmitt-trigger/>, ultima accesare: 28.06.2017.
10. Zte.com.cn,
http://wwwen.zte.com.cn/endata/magazine/ztecommunications/2008year/no2/articles/200806/t20080624_162464.html , ultima accesare: 20.06.2017.
11. Instructables.com <http://www.instructables.com/id/Very-simple-PWM-with-555Modulate-every-thing> , ultima accesare: 20.06.2017.
12. Dezmembrarimasini.ro, <http://www.dezmembrarimasini.ro/media/images/original/volantadacia-solenza-2003--vclKgf.jpg>, ultima accesare: 20.06.2017 .
13. Manual service Logan.
14. Manual de reparație pentru echipamentele electrice Dacia Solenza,
<https://www.slideshare.net/teodor0220/16234472-electricevolirox41solenza>.
15. Blogspot.com, http://1.bp.blogspot.com/-R51S44pEUok/TjRKDv4AJDI/AAAAAAAADBo/YQ0_onmTwfY/s1600/DSC02055.JPG , ultima accesare: 20.06.2017.