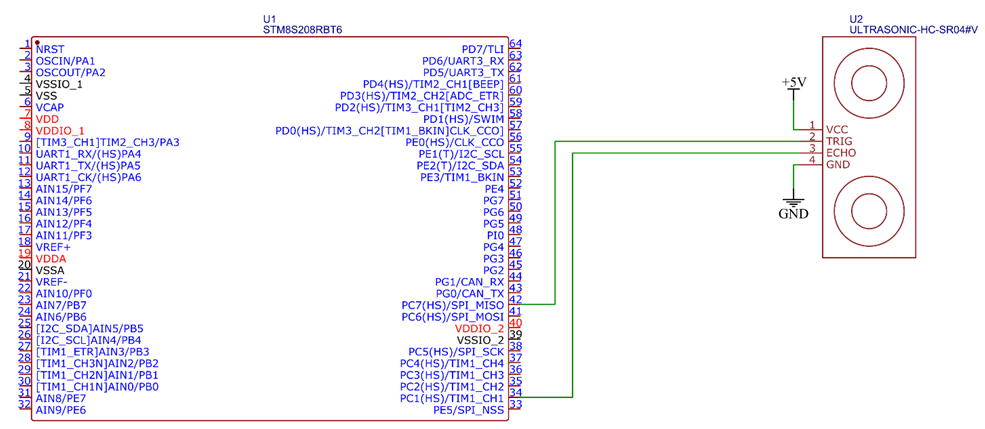
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | | |
| **Samostatný projekt MIT** | | | | | | | | | | | | | | |
| Název projektu | |  | | | | | | | | | | Číslo projektu | |  |
| Ultrazvukový měřič vzdálenosti | | | | | | | | | | | | 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 -- EasyEDA  Slovní popis zapojení  Vývojový diagram programu -- blokově  Slovní popis funkce programu  Zdrojové kódy (Céčko i EasyEDA 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?  **Ultrazvukový měřič**   1. Měření vzdálenosti   Potřeby: ultrazvukový měřič vzdálenosti | | | | | | | | | | | | |  | |
| Poř. č. | Příjmení a jméno | | | |  | | | | Třída | | Školní rok | | |  |
| 30 | Závodský Ondřej | | | | | | | | 4B | | 2021/22 | | | |
| Datum vypracování | | |  | Datum odevzdání | |  | Počet listů |  | Klasifikace |  | | | | |
| 25. 4. 2022 | | | | 26. 4. 2022 | | | 6 | |  | | | | | |

**1)Schéma zapojení**

****

**2)Vývojový diagram**

**START**

**init milis() + setup()**

**If(**

**while==1**

**)**

**Zapsání vzdálenosti pomocí UARTu**

**3)Blokové schéma**

**HC-SR04 STM8S208RB UART**

**4)Slovní popis funkce programu**

Při zapnutí se v hlavní funkci zapne init\_milis(). Ta začne počítat čas do proměnné milis.

setup() nastaví prescaler a inicializuje lcd. GPIO\_setup() nastaví nezbytné piny.   
 Ve whilu se jako první příjde na funkci proccess\_measurment(). Toto je na čem celý kód stojí. V prvním case se chvíli čeká, aby měření probíhalo v rychlosti vhodné pro ultrazvukový měřič, dále pouze jako UM. Na konci case 0 se vyšle trigger k UM.

V case 1 se již měří vzdálenost. Po uplynutí pulzu se trigger přestane vysílat.

V posledním case se čeká na vlajku vyvěšenou tim1, rozdíl času za zapíše do proměné capture a celý proces se opakuje. Jsou zde i pojistky, kdyby se něco pokazilo

**5)Slovní popis zapojení**

Stm8 samotné je zapojeno skrz usb na napájení. Vzdálenost se zobrazuje pomocí UARTu.

**6)Zhodnocení**

Výhoda takovýchto měřiče je, že s ním lze bezkontaktně měřit. Nevýhody jsou v ostatních vlastnostech, jako cena, spolehlivost a přesnost.

Žádné skutečné nedostatek tohoto systému mě nenapadají, ale mohl jsem přidat řádek kde by se ukládal posledně naměřená hodnota.

Naučil jsem se, jak hledat informace pro stm8 a prakticky jsem si vyzkoušel práci s flagy.

Možnost zlepšení:

Jelikož jediné, co měřič může dělat je měřit, tak není, co na něm zlepšovat. Avšak co bys se dalo přidat je pomůcka pro reálnou zákožku a permanentní konstrukce např.: měřič hladiny.

**Kód:**

#include "stm8s.h"

#include "milis.h"

#include "stm8\_hd44780.h"

#include "delay.h"

#include <stdio.h>

#define \_ISOC99\_SOURCE

#define \_GNU\_SOURCE

#define PULSE\_LEN 2             // délka spouštěcího (trigger) pulzu pro ultrazvuk

#define MEASURMENT\_PERIOD 100   // perioda měření ultrazvukem (měla by být víc jak (maximální\_dosah\*2)/rychlost\_zvuku)

uint16\_t capture = 1;               // tady bude aktuální výsledek měření (času)

uint8\_t capture\_flag = 0;   //tady se ukládá zda bylo provedeno měření času

uint16\_t vzdalenost = 0;

uint8\_t minuly\_stav=1; // zde si budeme ukládat minulý stav tlačítka (1=tlačítko stisknuté, 0=tlačítko uvolněné)

uint8\_t aktualni\_stav=1;  // zde si budeme ukládat aktuální stav tlačítka (1=tlačítko stisknuté, 0=tlačítko uvolněné)

// u funkce GPIO\_ReadInputPin, hodnota RESET znamená log.0 a tedy stisknuté tlačítko

static uint8\_t stage = 0;   // stavový automat

static uint16\_t time = 0;   // pro časování pomocí milis

//static nejde dát do funkce => build-STM8S208/main.asm:1621: Error: <r> relocation error

void setup(void){

    CLK\_HSIPrescalerConfig(CLK\_PRESCALER\_HSIDIV1);// taktování na 16MHz

    lcd\_init(); //inicializace LCD

}

void GPIO\_setup(void){

    GPIO\_Init(GPIOC, GPIO\_PIN\_7, GPIO\_MODE\_OUT\_PP\_LOW\_SLOW); //trig

    //GPIO\_Init(GPIOD, GPIO\_PIN\_4, GPIO\_MODE\_IN\_FL\_NO\_IT); // echo

    GPIO\_Init(GPIOC,GPIO\_PIN\_5,GPIO\_MODE\_OUT\_PP\_LOW\_SLOW); // nastavíme PC5 jako výstup typu push-pull (LEDka)

    GPIO\_Init(GPIOE, GPIO\_PIN\_4,GPIO\_MODE\_IN\_FL\_NO\_IT); // nastavíme PE4 jako vstup (tlačítko)

}

void TIM1\_setup(void)

{

       TIM1\_TimeBaseInit(15, TIM1\_COUNTERMODE\_UP, 0xffff, 0);      // timer necháme volně běžet (do maximálního stropu) s časovou základnou 1MHz (1us)

// Konfigurujeme parametry capture kanálu 1 - komplikované, nelze popsat v krátkém komentáři

    TIM1\_ICInit(TIM1\_CHANNEL\_1, TIM1\_ICPOLARITY\_RISING,

                TIM1\_ICSELECTION\_DIRECTTI, TIM1\_ICPSC\_DIV1, 0);

// Konfigurujeme parametry capture kanálu 2 - komplikované, nelze popsat v krátkém komentáři

    TIM1\_ICInit(TIM1\_CHANNEL\_2, TIM1\_ICPOLARITY\_FALLING,

                TIM1\_ICSELECTION\_INDIRECTTI, TIM1\_ICPSC\_DIV1, 0);

    TIM1\_SelectInputTrigger(TIM1\_TS\_TI1FP1);    // Zdroj signálu pro Clock/Trigger controller

    TIM1\_SelectSlaveMode(TIM1\_SLAVEMODE\_RESET); // Clock/Trigger má po příchodu signálu provést RESET timeru

    TIM1\_ClearFlag(TIM1\_FLAG\_CC2);      // pro jistotu vyčistíme vlajku signalizující záchyt a změření echo pulzu

    TIM1\_Cmd(ENABLE);

}

void process\_measurment(void){

    switch (stage) {

    case 0:

                          // čekáme než uplyne  MEASURMENT\_PERIOD abychom odstartovali měření

        if (milis() - time > MEASURMENT\_PERIOD \* 4) {

            time = milis();

            GPIO\_WriteHigh(GPIOC, GPIO\_PIN\_7);  // zahájíme trigger pulz

            stage = 1;          // a budeme čekat až uplyne čas trigger pulzu;

        }

        break;

    case 1:

                            // čekáme než uplyne PULSE\_LEN (generuje trigger pulse)

        if (milis() - time > PULSE\_LEN \* 4) {

            GPIO\_WriteLow(GPIOC, GPIO\_PIN\_7);

   // ukončíme trigger pulz

            stage = 2;          // a přejdeme do fáze kdy očekáváme echo

        }

        break;

    case 2:

           // čekáme jestli dostaneme odezvu (čekáme na echo)

        if (TIM1\_GetFlagStatus(TIM1\_FLAG\_CC2) != RESET) {

                   // hlídáme zda timer hlásí změření pulzu

            capture = TIM1\_GetCapture2();

                   // uložíme výsledek měření

            capture\_flag = 1;   // dáme vědět zbytku programu že máme nový platný výsledek

            stage = 0;          // a začneme znovu od začátku

        } else if (milis() - time > MEASURMENT\_PERIOD \* 4) {

      // pokud timer nezachytil pulz po dlouhou dobu, tak echo nepřijde

            stage = 0;          // a začneme znovu od začátku

        }

        break;

    default:                   // pokud se cokoli pokazí

        stage = 0;              // začneme znovu od začátku

    }

}

void main(){

    init\_milis();

    setup();

    GPIO\_setup();

    TIM1\_setup();

    lcd\_gotoxy(0,1);

    lcd\_puts("vz=");

    char text[32];

    while (1){

        //if(GPIO\_ReadInputPin(GPIOE,GPIO\_PIN\_4)==RESET){

        process\_measurment(); //zajištuje měření

        if(capture\_flag == 1){ //jakmile je nová hodnota tak se vypíše nový výsledek a flag se nastaví na 0

            vzdalenost = capture;

            //lcd\_clear();

            lcd\_gotoxy(0,1);

            sprintf(text,"vzd=%3u",vzdalenost/58);

            lcd\_puts(text);

            capture\_flag = 0;

            }

        //}

    }

}

#include "\_\_assert\_\_.h"