Střední průmyslová škola elektrotechnická Havířov	**PRG**	**Třída: 4B** **Skupina:** 1

Řídící jednotka automatické pračky

Den: 05.03.2025

Jméno učitele: Božena Ralbovská

Jméno: Vojtěch Lisztwan

Známka: snad jedna

1. Zadání

Pokyny

- Jednotlivé fáze mají fixní časový úsek, po který běží – ten se počítá z celkového času v následujících poměrech:
  - namáčení – zbytek po odečtení zbylých časů z celkového času
  - praní – 60% celkového času
  - opláchnutí – 20% celkového času
  - sušení – 15% celkového času.
- Po dokončení pracího cyklu se zobrazí název cyklu a fráze „KONEC“ na LCD a 1sekundu běží zámek, po který nelze otevřít dveře pračky.

- Program bude provádět následující:
  - Zobrazení menu s výběrem programu, ve kterém se uživatel pohybuje pomocí kláves „*“ a „#“. Klávesu „0“ použijte pro výběr programu:
    - 3 programy pračky: KRÁTKY (2min), DLOUHY(5min), TEST (1min).
  - Po zvolení programu se rozběhne cyklus praní, zobrazujte na:
    - 1. řádku LCD displeje celkový čas zbývající k dokončení cyklu (MM:SS)
    - 2. řádku LCD zobrazuje čas aktuální fáze.
  - Zobrazení aktuální fáze na externích LED diodách.
  - Zobrazujte stav pračky na interních LED diodách (otevřeno – zelená, pauza – oranžová, zamknuté dveře – červená, probíhající praní – modrá).
  - Přidejte funkci automatického zámku po dokončení cyklu praní.
  - Přidejte možnost pauzy pomocí uživatelského tlačítka.
  - Přidejte smyčku nebo jiný mechanismus, který zajistí, že program bude opakovatelný.

2. Teoretický rozbor - analýza

Struktura kódu

Knihovny a makra:

Zahrnuté knihovny poskytují funkcionalitu pro práci s hardwarem STM32, jako jsou LED, tlačítka, LCD a UART.

Knihovny jsou propůjčeny ze školního kitu.

Zadefinované datové typy:

```
typedef struct{
	int casCelkem;
	int namaceni;
	int prani;
	int oplachnuti;
	int suseni;
	char nazev[20];
}Tcyklus;		//struktura pro definování programů pračky
```

Globální proměnné:

```
int odStartu=0;		//globální proměnná pro počítání vteřin
int doKonceCyklu=0;	//proměnná počítající čas do konce cyklu
int doKonceFaze=0;	//proměnná počítající čas zbývající do konce fáze
int faze = 5;		//proměnná pro uložení, která fáze probíhá
int locked =0;		//proměnná, která symbolizuje zamknutí dvířek
int opened=0;		//proměnná která symbolizuje otevřené dvířka pračky
Tcyklus cykly[] = {
    {120,0,0,0,0, "Kratky"},
    {300,0,0,0,0, "Dlouhy"},
    {60,0,0,0,0, "Test"}
};			//proměnná, ve které jsou zadefinované programy pračky
```

Funkce:

`void led_set(pin_t pin, int value)`: Ovládají stav LED diod.

`int vyber(): funkce pro vyber programu pro praní`

RTOS tasky:

- `__task void Task();`
  - Task, který zajišťuje přepínání fází a cyklů. Zapisuje do globálních proměnných jako je locked nebo pause. Zajišťuje výpis na LCD.
- `__task void Timer();`
  - `Task, ve kterém se počítají vteřiny od začátku. Hlavní úlohou tohoto tasku je inkrementace proměnné odStartu;`
  - `Také obsluhuje uživatelské tlačítko. Po stitknutí se nastaví pause na 1, rouzvítí se příslušné LED diody. Následně Task čeká v nekonečné smyčce na stisknutí tlačítka a zrušení pauzy.`
- `__task void Leds();`
  - `Zajišťuje rozsvicování příslušných diod, jak externích tak interních.`

Hlavní funkce:

setup: Inicializuje hardware, globální proměnné a vytváří úkoly.

main: Spouští RTOS inicializaci.

Analýza funkcionality

1. Logika řídící jednotky automatické pračky

- Nejdříve uživatel musí vybrat program, který chce použít. To zajistí funkce vyber().
- Z vybraného programu se rozpočítá časy jednotlivých fází podle procent ze zadání.
- Po dokončení všech fází se počká 1 vetřinu a pračka se poté odemkne.

3. Rozhraní a vstupy

Klávesnice je použita pro simulaci ovládacích prvků na reálné pračce.

Tlačítko USER_BUTTON slouží pro pozastavení praní.

LCD displej slouží pro informování uživatele o aktuálním stavu a fázi pračky a zbývajícím čase do konce programu.

4. LED diody

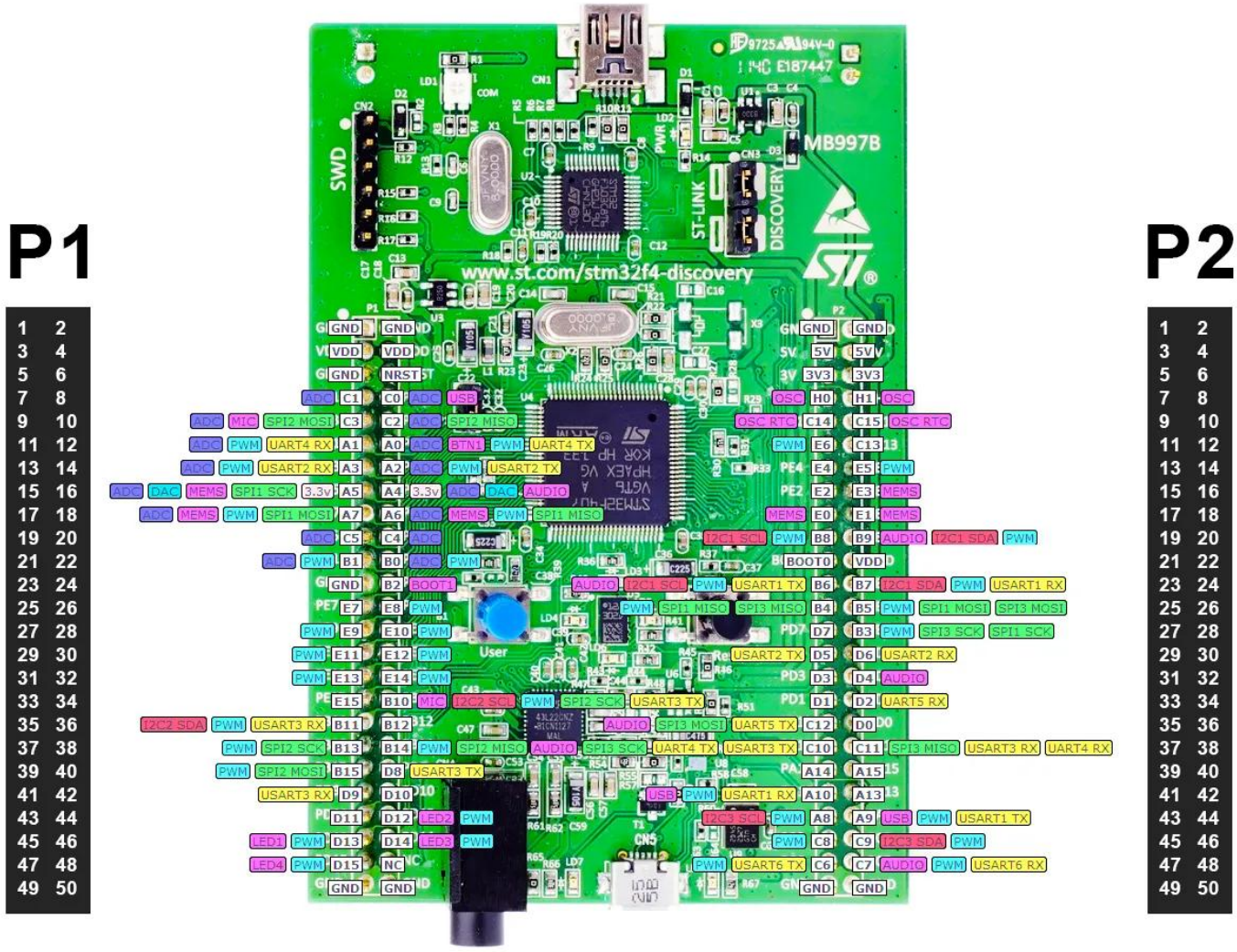
- Interní
  - Zelená-dvířka do pračky jsou otevřená
  - Oranžová-pozastavení programu
  - Červená-dvířka jsou zamknuta
  - Modrá-probíhající praní
- Externí
  - Zobrazují po řadě aktuální fáze praní.
  - Z levé strany do pravé

5. Displej

- LCD slouží pro výběr programu.
- Po spuštění informuje uživatele o průběhu programu.

Hardware

STM32F407



Obr. 1

- Jedná se o mikroprocesor, který programujeme pomocí ST-LINK. Pracuje na frekvenci 16MHz, ale můžeme ho spustit až na 160MHz. Podporuje RTOS, má zabudované uživatelské tlačítko a interní led diody. Tlačítko černé bary slouží k restartu mikroprocesoru. Podporuje sběrnice jako je I2C, SPI nebo třeba usb. Dokáže zpracovávat audio, má ADC převodník, generátor signálu PWM a mnoho dalšího.

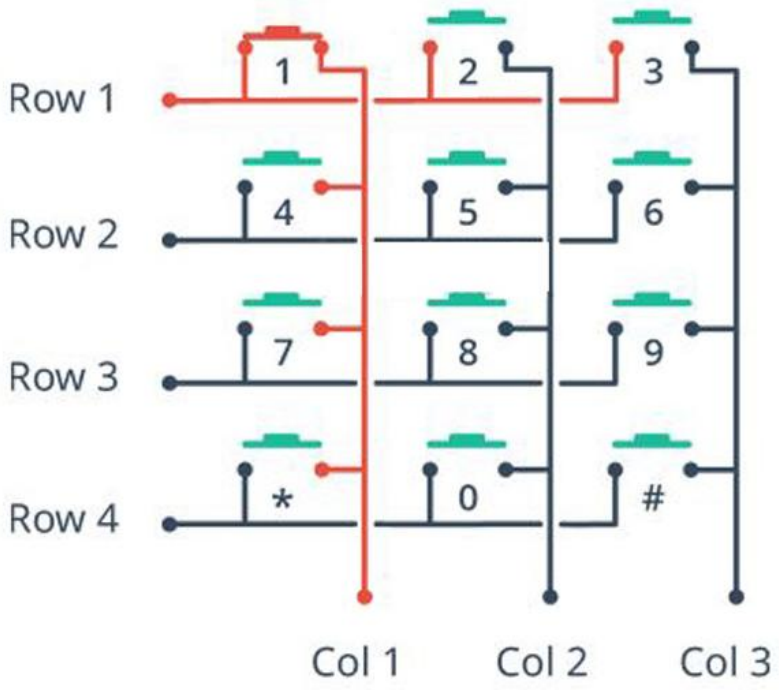
Interní led diody

- Připojené na port D, jedná se o diody smd.
- Přípojené na PD13 - PD15 v pořadí žlutá, zelená, červená a modrá.

Zabudované uživatelské tlačítko

- Připojeno na port A na pin PA0.

Klávesnice



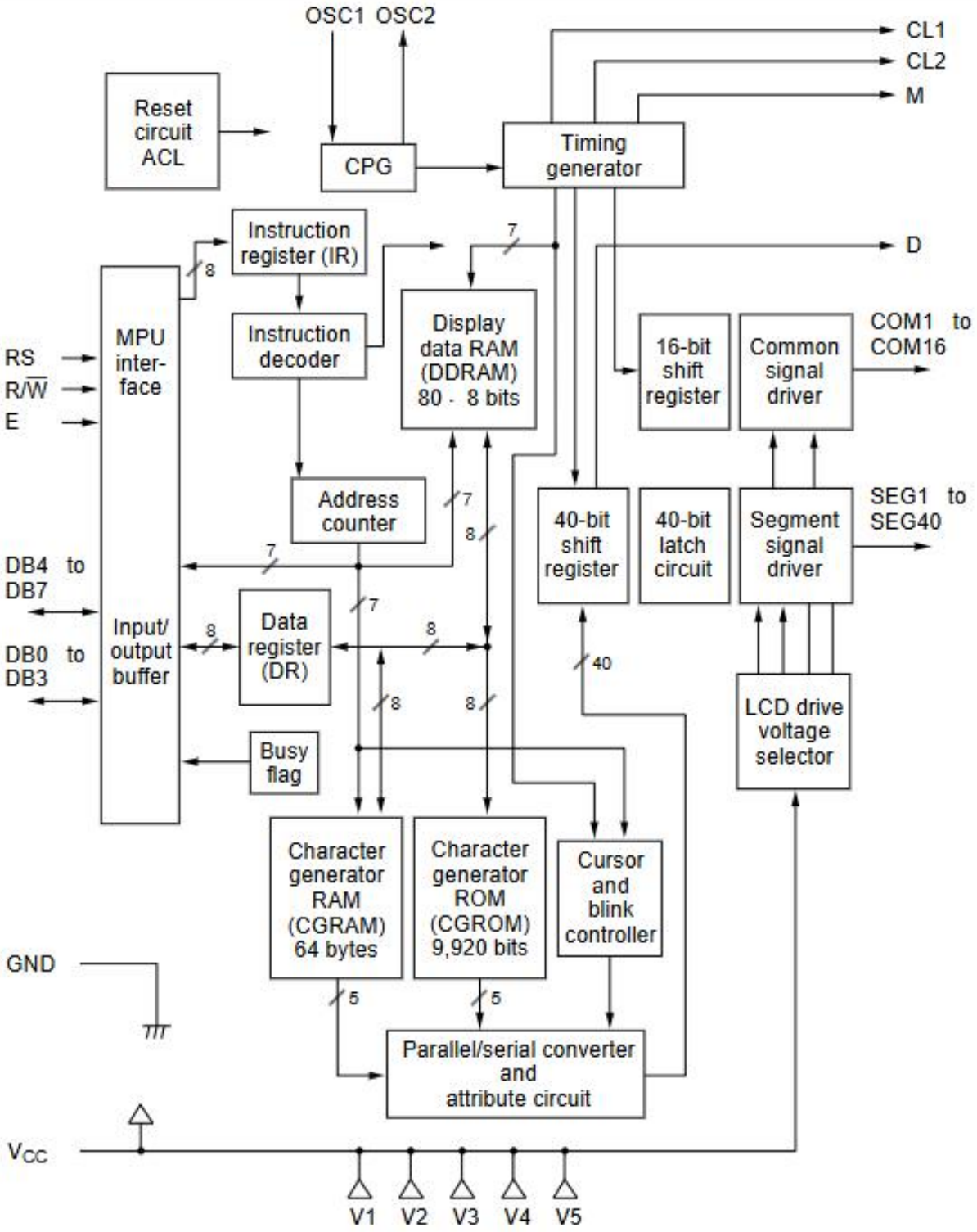
Obr. 2

Na obrázku Obr.2 je vyobrazena klávesnice 3x4, já jsem použil 4x4. Z obrázku lze jednoduše pochopit princip funguvání. Je zmáčknuto tlačítko 1. Postupně za sebou přivademe na piny ROW1 - ROW4 napětí, a čteme piny COL1-COL-3. Podle toho, kde naměříme napětí tak pomocí souřadnice xy zjistíme které tlačítko je přesně zmáčknuto. Z mapy KBD_MAP zjistíme přiřazený znak tomuto tlačítku.

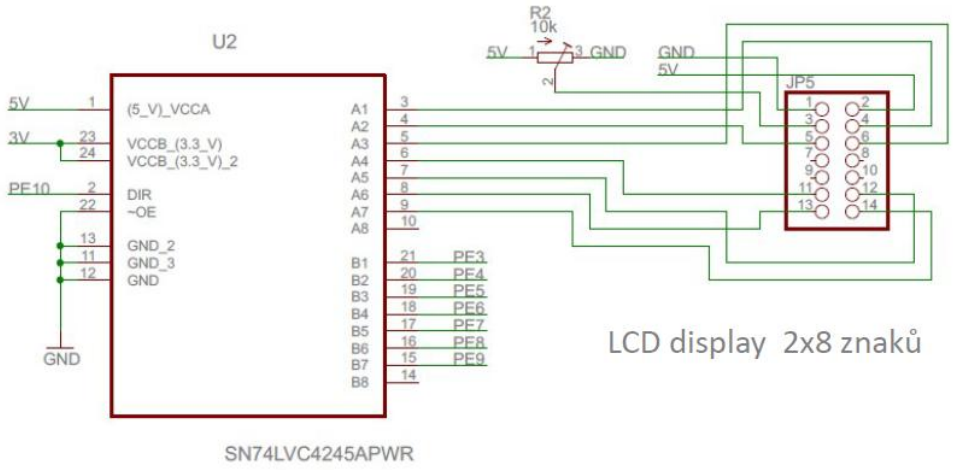
Standartně jsou piny sloupce COL0-COL3 připojeny na PD0-PD3, a piny řady ROW0-ROW3 připojeny na PD6-PD9. Ve školním kitu je ale o jeden sloupec méně, protože jsou použity klávesni 4x3.

LCD 16x4

Obr. 3


OSC1 OSC2
CL1
CL2
M
Reset circuit ACL
CPG
Timing generator
Instruction register (IR)
D
Instruction decoder
Display data RAM (DDRAM) 80 - 8 bits
16-bit shift register
Common signal driver
COM1 to COM16
RS
R/W
E
MPU inter-face
Address counter
40-bit shift register
40-bit latch circuit
Segment signal driver
SEG1 to SEG40
DB4 to DB7
DB0 to DB3
Input/ output buffer
Data register (DR)
LCD drive voltage selector
Busy flag
Character generator RAM (CGRAM) 64 bytes
Character generator ROM (CGROM) 9,920 bits
Cursor and blink controller
GND
Parallel/serial converter and attribute circuit
VCC
V1
V2
V3
V4
V5

Obr. 4



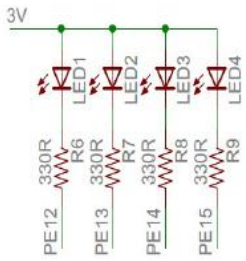
Obr. 5

Ve školením kitu je použit displej LCD 2x8. Pro práci s ním je nutno použít knihovnu LCD.h. Komunikace s řídícími obvody tohoto displeje probíhá na sedmi vodičích, tři jsou řídídí a čtyři datové. Pro rychlejší komunikaci jsme mohli požít osm datových vodičů. Změna by ale byla tak nepatrná, že radši šetříme volnými piny na mikroprocesoru.

Propojení:

LCD	ARM
RS	PE3
R/W	PE4
E	PE5
DB4	PE6
DB5	PE7
DB6	PE8
DB7	PE9

Externí ledky

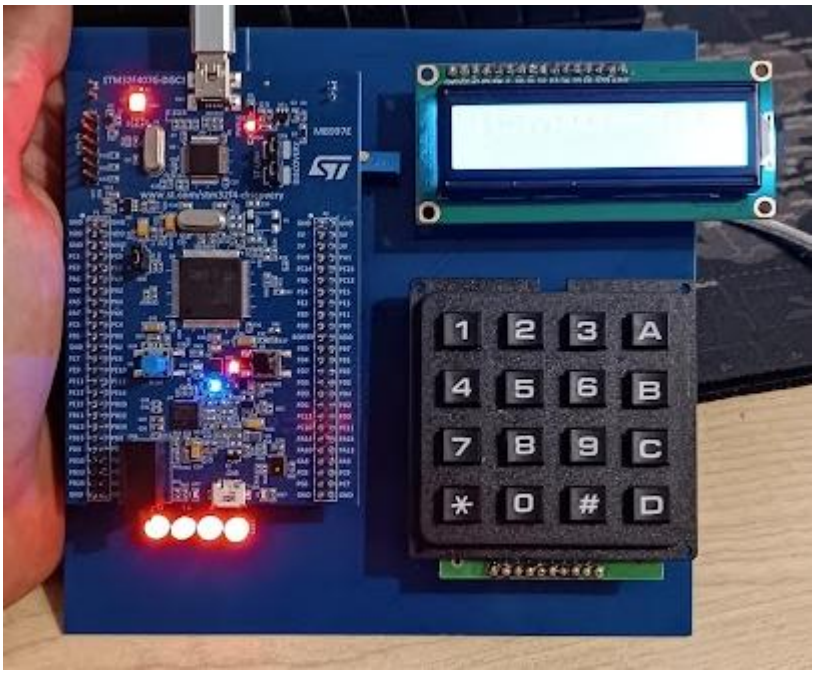


Obr. 6

Externí LEDky(Obr.6) jsou zapojeny způsobem PULL_UP. To znamená že je zapínáme přivedením nuly na příslušné piny. Z Obr. 6 je zřejmé, že ledky jsou připojeny na PE12-PE15. Všechny jsou červené barvy.

DESKA-vlastní výroby

Deska je zjednodušení pro zapojení.



Round Robin

Díky požití RTOS a Round Robinu se může program/zařízení chovat jako zařízení, které vykonává více věcí najednou. Například nečekáme na načtení vstupu z klávesnice, zatímco rozsvicujeme ledky. Plánovač každé úloze přidělí stejný čas. Úlohy se střídají tak rychle, že se systém jeví jako že dělá všechno v jednom časovém okamžiku(dělá více věcí najednou).

Může to způsobovat ale problémy se synchronizací procesů. Proto je v kritických místech(kritické sekce) nutné použít semafory(OS_SEM). Mě se podařilo program napsat bez kritické sekce, proto nemusím semafory používat.

3. Postup - popis vlastních funkcí

led_set(pin_t pin, int value)

Nastavuje stav konkrétní LED diody podle zadaného pinu. Funkce rozlišuje mezi interními a externími LED diodami. Pro externí LED je logika inverzní (0 zapnuto, 1 vypnuto).

vyber()

Funkce pro výber programu, vrátí číslo, které odpovídá indexu v poli vybraného programu.

__task void setup()

- Inicializace a nastavení periferií.
- Vypočet všech časů jednotlivých časů fází pomocí procent z celkového času.
- Inicializace a spuštění jednotlivých úloh.
- V posledním kroku se úloha `setup` sáma smaže.

__task void Task()

- Zpracovává vstup od uživatele(vyber programu).
- Zajišťuje přepínámí mezi fázemí a cylky.
- Vypisuje informace na LCD(průběh praní)

__task void Timer()

- Zajišťuje počítání vteřin od startu programu.
- Kontroluje zmáčknutí tlačítka pauzy, v případě zmáčknutí nastaví promennou `pause` na 1 a čeká v nekončné smyčce na opakované zmáčknutí.

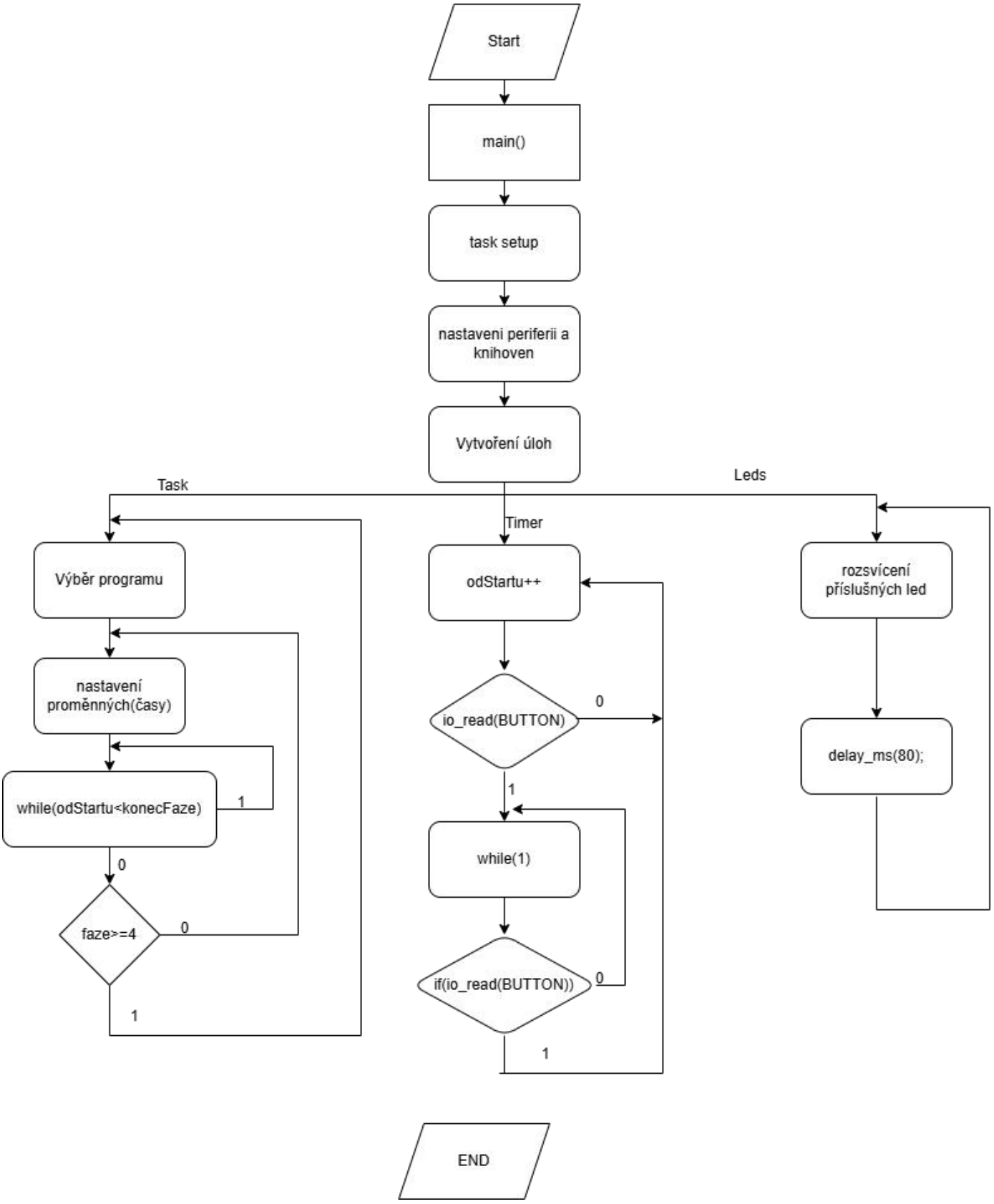
__task void Leds()

- Rozsvicuje a zhasíná příslušné diody. Kontroluje zmáčknutí tlačítka pro otevření pračky.

int main()

Spouští operační systém a inicializační úlohu `setup`. Po spuštění systému jsou zajištěny všechny potřebné funkce prostřednictvím vytvořených úloh.

6. Vývojový diagram



7. program + blokový komentář

```
/**
 * @file   pracka_Rala.c
 * @author Vojtěch Lisztwan
 *
 */


#include "stm32_kit.h"
#include "stm32_kit/led.h"
#include "stm32_kit/button.h"
#include "stm32_kit/keypad.h"
#include "stm32_kit/lcd.h"
#include "stm32_kit/chrono.h"
#include "string.h"
#include "stdio.h"
//Nastavení mapy pro přiřazení znaků pro jednotlivé tlačítka
static uint8_t KBD_MAP[KEYPAD_ROWS][KEYPAD_COLS] = {
        '1', '2', '3', 'A',
        '4', '5', '6', 'B',
        '7', '8', '9', 'C',
        '*', '0', '#', 'D'
};


//výběr pinů pro interní led
enum pin interni[]={
    LED_IN_0,
    LED_IN_1,
    LED_IN_2,
    LED_IN_3
};
```

```
//výběr pinů pro externí led
enum pin externi[]={
    LED_EX_3,
    LED_EX_2,
    LED_EX_1,
    LED_EX_0
};

//deklarace úloh
OS_TID id_task,id_task1,id_task2;

//deklarace globálních proměnných
int odStartu=0;
int doKonceCyklu=0;
int doKonceFaze=0;
int faze = 5;
int locked =0;
int opened=0;

//definici datového typu Tcyklus
typedef struct{
    int casCelkem;
    int namaceni;
    int prani;
    int oplachnuti;
    int suseni;
    char nazev[20];
}Tcyklus;

//Deklarace a inicializace programů pro praní, realizováno jako pole cyklů
Tcyklus cykly[] = {
    {120,0,0,0,0, "Kratky"},
```

```
    {300,0,0,0,0, "Dlouhy"},
    {60,0,0,0,0, "Test"}
};

//funkce pro jednodužší zapínání led
//použití Inverzní logiky pro externí led, jsou zapojeny s PULL_UP rezistorem
void led_set(pin_t pin, int value){
    //proměnná pro inverzní logiku
    int on = 1;
    if(io_port(pin)==io_port(LED_EX_0))on =0;
    io_set(pin, on ? value : !value);
}

//funkce pro výběr programu pro praní, vrátí index z pole cykly
int vyber(){
    int position=0;
    int prev=2;
    uint8_t znak;
    char buff[21];
    LCD_set(LCD_CLR);

    //smyčka, díky které se uživatel může pohybovat v menu
    //pro pohyb použita klávesnice a znaky '*' a '#'
    do{
        if(prev!=position){
            LCD_set(LCD_CLR);
            LCD_set(LCD_LINE1);
            sprintf(buff, "%s",cykly[position].nazev);
            LCD_print(buff);
            LCD_set(LCD_LINE2);
            sprintf(buff, "%02d:%02d",(cykly[position].casCelkem/60),cykly[position].casCelkem%60);
            LCD_print(buff);
```

```
            prev = position;
        }
        znak = KBD_read();
        if(znak=='#')position++;
        if(znak=='*')position=position - 1;
        if(position < 0)position = /*((sizeof(cykly)/sizeof(cykly[0]))-1)*/ 2;
        if(position>=sizeof(cykly)/sizeof(cykly[0])) position=0;

        delay_ms(100);
    }while(znak!='0');
    return position;
}

//Úloha, která zajišťuje výběr programu pro praní a přechod mezi fázemi programu
//zapíná a vypíná některé kontrolní ledky
__task void Task() {
    for (;;) {
        int index = vyber();
        int end;
        int konecFaze;
        char buff[20];
        char action[20];

        //kontrola, jetli není pračka otevřená před spuštěním praní
        if(opened){
            //kontrola zavřené pračky před spuštěním praní
            LCD_set(LCD_CLR);
            LCD_set(LCD_LINE1);
            LCD_print("Zavrete pracku");
            while(opened){;}//nekonečná smyčka čekající na zavření pračky
        }
```

```
end = odStartu+cykly[index].casCelkem;//výpočet konce programu
locked =1; //uzamčení pračky
led_set(interni[3], 1); // zapnutí led diody signalizující praní

//smyčka pro fáze programu
for(faze = 1;faze<5;faze++){

    switch(faze){
        case 1:
            konecFaze = odStartu+cykly[index].namaceni;
            sprintf(action, "namaceni");
            break;
        case 2:
            konecFaze = odStartu+cykly[index].prani;
            sprintf(action, "prani");
            break;
        case 3:
            konecFaze = odStartu+cykly[index].oplachnuti;
            sprintf(action, "oplachnuti");
            break;
        case 4:
            konecFaze = odStartu+cykly[index].suseni;
            sprintf(action, "suseni");
            break;
        default:
            break;
    }
    LCD_set(LCD_CLR);
    do{
        LCD_set(LCD_LINE1);
        sprintf(buff, "%02d:%02d %s", (end-odStartu)/60,(end-odStartu)%60, cykly[index].nazev);
        LCD_print(buff);
```

```
                LCD_set(LCD_LINE2);
                sprintf(buff, "%02d:%02d %s", (konecFaze-odStartu)/60,(konecFaze-odStartu)%60, action);
                LCD_print(buff);
                delay_ms(80);
            }while(konecFaze>odStartu);//aktualizace výpisů časů na LCD.
        }
        //vynulování časů-konec programu praní
        odStartu=0;
        doKonceCyklu=0;
        doKonceFaze=0;
        led_set(interni[3], 0); //zhasnutí signalizace praní
        //výpis na LCD
        LCD_set(LCD_CLR);
        LCD_set(LCD_LINE1);
        sprintf(buff, "%s KONEC",cykly[index].nazev);
        LCD_print(buff);

        delay_ms(1000);
        locked=0; //odemčení pračky

    }
}

__task void Timer() {
    for (;;) {
            delay_ms(1000); //čekání 1 vteřiny
            odStartu++; //inkrementace čítače vteřin

        //nulování čítačů, kdybychom nenulovali, po nějaké době by mohly přetéct
        if(odStartu>500000&&doKonceCyklu==0){
            odStartu=0;
            doKonceCyklu=0;
```

```
            doKonceFaze=0;
        }
        //signalizace zamknutí pračky a čekání v nekočné smyčce, doku ji znovu nepustíme
        if(io_read((USER_BUTTON)==1)&&(locked ==1)){
            led_set(interni[1], 1);
            led_set(interni[3], 0);
            delay_ms(1000);
            while(1){
                if(io_read(USER_BUTTON)==1){
                    led_set(interni[1], 0);
                    led_set(interni[3], 1);
                    break;
                }
            }
        }

    }
}

//úloha pro správu LED diod
__task void Leds() {
    for (;;) {
        for(int i=0;i<4;i++){
            io_set(externi[i],((faze-1)==i)? 0:1);//signalizace aktuální fáze praní
        }
        io_set(interni[2], locked); //signalizace zamknuté pračky led diodou(červená)



        uint8_t znak;
        //čtení klávesnice, jestli není zmáčknuto D pro otevření pračky
        znak=KBD_read();
        if(znak=='D'&&!locked){
```

```
            opened = !opened;
            delay_ms(100);
        }
        io_set(interni[0], opened); //signalizace zelenou diodou otevření pračky

    }
}

//inicializace úloh, nastavení periferií a jejich knihoven
//po proběhnutí úlohy se úloha sama smaže
__task void setup() {
    LCD_setup();
    LCD_set(LCD_CLR);
    LCD_set(LCD_CUR_OFF);
    LED_setup();
    BTN_setup();
    KBD_setup();
    for(int i=0;i<(sizeof(cykly)/sizeof(cykly[0]));i++){
        cykly[i].prani=(int)((double)cykly[i].casCelkem*0.6);
        cykly[i].oplachnuti=(int)((double)cykly[i].casCelkem*0.2);
        cykly[i].suseni=(int)((double)cykly[i].casCelkem*0.15);
        cykly[i].namaceni=cykly[i].casCelkem-cykly[i].oplachnuti-cykly[i].suseni-cykly[i].prani;
    }

    id_task = os_tsk_create(Task, 0);
    id_task1 = os_tsk_create(Timer, 0);
    id_task2 = os_tsk_create(Leds, 0);
    os_tsk_delete_self();
}

//hlavní program, který spustí úlohu setup, zbytek se odehrává v jednotlivý úlohách
```

```
int main () {

    os_sys_init(setup);



}
```

8. Zhodnocení

Projekt Řídící jednotka automatické pračky byl úspěšně realizován na platformě STM32 za použití RTOS (Real-Time Operating System). Cílem bylo nasimulovat řídící jednotku automatické pračky, což si myslím, že se povedlo. Samozdřejmě by pro reálnou implementaci bylo potřeba projekt více rozšířit, přidat více programů a hlavně přizpůsobit konrétnímu hardwaru použitém v pračce. Například řízení ohřívače vody, čerpadel nebo motoru. Nicméně tu podstatnou část se podařilo úspěšně nasimulovat.

Díky desce vlastní výroby bylo jednodužší periferie propojit dohromady a nemusel jsem mít strach z povytaženého vodiče.

Klíčové vlastnosti projektu

Rozšířitelnost:

Kód je velmi dobře připraven pro rozšíření. Přidání pracích programů, změnu jejich délky atd. Byl navržen s ohledem na možnost rozšíření o další moduly, například řízení motoru atd.

Informování uživatele:

Program má velmi dobře zpracovaný sestém informování uživatele o aktuálním stavu pračky, kolik času je potřeba pro dokončení atd.

Použití periferií:
Byly využity různé periférie dostupné na STM32, jako je LCD displej pro vizualizaci času a stavu, LED diody pro indikaci aktuální fáze a simulaci jednotlivých prvků pračky, klávesnice pro výběr programu a simulaci otevření dveří.

Real-Time Management:
Díky použití RTOS je zajištěn plynulý chod jednotlivých úkolů, včetně inkrementace času, zpracování vstupu a aktualizace displeje.

Silné stránky projektu

- **Efektivní využití RTOS**: Rozdělení funkcionality do úkolů zlepšuje čitelnost a správu kódu.
- **Robustní správa času**: Systém správně počítá zbývající čas a umožňuje uživateli sledovat průběh programu.
- **Přehledný kód**: Projekt je dobře komentovaný (nyní bez diakritiky), což usnadňuje orientaci v kódu i jeho případné rozšíření.

Možnosti vylepšení

1. **Optimalizace času delay**: Některé části kódu (např. delay_ms) mohou být optimalizovány, aby systém reagoval rychleji na uživatelské vstupy.
2. **Přesné časování:** Časování (počítání sekund) je v tomto případě docela dost nepřesné. Pro větší přesnost by bylo vhodnější použít jiný interní časovač(třeba TIM4) a trochu jiný způsob počítání. Třeba použití Handleru pro přičtení sekundy do čítače.
3. **Energetická efektivita**: Při delší nečinnosti by bylo vhodné implementovat úsporný režim pro minimalizaci spotřeby energie. Například při doprání by bylo vhodné úlohy uspat a počkat, až uživatel pračku vypne.
4. **Rozšíření funkcí**: Například řízení skutečných hardwarových součástí pračky.

Závěr

Základ programu je napsán robustně a srozumitelně. Je připraven pro další rozšířování(přidání programů) atd. Zadání se mi podařilo splnit bez větších promblémů. Jako rozšíření, které by docela dobře šlo vyzkoušet na výukovém kitu, by šlo udělat odložení startu programu.