Calculator

Student: **Calcan Sorin Andrei - 334CC**

Introducere

In cadrul acestui proiect imi doresc sa implementez un calculator de buzunar capabil sa evalueaza expresii matematice infixate incomplet parantezate (ex: 32 / (3 + 1) - 1 / 2 + 3 * 3). Astfel, modulul va avea o interfata interactiva in care utilizatorul poate sa isi introduca expresia de la tastatura si, in timp real, aceasta va fi afisata pe ecran pentru a permite corectura, iar la final utilizatorul primeste rezultatul evaluarii expresiei.

Ideea initiala provine dintr-o incercare de implementare de algoritm in cadrul laboratorul de Structuri de Date imbinata cu o curiozitate pentru astfel de dispozitive de cand eram mic (printre primele mele jucarii a fost un calculator de birou de-al lui tata). Consider ca este un proiect util pentru oricine are nevoie imediata de rezultatul unui calcul, eliminand factorul de eroare umana sau nevoia unui pix si foaie. De asemenea, fiind o proiect ce foloseste Arduino, in mod natural acesta este portabil, iar interfata simplista face dispozitivul usor de utilizat.

Descriere generală

Intr-un ciclu de functionare normala, sistemul incepe in etapa de introducere a inputului, unde utilizatorul poate sa actioneze ansamblul de butoane pentru cifre (0 - 9), operatori (+, -, *, /), paranteze, virgula si de control (=, del). Apasarea acestor butoane are feedback instant pe ecran pentru a permite ajustarea. Odata cu apasarea butonului ”=”, inputul utilizatorului va fi procesat de algoritmul de parsare de expresii, iar pe ecran se va afisa rezultatul in cazul unui input valid, sau “ERR” in caz de eroare.

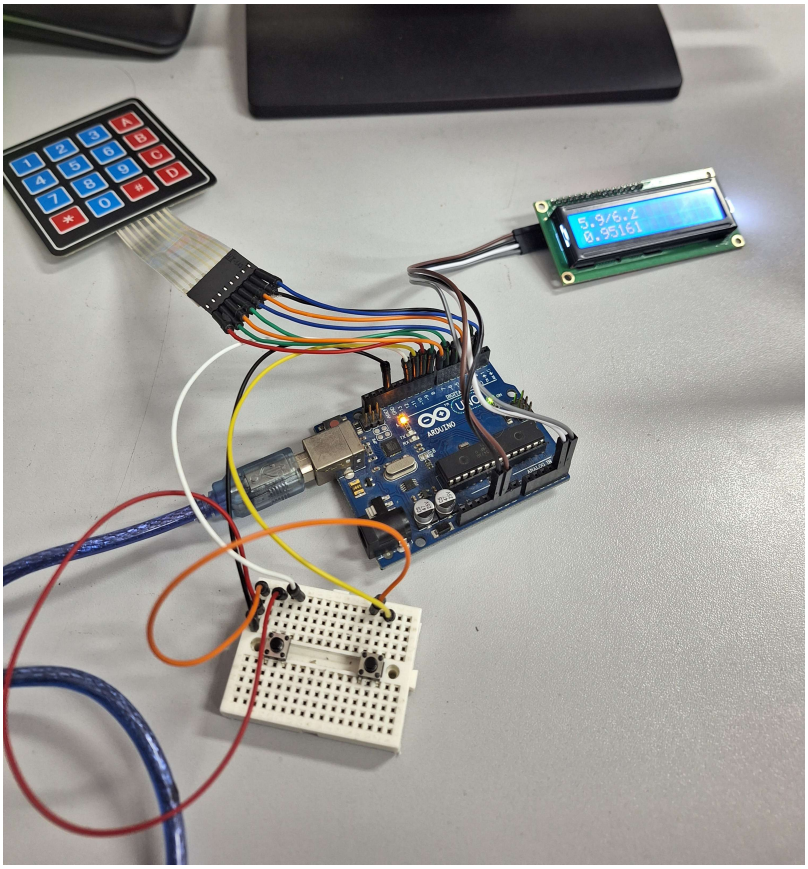
Schema bloc

Hardware Design

Piese

- Arduino UNO
- Tastatura Matriciala: see
- Butoane
- Ecran LCD 1602
- Breadboard, fire, rezistente

Montaj



Schema electrica

Software Design

Mediu de dezvoltare: PlatformIO

Biblioteci externe utilizate: LiquidCrystal_I2C.h

Input

In cadrul implementarii calculatorului, pentru ca utilizatorul sa poate interactiona cu ansamblul, este esentiala identificarea apasarii oricarui dintre butoanele puse la dispozitie pentru preluarea inputului. Aceasta logica este implementata de catre functia read_keypad() care ruleaza in bucla principala a sistemului pentru a asigura o permanenta disponibilitate catre utilizator.

Interfatarea cu tastatura matriciala 4×4 se realizeaza folosind 8 pini digitali (4 linii si 4 coloane). Pinii corespunzatori liniilor sunt setati ca pini de output cu valoarea default HIGH, iar pinii de coloane sunt setati ca input, legati la nivelul HIGH prin activarea rezistentei de pull-up. Pentru a prelua butoanele apsate de pe tastatura este necesare o scanare continua a fiecarei linii din matrice prin coborarea temporara a nivelului logic pentru pinul corespunzator liniei la LOW si apelarea functiei scan_row(). Apasarea unui buton are ca efect scurtcircuitarea rezistentei de pull-up pentru pinul de input al coloanei, ducand valoarea acestuia la valoarea logica a liniei pe care se afla butonul, de aceea este esential ca doar o singura linie sa fie legata la masa la o citire (linia scanata in acel moment). Astfel la scanarea unei linii se itereaza prin toti registrii de input ai coloanelor si se identifica o apasare in cazul unei valori de 0 (LOW).

Dupa identificarea butonului apasat de la tastatura, se preia valoarea caracterului corespunzator din matricea de caractere initializata inca de la pornirea sistemului, se verifica conditii de eroare (in cazul unui input prea lung), se face handling in cazul apasarii butonului de paranteze pentru a decide ce tip de paranteza intentioneaza utilizatorul sa puna, se actualizeaza bufferul de input si se afiseaza pe ecranul LCD printr-o transmisie prin I2C.

Pentru interfatarea cu butoanele de control, lucrurile stau mai simplu: Fiecare dintre cele 2 butoane are un pin de date aferent care este initializat ca pin de GPIO, de intrare, care este legat by default la HIGH prin pull-up. La apasare, circuitul se scurtcircuiteaza, punand acel pin pe nivel logic LOW. Acest lucru este identificat prin citirea registrului corespunzator si executand protocolul subscris de acel buton.

Pentru primul buton de control, acesta are rolul de buton de stergere, dand overwrite la ultimul caracter scris cu caracterul nul si actualizand counterul intern pentru buffer si retransmitand outputul la ecran.

Al doilea buton este practic butonul de ”=” sau “enter”. Odata cu apasarea lui, bufferul este preluat de functia de parsare de expresii, iar rezultatul este afisat pe a doua linia a ecranului, resetand in spate bufferul de input pentru a pregati un nou calcul de la utilizator.

Dupa apasarea oricarui buton, sistemul este pus intr-un busy wait printr-un delay pe perioada de debounce, pentru a evita in acelasi timp inregistrarea multipla a unei singure apasari sau a unei apasar accidentale. Acest lucru de putea implementa mai elegant cu intreruperi si Timer insa sistemul neavand alte taskuri importante sa se intample intre apasari, busy waitul este mai simplu si isi atinge scopul.

Parser

Pentru algoritmul de parsare a expresiilor aritmetice infixate, incorect parantezate, am implementat o versiune de algoritm bazata pe Shunting Yard al lui Edgar Dijkstra. Parserul se gaseste in fisierul eval.c, care ca prerequisite foloseste o implementare eficienta de stiva generica in C.

Am folosit o stiva in C deoarece am vrut sa optez pentru ceva low-level cu putin overhead, fiind vorba de resursele limitate puse la dispozitie de microcontrollerul Atmega328P, am evitat pe cat posibil

folosirea unei implementari mai high level prin C++. Singurul motiv pentru care fisierul main este in cpp este pentru a putea interfata cu libraria de LCD care este implementata in cpp. Stiva este generica deoarece parserul utilizeaza in structurile sale interne 2 stive cu tipuri de date diferite (una pentru operatori si cealalta pentru operanzi).

Parserul incepe prin a-si initializa structurile interne, urmand sa parcurga fiecare caracter din bufferul expresiei primite ca parametru sub forma unui string (char * in C):

- Pentru un caracter numeric, parserul incepe protocolul de construire a unui operand cifra cu cifra prin inmultirea repetitiva cu 10 si adunarea acesteia la total. In cazul intalnirii virgulei, protocolul trece la crearea partii fractionare in acelasi mod, la final fiind asamblate adunand partea intreaga la cea fractionara. Astfel, toate caracterele numerice insirate cu o virgula intre ele sunt parcurse si adaugate in stiva de operanzi.
- In cazul unui operator ”+”, ”-”, “*”, ”/”, se verifica initial cazul particular in care ”-” are rolul de a inversa un operand (e.g ”(-2) * 3”), caz in care urmatorul operand este introdus in stiva cu semn schimbat. Inainte ca operatorul sa fie introdus in stiva corespunzatoare, algoritmul verifica daca in varful stivei de operatori se afla operatii cu precedenta mai mare sau egala cu cea curenta, caz in care isi permite evaluarea imediata a acestui calcul scotand ultimii doi operatanzi din stiva si apeland applyOp() pentru a obtine rezultatul si a-l introduce inapoi in stiva de numere. In final noul operator parsat este introdus in stiva sa.
- Pentru ”(” aceasta se adauga simplu in stiva de operator.
- Pentru ”)” se forteaza evaluarea tuturor operatiilor din stiva de operatori pana la intalnirea ”(”.

Pentru a determina precedenta unui operator s-a utilizat functia precedence() care atribuie precedenta 2 pentru inmutire si impartire, si precedent 1 pentru adunare si scadere.

Debug

Pentru o experienta mai usoara de dezvoltare, o functionalitate suplimentara adaugata calculatorului o reprezint modul de DEBUG care porneste, prin registrii, o comunicare seriala prin UART, la care face forward la standard output pentru a putea folosi functii de I/O din libraria standard “avr/io.h” (printf). Astfel, daca sistemul ruleaza in modul DEBUG, la fiecare apasare de buton, acest lucru este transmis la un monitor serial printr-un mesaj de tipul “S-a apasat x”. De asemenea sunt trimise la monitor si edge caseurile relevante precum inversarea unui operand sau depasirea lungimii bufferului de input.

Rezultate Obţinute

Proiectul se afla intr-o stare functionala, fiind posibila introducerea unor expresii aritmetice valide prin interactiunea cu tastatura matriciala 4×4 si butonele de control pentru a obtine rezultate corecte.

Posibile imbunatatiri la calitatea proiectului se pot face atat la nivel fizic, prin proiectarea unei carcase compacte care sa inmagazineze toate componentele pentru a pune in aplicarea avantajele dimensiunilor mici ale place Arduino, cat si la nivel sofware prin adaugarea unui modul de input proofing pentru filtrarea expresiilor gresite.

Concluzii

In urma acestui proiect am inteles importanta optimizarilor pentru sisteme cu resurse limitate, fiind nevoie sa limitez structurile de date utilizate si sa le optimizez pe cele folosite. De asemenea s-a facut evidenta importanta unei abordari robuste, in absenta unui sistem de operare microcontrollerul avand posibilitatea sa intre intr-un mod de functionare nedeterminista prin operatii gresite cu memoria.

De altfel, am prins si lectii in ceea ce priveste designul hardware: Inainte de a comanda piese ar fi recomandabil sa trec in revista absolut tot ce este necesar functionarii, inclusiv elemente necesare interfatarii intre module si sa ma asigur ca totusi aceasta interfatarea este posibila/compatibila (uitand sa ma uit pe documentatia ecranului pentru nivelele de tensiune pentru functionare, dar din fericire norocul a fost de partea mea).

Download

calculator_arduino.zip

Jurnal

18.04.2024 - Prima iteratie de algoritm de parsare functional pe PC

19.04.2024 - Adaptarea algoritmului pentru embedded limitand memoria folosita

13.05.2024 - Piesele comandate au fost livrate

14.05.2024 - Testarea ecranului si implementarea functiei de interfatare cu tastatura

15.05.2024 - Integrarea parserului cu proiectul

22.05.2024 - Ultimele retusuri si edge caseuri

Bibliografie/Resurse

https://github.com/johnrickman/LiquidCrystal_I2C

https://en.wikipedia.org/wiki/Shunting_yard_algorithm

https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/

https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Export to PDF