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

LUXMETR

Den: 05.03.2025

Jméno učitele: Terezaa Hermanová

Jméno: Vojtěch Lisztwan

Známka: snad jedna

1. Zadání

1. Napište program pro STM32F4 v jazyce C, který průběžně zobrazuje na LCD napětí na vstupu A/D napojeného na dělič s foto odporem.

2. pro vyšší přesnost skenujte 128krát s periodou 1ms (Fs = 1kHz).

3. A/D je 12bit s rozsahem 0 a 3V.

4. Kalibrujte čidlo nejméně v 6ti bodech, vypočtěte kalibrační funkci v excelu s odchylkou lepší než 0,95.

5. Pomocí menu volíme na klávesnici jednotky: lux nebo candelu a zdroje světla: žárovka nebo sluneční světlo.

6. Aktivujte LED po překročení osvětlení nad 400 Lx.

2. Teoretický rozbor - analýza

Struktura kódu

Kód je napsán v jazyce C a je určen pro mikrokontrolér STM32. Hlavním účelem programu je měření osvětlení (v luxech nebo kandela) pomocí ADC (Analog-to-Digital Converter) a zobrazování výsledků na LCD displeji. Kromě toho využívá klávesnici pro uživatelský vstup a LED diody pro signalizaci.

Hlavní části kódu:

- Inkluze knihoven – zahrnutí potřebných souborů pro práci se STM32 periferiemi (tlačítka, LCD, ADC, LED, časování).
- Definice konstant – mapování klávesnice a definování pole LED pinů.
- Funkce led_set() – slouží k ovládání LED diod s ohledem na jejich zapojení.
- Funkce setup() – inicializuje systémové hodiny, přerušení SysTick, LCD, klávesnici, ADC, tlačítka a LED diody.
- Funkce main() – hlavní smyčka programu:
  - Volba jednotky měření (lux/kandela) pomocí klávesnice.
  - Volba světelného zdroje (žárovka/slunce).
  - Periodické měření napětí pomocí ADC a výpočet odpovídající hodnoty osvětlení.
  - Výpis výsledků na LCD displej.
  - Rozsvícení LED diod, pokud je intenzita osvětlení nad 400 luxů.

Funkce:

`void led_set(pin_t pin, int value)`: Nastavuje hodnotu na daném pinu pro rozsvícení ledky. Testuje jestli je ledka interní nebo externí.

`BOARD_SETUP void setup(void): Importuje knihovny a nastavuje periferie.`

Hlavní funkce(main):

Výběr jednotky měření

Uživatel si pomocí klávesnice vybere, zda chce měřit v luxech nebo kandelách.

Hodnota se ukládá do proměnné `jednotka`.

Výběr zdroje světla

Uživatel si zvolí, zda měření probíhá u žárovky nebo na slunci.

Výsledek se ukládá do proměnné `zdroj` a následně ovlivňuje korekční koeficient měření.

Měření intenzity světla

Pro získání stabilní hodnoty se provede průměrování ze 128 vzorků ADC.

Hodnota ADC se převede na milivolty (`mili`), ze kterých se vypočítají luxy pomocí exponenciálního vztahu.

Pokud je měření ve `candela`, provede se další přepočet.

Zobrazení na LCD

Nejprve se zobrazí napětí v mV na prvním řádku LCD.

Na druhém řádku se zobrazí vypočtená hodnota osvětlení a její jednotka.

Ovládání LED diod

Pokud je intenzita osvětlení vyšší než 400 luxů, rozsvítí se všechny LED diody.

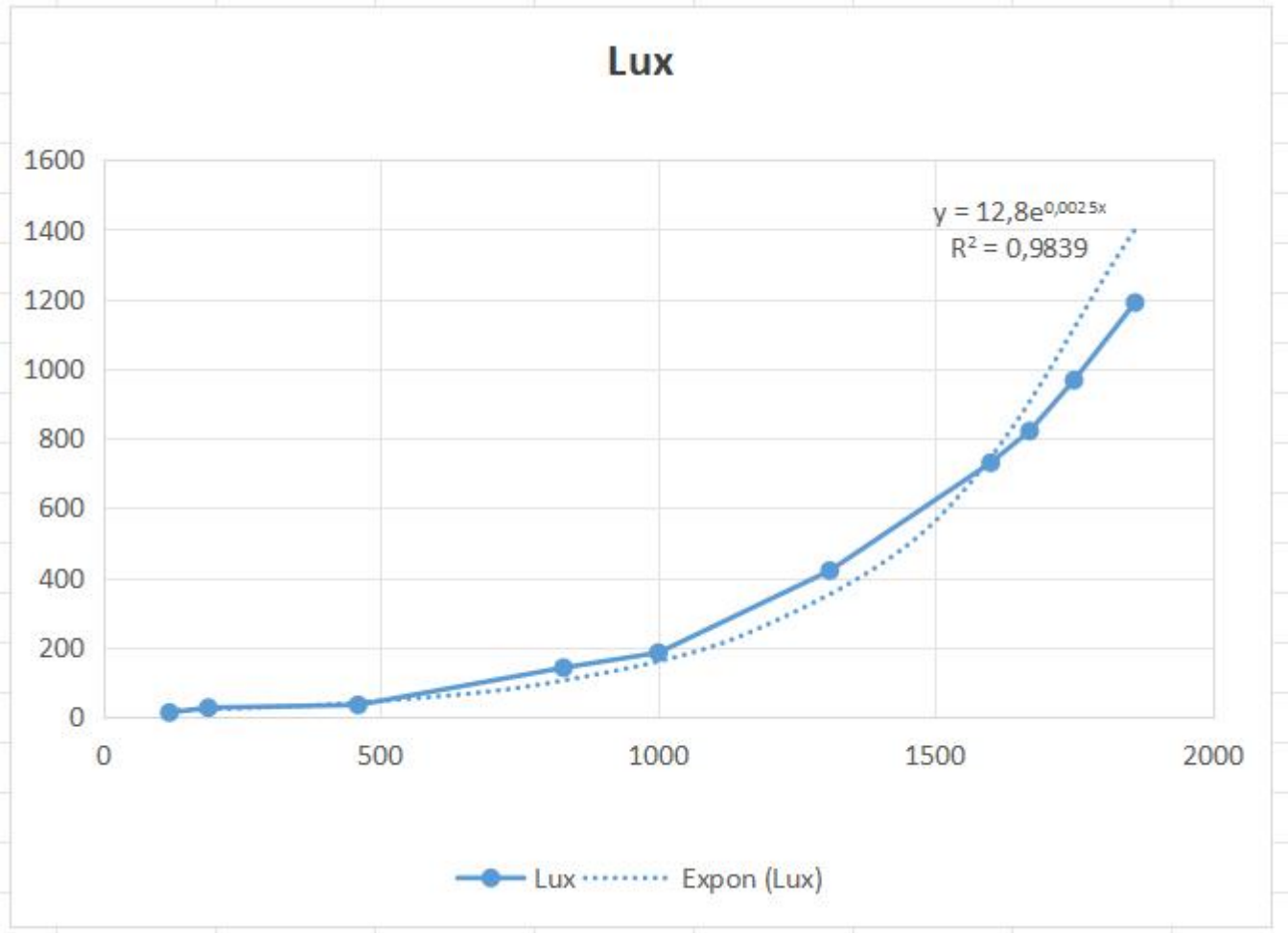
Jinak se všechny LED diody vypnou.

Analýza výpočtů

- `mili = ((sum/128)*3000)/4096;`
- Převádí průměrnou hodnotu ADC na napětí v mV.
- Rozsah ADC je 0–4096 (12bit), referenční napětí je 3V (3000 mV).
- `luxes = 12.8 * exp(0.0025 * mili);`

mV	Lux
120	12
190	26
460	34
830	141
1001	184
1310	420
1600	730
1670	821
1750	967
1860	1190

- Jelikož má každý fotorezistor jiné vlastnosti, je nutné udělat kalibrační tabulku, za které sestrojíme graf a určíme parametry exponenciální funkce pro správný výpočet Luxů/candella.



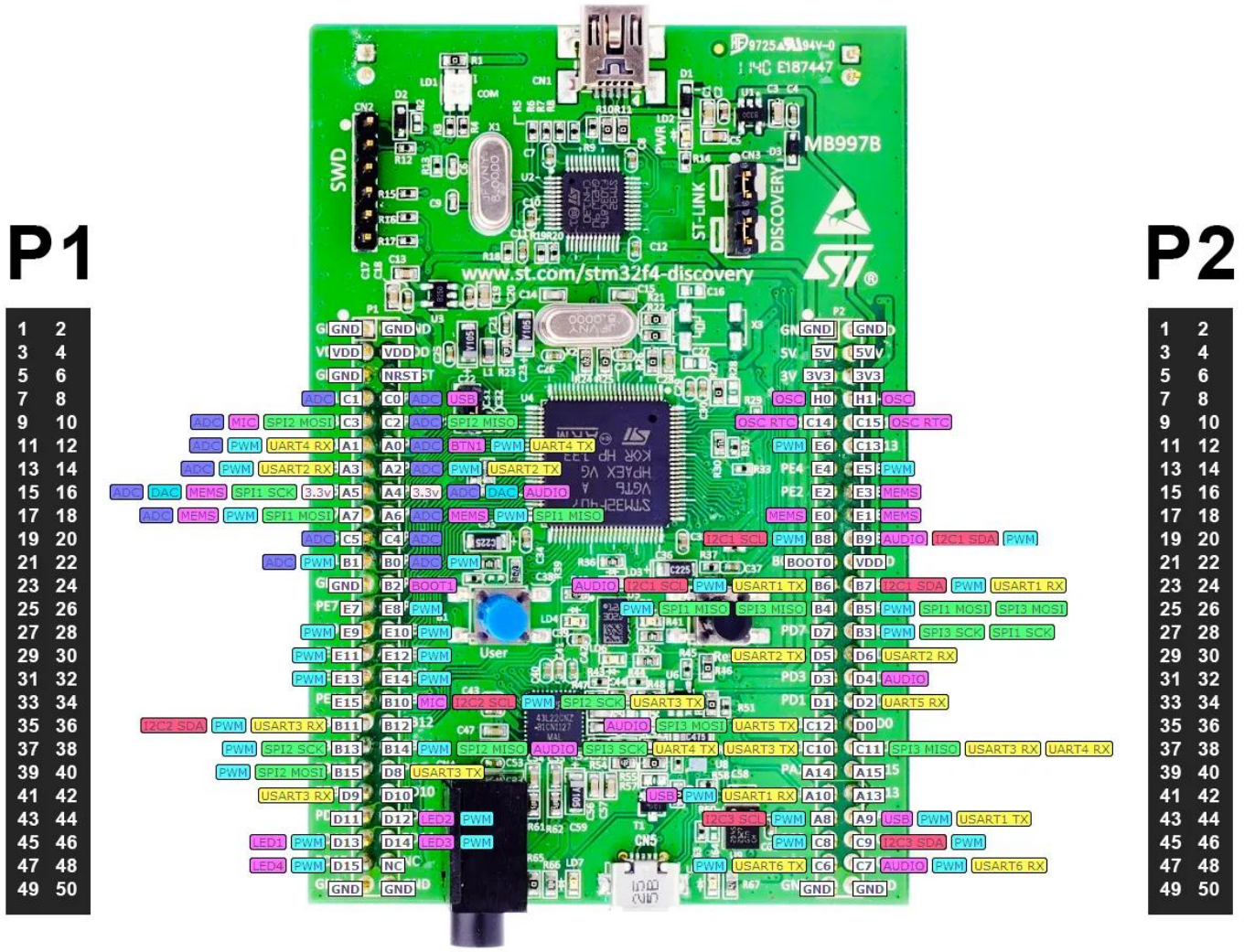
- Odchylka 0,98 splňuje zadání.

- `luxes *= zdroj ? 0.95 : 1.05;`
  - Korekční faktor podle typu světelného zdroje.

- `candela = luxes * 0.009290304;`
  - Převod luxů na kandely.

Hardware

STM32F407



Obr. 1

- Jedná se o mikroprocesor, který programujeme pomocí ST-LINK. Pracuje na frekvenci 16MHz, ale můžeme ho spustit až na 160MHz. Je 32 bitový. 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.

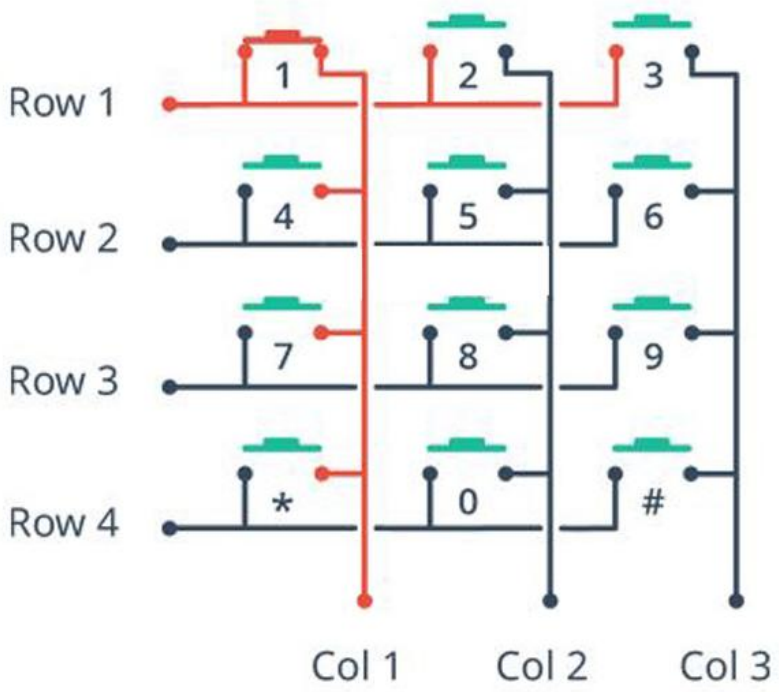
Interní led diody

- 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

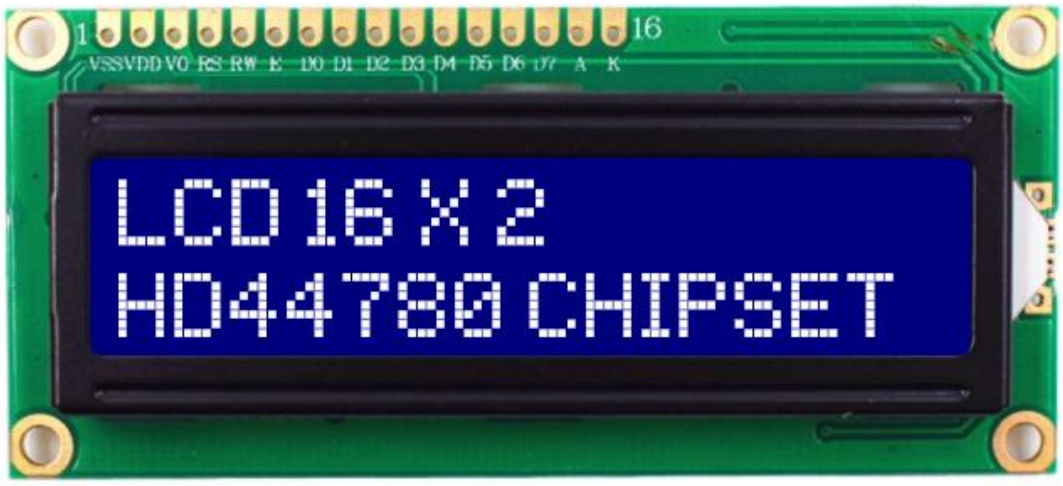


Obr. 2

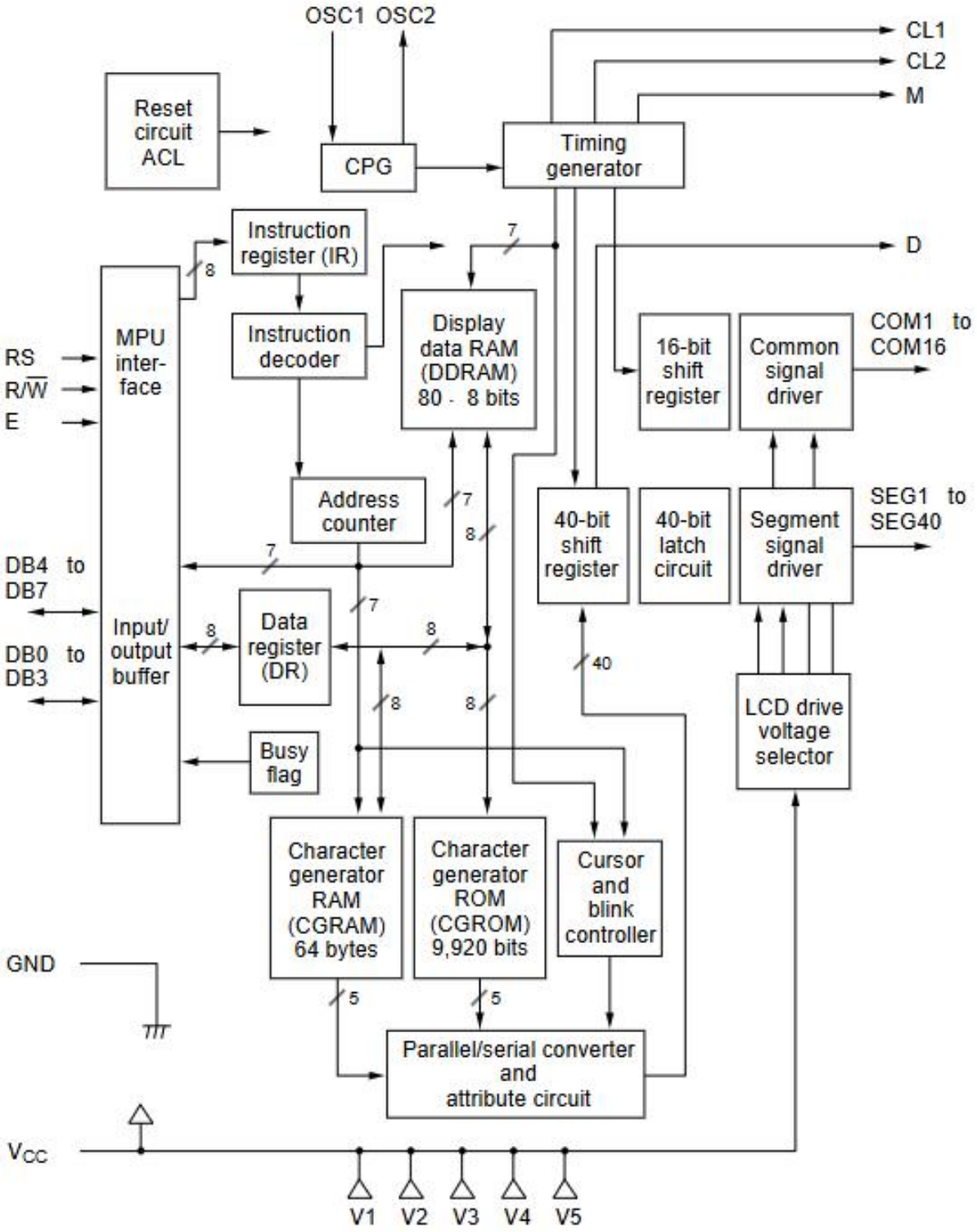
Na obrázku Obr.2 je vyobrazena klávesnice 3x4, já jsem použil 4x4. Z obrázku lze jednoduše pochopit princip funguvání. Je zmáčknuto tlačítko 1. Postupně za sebou přivademe na piny ROW1 - ROW4 napětí, a čteme piny COL1-COL-3. Podle toho, kde naměříme napětí tak pomocí souřadnice xy zjistíme které tlačítko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

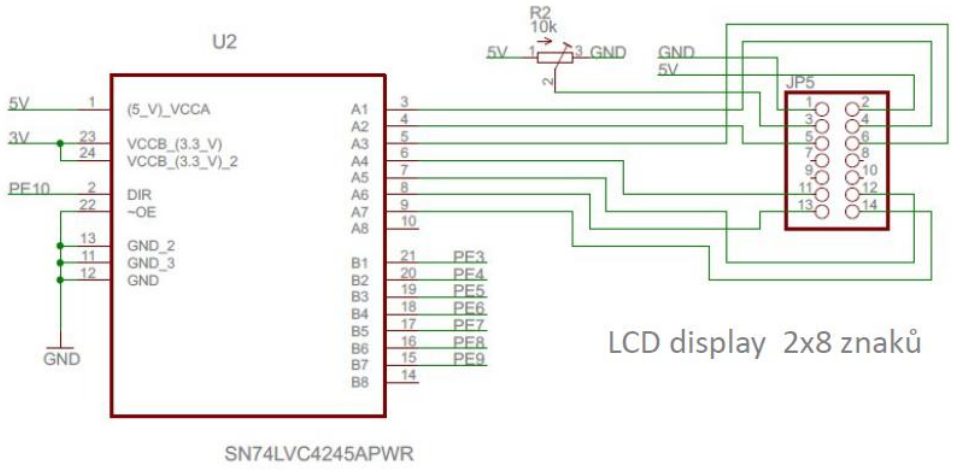
LCD 16x4



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



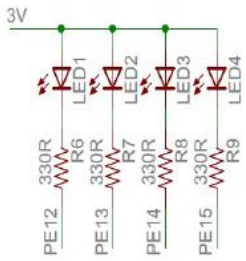
Obr. 5

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

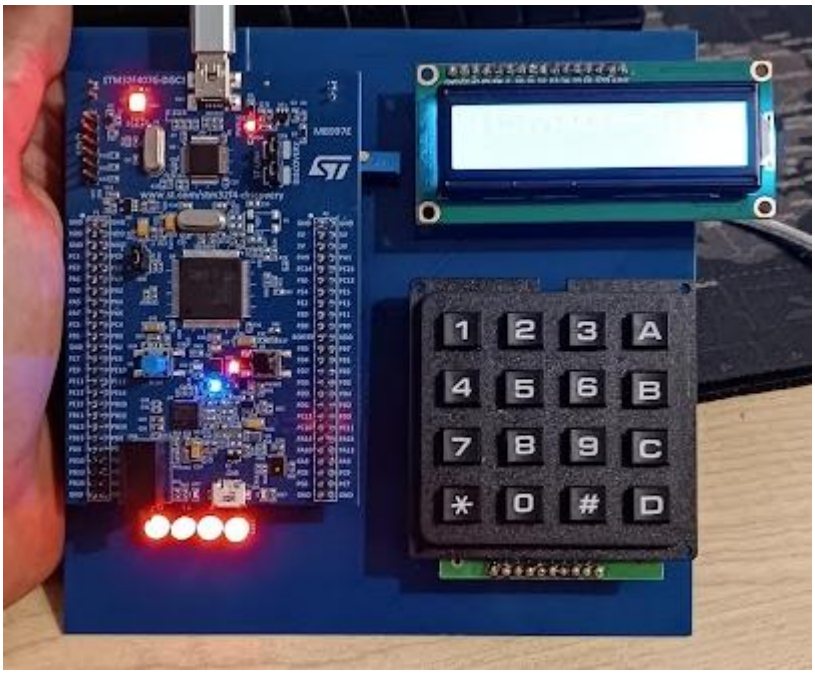


Obr. 6

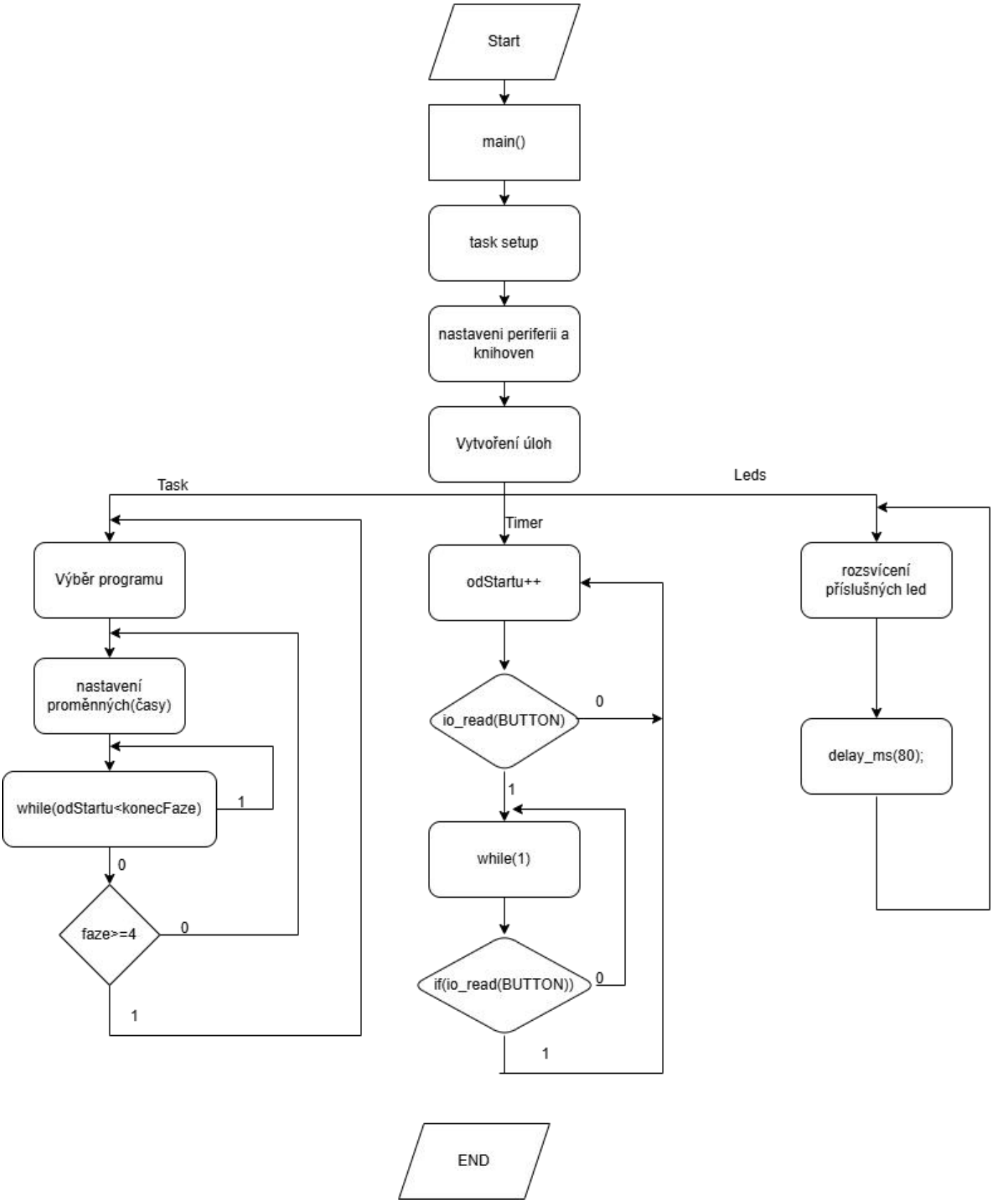
Externí LEDky(Obr.6) jsou zapojeny způsobem PULL_UP. To znamená že je zapínáme přivedením nuly na příslušné piny. Z Obr. 6 je zřejmé, že ledky jsou připojeny na PE12-PE15. Všechny jsou červené barvy.

DESKA-vlastní výroby

Deska je zjednodušení pro zapojení.



3. Vývojový diagram



4. program + blokový komentář

```c
/**
 * @file   luxmetr.c
 * @author <Vojtech Lisztwan>
 *
 * Program pro měření osvětlení pomocí ADC na STM32F4.
 * Umožňuje měřit osvětlení v luxech nebo candelách a zobrazuje hodnoty na LCD displeji.
 * Pokud je naměřená hodnota luxů vyšší než 400, rozsvítí všechny LED diody.
 */

#include "stm32_kit.h"
#include "stm32_kit/button.h"
#include "stm32_kit/lcd.h"
#include "stm32_kit/keypad.h"
#include "stm32_kit/adc.h"
#include "stm32_kit/led.h"
#include "stm32_kit/chrono.h"
#include <stdio.h>
#include <math.h>

// Mapování klávesnice na znaky
static uint8_t KBD_MAP[KEYPAD_ROWS][KEYPAD_COLS] = {
        '1', '2', '3', 'A',
        '4', '5', '6', 'B',
        '7', '8', '9', 'C',
        '*', '0', '#', 'D'
};

// Seznam vnitřních LED diod
enum pin interni[] = {
```

```
    LED_IN_0,
    LED_IN_1,
    LED_IN_2,
    LED_IN_3
};

// Seznam externích LED diod
enum pin externi[] = {
    LED_EX_3,
    LED_EX_2,
    LED_EX_1,
    LED_EX_0
};

/**
 * Nastavení LED diody na požadovanou hodnotu.
 * @param pin - pin LED diody
 * @param value - hodnota (0 = zhasnuto, 1 = rozsvíceno)
 */
void led_set(pin_t pin, int value) {
    int on = 1;
    if(io_port(pin) == io_port(LED_EX_0)) on = 0; // Pokud je LED externí, invertujeme logiku
    io_set(pin, on ? value : !value);
}

/**
 * Inicializace desky a periferií.
 */
BOARD_SETUP void setup(void) {
    SystemCoreClockUpdate();  // Aktualizace hodin systému
    SysTick_Config(SystemCoreClock / 10000); // Konfigurace časovače SysTick
```

```
    LCD_setup();  // Inicializace LCD
    KBD_setup();  // Inicializace klávesnice
    ADC_setup();  // Inicializace ADC převodníku
    BTN_setup();  // Inicializace tlačítek
    LED_setup();  // Inicializace LED diod
}

/**
 * Hlavní program.
 */
int main(void) {
    LCD_set(LCD_CLR);
    LCD_set(LCD_LINE1);

    uint16_t value;  // ADC hodnota
    double mili;     // Napětí v mV
    double sum;      // Součet vzorků ADC
    double luxes;    // Vypočtená hodnota osvětlení v luxech
    double candela;  // Vypočtená hodnota v candelách
    char buff[20];   // Buffer pro textový výstup
    int jednotka;    // Jednotka měření (0 = luxy, 1 = candely)
    int zdroj;       // Typ zdroje světla (0 = žárovka, 1 = slunce)
    uint8_t znak;    // Proměnná pro uchování stisknuté klávesy
    char jedn[20];   // Řetězec jednotky měření

    // Výběr jednotky měření
    LCD_set(LCD_CLR);
    LCD_set(LCD_LINE1);
    LCD_print("1-LUX");
    LCD_set(LCD_LINE2);
    LCD_print("2-Candel");
    do {
```

```
    znak = KBD_read();
    if(znak == '1') {
        jednotka = 0;
        break;
    }
    if(znak == '2') {
        jednotka = 1;
        break;
    }
} while(1);

// Výběr zdroje světla
LCD_set(LCD_CLR);
LCD_set(LCD_LINE1);
LCD_print("1 zarovka");
LCD_set(LCD_LINE2);
LCD_print("2-slunce");
do {
    znak = KBD_read();
    if(znak == '1') {
        zdroj = 0;
        break;
    }
    if(znak == '2') {
        zdroj = 1;
        break;
    }
} while(1);

// Hlavní smyčka měření
while (1) {
    sum = 0;
```

```
// Průměrování 128 vzorků z ADC
for(int i = 0; i < 128; i++) {
    value = ADC_read();  // Čtení hodnoty z ADC
    sum += value;
    delay_ms(1);  // Krátká prodleva
}

// Převod ADC hodnoty na napětí v mV
mili = ((sum / 128) * 3000) / 4096;

// Výpočet luxů podle empirického vzorce
luxes = 12.8 * exp(0.0025 * mili);

// Korekce hodnoty podle zdroje světla
luxes *= zdroj ? 0.95 : 1.05;

// Převod luxů na candely
candela = luxes * 0.009290304;

// Zobrazení napětí na LCD
sprintf(buff, "%4.0f mV", mili);
LCD_set(LCD_LINE1);
LCD_print(buff);

// Nastavení jednotky pro zobrazení
if(jednotka) {
    sprintf(jedn, "Can");
} else {
    sprintf(jedn, "Lux");
}
```

```
            // Zobrazení vypočtené hodnoty na LCD
            sprintf(buff, "%4.0f %s", jednotka ? candela : luxes, jedn);
            LCD_set(LCD_LINE2);
            LCD_print(buff);

            delay_ms(200);  // Krátká prodleva

            // Rozsvícení LED při vysoké hodnotě luxů
            if(luxes > 400) {
                for(int i = 0; i < 4; i++) {
                    led_set(interni[i], 1);
                    led_set(externi[i], 1);
                }
            } else {
                for(int i = 0; i < 4; i++) {
                    led_set(interni[i], 0);
                    led_set(externi[i], 0);
                }
            }
        }
    }
```

5. 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, přídat 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.