Universitatea Ștefan cel Mare Suceava
Facultatea de Inginerie Electrică și Știința Calculatoarelor

Implementarea unui calculator cu display touch sensitive

Microcontrolere
Microcontrolere – arhitecturi și programare

Maciuc Simon-Gabriel
3131A

Suceava, 2023

Cuprins

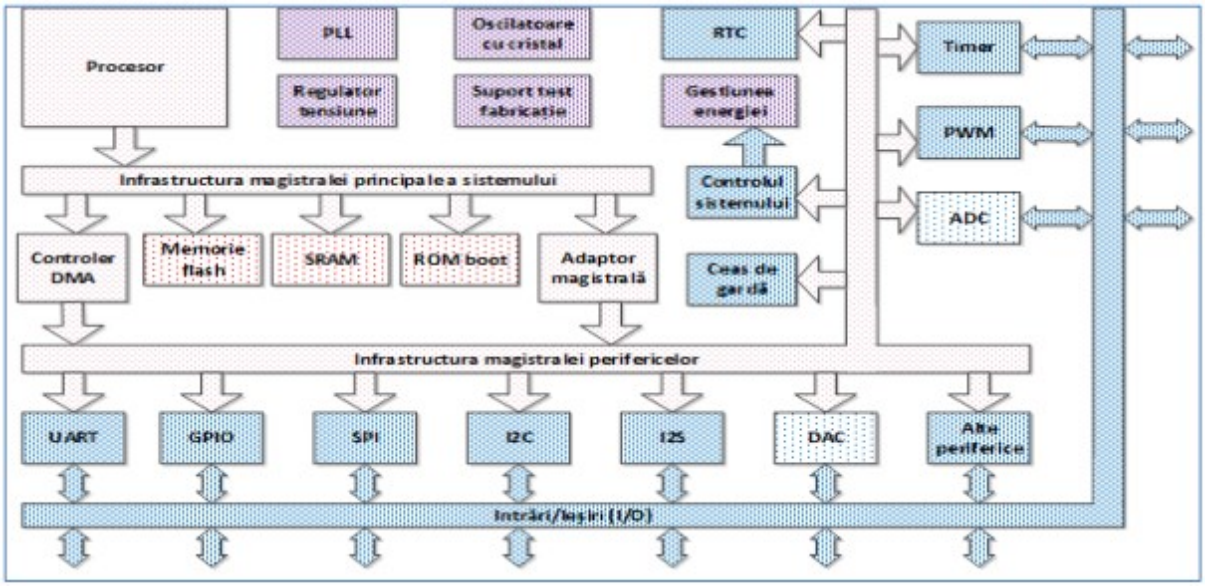
1. Noțiuni teoretice

Pentru realizarea acestui proiect s-a folosit display-ul **TFT LCD** (Thin-film-transistor liquid-crystal display) de pe microcontroler-ul **Cortex-M4 STM32F429I-DISCO**.

Un **microcontroler este un mic calculator de preț redus**, construit cu scopul de a executa sarcini specifice. Dintre beneficiile utilizării microcontrolerelor amintim **avantajul costului**, **consum redus de energie** și **totul în unul** (All-in-one). Cel mai mare avantaj al microcontrolerelor față de microprocesoare se referă la costurile de proiectare și hardware care sunt mult mai mici și pot fi păstrate la un nivel minim. Este ieftină înlocuirea unui microcontroler, în timp ce microprocesoarele sunt de zeci de ori mai scumpe. Microcontrolerele sunt în general construite folosind o tehnologie cunoscută sub numele de **Complementary**

Metal Oxide Semiconductor (CMOS). Această tehnologie este un sistem de fabricație competent, care utilizează mai puțină energie și este mai imun la pulsuri (creșteri de scurtă durată) a tensiunii de alimentare decât alte tehnici. Consumul mai redus este determinat și de arhitectura mai simplă a acestora în raport cu
microprocesoarele (memorie cache lipsă sau de capacitate mai mică, pipeline mai redus, etc.). Un microcontroler cuprinde, de obicei, o UCP (Unitate Centrală de Prelucrare), ROM (memorie nevolatilă numai în citire), RAM (memorie volatilă cu acces aleatoriu în citire/scriere) și porturi I/O, integrate pe același chip, pentru a executa o sarcină unică și dedicată. Pentru nucleele cunoscute sub denumirea **Cortex**, convenția de denumire este diferită și ușor de urmat. Există trei familii Cortex: Cortex-A, Cortex-R și Cortex-M.

Familia Cortex-M este familia cu consum de putere ultra redus, de mici dimensiuni, seria de microcontrolere. Aceasta funcționează, în general, într-un punct de performanță mai mică decât seriile A sau R, dar poate rula bine peste 100 MHz. De obicei, seria este construită sub formă de microcontrolere cu mai multe linii de intrare și de ieșire și este proiectat pentru sistemele de dimensiuni și consum redus care se bazează pe multe intrări/ieșiri digitale.



Structura generala a unui microcontroller Cortex-Mx

Liniile **STM32F429** oferă performanța nucleului Cortex-M4 (cu unitate în virgulă mobilă) care rulează la 180 MHz, atingând în același timp un consum de energie statică mai scăzut (mod Stop). La 180 MHz, STM32F429 oferă performanță CoreMark de 225 DMIPS/608 executând din memoria Flash, cu stări de așteptare 0 datorită **acceleratorului ART** de la ST.

Noua interfață de **controler LCD-TFT** cu suport dual-layer profită de Chrom-ART Accelerator™. Acest accelerator grafic realizează crearea de conținut de două ori mai rapid decât nucleul singur. Pe lângă copierea eficientă a datelor brute 2D, funcționalitățile

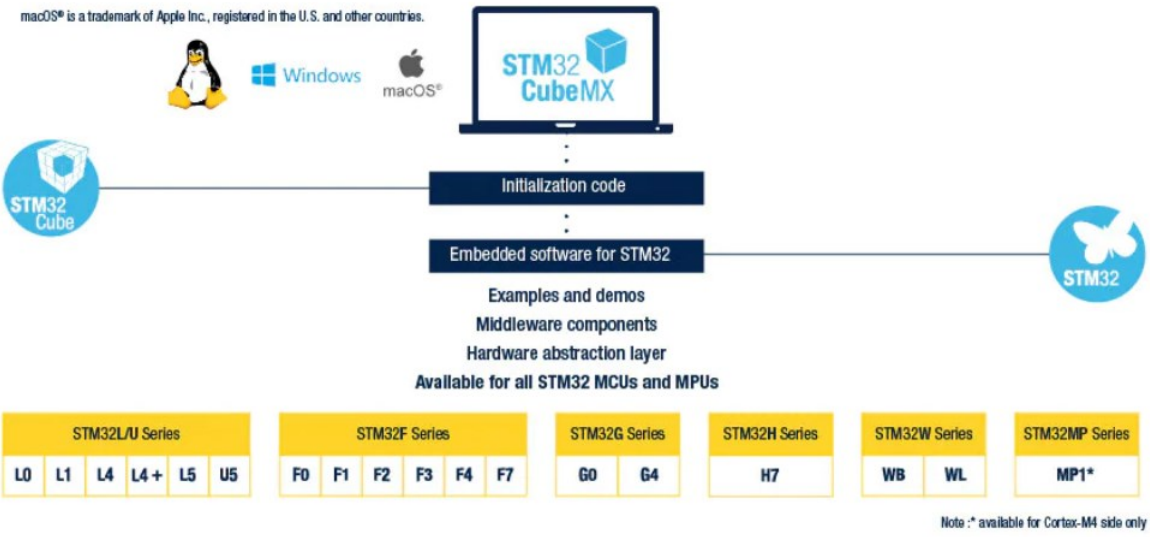
suplimentare sunt acceptate de Chrom-ART Accelerator, cum ar fi conversia formatului de imagine sau amestecarea imaginilor (amestecarea imaginii cu o anumită transparență). Acest LCD este un afișor de 2,41" cu 262 K culori, QVGA (240 x 320 pixeli), comandat direct de **STM32F429** folosind protocolul RGB și controlerul LCD ILI9341.

2. Descrierea temei alese

Tema aleasa reprezinta recreerea unui „device” pe care il folosim din reflex cand este vorba de calculat o expresie simpla matematica, si anumte calculatorul. Scopul acestui proiect a fost de a invata cum se foloseste un display care este touch sensitive si, evident, cum se implementeaza corect algoritmul din spatele unui calculator, care la prima impresie pare banal.

3. Proiectarea aplicatiei

Pentru inceput se creeaza un proiect pentru gestiunea display-ului de pe microcontroler. Acest lucru se poate face automat folosind **STM32CubeMX**.



STM32CubeMX este un instrument grafic care permite o configurare foarte ușoară a microcontrolerelor și microprocesoarelor STM32, precum și generarea codului C de inițializare corespunzător pentru nucleul Arm® Cortex®-M sau un arbore de dispozitiv Linux® parțial pentru Arm® Cortex®- Un nucleu, printr-un proces pas cu pas.

Primul pas constă în selectarea fie a unui microcontroler STMicrolectronics STM32, a unui microprocesor sau a unei platforme de dezvoltare care se potrivește cu setul necesar de periferice, fie a unui exemplu care rulează pe o anumită platformă de dezvoltare.

Al doilea pas constă în configurarea fiecărui software încorporat necesar datorită unui rezolvator de conflicte de pinout, unui ajutor de setare a arborelui ceasului, unui calculator de consum de energie și unui utilitar care configurează perifericele (cum ar fi GPIO sau USART) și stivele de middleware (cum ar fi USB sau TCP/IP).

Stivele implicite de software și middleware pot fi extinse datorită pachetelor de expansiune STM32Cube îmbunătățite. STMicrolectronics sau pachetele partenere ale STMicrolectronics pot fi descărcate direct dintr-un manager de pachete dedicat disponibil în STM32CubeMX, în timp ce celelalte pachete pot fi instalate de pe o unitate locală.

Acest proces a fost deja realizat, iar proiectul se poate gasi in documentatia laboratorului 3 de la disciplina Microcontrolere.

4. Implementarea si testarea aplicației

```
HAL_Init();
/* Configure the system clock */
SystemClock_Config();
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DMA2D_Init();
MX_FMC_Init();
MX_I2C3_Init();
MX_LTDC_Init();
MX_SPI5_Init();
MX_TIM1_Init();
MX_USART1_UART_Init();
```

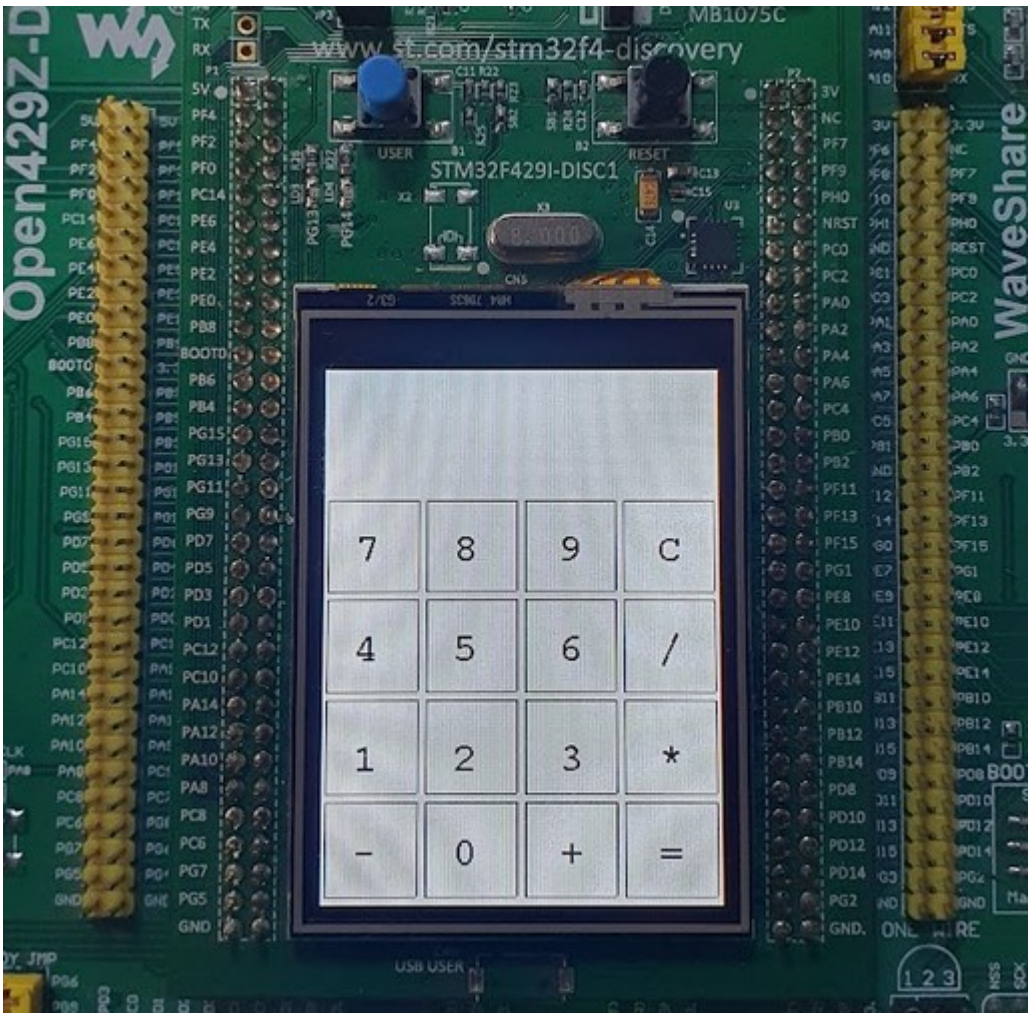
Acest cod initializeaza diferite periferice precum GPIO, DMA2D, FMC, I2C3, LTDC, SPI5, TIM1 si USART1. Aceste functii sunt oferite de bibliotecile HAL (Hardware Abstraction Layer) si sunt utilizate pentru a configura si controla diferite periferice din microcontrollerul dvs. Biblioteca HAL (Hardware Abstraction Layer) este o biblioteca oferită de STMicroelectronics pentru utilizarea cu facilitățile de microcontroller-ul STM32. Aceasta oferă o interfață de programare înalt nivel pentru accesarea facilităților hardware ale microcontroller-ului, cum ar fi perifericele de intrare/ieșire, comunicațiile și timer-ele. Aceasta face ca programarea să fie mai ușoară și mai portabilă, deoarece utilizatorul nu trebuie să se concentreze asupra detaliilor specifice ale hardware-ului și poate utiliza aceeași interfață de programare pentru diferite tipuri de microcontroller-e STM32.

In primul rand interfata calculatorului a fost creata cu urmatorul cod:

```
BSP_LCD_DisplayStringAt(-32, 100, (uint8_t*)"8", CENTER_MODE);
BSP_LCD_DisplayStringAt(-92, 100, (uint8_t*)"7", CENTER_MODE);

BSP_LCD_DisplayStringAt(-32, 160, (uint8_t*)"5", CENTER_MODE);
BSP_LCD_DisplayStringAt(-92, 160, (uint8_t*)"4", CENTER_MODE);

BSP_LCD_DisplayStringAt(-32, 225, (uint8_t*)"2", CENTER_MODE);
BSP_LCD_DisplayStringAt(-92, 225, (uint8_t*)"1", CENTER_MODE);

BSP_LCD_DisplayStringAt(-32, 280, (uint8_t*)"0", CENTER_MODE);
BSP_LCD_DisplayStringAt(-92, 280, (uint8_t*)"-", CENTER_MODE);

BSP_LCD_DisplayStringAt(32, 100, (uint8_t*)"9", CENTER_MODE);
BSP_LCD_DisplayStringAt(92, 100, (uint8_t*)"C", CENTER_MODE);

BSP_LCD_DisplayStringAt(32, 160, (uint8_t*)"6", CENTER_MODE);
BSP_LCD_DisplayStringAt(92, 160, (uint8_t*)"/", CENTER_MODE);

BSP_LCD_DisplayStringAt(32, 225, (uint8_t*)"3", CENTER_MODE);
BSP_LCD_DisplayStringAt(92, 225, (uint8_t*)"*", CENTER_MODE);

BSP_LCD_DisplayStringAt(32, 280, (uint8_t*)"+", CENTER_MODE);
BSP_LCD_DisplayStringAt(92, 280, (uint8_t*)"=", CENTER_MODE);
```

```
BSP_LCD_DrawRect(2,80,55,55);
BSP_LCD_DrawRect(62,80,55,55);
BSP_LCD_DrawRect(122,80,55,55);
BSP_LCD_DrawRect(182,80,55,55);

BSP_LCD_DrawRect(2,140,55,55);
BSP_LCD_DrawRect(62,140,55,55);
BSP_LCD_DrawRect(122,140,55,55);
BSP_LCD_DrawRect(182,140,55,55);

BSP_LCD_DrawRect(2,200,55,55);
BSP_LCD_DrawRect(62,200,55,55);
BSP_LCD_DrawRect(122,200,55,55);
BSP_LCD_DrawRect(182,200,55,55);

BSP_LCD_DrawRect(2,260,55,55);
BSP_LCD_DrawRect(62,260,55,55);
BSP_LCD_DrawRect(122,260,55,55);
BSP_LCD_DrawRect(182,260,55,55);
```

Rezultand astfel interfata urmatoare:



Programul se bazeaza pe o bucla infinta in care se tot citesc caractere, in functie de coordonatele de pe display, si se adauga intr-un string. Pe parcursul citirii se valideaza continutul, adica:

-sa nu existe doua semne de operatie unul langa altul (exceptie -), moment in care se va sterge automat ultimul semn introdus

-sa nu se imparta la 0, caz in care se afiseaza „INVALID”, se sterge cifra 0, si se continua citirea

De asemenea, se asigura citire corecta in momentul apasarii, adica sa nu se citeasca de mai multe ori daca se tine apasat pe ecran.

```
while (1)
{
    TS_StateTypeDef Prev_State = TS_State;
    BSP_TS_GetState(&TS_State);

    if (strstr(expresie_citita,"/0")!=NULL)
    {
            BSP_LCD_DisplayStringAt(0, 40, (uint8_t*)"                    ", CENTER_MODE);
            BSP_LCD_DisplayStringAt(0, 40, (uint8_t*)"INVALID", CENTER_MODE);

            HAL_Delay(1500);
            expresie_citita[strlen(expresie_citita)-1] = '\0';

            BSP_LCD_DisplayStringAt(0, 40, (uint8_t*)"                    ", CENTER_MODE);
            BSP_LCD_DisplayStringAt(0, 40, (uint8_t*)expresie_citita, CENTER_MODE);
    }
    if (strstr(expresie_citita,"/*")!=NULL || strstr(expresie_citita,"*/")!=NULL ||
            strstr(expresie_citita,"//")!=NULL || strstr(expresie_citita,"**")!=NULL ||
        strstr(expresie_citita,"+*")!=NULL || strstr(expresie_citita,"*+")!=NULL ||
    strstr(expresie_citita,"+/")!=NULL || strstr(expresie_citita,"/+")!=NULL)
    {
            expresie_citita[strlen(expresie_citita)-1] = '\0';
            refresh();
    }

    if(TS_State.TouchDetected && Prev_State.TouchDetected != TS_State.TouchDetected)
    {
        x1 = TS_State.X;
        y1 = TS_State.Y;
```

Mai sus este codul pentru validari, iar in partea de jos a pozei x1 si y1 reprezinta coordonatele locului unde s-a detectat atingere pe ecran. In functie de aceste coordonate se se face adaugarea in string a caracterelor specifice zonelor.

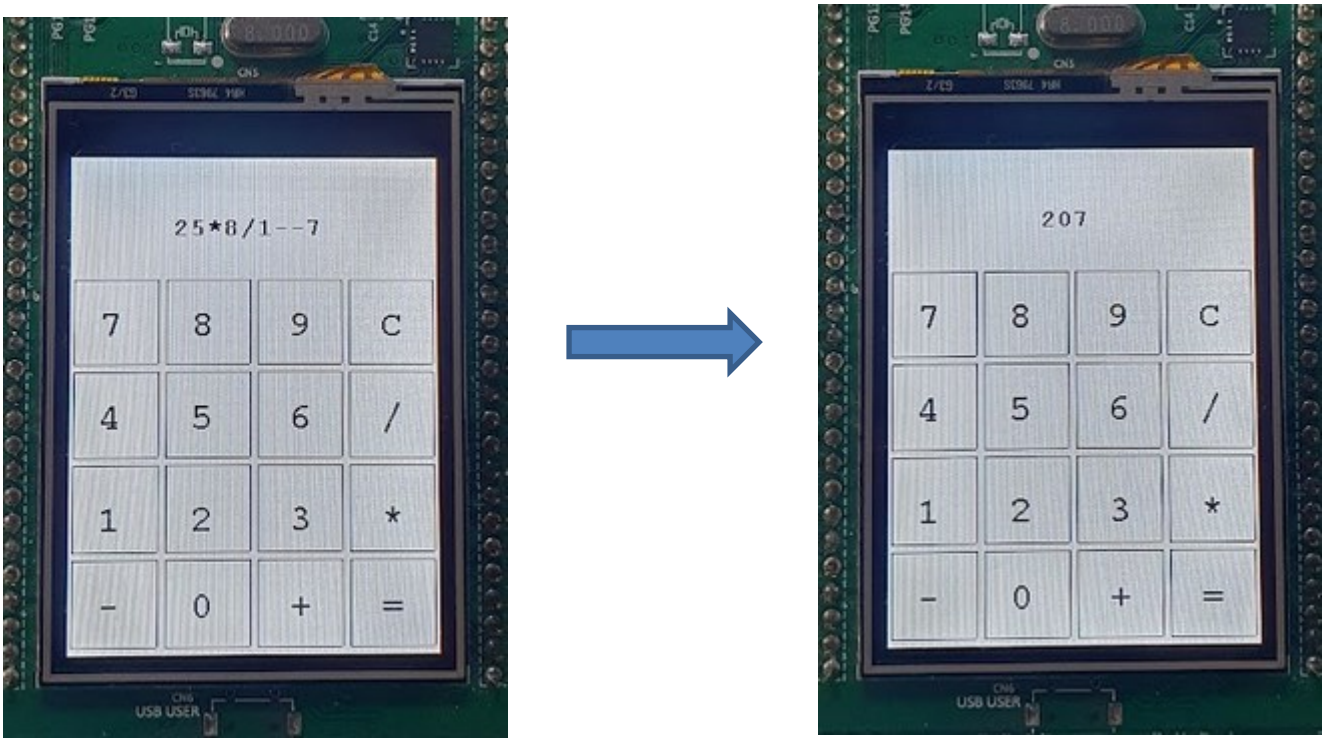
Alaturi este codul pentru calcularea expresiei matematice introduse care rezulta din apelarea functiei „expression()”.

```
char *expresie;

char peek()
{
    return *expresie;
}

char get()
{
    return *expresie++;
}

double expression();

double number()
{
    double result = get() - '0';
    while (peek() >= '0' && peek() <= '9')
    {
        result = 10*result + get() - '0';
    }
    return result;
}

double factor()
{
    if (peek() >= '0' && peek() <= '9')
        return number();
    else if (peek() == '(')
    {
        get(); // '('
        double result = expression();
        get(); // ')'
        return result;
    }
    else if (peek() == '-')
    {
        get();
        return -factor();
    }
    return 0; // error
}
```

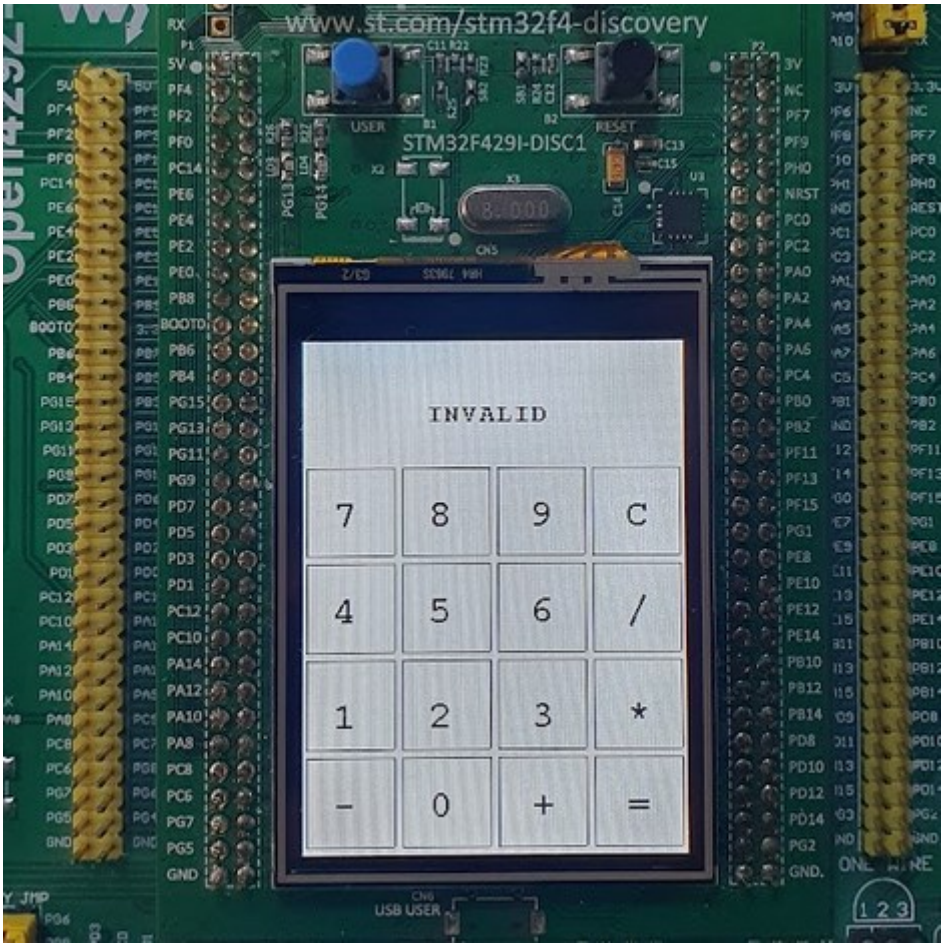
```
double term()
{
    double result = factor();
    while (peek() == '*' || peek() == '/')
        if (get() == '*')
            result *= factor();
        else
            result /= factor();
    return result;
}

double expression()
{
    double result = term();
    while (peek() == '+' || peek() == '-')
        if (get() == '+')
            result += term();
        else
            result -= term();
    return result;
}
```

Codul de mai jos reprezinta momentul cand se apasa pe caracterul „=”. Se apeleaza functia expression(), iar rezultatul se pune in variabila double „numar_calculat”. Aceasta variabila se converteste la string, in functie de rest (daca exista rest se afiseaza si restul, in caz contrat, nu).

```
else if((xl>=182 && xl<=237) && (yl>=260 && yl<=315))
{
        double numar_calculat=0;
        char string_numar[200];

        expresie = expresie_citita;
        numar_calculat=expression();

        if (numar_calculat!= (long long) numar_calculat)
            sprintf(string_numar,"%.3f",numar_calculat);
        else
        {
            sprintf(string_numar,"%.0f",numar_calculat);
            strcpy(expresie_citita,string_numar);
        }
        BSP_LCD_DisplayStringAt(0, 40, (uint8_t*)"                    ", CENTER_MODE);
        BSP_LCD_DisplayStringAt(0, 40, (uint8_t*)string_numar, CENTER_MODE);  //=
}
```



Demonstrare functionare: expresie matematica si rezultat



Demonstrare impartire la 0

5. Interpretarea rezultatelor și concluzii

In concluzie calculatorul cu display tocuh sensitive functioneaza in mod corespunzator. In cazul introducere unei expresii gresite, acesta corecteaza, iar expresie se calculeaza in mod corespunzator (chiar si in functie de semn).

Singurul minus al acestui calculator ar fi afisarea expresie pe display, deoarece spatiul acordat pentru afisare este cam limitat, iar in cazul iesirii din ecran, display-ul nu actualizeaza expresia.

Bibliografie

https://www.st.com/en/development-tools/stm32cubemx.html

https://www.st.com/en/microcontrollers-microprocessors/stm32f429-439.html

https://stackoverflow.com/questions/9329406/evaluating-arithmetic-expressions-from-string-in-c

Anexe

https://classroom.google.com/u/1/c/NTU1MTAyMTE3MTc0/m/NTU1MTA2Nzk5OTU2/details

https://classroom.google.com/u/1/c/NTU1MTAyMTE3MTc0/m/NTU1MTA2Nzk5OTU2/details