Vyšší odborná škola a
Střední průmyslová škola elektrotechnická
Božetěchova 3, 772 00 Olomouc

Samostatný projekt MIT

Název projektu	Číslo projektu
Laserové ukazovátko	MIT 01

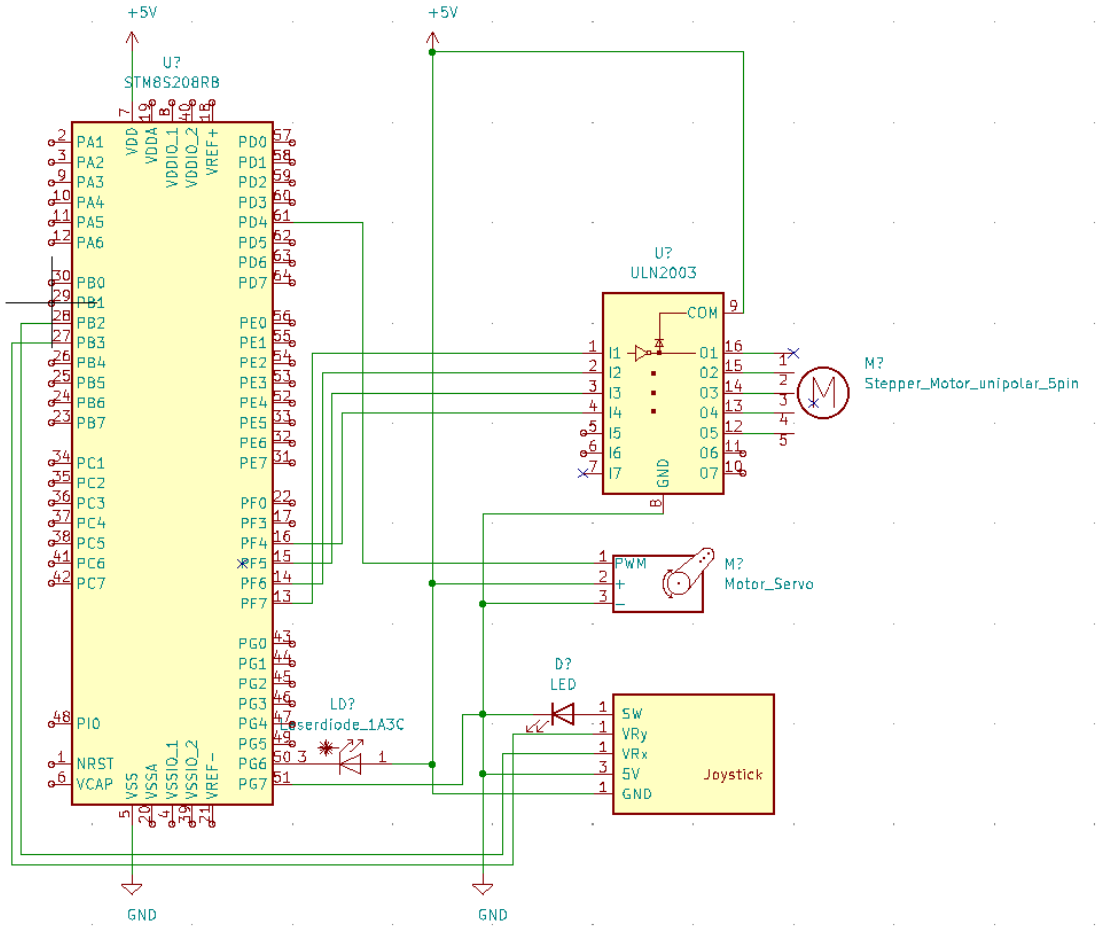
Zadání
Projekty lze realizovat pomocí vývojového kitu anebo modulem s vlastním mikrokontrolerem například na pájecím poli.
Kompletní vlastní konstrukce (bez vývojového kitu) bude lépe hodnocená. K zhotovení vlastní konstrukce budete potřebovat modul s mikrokontrolerem a programátor/debugger. Případně můžete zhotovit i vlastní DPS na školní fréze, pak místo modulu s mikrokontrolerem stačí samostatný mikrokontroler.

Vypracování:
Schéma zapojení -- KiCad
Slovní popis zapojení
Vývojový diagram programu -- blokově
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
Co jsem se naučil a v čem to pro mě mělo přínos a jak to vidím dál?

Poř. č.	Příjmení a jméno	Třída	Školní rok
17	Prokop Spurný	4B	2021/22

Datum vypracování	Datum odevzdání	Počet listů	Klasifikace
13.02.2022	13.02.2022	7	

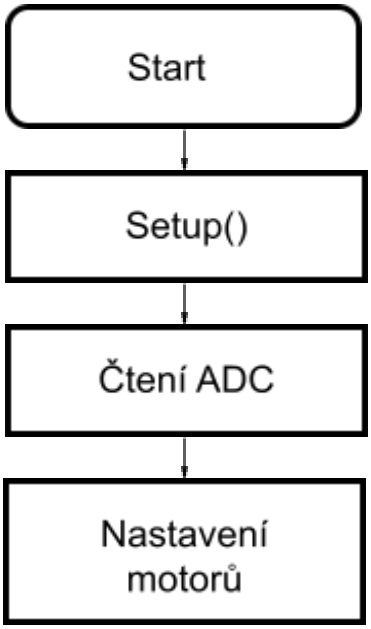
1)Schéma zapojení



2)Slovní popis zapojení

K stm8 je připojen joystick hodnoty z os x a y jsou připojeny na B2 a B3 tlačítko je připojeno na G7 napájení je připojeno naopak než říká deska. Krokový motor je připojen na napájení a PWM je na pin D4. Řadič krokového motoru je připojen na napájení a signáli dostává z pinů F7 - F4. Laser je připojen na G6 a na 5V. Signalizační dioda tlačítka je připojena na SW a na GND.

3)Vývojový diagram



4)Blokové schéma

5)Slovní popis funkce programu

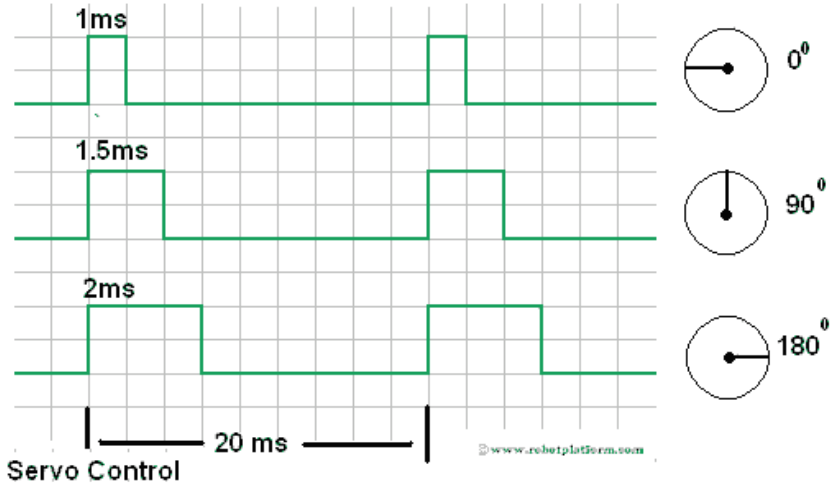
Na začátku programu jsem vytvořil makra pro výstupy. Void setup nastaví vstupy a výstupy také zavolá adc, milis, tim2. Ve void main vytvoříme proměnné a zavoláme setup a uart. While je nekonečně se opakující smyčka ve které nejprve načte hodnotu adc a uložíme ji do adc_valuex nebo y. Hodnoty se vyhodnotí ve 4 if funkcích které rozhodují jestli se pohnou motory na osách x,y. V poslední If funkci je LEDa_TOGG; což přepíná laser.

Krokový motor:

U krokového motoru jsem použil wave drive a vypínám a zapínám piny F7 - F4 podle tabulky.

METHODS	PHASES	1	2	3	4	5	6	7	8
WAVE DRIVE One phase at a time Simplest, but least used	BLUE								
	PINK								
	YELLOW								
	ORANGE								

Servo:
Řídíme pomocí pwm které nastavují od 1000 do 5000 tímto nastavuje me dobu v H.



6)3D tisk
Modely jsem vytvořil pomocí programu Blender. Stl jsem protáhl Prusaslicerem. Tiskl jsem na prusa mini+. Použitý filament je prusament PLA.

7)Zhodnocení
Tento projekt si myslím že mě toho hodně naučil. vyzkoušel jsem si práci z mnoha prvky se kterými jsem v obvodu nikdy nepracoval jako servo nebo krokový motor. a tím jsem zjistil jak fungují a jaké jsou jejich limitace.
Výhoda nevím jestli tento projekt má přímo nějaké výhody stavěl jsem ho tak aby ověřil funkčnost a možnosti použití reálné použití asi nemá ale určitě by se dal modifikovat do něčeho použitelného.
Nevýhoda nedostatky tohoto systému jsou asi to že jsem použil levný krokový motor který se zasekne o překážku a nemá skoro žádnou sílu. Také by možná bylo dobré vytvořit konstrukci která by držela motor osy x který se momentálně může hýbat kam se mu zachce.
Výhled do budoucna možná bych mohl tento projekt vylepšit nahrazením joysticku gyroskopem a přidání displeje by mohl být dobrý projekt na druhé pololetí.

Kód:

```
#include "stm8s.h"
#include "milis.h"
#include "stm8s_i2c.h"
#include "delay.h"
#include "spse_stm8.h"
#include <stdio.h>

#include "../lib/uart.c"

#define _ISOC99_SOURCE
#define _GNU_SOURCE

#define LED_PORT GPIOC
#define LED_PIN  GPIO_PIN_5
#define LED_HIGH   GPIO_WriteHigh(LED_PORT, LED_PIN)
#define LED_LOW  GPIO_WriteLow(LED_PORT, LED_PIN)
#define LED_TOGG GPIO_WriteReverse(LED_PORT, LED_PIN)



#define LEDa_PORT GPIOG
#define LEDa_PIN  GPIO_PIN_6
#define LEDa_HIGH   GPIO_WriteHigh(LEDa_PORT, LEDa_PIN)
#define LEDa_LOW  GPIO_WriteLow(LEDa_PORT, LEDa_PIN)
#define LEDa_TOGG GPIO_WriteReverse(LEDa_PORT, LEDa_PIN)
////////////////////////////////////////////////////////////// stepper motor
#define A1_PORT GPIOF
#define A1_PIN  GPIO_PIN_7
#define A1_HIGH   GPIO_WriteHigh(A1_PORT, A1_PIN)
#define A1_LOW  GPIO_WriteLow(A1_PORT, A1_PIN)
#define A1_R GPIO_WriteReverse(A1_PORT, A1_PIN)

#define A2_PORT GPIOF
#define A2_PIN  GPIO_PIN_6
#define A2_HIGH   GPIO_WriteHigh(A2_PORT, A2_PIN)
#define A2_LOW  GPIO_WriteLow(A2_PORT, A2_PIN)
#define A2_R GPIO_WriteReverse(A2_PORT, A2_PIN)

#define A3_PORT GPIOF
#define A3_PIN  GPIO_PIN_5
```

```
#define A3_HIGH   GPIO_WriteHigh(A3_PORT, A3_PIN)
#define A3_LOW  GPIO_WriteLow(A3_PORT, A3_PIN)
#define A3_R GPIO_WriteReverse(A3_PORT, A3_PIN)

#define A4_PORT GPIOF
#define A4_PIN  GPIO_PIN_4
#define A4_HIGH   GPIO_WriteHigh(A4_PORT, A4_PIN)
#define A4_LOW  GPIO_WriteLow(A4_PORT, A4_PIN)
#define A4_R GPIO_WriteReverse(A4_PORT, A4_PIN)
////////////////////////////////////////////////////////////////////////////////
//
#define BTN_PORT GPIOG
#define BTN_PIN  GPIO_PIN_7
#define BTN_PUSH (GPIO_ReadInputPin(BTN_PORT, BTN_PIN))

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);
    }
}

void tim2_setup(void){
     TIM2_TimeBaseInit(TIM2_PRESCALER_8, 40000);
    //TIM2_ITConfig(TIM2_IT_UPDATE, ENABLE);
    TIM2_OC1Init(               // inicializujeme kanál 1 (TM2_CH1)
        TIM2_OCMODE_PWM1,       // režim PWM1
        TIM2_OUTPUTSTATE_ENABLE, // Výstup povolen (TIMer ovládá pin)
        3000,                   // výchozí hodnota šířky pulzu (střídy)
1056/1600 = 66%
        TIM2_OCPOLARITY_HIGH     // Polarita LOW protože LED rozsvěcím 0
(spol. anoda)
     );


     TIM2_OC1PreloadConfig(ENABLE);
```

```
     TIM2_Cmd(ENABLE);
}


void ADC_init(void){
// na pinech/vstupech ADC_IN2 (PB2) a ADC_IN3 (PB3) vypneme vstupní buffer
ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL2,DISABLE);
ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL3,DISABLE);
// 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);
// rozběhneme AD převodník
ADC2_Cmd(ENABLE);
// počkáme než se AD převodník rozběhne (~7us)
ADC2_Startup_Wait();
}



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(LEDa_PORT, LEDa_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);

    GPIO_Init(A1_PORT, A1_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(A2_PORT, A2_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(A3_PORT, A3_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(A4_PORT, A4_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(GPIOB, GPIO_PIN_4,GPIO_MODE_IN_FL_NO_IT);

    tim2_setup();
    init_milis();
    ADC_init();
}
```

```c
int main(void)
{
    uint32_t time = 0;
    uint32_t ss = 0;
    uint32_t cum = 3000;
    //init();
    uint16_t adc_valuex;
    uint16_t adc_valuey;

    setup();
    init_uart();

    while (1) {

        adc_valuex = ADC_get(ADC2_CHANNEL_2); // do adc_value ulož výsledek
převodu vstupu ADC_IN2 (PB2)
        adc_valuey = ADC_get(ADC2_CHANNEL_3);

        printf("x:""%d",adc_valuex );
        printf("  y:""%d",adc_valuey );

        if (adc_valuex > 1000) {
            printf("  LEFT ");

            //1 step
            A1_HIGH;
            delay_ms(3);
            //2 step
            A1_LOW;
            A2_HIGH;
            delay_ms(3);
            //3 step
            A2_LOW;
            A3_HIGH;
            delay_ms(3);
            //4 step
            A3_LOW;
            A4_HIGH;
            delay_ms(3);
```

```
            A4_LOW;

        }
        if (adc_valuex < 500) {
            printf("  RIGHT ");

            //1 step
            A4_HIGH;
            delay_ms(3);
            //2 step
            A4_LOW;
            A3_HIGH;
            delay_ms(3);
            //3 step
            A3_LOW;
            A2_HIGH;
            delay_ms(3);
            //4 step
            A2_LOW;
            A1_HIGH;
            delay_ms(3);
            A1_LOW;

        }
        if (adc_valuey < 500) {
            printf("  DOWN ");

            TIM2_SetCompare1(cum);
            if (cum < 5000) {
            cum += 2;
            }

        }
        if (adc_valuey > 1000) {
            printf("  UP ");

            TIM2_SetCompare1(cum);
            if (cum > 1000) {
            cum -= 2;
            }
```

```
        }

        if( BTN_PUSH){
            if (ss){
            LEDa_TOGG;
            ss = 0;
            }
        }
        else{
            ss = 1;
        }
        printf("\r\n");

    }
}

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