УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
Трг Доситеја Обрадовића 6, 21000 Нови Сад, Република Србија
Деканат: 021 6350-413; 021 450-810; Централа: 021 485 2000
Рачуноводство: 021 458-220; Студентска служба: 021 6350-763
Телефакс: 021 458-133; e-mail: ftndean@uns.ac.rs
ИНТЕГРИСАНИ СИСТЕМ МЕНАЏМЕНТА СЕРТИФИКОВАН ОД:

PROJEKAT IZ PREDMETA M2M ELEKTRONSKI SISTEMI

NAZIV PROJEKTA:

Testiranje GSM modema i razvoj kalkulatora

TEKST ZADATKA:

Prvi deo zadatka podrazumeva da se pomoću TFT displeja napravi kalkulator sa osnovnim funkcijama (sabiranje, oduzimanje, množenje, deljenje, koren broja i kvadrat broja). Drugi deo zadatka podrazumeva pisanje programa koji aktivira *GSM Click 3* modem i simulira pretplatu na parking. Kada se pošalje poruka „+PARKING“ broj pretplatnika se memoriše u bazu. Ako je broj u bazi, kad stigne dolazni poziv sa tog broja, otvara se rampa (uključuje se prva *LED* porta B). Za svakog sledećeg pretplatnika uključuje se naredna slobodna *LED*, sve dok ima slobodnog mesta. Ukoliko korisnik želi da se obriše iz baze, treba da pošalje *SMS* sadržine „-PARKING“ i modem treba da prepozna broj, obriše ga i korisnik više nema mogućnost parkiranja.

MENTOR PROJEKTA:

Vanr. Prof. dr Vladimir Rajs
dr Branislav Batinić

PROJEKAT IZRADILI:

Panić Milica E1 96/2022
Stefanov Miljana E1 89/2023

DATUM ODBRANE PROJEKTA:

April 2024.

Sadržaj

1. Uvod

Projektni zadatak se sastoji iz dva glavna dela:

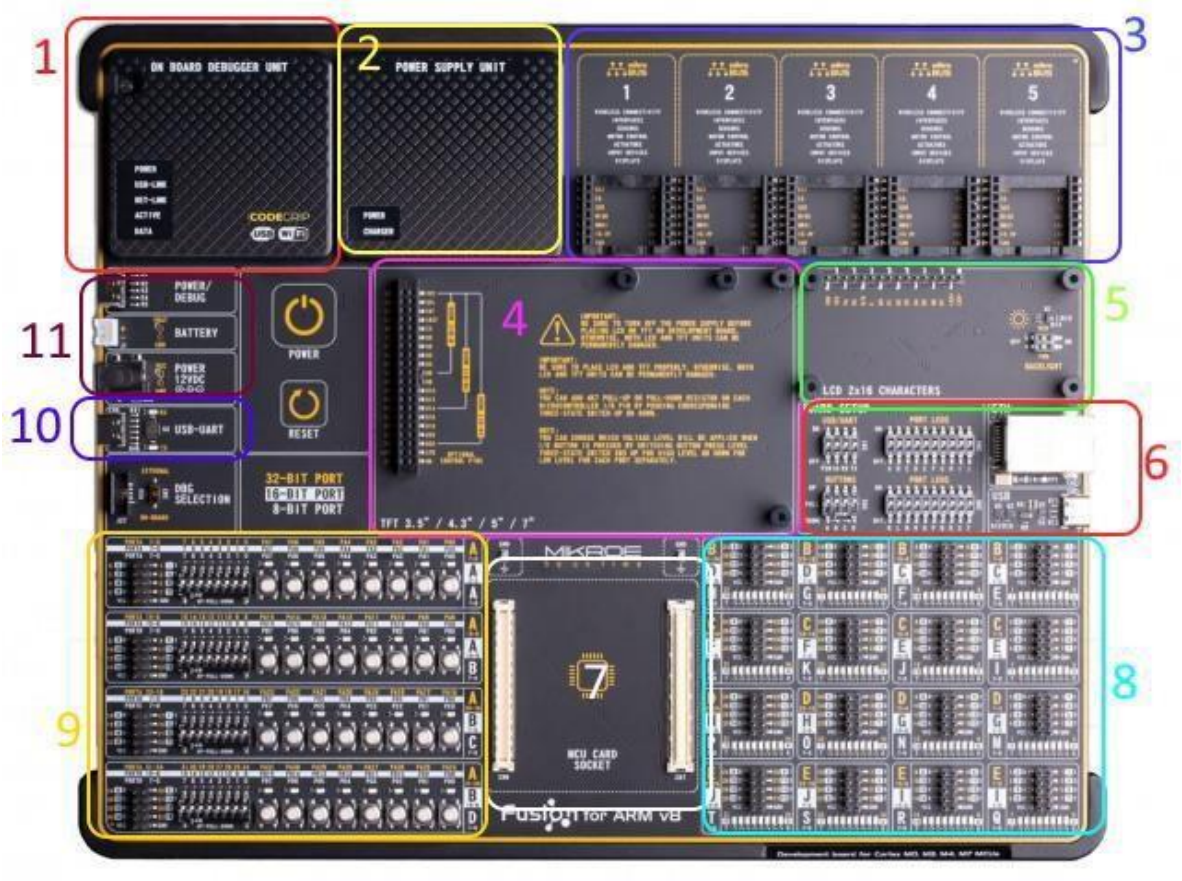
1) Prvi deo projekta podrazumeva da se pomoću 5“ TFT displeja napravi kalkulator. Funkcije koje kalkulator mora da sadrži su: sabiranje, oduzimanje, množenje, deljenje, koren broja i kvadrat broja.

2) Drugi deo projekta se odnosi na testiranje *GSM* modema. Za početak je neophodno aktivirati modem koji će čekati dolaznu poruku, zatim, kad poruka pristigne, proverava se njen sadržaj. Ukoliko je sadržaj poruke : „+PARKING“, broj se automatski memoriše u bazu mobilnih pretplatnika. Ako je broj u bazi kad stigne dolazni poziv sa tog broja, otvara se rampa (uključuje se prva *LED* na portu B). Za svakog sledećeg registrovanog pretplatnika uključuje se sledeća *LED* porta B, sve dok ima slobodnog mesta. Kad neki korisnik želi da napusti zauzeto mesto (simuliramo tako sto pritisne neki univerzalni taster), isključuje se prethodno uključena *LED* na odgovarajućem mestu. Ako korisnik želi da se obriše iz baze, treba da pošalje *SMS* sadržine „- PARKING“, i modem treba da prepozna broj, obriše ga i sada taj korisnik više nema mogućnost parkiranja, tj. neće se više reagovati na njegov poziv.

Oba dela projektnog zadatka potrebno je realizovati na razvojnom sistemu *Fusion Board for ARM v8*, kompanije Mikroelektronika, koristeći razvojno okruženje za pisanje programa *MikroC PRO for ARM*.

2. Korišćeni razvojni sistem

Za realizaciju ovog projekta korišćen je razvojni sistem *Fusion for ARM* v8 sa mikrokontrolerom *MK64FN1MOVDC12*. Korišćeni razvojni sistem na kom su obeleženi pojedini delovi prikazan je na slici 2.1.



Slika 2.1: Izgled razvojnog sistema Fusion for ARM v8

Prikazani razvojni sistem sastoji se od nekoliko većih celina, obeleženim sledećim redosledom:

1. *CodeGrip* Programator
2. Jedinica za napajanje razvojnog sistema
3. MikroBus kućišta za smeštanje *click* pločica
4. Kućište za smeštanje kapacitivnog 5“ *TFT* displeja
5. Kućište za smeštanje LCD 2x16 displeja
6. *Ethernet* i *USB* komunikacija
7. Mesto za povezivanje mikrokontrolera
8. Izvučeni pinovi mikrokontrolera
9. Tasteri i prekidači
10. Priključak za serijsku komunikaciju
11. Konektori za različite vrste napajanja

2.1.*CodeGrip* Programator

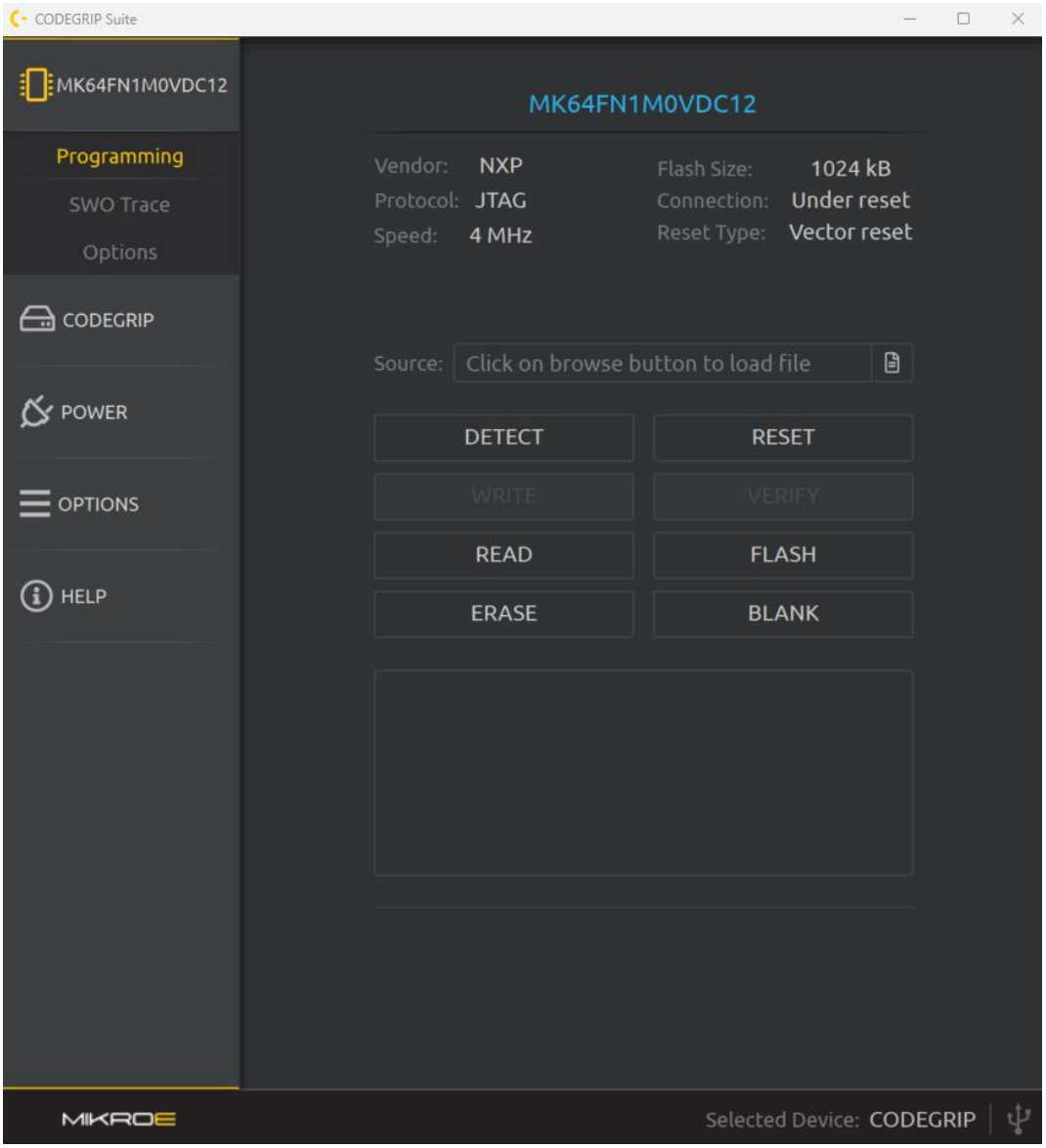
Za programiranje korišćenog mikrokontrolera odnosno za učitavanje programskogkoda u memoriju mikrokontrolera korišćen je *CodeGrip* programator (slika 2.1.1). Ovim programatorom je omogućeno programiranje mikrokontrolera putem *WiFi* mreže kao i putem *USB* kabla. U okviru programatora se nalaze indikatorske *LED* diode koje daju informacije o stanju same ploče (da li je uključena na napajanje), kao i o statusu programatora:

- ***POWER*** dioda označava da li je ploča povezana na napajanje.
- ***USB-Link*** dioda govori da je uspostavljena komunikacija putem *USB*-a.
- ***NET-Link*** dioda govori da je uspostavljena komunikacija putem *WiFi* mreže.
- ***ACTIVE*** dioda daje informaciju o stanju programatora (da li je uključen ili ne).
- ***DATA*** dioda se odnosi na sam proces programiranja mikrokontrolera i u trenutku spuštanja koda ova dioda treperi.

Slika 2.1.1. CodeGrip programator

2.2.*CodeGrip* softverski paket

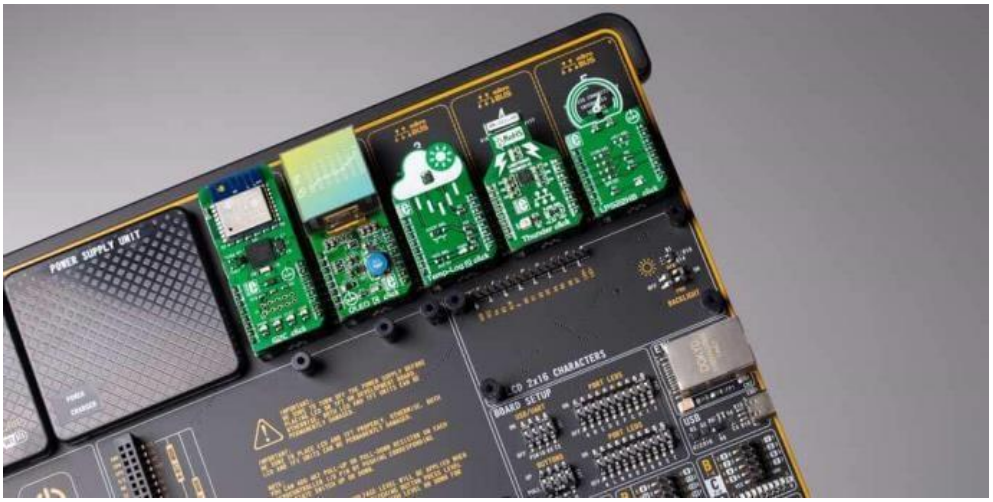
Uz dati programator, za spuštanje koda na razvojni sistem korišćen je i softverski paket *CodeGrip Suite*, takođe razvijen od strane kompanije Mikroelektronika. Radi se o moćnom paketu koji pruža potpunu kontrolu nad *Fusion Board* razvojnim sistemom. Specijalno je namenjen upravo za navedenu razvojnu ploču. Koristi se da na pametan način vrši kontrolu nad taskovima koji se bave debagovanjem i programiranjem mikrokontrolera, a takođe nudi i širok spektar opcija i podešavanja razvojnog sistema. Na slici 2.2.1. nalazi se grafički prikaz glavnog ekrana za povezivanje ploče sa ovim okruženjem, gde se vide razne mogućnosti koje *CodeGrip* softverski paket nudi.



Slika 2.2.1. Izgled CodeGrip softverskog okruženja

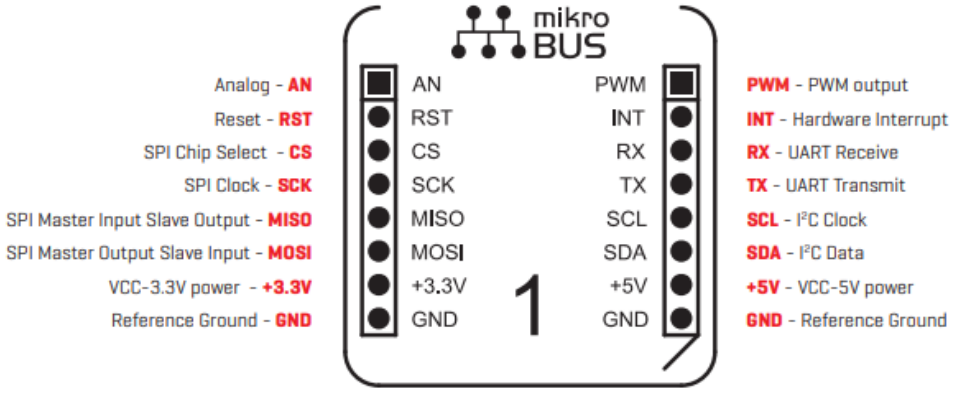
2.3. Kućišta za povezivanje *click* pločica

Razvojni sistem *Fusion for ARM* pored klasičnog povezivanja periferija pomoću kratkospojnika sadrži i mikroBUS kučišta za smeštanje *click* pločica koje se proizvode od strane firme *Mikroelektronika* (slika 2.3.1). Utičnice se prave prema *mikroBUS* standardu koji će biti opisan u nastavku.



Slika 2.3.1. Prostor za dodavanje click pločica na MikroBus kućišta

Na slici 2.3.2. je prikazan tipičan izgled jednog *mikroBus* kućišta kao i raspored pinova koji su raspoređeni u 2x8 strukturu. Kao što pokazuje slika 2.3.1, pločice koje se povezuju na *mikroBus* kućišta podržavaju različite vrste komunikacionih protokola: *UART, SPI, I2C.* Takođe, postoji i *PWM* pin za svaku pločicu kao i pin za reset *click* pločice. Ova kućišta imaju izvučene pinove za dve vrednosti napona napajanja: 3.3V i 5V. Razvojni sistem sadrži 5 ovakvih kućišta koji su numerisani brojevima 1-5 i kada se radi sa *click* pločicama bitno je u kodu definisati na koje kućište je neka pločica priključena.



Slika 2.3.2. Izgled mikroBUSTM standarda za utičnice i click pločice

2.4.Korišćen mikrokontroler

Mikrokontroler se povezuje na razvojni sistem koristeći *MCU CARD socket.* U zavisnosti od tipa mikrokontrolera postoji više vrsta ovih kućišta. Za potrebe ovog projekta korišćen je MK64FN1M0VDC12 *Kinetis* tip mikrokontrolera. Slika 2.4.1 prikazuje izgled *MCU CARD socketa* i mikrokontrolera u okviru tog *socketa*.

Slika 2.4.1. Izgled socketa i samog mikrokontrolera

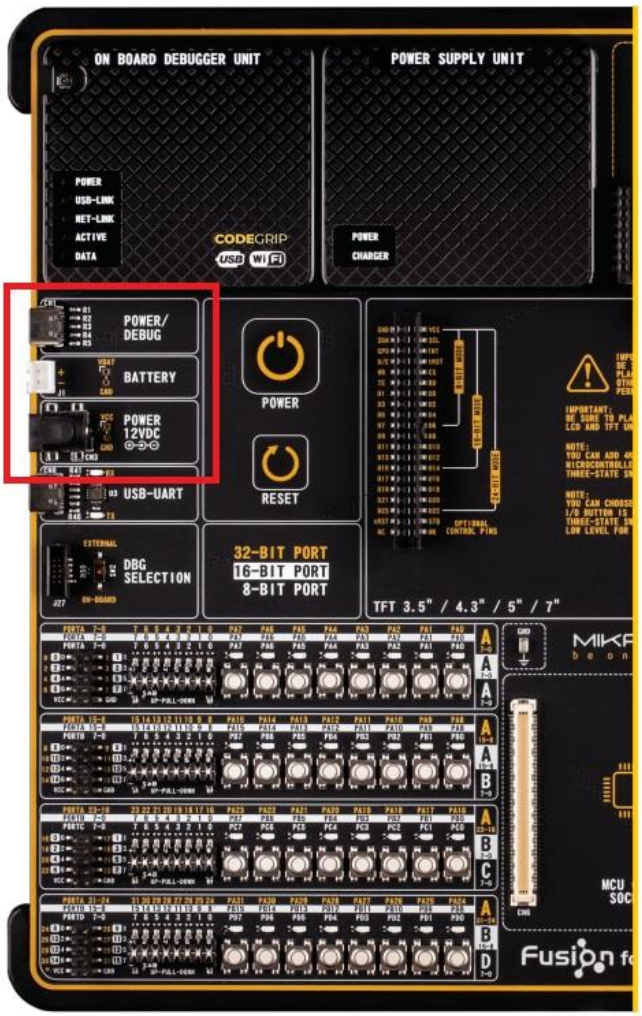
Neke od osnovnih osobina korišćenog mikrokontrolera su sledeće:

- Poseduje *ARM Cortex-M4* procesor
- 1024kB *Flash* memorije
- Ima 121 pin
- 262kB *RAM* (eng *Random Access Memory)*
- Radni napon od 3.3V
- 6 *UART* modula
- 3 *SPI* modula
- 3 *I2C* modula

2.5.Napajanje razvojnog sistema *Fusion for ARM*

Razvojni sistem se, kao što prikazuje uokvireni deo na slici 2.5.1, moze napajati na trinačina:

- Pomoću USB-C kabla i računara
- Pomoću baterije
- Pomoću spoljašnjeg izvora napajanja 12V



Slika 2.5.1. Napajanje razvojnog sistema Fusion for ARM

3. Prvi deo projekta

Kao što je već rečeno u uvodnom delu, prvi deo projekta se odnosi na formiranje kalkulatora. U skladu sa tim, komponente koje su potrebne za uspešnu realizaciju prvog dela projekta su:

- Razvojni sistem *Fusion for ARM* v8
- *TFT* 5'' kapacitivni displej

U nastavku će se detaljno opisati *TFT* displej, nakon čega se prelazi na opis programskog koda koji se koristi za zadatak.

3.1. *TFT* 5'' kapacitivni displej

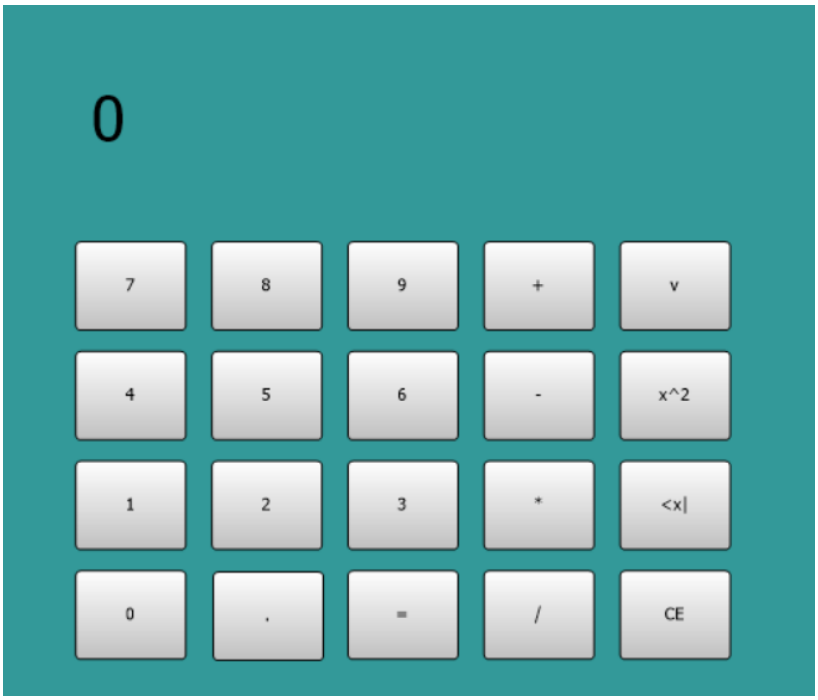
Kako bi se prikazivao izgled kalkulatora, koristi se *TFT* 5" kapacitivni displej (slika 3.1.1). Osim za prikaz, ekran koristimo i za izvršavanje funkcionalnosti. Obzirom da ovaj ekran ima opciju detektovanja pritiska (*touch scrren*), omogućeno je korišćenje kalkulatora pritiskom na određene *button*-e.

Slika 3.1.1. Kapacitivni TFT 5'' displej

Povezivanje ekrana na razvojni sistem je ostvareno pomoću konektora sa 2x20 pinova. Ovaj konektor je standarizovan i na razvojnom sistemu već se nalazi odgovarajući priključak u koji je potrebno postaviti ektran. Za reprezentaciju boja ekran koristi 24 bita čime je obezbeđeno 224 (približno 16,77 miliona) različitih boja. Na ekranu je ugrađen grafički kontroler *SSD1963*. Rezolucija ekrana je 800 x 480 piksela. Kontrast, saturacija i osvetljenje ekrana su programabilni. Za komunikaciju sa mikrokontrolerom na razvojnom sistemu koristi se *I2C* komunikacija. Za napajanje ekrana se koristi napon od 3.3V koji je već dostupan na samom razvojnom sistemu i nije potrebno eksterno napajanje.

3.2.*Softverska implementacija prvog dela projekta*

Vizuelni prikaz ekrana se nalazi na slici 3.2.1. Za prikaz, korišćene su samo dve komponente: *label* za prikaz trenutnog računa i *buttons* za svaki taster kalkulatora. Pored osnovnih brojeva i operacija, nalazi se i taster sa oznakom *v* - koren broja, *<x|* - brisanje jednog karaktera i *CE* – brisanje čitavog računa.



Slika 3.2.1. Vizuelni prikaz na TFT ekranu za prvi deo projekta

U glavnom programu je potrebno dodati neophodne biblioteke (slika 3.2.2). U *Kalkulator_objects* biblioteci se nalaze svi objekti koji se mogu prikazivati na ekranu. U te objekte spadaju labele, linije, tasteri i ekran. Svi ovi objekti imaju u svojoj strukturi definisane parametre koji bliže opisuju isti, poput fonta, boje, naziva, pozicije na ekranu i slično. Ova biblioteka je neophodna radi iscrtavanja kalkulatora. Dodate su i četiri standardne biblioteke : *string, stdint, stdlib, stddef*. Osim ovih biblioteka, potrebno je uključiti i biblioteku *math.h* koja omogućava sve neophodne matematičke operacije.

```
#include "Kalkulator_objects.h"
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <stddef.h>
#include <math.h>
```

Slika 3.2.2. Biblioteke koje je potrebno uključiti

Na slici 3.2.3. su izlistane sve promenljive koje su korišćenje i njihove inicijalne vrednosti, zajedno sa opisima čemu služe.

```
char labell[50] = {0}; //string koji pokazuje trenutni racun kalkulatora
int duzina = 0; //duzina stringa label
int i; //brojacka promenljiva
float rezultat=0.0; //rezultat racuna koji se ispisuje na ekranu
int zakljucano = 0; //fleg koji se koristi kod ispisa
float result =0.0;  //cuvanje rezultat
float operand ; //za cuvanje trenutnog broja koji se obradjuje u izrazu
```

Slika 3.2.3. Promenljve korištene u projektu

Glavna funkcija *calculate()* je prikazana na slici 3.2.4. Prolazi se kroz svaki karakter koji je dat kao naš prosleđen izraz i primenjuju se odgovarajuće operacije. Lokalne promenljive koje su definisane nam pomažu tokom obrađivanja izraza kako bi dobili konačan rezultat na kraju.

```
float calculate(char *expression) {
    float result = 0;
    float current_num = 0.0;
    char last_operator = '+';
    int decimal_flag = 0;
    float decimal_multiplier = 0.1;
    float intermediate_result = 0.0;

    while (*expression) {
        if (isdigit(*expression)) {
            if (decimal_flag) {
                current_num += (*expression - '0') * decimal_multiplier;
                decimal_multiplier *= 0.1;
            } else {
                current_num = current_num * 10.0 + (*expression - '0');
            }
        } else if (*expression == '.') {
            decimal_flag = 1;
        } else if (*expression == '+' || *expression == '-' || *expression == '*' || *expression == '/') {
            switch (last_operator) {
                case '+':
                    intermediate_result += current_num;
                    break;
                case '-':
                    intermediate_result -= current_num;
                    break;
                case '*':
                    intermediate_result *= current_num;
                    break;
                case '/':
                    if (current_num != 0) {
                        intermediate_result /= current_num;
                    } else {
                        intermediate_result = 0;
                    }
                    break;
            }
            last_operator = *expression;
            current_num = 0;
            decimal_flag = 0;
            decimal_multiplier = 0.1;
        } else if (*expression == 'r') { // Square root
            intermediate_result += current_num;
            intermediate_result = sqrt(intermediate_result);
            current_num = 0;
        } else if (*expression == 's') { // Square
            intermediate_result += current_num;
            intermediate_result = intermediate_result * intermediate_result;
            current_num = 0;
        }
        expression++;
    }
```

Slika 3.2.4. Deo glavne funkcije calculate() – prolaženje kroz izraz

Na slici 3.2.5. prikazan je nastavak funkcije *jednako()* u kojoj provarava da li zaostalih operacija u izrazu, ako ima još jednom prolazi kroz ceo izraz i povećava vrednost *intermediate_result*. Nakon što se završi ovaj deo, data funkcija vraća nam vrednost promenljive intermediate_result.

```
    switch (last_operator) {
        case '+':
            intermediate_result += current_num;
            break;
        case '-':
            intermediate_result -= current_num;
            break;
        case '*':
            intermediate_result *= current_num;
            break;
        case '/':
            if (current_num != 0) {
                intermediate_result /= current_num;
            } else {
                intermediate_result = 0;
            }
            break;
    }

    return intermediate_result;
}
```

Slika 3.2.5. Nastavak glavne funkcije calculate() – provara da li ima još operacija

Ispis rezultata računanja prikazan je na slici 3.2.6. Na početku izračuvamamo kolika je vrednost našeg izraza. Posle se nalazi *if* uslov gde se proverava stanje promenljive zaključano. Ova promenljiva predstavlja *flag* koji zaključava unos broja na već dobijeni rezultat prikazan na kalkulatoru, dok je unos operatora omogućen. Sledeći if uslov proverava da li je rezultat ceo broj. Ukoliko je to slučaj, rezultat se na displeju prikazuje kao ceo broj, odnosno tipa integer. U slučaju da rezultat nije ceo broj, izvršiće se konverzija rezultata u tip sa pokretnim zarezom, float. Formatni specifikator %g omogućava da se broj decimalnih mesta prilagodi rezultatu.

```
void jednako() {

     result = calculate(label1);
    rezultat = result;

    if ((int)rezultat == rezultat) {
        sprintf(Label1.Caption, "%d", (int)rezultat);
    } else {
        sprintf(Label1.Caption, "%g", rezultat);
    }
    ClearRect(2, 2, 840, 130);
    DrawLabel(&Label1);
    zakljucano = 1;
}
```

Slika 3.2.6. Deo glavne funkcije jednako() – ispis rezultata

Na kraju, potrebno je i spomenuti main funkciju prikazanu na slici 3.2.7, koja se nalazi unutar fajla Kalkulator_main.c. Unutar main rutine, pokreće se *Touch Panel* funkcijom *Start_TP* u kojoj se vrše potrebne inicijalizacije ekrana. Unutar *while* petlje se proverava da li je detektovan pritisak na ekranu funkcijom *Check_TP().*

```
void main() {

  Start_TP();

  while (1) {
    Check_TP();
  }
}
```

Slika 3.2.7. Promenjljve korištene u projektu

4. DRUGI DEO PROJEKTA

Nakon što je prvi deo projekta uspešno realizovan, prelazi se na drugi deo projekta koji podrazumeva da se testira *GSM* modul. Radi se o sledećem – koristeći *GSM Click* 3 pločicu kompanije *Mikroelektronika*, potrebno je napisati program koji će prvobitno aktivirati modul, zatim osposobiti pristizanje poruka na modul, nakon toga će omogućiti čitanje pristigle poruke. Kada se pošalje poruka „+PARKING“ broj pretplatnika se memoriše u bazu. Ako je broj u bazi, kad stigne dolazni poziv sa tog broja, otvara se rampa (uključuje se prva *LED* porta B). Za svakog sledećeg pretplatnika uključuje se naredna slobodna *LED*, sve dok ima slobodnog mesta. Ukoliko korisnik želi da se obriše iz baze, treba da pošalje *SMS* sadržine „-PARKING“, i modem treba da prepozna broj, obriše ga i korisnik više nema mogućnost parkiranja

Da bi se opisani postupak izvršio, neophodno je posedovati sledeće komponente (celine):

- *GSM Click* 3 pločica
- *SIM* kartica koju će koristiti *GSM* modul
- Mobilni telefon sa svojom *SIM* karticom
- *OLED* 0.96’’ ekran

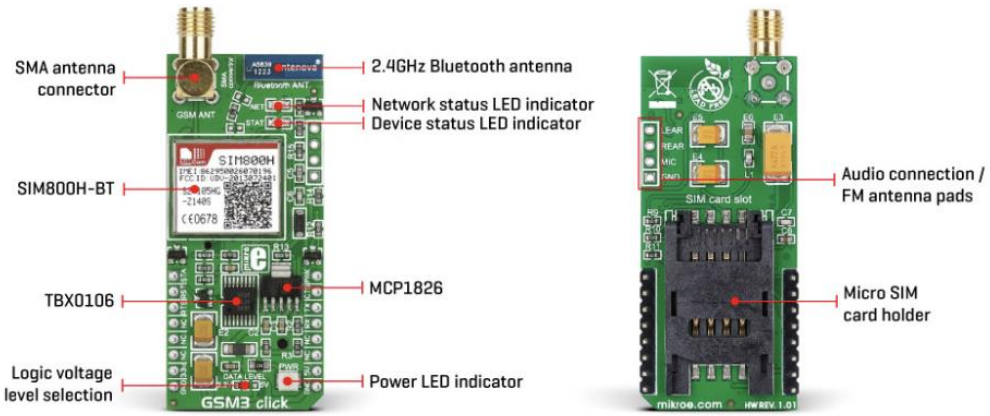
U nastavku će se opisati samo funkcionalost GSM Click 3 pločice i OLED 0.96’’ displeja, obzirom da svaki mobilni telefon sa SIM karticom na isti način obavlja funkcionalnost. Međutim, važno je napomenuti da prilikom ubacivanja SIM kartice u modem, potrebno je isključiti *PIN* kod ukoliko je aktiviran na kartici.

4.1 GSM 3 click pločica

GSM Click 3 je sveobuhvatno rešenje za komunikaciju putem *GSM* mobilne mreže. Ključni deo ove pločice je SIM800H-BT, četvorofrekventni 2G GSM/GPRS modul, koji podržava sve četiri glavne GSM frekvencije - 850MHz, 900MHz, 1800MHz i 1900MHz. Modul je usklađen sa *GSM* fazom 2/2+, pružajući širok spektar opcija za mobilnu umrežavanje i komunikaciju.

Navedeni modul nudi indikaciju statusa mreže, detekciju ometanja i ugrađene internet protokole kao što su TCP/IP, UDP, FTP, PPP, HTTP, e-pošta, MMS i mnoge druge. Takođe, sadrži i napredne glasovne/audio funkcije, uključujući FM radio interfejs. Brzina prenosa podataka modula iznosi prosečno 85.6 kbps. Posebno značajna karakteristika je podrška za Bluetooth 3.0+ EDR protokol, omogućavajući prenos veće količine podataka između uređaja. Na slici 4.1.1. prikazan je izgled opisanog modula.

Obzirom da se radi o *GSM*/*GPRS* modulu, on može da se koristi širom sveta. Širok spektar mogućih funkcionalnosti, upakovanih u kompaktan modul tako da se pokreću preko standardnih AT komandi (AT potiče od engleske reči *Attention*) čini modul izuzetno korisnim u raznim *M2M* (eng. *Machine to Machine*) aplikacijama, uključujući mobilne internet terminale, automatsko očitavanje brojila, aplikacije za nadzor i bezbednost, cene puteva, praćenje imovine kao i mnoge druge aplikacije koje koristite mobilnu mrežu.



Slika 4.1.1. Izgled GSM 3 click pločice – prednja i zadnja strana

SIM800H-BT modul mora da se napaja stabilnim naponom napajanja. Da bi modul pravilno funkcionisao, potrebno mu je napajanje od oko 4V koje je dovedeno sa 5V *mikroBus* utičnice i prethodno snižen na oko 4V pomoću 1A MCP1826 LDO regulatora. Iako je SIM800H uređaj male snage, moduli celularne mreže su generalno poznati po velikoj potrošnji energije pa je stoga morao da se koristi 1A LDO.

UART interfejs podržava standardne brzine prenosa od 1200bps do 115,2 kbps i automatska detekcija *baud rate* je podešena tako da se ne mora dodatno podešavati i usklađivati *baud rate* između modema i upravljačkog uređaja.

SIM800H-BT je dizajniran kao tradicionalni *DCE* (*Data Communication Equipment*) uređaj, nudeći sve *UART* pinove, uključujući i pinove za kontrolu toka *CTS* (*Clear to Send)* i *RTD* (*Request to Send*), pored *RX* i *TX* pinova.

Crvena *LED* dioda označena kao *NET* se koristi za označavanje statusa mreže. Status mreže je prikazan ovom diodom koristeći sledeći obrazac:

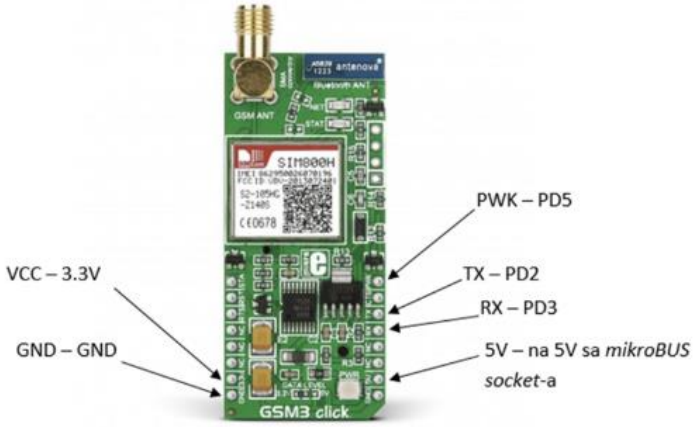
- Uključena *LED* – uređaj se napaja.
- Uključena 64ms a zatim isključena 800ms – uređaj traži mrežu.
- Uključena 64ms a zatim isključena 3s – uređaj registrovan na mrežu.
- Uključena 64ms a zatim isključena 300ms – komunikacija je ostvarena.

Žuta *LED* dioda obeležena kao *STAT*, koristi se za indikaciju statusa uređaja, da li je uređaj u funkciji i da signalizira završetak interne inicijalizacije modula, odnosno da je modul spreman za rad.

PWRKEY pin je rutiran na *mikroBUS PWM (pulse-width modulation*) pin i koristi se za ručno uključivanje/isključivanje klik pločice. Držanje ovog pina na logičkoj nuli najmanje 1 s, dovešće do promene power stanja SIM800H modula.

Pored *PWRKEY* pina, ovaj modul takođe ima i *RESET* pin i koristi se u slučaju kada *PWRKEY* ili odgovarajuća AT komanda za isključivanje ne funkcioniše. Logička nula na *RESET* pinu će resetovati uređaj. Ovaj pin je povezan sa *mikroBUS RST* pinom. Na slici 4.1.2. su prikazani pinovi od interesa odnosno pinovi koje GSM3 *click* pločica koristi za inicijalizaciju i komunikaciju sa mikrokontrolerom.

Komunikacija između mikrokontrolera i GSM3 *click* pločice se vrši pomoću UART komunikacije. Korišćeni mikrokontroler sadrži više UART modula a za potrebe ovog dela projekta je iskorišćen UART2 modul čiji su RX i TX pinovi na portu D (pinovi PD2 i P3), kao što pokazuje slika 4.1.2. Pin *PWK* se koristi za dovođenje start impulsa kojim se omogućava početak rada *GSM* modula i njegova ispravna inicijalizacija. Pored napona napajanja od 3.3V, potrebno je dovesti 5V sa mikroBus utičnice kako bi uređaj ispravno funkcionisao.

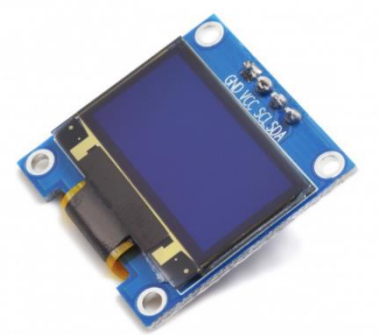


Slika 4.1.2. Pinovi koji se koriste u okviru drugog dela projekta

Neophodno je napomenuti da je pre povezivanja *GSM* modula sa razvojnim sistemom neophodno ubaciti važeću *SIM* karticu u slot predviđen za to. Na zadnjoj strani pločice, primećuje se crni plastični deo gde je za početak neophodno otvoriti ga, a zatim ubaciti *SIM* karticu tako da, kada se postavi, „odsečeni ćošak“ kartice bude u donjem desnom uglu. Zatim se kartica osigurava dodatnim metalom koji se prevuče preko nje i modul je tada spreman za upotrebu. Bitno je napomenuti da ovaj modul podržava isključivo najveće dimenzije *SIM* kartice (standardna veličina), te ako korisnik inače koristi karticu veličine mikro ili nano, neophodno je da nabavi adapter na najveću moguću veličinu kartice.

4.2 OLED 0.96'' displej

Kako bi se pratilo stanje u kom se *GSM* modul trenutno nalazi, korišćen je *OLED* 0.96'' displej rezolucije 128x64 piksela. Na displeju se ispisuju sve neophodne informacije u vidu stringa kako bi se moglo prepoznati trenutno stanje modula. Komunikacija između displeja i mikrokontrolera podrazumeva *I2C* komunikaciju koja je uspostavljena povezivanjem displeja i porta E razvojnog sistema putem kratkospojnika. Korišteni pinovi su PE0 i PE1. *OLED* 0.96'' displej je prikazan na slici 4.2.1.



Slika 4.2.1. OLED 0.96'' displej

OLED displej se povezuje po sledećem pravilu:

- *SDA* – PE0
- *SCK* – PE1
- *VCC* – 3.3V
- *GND* – GND

4.3 Softverska implementacija drugog dela projekta

Fokus projekta je baziran na testiranju *GSM* modema. Stoga je neophodno uspostaviti komunikaciju mikrokontrolera razvojne ploče sa *GSM click* 3 pločicom koja se koristi. Pošto se za uspostavu komunikacije koristi *UART2* modul na razvojnom sistemu, potrebno je dodati biblioteku za komunikaciju *GSM* modema sa mikrokontrolerom, a takođe da bi korisnik mogao da ima vizuelni prikaz onoga šta se događa sa modulom dodaje se biblioteka sa potrebnim funkcijama za rad sa *OLED* ekranom. Sve dodate biblioteke prikazane su na slici 4.3.1.

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include "uart2.h"
#include "OLED_096.h"
```

Slika 4.3.1. Uključene biblioteke

Potrebno je potom definisati makroe, prikazane na slici 4.3.2. Makroi STIGAO i PONISTI se koriste kako bi se signaliziralo da li je stigla neka poruka sa UART2 modula razvojne ploče. Makro BROJ označava broj cifara broja telefona, dok makro PARKING_MESTA označava maksimalan broj parking mesta.

```
#define STIGAO 1
#define PONISTI 0

#define BROJ 13
#define PARKING_MJESTA 10
```

Slika 4.3.2. Makroi za indikaciju pristigle poruke sa UART2 *modula i za definisanje baze pretplatnika*

Na slici 4.3.3. su date promenljive koje se koriste u main.c fajlu zajedno sa njihovim opisima.

```
uint8_t stringUART=0, i=0,a=0; //U stringUART smestamo makroe STIGAO i PONISTI, dok su i,a brojacke promenljive
uint8_t rx_buffer[256];  //bafer za Rx, maximalne velicine 256 karaktera
uint8_t receive;  //promenjiva u koju cemo smestati primljni karakter iz UART5_D registra
uint8_t stanje=0; //promenljiva koja govori u kom se stanju nalazi GSM modul
char phoneNumber[13]; //niz rezervirsan za broj telefona posiljaoca +PARKING poruke
char phoneNumber_minus[13];  //niz rezervirsan za broj telefona posiljaova -PARKING poruke
char baza[PARKING_MJESTA][BROJ];   //baza sa brojevima telefona
char *br; //jedan po jedan broj koji će popunjavati phoneNumber
char *br_minus; //jedan po jedan broj koji će popunjavati phoneNumber_minus
int index = -1; //flag za indikaciju zauzetosti parking mesta
int x; //brojacka promenljiva
char mob[13]; //niz rezervisan za cuvanje broja pretplatnika tokom poziva
```

Slika 4.3.3. Promenljive u okviru main.c fajla

Prekidna rutina za *UART2* prikaza je na slici 4.3.4. Postupak je takav da se u promenljivu receive smešta svaki pristigli karakter, pa se zatim niz *rx_buffer* puni smeštajući vrednost promenljive *receive* u dati niz. Ukoliko nije završeno slanje poruke, to jest ako receive ne poprimi vrednost 0x0D što je u ovom slučaju indikator za kraj poruke (eng. *Carriage Return, CR*), u tom slučaju se nastavlja popunjavanje niza *rx_buffer*. U suprotnom, *receive* se postavlja na nultu vrednost a podiže se *flag* da je poruka stigla dodeljivanjem makroa STIGAO promenljivoj *stringUART*.

```
void INT_UART2_RX() iv IVT_INT_UART2_RX_TX ics ICS_AUTO
{
  while ( ( UART2_S1 & UART_S1_RDRF_MASK) == 0 );

    (void) UART2_S1;
    receive = UART2_D;

    if(receive!=0x0D)
    {
        rx_buffer[i]=receive;
        i++;
    }
    else
    {
        receive=0;
        i=0;
        stringUART=STIGAO;
    }
}
```

Slika 4.3.4. Prekidna rutina za UART2 modul

Pored prekidne rutine, dodata je i funkcija koja niz *rx_buffer* prazni i postavlja u inicijalno stanje. Funkcija je prikazana na slici 4.3.5. Napravljena je tako da se svaki element niza *rx_buffer* postavlja na *null*, što može da se interpretira i kao kraj stringa. Na taj način se osigurava da je niz *rx_buffer* postavljen u inicijalno stanje.

```
void clear_rx_buffer(){
   for(a=0;a<29;a++){
     rx_buffer[a]='\0';
   }
}
```

Slika 4.3.5 .Funkcija za pražnjenje niza rx_buffer

Unutar *main.c* fajla, u glavnoj main rutini, nalaze se osnovna podešavanja i inicijalizacije koje je potrebno izvršiti prije beskonačne *while* petlje. Na početku se definišu pinovi za *UART2* komunikaciju, PD0 i PD1 tako da budu izlazni pinovi. Zatim se poziva funkcija *UART2_Inicijalizacija()* koja će inicijalizovati komunikaciju – postaviće odgovarajući *baud rate* i aktiviraće prijemnu i predajnu stranu, a potom se dozvoljava *interrupt* za *UART2* komunikaciju.

Kako bi rad sa OLED 0.96“ bio omogućen, potrebno je podesiti pinove PE0 i PE1 kao izlazne pinove kako bi se *I2C* komunikacija uspostavila. Funkcija *I2C1_Init_Advanced()* će izvršiti inicijalizaciju *I2C1* modula.

Pošto su pinovi za *OLED* definisani, sledeći korak je inicijalizacija *OLED* displeja. Posle kraće pauze, displej se prazni od bilo kakvog prethodnog sadržaja i spreman je za upotrebu. Dodatni korak pre uključivanja *GSM* modema je da se reset pinom modema upravlja kao izlaznim pinom kontrolera, to jest modem se uključuje promenom stanja na njegovom *PWRKEY* pinu. Ovaj pin modema je povezan na PD5 pin na razvojnoj ploči, te se PD5 definiše kao izlazni pin. Funkcijom *GSM_PowerON*(), modem se uključuje.

Na parkingu koje simuliramo unutar projekta, svako parking mesto je predstavljeno jednom *LED* koja se nalazi na portu B. Pošto je potrebno da upravljano *LED*, odnosno da ih isključujemo i uključujemo po potrebi, potrebno je definisati deset dioda porta B kao izlazne pinove. Na početku sva parking mesta trebaju da budu slobodna, stoga setujemo sve bite registra *GPIOB_PCOR*, čime ćemo zapravo da postavimo sve bite porta B na nulu, čineći *LED* iskjučenim.

Unutar niza baza, smeštaju se brojevi telefona svih pretplatnika parkinga. Stoga, potrebno je da se baza inicijalizuje kao prazna.

```
void main()
  {

       GPIO_Digital_Output(&PTD_PDOR, _GPIO_PINMASK_3);   //Tx pin za UART5 izlazni
       GPIO_Digital_Input(&PTD_PDIR, _GPIO_PINMASK_2);  //Rx pin za UART5 ulazni

       UART2_Inicijalizacija();
       delay_ms(100);                 /
       NVIC_IntEnable(IVT_INT_UART2_RX_TX);    //omoguci interrupt za UART5 Rx Tx

       GPIO_Digital_Output(&PTE_PDOR, _GPIO_PINMASK_0);   //i2c1 SDA
       GPIO_Digital_Output(&PTE_PDOR, _GPIO_PINMASK_1);   //i2c1 SCK

       I2C1_Init_Advanced(4000000, &_GPIO_Module_I2C1_PE1_0);

       Init_OLED();
       delay_ms(100);
       oledClear();

       GPIO_Digital_Output(&PTD_PDOR, _GPIO_PINMASK_5);   //reset pin modema, kao izlazni pinom kontrolera
       GSM_PowerON();     //ukljucivanje modema

       GPIO_Digital_Output(&PTB_PDOR, _GPIO_PINMASK_ALL) ;

       GPIOB_PCOR |= 0xffffffff;  //Sva parking mjesta su slobodna na pocetku

       for ( i = 0; i < PARKING_MJESTA; i++) {
        strcpy(baza[i], ""); // Inicijalno sva mjesta u bazi su prazna
       }
```

Slika 4.3.6. Inicijalizacija unutar main rutine

U *while(1)* petlji dešava se sva ostala funkcionalnost ovog dela projekta. Program je napisan kao mašina stanja, tako da se redosled izvršavanja zna i ciklično se ponavlja. Na osnovu promenljive *stanje*, realizovani sistem se može naći u pet različitih stanja. U nastavku će biti opisano svako od stanja.

Prilikom definisanja promenljive *stanje*, postavljeno je da ona ima inicijalnu vrednost 0. Na početku kako bi se GSM modul aktivirao, mikrokontroler putem *UART2* modula šalje poruku GSM modulu u vidu stringa „*AT*“, te ako je za početak stigla poruka sa modula i ako je ta poruka sadržaja „*OK*“, može se preći na naredno stanje modema, a to je aktivno stanje. Poruka „*Ukljucen je modem*“ se ispisuje u nultom stanju ako su uslovi zadovoljeni. Zatim se promenljivoj stringUART dodeljuje vrednost PONISTI i niz *rx_buffer* se prazni i prelazi se na naredno, prvo stanje. Nulto stanje prikazano je na slici 4.3.7.

```
switch(stanje){
  case 0:
       Uart2_WriteString("AT\r");
       Delay_ms(500);
       if(stringUART==STIGAO){
          oledClear();
          oledGotoYX(0,0);
          if(strstr(rx_buffer, "OK")){
            oledPrint("Ukljucen je");
            oledGotoYX(1,0);
            oledPrint("modem");
            Delay_ms(500);
            stanje = 1;
          }
          clear_rx_buffer();
          stringUART=PONISTI;
       }
```

4.3.7. Nulto stanje modema

Sledeće stanje predstavlja aktivno stanje modema i u tom slučaju promenljiva *stanje* ima vrednost 1. Iako je GSM modul aktivan, to ne znači i da je povezan na mrežu. Upravo ovo stanje opisuje registrovanje uređaja na mrežu. Registrovanje se vrši komandom „*AT+CREG=1*“ i ako je odgovor modula potvrdan („*OK*“), ispisuje se poruka: „Prijavljen na mrezu“ i prelazi se u naredno, drugo stanje. Prvo stanje modema je prikazano na slici 4.3.8.

```
case 1:
     clear_rx_buffer();
     Uart2_WriteString("AT+CREG=1\r");
     Delay_ms(500);
     if(stringUART == STIGAO){
       oledClear();
       oledGotoYX(0,0);
       if(strstr(rx_buffer, "OK")){
         oledPrint("Prijavljen na");
         oledGotoYX(1,0);
         oledPrint("mrezu");
         Delay_ms(1000);
         stanje = 2;
       }
       clear_rx_buffer();
       stringUART=PONISTI;
     }
```

4.3.8. Prvo stanje modema

Nakon što se uređaj registrovao na mrežu, prelazi se na glavnu funkcionalnost datog zadatka, a to je čekanje poruka. Da bi modul uopšte mogao da čita poruke, mora mu se proslediti odgovarajuća komanda, što je u ovom slučaju „*AT+CMGF=1*“ U ovom slučaju je modem u modu za poruke i može njima i da manipuliše. Da se ne bi desilo da modul pokuša da zove nešto što je zaostalo u njegovoj memoriji, prvo se briše sve iz njegove memorije pomoću komande „*AT+CMGD=X*“, gde *X* predstavlja broj između 0 i 4 i ispisivanje komandi tako da na kraju stoje svi brojevi između 0 i 4 označava brisanje svih poruka iz sistemske memorije GSM modula. Pošto su sve poruke obrisane, može se preći u naredno, treće stanje. Na slici 4.3.9. prikazano je drugo stanje modema.

```
case 2:

      Uart2_WriteString("AT+CMGF=1\r");
      Delay_ms(100);
      Uart2_WriteString("AT+CMGD=0\r");
      Delay_ms(100);
      Uart2_WriteString("AT+CMGD=1\r");
      Delay_ms(100);
      Uart2_WriteString("AT+CMGD=2\r");
      Delay_ms(100);
      Uart2_WriteString("AT+CMGD=3\r");
      Delay_ms(100);
      Uart2_WriteString("AT+CMGD=4\r");
      Delay_ms(100);

 if (strstr(rx_buffer, "OK")){
   clear_rx_buffer();
   stanje=3;
   }
break;
```

4.3.9. Drugo stanje modema

U stanju kada promenljiva stanje ima vrednost 3 vrši se provera pristiglih poruka. Prvobitno se isključuje eho komandom „*ATE0*“ i ispisuje se odgovarajuća poruka na *OLED* 0.96’’ ekranu da modem očekuje poruku. Potom, unutar *while* petlje, vrši se neprekidno slanje poruke o čekanju, sve dok korisnik ne pošalje poruku sadržine: „+PARKING“ ili „-PARKING“. Komanda „*AT+CNMI=1,2,0,0,0*“ služi za odgovarajuće slanje pristigle poruke ka mikrokontroleru radi njene dalje obrade. Ukoliko je poruka odgovarajućeg sadržaja, uslov za izlazak iz *while* petlje je zadovoljen i prelazi se na dalje izvršavanje koda koje je unutar stanja 3. Stanje broj tri je prikazano na slici 4.3.10.

```
case 3:
    Uart2_WriteString("ATE0\r");
    Delay_ms(100);
    clear_rx_buffer();
    Delay_ms(100);
    oledClear();
    oledGotoYX(0,0);
    oledPrint("Cekam poruku");
    Delay_ms(500);
       while (!(strstr(rx_buffer, "+PARKING") || strstr(rx_buffer, "-PARKING"))){
       clear_rx_buffer();
       Uart2_WriteString("AT+CNMI=1,2,0,0,0\r");
       Delay_ms(500);
       oledClear();
       oledGotoYX(0,0);
       oledPrint("Cekam poruku");
       oledGotoYX(1,0);
       Delay_ms(500);
       }
```

4.3.10. Treće stanje modema

Na slici 4.3.11.prikazan je deo koda koji se izvršava u stanju tri u slučaju da je pristigla poruka sadržaja: „+PARKING“. Kad je očitana poruka, očitava se broj telefona, gde se traži svaki broj unutar niza *rx_buffer* i to pomoću funkcije *strstr()* gde se kao prvi parametar prosleđuje niz odakle tražimo podstring, a drugi parametar je početak podstringa koji se traži. Kada je formiran broj telefona unutar stringa p*honeNumber*, vrši se provera postojanja datog broja u bazi, uz pomoć funkcije *proveriBrojUBazi.* Shodno ishodu, ispisuje se odgovarajuća poruka na displeju.

U slučaju da se broj ne nalazi u bazi i da postoji slobodno mesto u bazi pretplatnika, broj se dodaje pomoću funkcije *dodajBrojUBazu* i prelazi se na sledeće, četvrto stanje. U slučaju da je baza puna ili da se broj već nalazi u bazi pretplatnika, prelazi se na početak stanja tri, na čekanje nove poruke.

```
if((strstr(rx_buffer,"+PARKING"))){
  br=strstr(rx_buffer, "381");
  for(i=0; i<12; i++){
      phoneNumber[i]=*br;
      br++;
  }
  if (provjeriBrojUBazi(phoneNumber)) {
    oledClear();
    oledGotoYX(0,0) ;
    oledPrint("Broj u bazi.");
    Delay_ms(1000);
    stanje = 3;
    break;
  }
  else {
    oledClear();
    oledGotoYX(0,0);
    oledPrint("Broj nije u");
    oledGotoYX(1,0);
    oledPrint("bazi");
    Delay_ms(1000);
    if(dodajBrojUBazu(phoneNumber)) {}
    else{
     oledClear();
     oledGotoYX(0,0);
     oledPrint("Nema prostora");
     oledGotoYX(1,0) ;
     oledPrint("u bazi.");
     Delay_ms(1000);
     stanje = 3;
     break;
     }
   }
clear_rx_buffer();
stanje=4;
break;
}
```

4.3.11. Treće stanje modema – „+PARKING“

Kod koji se izvršava u slučaju da je primljena poruka „-PARKING“ je prikazan na slici 4.3.12. Kao i u prethodnom slučaju, prvo se proverava broj telefona sa kojeg je poruka poslata i čuva se unutar stringa *phoneNumber_minus*. Zatim, dati broj se provera u bazi i ukoliko je on pronađen potrebno je izbrisati broj iz baze i isključiti *LED* na ogovarajućoj poziciji koja odgovara poziciji broja telefona iz baze. Brisanje broja ostvareno je funkcijom *obrisiBrojIzBaze*.

```
else if((strstr(rx_buffer,"-PARKING"))){
  br_minus=strstr(rx_buffer, "381");
  for(i=0; i<12; i++){
      phoneNumber_minus[i]=*br_minus;
      br_minus++;
  }
  if (provjeriBrojUBazi(phoneNumber_minus)) {
   GPIOB_PCOR |= (1<<i+8);
   obrisiBrojIzBaze(phoneNumber_minus);
   oledClear();
   oledGotoYX(0,0)   ;
   oledPrint("Broj obrisan");
   oledGotoYX(1,0)   ;
   oledPrint("iz baze");
   Delay_ms(1000);
   stanje = 3;
   break;
}
else{
  stanje = 3;
    }
}
```

4.3.12. Treće stanje modema – „-PARKING“

Poslednje stanje u kom se GSM modul može naći je stanje kada je promenljiva *stanje* jednaka 4. U ovom stanju se čeka na poziv nakon uspešnog slanja poruke sadržaja „+PARKING“. Prikazana *while* petlja će biti aktivna sve dok ne pristigne string koji sadrži niz „*381*“, kako bi se detektovao poziv. Broj telefona koji poziva se potom očitava i čuva unutar promenljive mob. Opisani deo koda je prikazan na slici 4.3.13.

```
case 4:
        clear_rx_buffer();
        while(!(br=strstr(rx_buffer, "381")))
        {
          oledClear();
          oledGotoYX(0, 0);
          oledPrint("Pozovite");
          Delay_ms(1000);
        }
           if(stringUART == STIGAO)
           {
             oledClear();
             oledGotoYX(0,0);
             oledPrint("Zvanje u toku");
             Delay_ms(1000);
               for(x=0; x<12; x++)
               {
                if((*br) != ('\r'))
                  mob[x]=*br;
                else mob[x]=0;
                  br++;
               }
```

4.3.13. Četvrto stanje modema

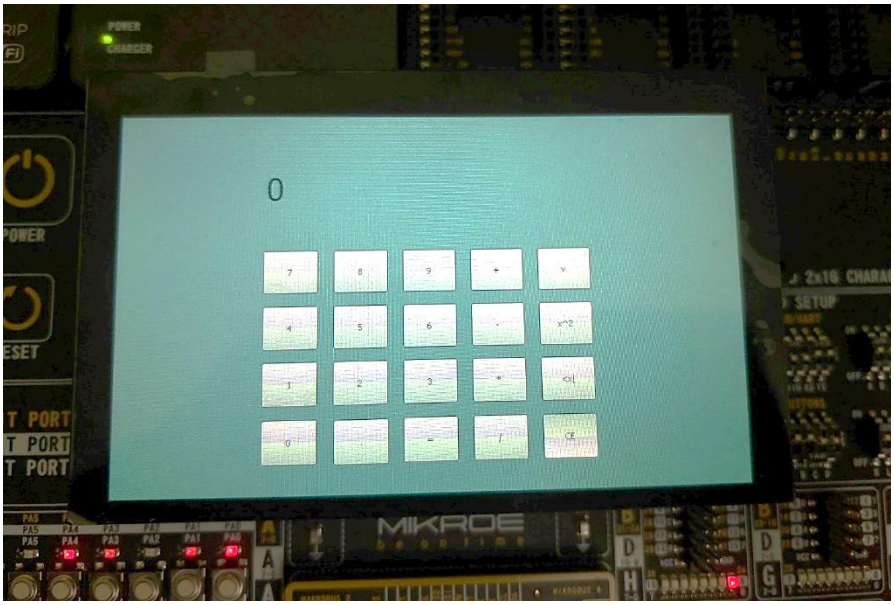
Zatim, potrebno je proveriti da li se broj pozivaoca nalazi u bazi. Provera se vrši pomoću funkcije *strcmp()* koja poredi dva stringa, konkretno u našem slučaju, broj telefona pozivaoca i svaki element unutar baze, odnosno svaki sačuvani broj pretplatioca. U slučaju da je broj pronađen, traži se slobodno parking mesto, odnosno, prva isključena *LED* porta B pomoću funkcije *pronadjiSlobodnuLed()*. Ukoliko postoji slobodno mesto, odgovarajuća *LED* se uključuje, parking mesto je uspešno zauzeto i prelazi se na početak stanja 3, čekanje poruke novog pretplatioca. Ukoliko nije pronađeno ni jedno slobodno parking mesto, ispisuje se odgovarajuća poruka na displeju i prelazi se takođe, na početak stanja 3, gde se čeka sledeća poruka. Važno je napomenuti da prva dioda koju koristimo se nalazi na PB8, te iz tog razloga koristimo poziciju *i+8* unutar koda. Funkcijom *memset* brišemo sadržaj niza *phoneNumber*. Završni deo koda, prikazan je na slici 4.3.14.

```
for ( i = 0; i < PARKING_MJESTA; i++) {
   if(!strcmp(mob, baza[i]))
   {
      oledClear();
      oledGotoYX(0, 0);
      oledPrint("Parkiranje");
      Delay_ms(4000);
      index = i ;
      stanje = 3;
      break;
   }
   }
   if (index != -1) {
       // Ako je pronaden broj u bazi podataka, pronadi slobodnu LED diodu
      int slobodnaLedDioda = pronadjiSlobodnuLed();
      if (slobodnaLedDioda != -1) {
      // Ukljuci prvu slobodnu LED diodu
       GPIOB_PSOR |= (1 << i+8);
       stanje = 3;
      }
      else {
       oledClear();
       oledGotoYX(0, 0);
       oledPrint("Nema slobodnih");
       oledGotoYX(1, 0);
       oledPrint("mjesta");
       stanje = 3;
     }
   }
}memset(phoneNumber, 0, sizeof(phoneNumber));
break;
```

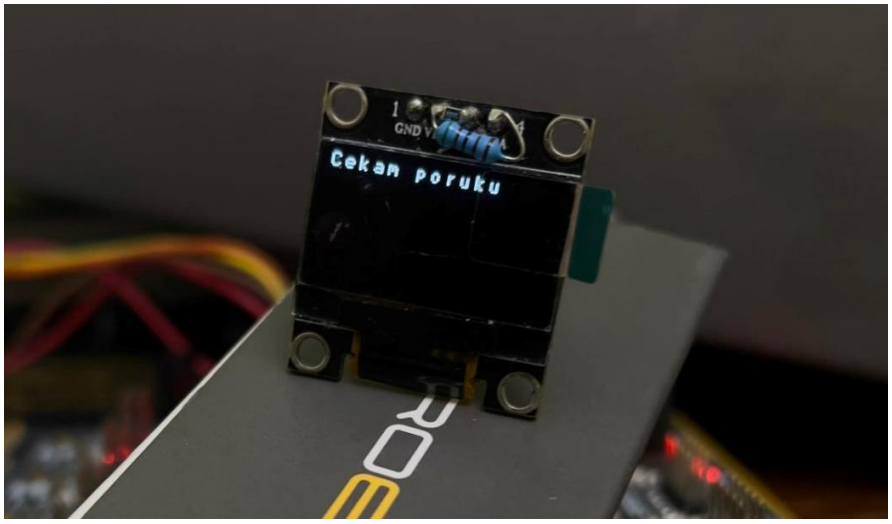
4.3.14. Četvrto stanje modema-parkiranje

5. Fizički prikaz prvog i drugog dela projekta

Na slici 5.1. i 5.2. nalaze se realizovani delovi projekta. Slika 5.1. opisuje prvi deo projekta, gde je traženo da se na kapacitivnom *TFT* 5'' ekranu prikaže kalkulator, dok se na slici 5.2. nalazi prikaz *OLED* 0.96'' ekrana na kom je ispisano jedno od stanja u kom se može naći *GSM* modul.

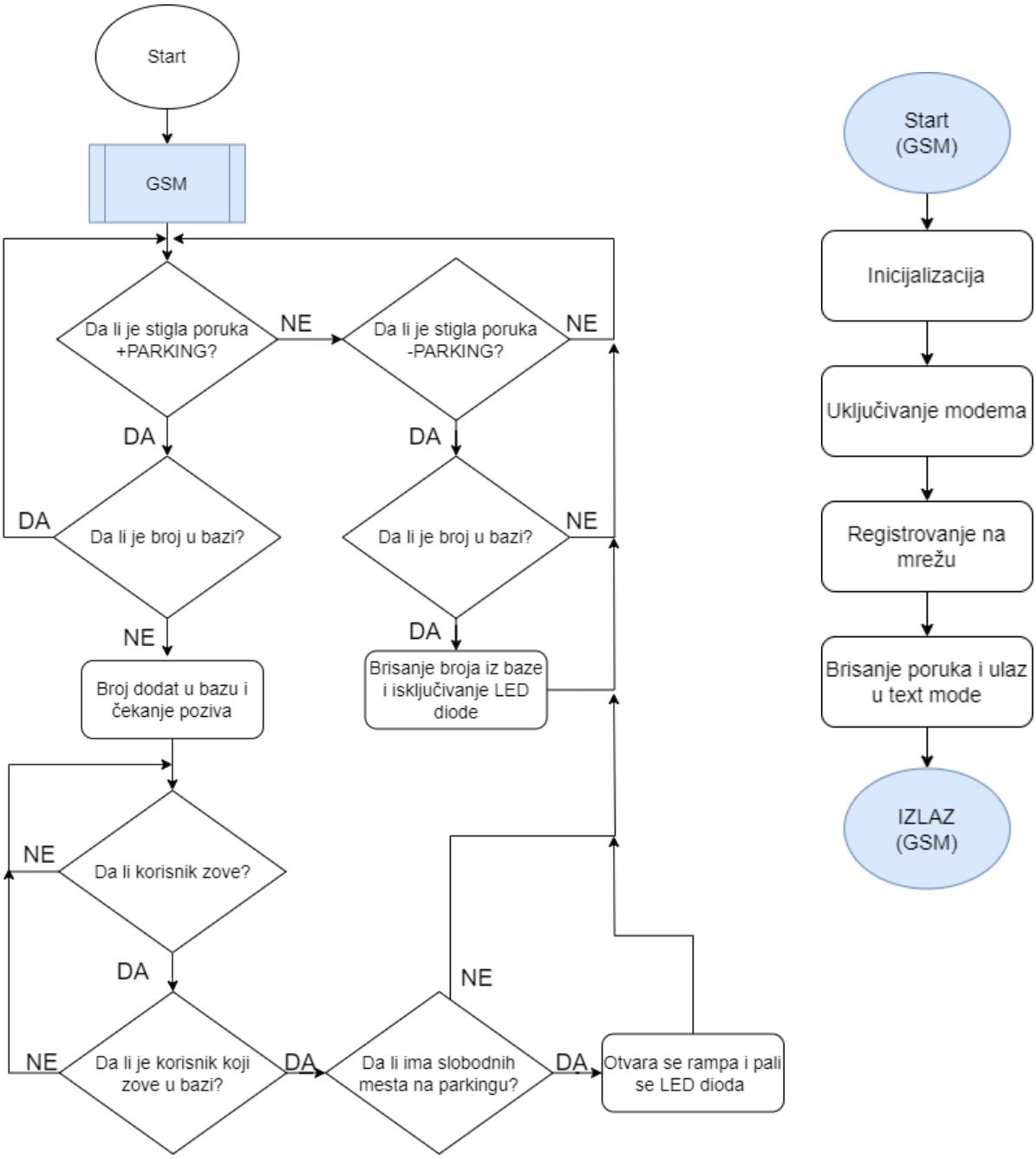


Slika 5.1. Prikaz TFT 5'' kapacitivnog ekrana sa kalkulatorom



Slika 5.2. 0.96'' OLED ekran koji opisuje stanje u kom je GSM modul u stanju čekanja

6. Algoritam rada



Slika 6.1. Glavni blok dijagram algoritma rada vezan za GSM modul (levo) i blok dijagram algoritma rada podsistema GSM (desno)

7. Zaključak

Izrađeni projekat u potpunosti odgovara postavljenim ciljevima, pružajući zadovoljavajuću funkcionalnost. Projekat je detaljno opisan, omogućavajući korisniku da prilagodi sistem prema vlastitim potrebama.

Tokom izrade projekta, nismo naišli na značajne poteškoće, osim u estetskom aspektu kalkulatora. Promena u prikazu simbola korena nastala je zbog tehničkog ograničenja na dugmetu koje dozvoljava samo unos brojeva i slova. Iz tog razloga, nije bilo moguće direktno ubaciti simbol korena na dugme, dok je prikaz simbola za kvadriranje broja ograničen na istu veličinu (x^2) zbog istog tehničkog ograničenja. Ovaj izbor je bio neophodan kako bismo održali doslednost u prikazu na tastaturi, usled ograničenja u prikazu simbola. Iako ove estetske detalje korisnici mogu smatrati neobičnim na početku, funkcionalnost samog kalkulatora je zadovoljavajuća. Ograničili smo i dužinu brojeva na 15 kako bismo održali proporcije i estetiku, pošto je deo za ispis rezultata veći od dela sa tastaturom.

U drugom delu projekta, koji se odnosi na *GSM* komunikaciju, dodali smo *OLED* displej kako bismo omogućili korisnicima vizualno praćenje tok projekta. Najizazovniji deo u vezi sa *GSM*-om bio je identifikacija odgovarajuće *AT* komande koja može pravilno pročitati poruku i oblikovati je unutar bafera. Ponekad, usled lošeg signala, modem ne prima poruku kako treba, pa je potrebno više pokušaja za uspešno slanje.

Moguća unapređenja u vezi sa GSM modemom uključuju implementaciju povratnih informacija. Iako sve informacije već budu prikazane na displeju, poboljšanje bi moglo obuhvatiti slanje odgovarajućih poruka korisnicima. Na primer, kada korisnik pošalje poruku +PARKING, mogao bi primiti automatski odgovor sa sledećim sadržajem: "Hvala Vam što ste izabrali našu garažu za parking. Želimo Vam ugodan dan." Slično tome, slanje -PARKING moglo bi rezultirati automatskim odgovorom: "Uspešno ste se odjavili, srećan put." Ove personalizovane poruke dodale bi dodatni nivo korisničkog iskustva i pružile bi korisnicima potvrdu o uspešnoj interakciji sa sistemom.

Dalja unapređenja, posebno u vezi sa kalkulatorom, mogla bi obuhvatiti dodavanje dodatnog prozora pored već postojećeg, pružajući proširene mogućnosti kao kod naučnog kalkulatora. U novom prozoru korisnici bi imali pristup raznovrsnim dodatnim operacijama koje bi obogatile funkcionalnost kalkulatora. Neki od predloženih dodataka uključuju:

- Simbol π
- Ln (prirodni logaritam)
- Sinus (Sin)
- Kosinus (Cos)
- Tangens (Tan)
- Hiperbolični sinus (Sinh)
- Hiperbolični kosinus (Cosh)
- Logaritam po bazi 10 (Log)
- Faktorijel
- Zagrade za grupisanje izraza
- Stepenovanje (x^y)
- Eksponencijalna funkcija sa proizvoljnim eksponentom (ex), itd

Ova nadogradnja bi omogućila korisnicima pristup bogatijem skupu matematičkih funkcija, čineći kalkulator još korisnijim za različite matematičke proračune.

8. Literatura

[1]-,*Datasheet* mikrokontrolera, URL:
https://www.nxp.com/docs/pcn_attachments/16530_K64P144M120SF5-rev4-final.pdf,
[Pristupljeno: april 2024]

[2]-, Razvojni sistem fusion for ARM v8,URL:
https://www.mikroe.com/fusion-for-arm,
[Pristupljeno: april 2024]

[3]-, Osnovne informacije o kapacitivnom TFT 5" displeju, URL:
https://www.mikroe.com/tft-board-5-capacitive,
[Pristupljeno: april 2024]

[4]-, Osnovne informacije o GSM3 *click* pločici,
URL:https://www.mikroe.com/gsm-3-click,
[Pristupljeno: april 2024]

[5]-, *MikroBUS standard specifications*, URL:
https://download.mikroe.com/documents/standards/mikrobus/mikrobus-standard-specification-v200.pdf, *[Pristupljeno: april 2024]*

[6]-, *SIM800 Series_AT Command Manual_V1.09, SIMCom Wireless Solutions Ltd*,
URL:
https://wiki.elecrow.com/images/2/20/SIM800_Series_Command_Manual_V1.09.pdf,
[Pristupljeno: april 2024]

[7]-, *OLED 0.96''* displej, URL:
https://www.smart-prototyping.com/0-96-OLED-Display-IIC-New-Version,
[Pristupljeno: april 2024]