Vyšší odborná škola a Střední průmyslová škola elektrotechnická
Božetěchova 3, Olomouc

Samostatný projekt MIT

Název úlohy	Číslo úlohy
Sun tracking, solární nabíječka	-

Zadání

1. Každý student odevzdá dvě až tři projektové úlohy, dle vlastního výběru a volby
2. Dohromady musí každý student získat 7 bodů.
3. Každá periferie se počítá **jen jednou**

Vypracování:

- Schéma zapojení -- KiCad
- Slovní popis zapojení
- Vývojový diagram programu -- blokově
- Blokové schéma
- Slovní popis funkce programu
- Zdrojové kódy (Céčko i KiCad i vše ostatní...) budou **jako příloha**, soubor main.c vložíte na konec textové části (až za zhodnocení)
- Zhodnocení: výhody a nevýhody, dostatky a nedostatky, výhled do budoucna

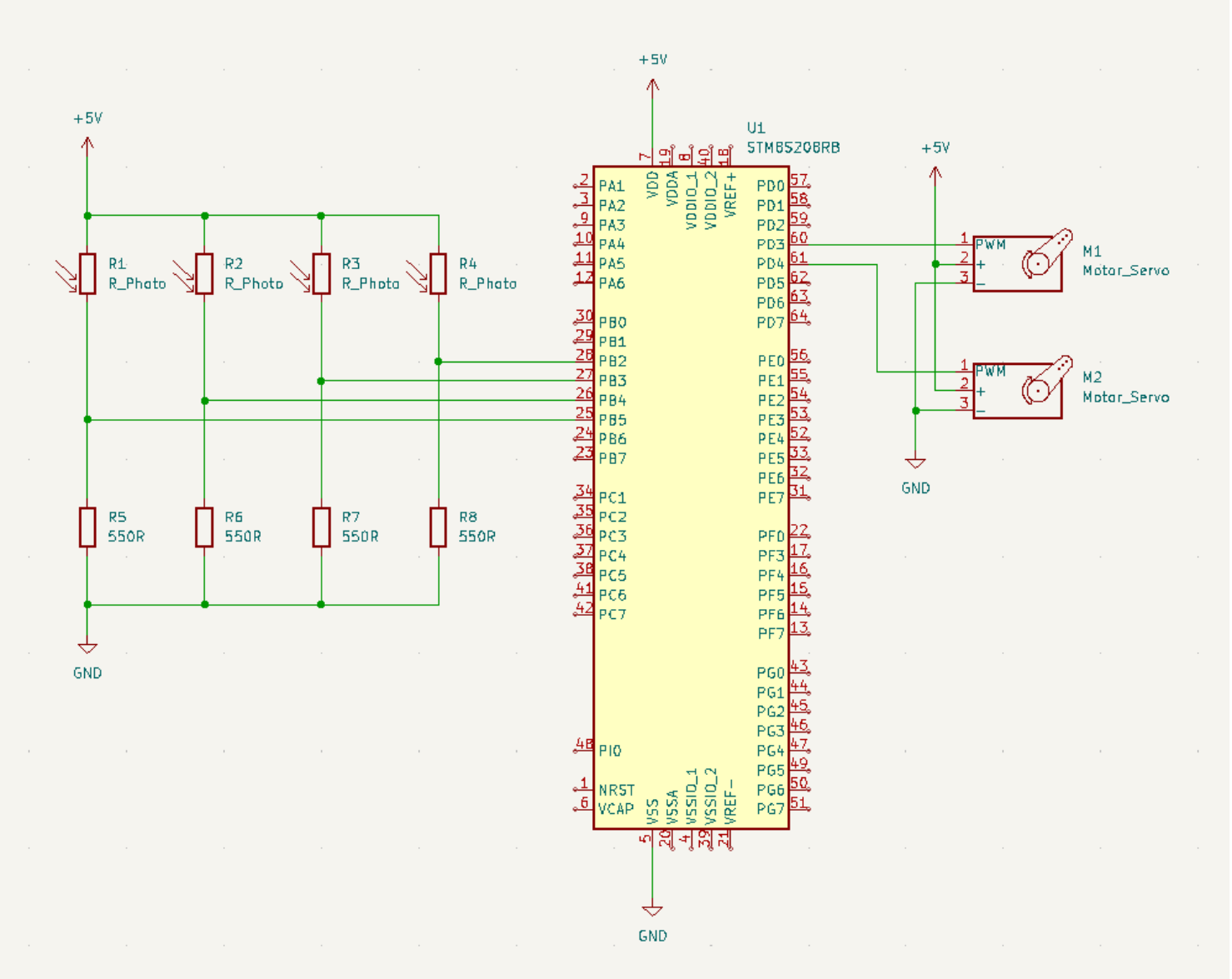
"Sun tracking" solární nabíječka

- Solární panel umístěný na servomotorku, mikrokontrolér snímá polohu slunce na obloze a natáčí panel tak aby zachytával maximální množství energie
- Potřeby: solární panel, servomotorek, mechanika

Poř. č.	Příjmení a jméno	Třída	Skupina	Školní rok
3	DVOŘÁK Roman	4B	1	2021/22

Datum vypracování	Datum odevzdání	Počet listů	Klasifikace			
			příprava	měření	protokol	obhajoba
13. 2. 2022	13. 2. 2022	7				

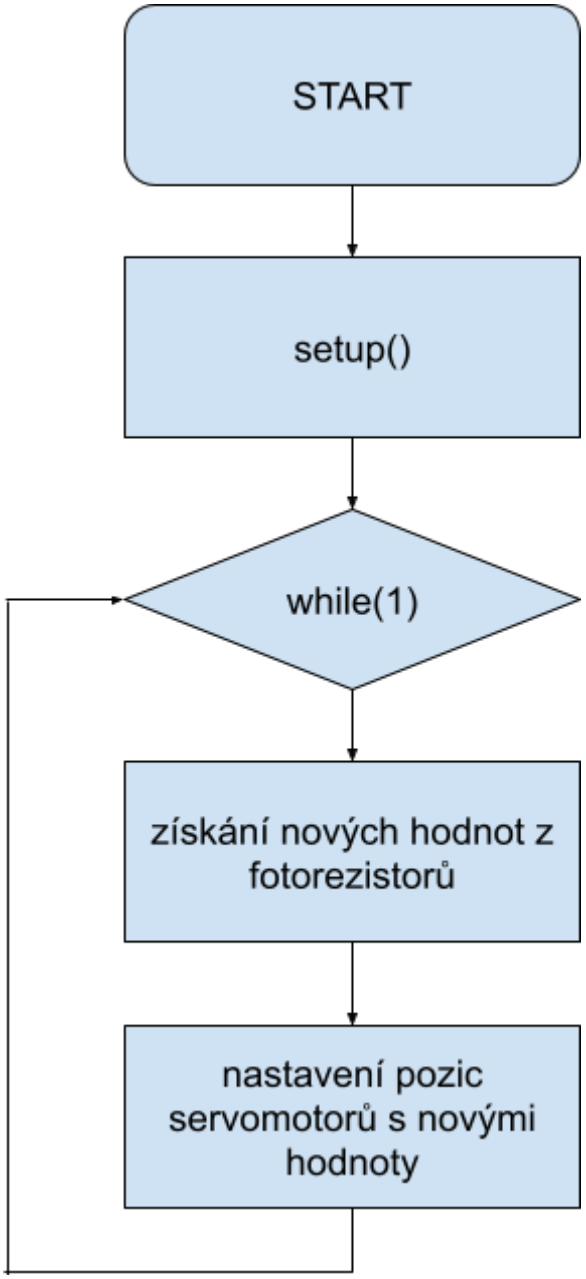
Schéma zapojení:



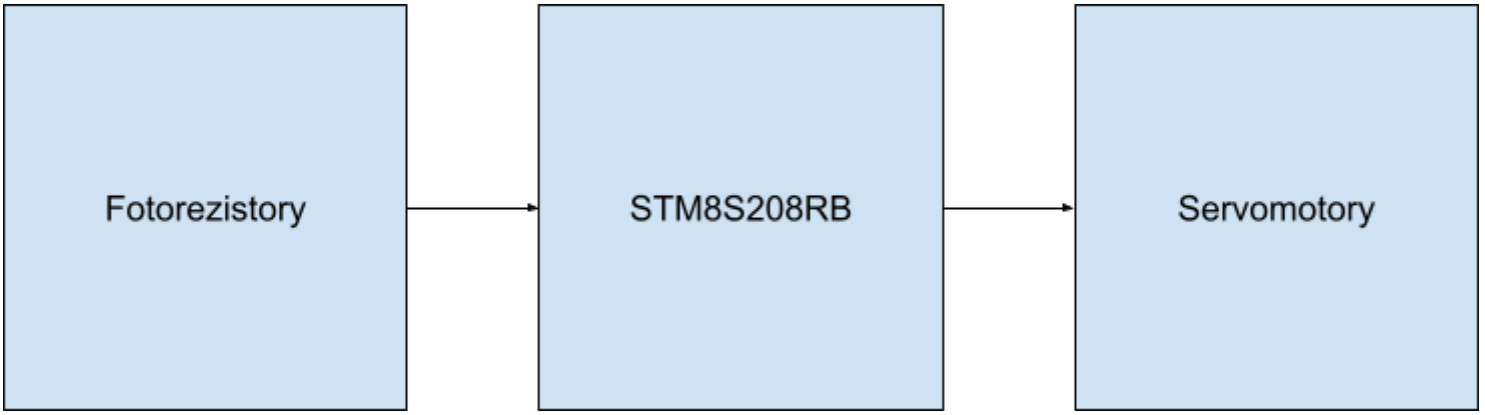
Slovní popis zapojení:

První servo je připojený na pin PD3 a +5V, GND. Zatímco druhý servo je připojený na PD4 a +5V a GND. Fotorezistory jsou zezhora připojený na +5V a ze zdola jsou propojený s pinem na STM(první je na pinu PB2, druhý je na pinu PB3, třetí je na pinu PB4 a čtvrtý je na pinu PB5) a také s odporem (550R) který je ze zdola propojený na GND.

Vývojový diagram programu:



Blokové schéma:



Slovní popis funkce programu:

Fotorezistory jsou připojené na ADC vstupy, tam dají programu hodnotu kolik je tam světla. Podle této hodnoty program vyhodnotí zda má serva posunout nahoru, dolů, doleva a nebo doprava. Program jde do nekonečna a nikdy nemá konec, je to tak z toho důvodu aby mohl pořád upravovat hodnoty na servech.

Zhodnocení:

Je to strašně pěkný projekt ale myslím si že má pár nevýhod. Jedna z věcí které se mi moc nelíbily byla ta že nakonec jsem přepojil serva ze zdroje t STMka na zdroj oddělený neboť STMka prostě nejsou schopny dávat dostatečný proud aby se serva hýbala. Dále můj výběr serva také nebyl zrovna nejlepší. Kdybych to měl dělat znovu tak bych si vybral serva které jsou o něco silnější. Až na tyto nedostatky bych ale řekl že je to velmi pěkný projekt a že je také velmi zajímavý.

Zdrojový kód:

```
#include "stm8s.h"
#include "milis.h"
#include "spse_stm8.h"

#include <stdio.h>
#include "stm8s_adc2.h"
#include "uart1.h"

#define _ISOC99_SOURCE
#define _GNU_SOURCE

void tim2_setup(void){
     TIM2_TimeBaseInit(TIM2_PRESCALER_8, 40000);
    //TIM2_ITConfig(TIM2_IT_UPDATE, ENABLE);
    TIM2_OC1Init(                // inicializujeme kanál 1 (TM2_CH1) PD4
        TIM2_OCMODE_PWM1,        // režim PWM1
        TIM2_OUTPUTSTATE_ENABLE, // Výstup povolen (TIMer ovládá pin)
```

```
        3500,                   // výchozí hodnota šířky pulzu (střídy) 1056/1600 =
66%
        TIM2_OCPOLARITY_HIGH      // Polarita LOW protože LED rozsvěcím 0 (spol. anoda)
     );
    TIM2_OC2Init(TIM2_OCMODE_PWM1,TIM2_OUTPUTSTATE_ENABLE,
                 4000, TIM2_OCPOLARITY_HIGH); //PD3
    // ošetření nežádoucích jevů při změně PWM
    TIM2_OC1PreloadConfig(ENABLE);
    TIM2_OC2PreloadConfig(ENABLE);

    TIM2_Cmd(ENABLE);
}

void setup(void)
{
    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);      // taktovani MCU na 16MHz
    GPIO_Init(LED_PORT, LED_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(BTN_PORT, BTN_PIN, GPIO_MODE_IN_FL_NO_IT);

    init_milis();
    tim2_setup();

    // inicializace ADC //
    // na pinech/vstupech ADC_IN2 (PB4) a ADC_IN3 (PB5) vypneme vstupní buffer
    ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL2, DISABLE);
    ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL3, DISABLE);
    ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL4, DISABLE);
    ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL5, DISABLE);

    // při inicializaci volíme frekvenci AD převodníku mezi 1-4MHz při 3.3V
    // mezi 1-6MHz při 5V napájení
    // nastavíme clock pro ADC (16MHz / 4 = 4MHz)
    ADC2_PrescalerConfig(ADC2_PRESSEL_FCPU_D4);

    // volíme zarovnání výsledku (typicky vpravo, jen vyjmečně je výhodné vlevo)
    ADC2_AlignConfig(ADC2_ALIGN_RIGHT);

    // nasatvíme multiplexer na některý ze vstupních kanálů
    ADC2_Select_Channel(ADC2_CHANNEL_2);
    ADC2_Select_Channel(ADC2_CHANNEL_3);
    ADC2_Select_Channel(ADC2_CHANNEL_4);
    ADC2_Select_Channel(ADC2_CHANNEL_5);
    // rozběhneme AD převodník
    ADC2_Cmd(ENABLE);
```

```
    // počkáme než se AD převodník rozběhne (~7us)
    ADC2_Startup_Wait();
}

void delay_ms(uint16_t ms) {
    uint16_t  i;
    for (i=0; i<ms; i = i+1){
        _delay_us(250);
        _delay_us(248);
        _delay_us(250);
        _delay_us(250);
    }
}

    uint32_t time = 0;
    uint16_t topright;
    uint16_t topleft;
    uint16_t downleft;
    uint16_t downright;
    uint16_t OCR1A;
    uint16_t OCR1B;

int main(void)
{

    setup();

    while (1) {

        topright = ADC_get(ADC2_CHANNEL_4);
        topleft = ADC_get(ADC2_CHANNEL_5)+20;
        downleft = ADC_get(ADC2_CHANNEL_3);
        downright = ADC_get(ADC2_CHANNEL_2);

        if (topleft > topright) {
            OCR1A = OCR1A - 1;
            delay_ms(1);
        }
        if (downleft > downright) {
            OCR1A = OCR1A - 1;
            delay_ms(1);
        }
        if (topleft < topright) {
            OCR1A = OCR1A + 1;
```

```
            delay_ms(1);
        }
        if (downleft < downright) {
            OCR1A = OCR1A + 1;
            delay_ms(1);
        }
        if (OCR1A > 4900) {
            OCR1A = 4900;
        }
        if (OCR1A < 2100) {
            OCR1A = 2100;
        }
        if (topleft > downleft) {
            OCR1B = OCR1B + 1;
            delay_ms(1);
        }
        if (topright > downright) {
            OCR1B = OCR1B + 1;
            delay_ms(1);
        }
        if (topleft < downleft) {
            OCR1B = OCR1B - 1;
            delay_ms(1);
        }
        if (topright < downright) {
            OCR1B = OCR1B - 1;
            delay_ms(1);
        }
        if (OCR1B > 4900) {
            OCR1B = 4900;
        }
        if (OCR1B < 3000) {
            OCR1B = 3000;
        }

        TIM2_SetCompare1(OCR1A);
        TIM2_SetCompare2(OCR1B);

    }
}

/*------------------------------  Assert -----------------------------------*/
#include "__assert__.h"
```