SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

SEMESTRÁLNY PROJEKT - RC VYSIELAČ

(Programátorský manuál)

Študijný program: Robotika a kybernetika

Bratislava 2016

Bc. Jakub Vydra

Bc. Tomáš Kolek

Bc. Anna Pružinská

Bc. Branislav Dlugoš

Obsah

1 HW a SW prostriedky

Hardvér

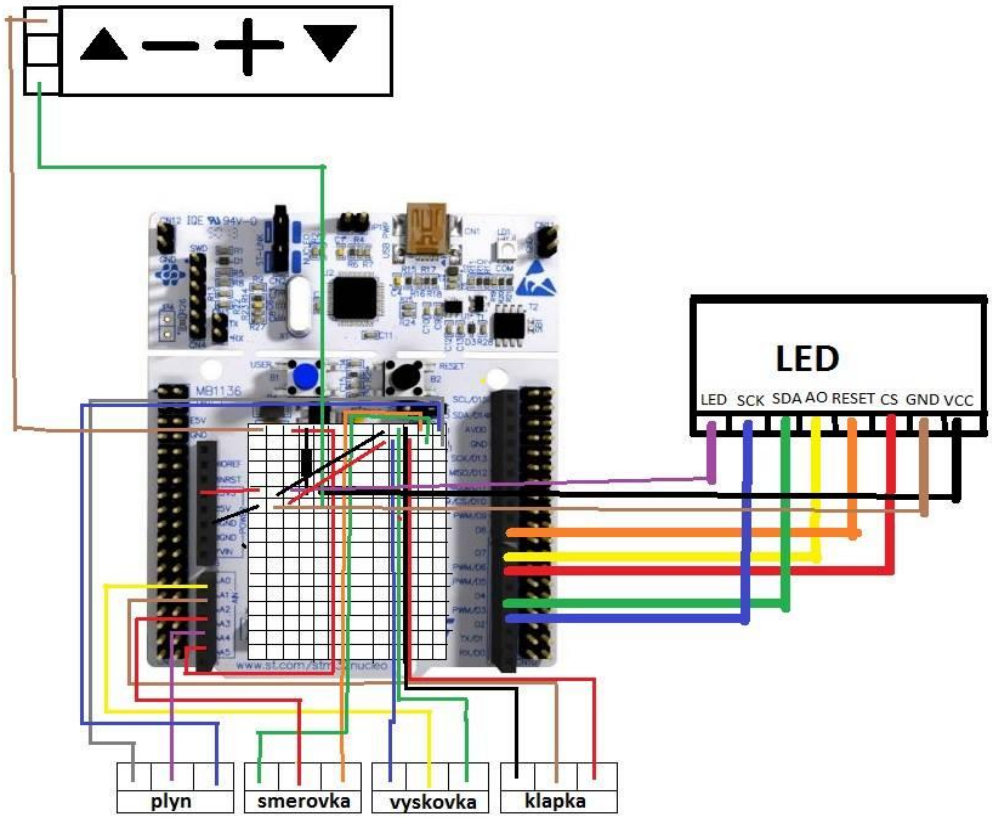
1. STM32L152RE
2. 4x potenciometer na ovládanie
3. 2x 3-polohové prepínače
4. 8k2 rezistor
5. 4 tlačidlá, display
6. Vodiče

Softvér

1. Atollic Studio
2. GitHub

1.1 Schéma zariadenia

Na obrázku číslo 1 môžeme vidieť schematický znázornený RC vysielač, ktorý obsahuje všetky časti potrebné na realizáciu nášho projektu.



Obrázok 1 Schéma

2 Popis funkčných častí

Základ bude tvoriť doska STM32L152RE. K doske budú pripojené potenciometre, ktorými sa bude simulovať ovládanie jednotlivých kanálov RC vysielačky. Takisto pomocou kontaktného poľa budú pripojené na dosku aj 4 tlačidlá, 2x 3-polohové prepínače a display.

2.1 Páky na ovládanie kanálov

Štyri kanály zabezpečujú ovládanie motora, výškovky, smerovky a klapiek. Každý kanál okrem ovládania motora sa jeho „knypel“ vráti do počiatočnej polohy (0, 0).

2.2 Troj-polohový prepínač

Ďalší kanál slúži na ovládanie podvozku, kde pre jednotlivé pozície prepínača sú zadefinované stavy podvozku ako vysunutý, medzi poloha a zasunutý.

2.3 Štyri tlačidlá

Tieto tlačidlá slúžia na pohyb v menu ovládača. Tlačidlá reprezentujú pohyb šípky v menu hore, dole, „do“ menu (enter) a „z“ menu (back).

2.4 Display

Display slúži na zobrazovanie informácií a menu, ktoré je popísané vyššie.

2.5 SPI

Komunikácia je zabezpečená pomocou SPI rozhrania. SPI je štvorvodičová synchrónna sériová zbernica pracujúca v móde full duplex slúžiaca na prepojenie periférií s mikropočítačmi. Zariadenia komunikujú spôsobom master/slave.

2.6 Výstupný signál

Výstupom aplikácie sú hodnoty posielané na jednotlivé kanály v ich normalizovaných rozsahoch.

3 Funkcie

Náš semestrálny projekt sa skladá z viacerých funkcií, ktoré budú popísané v jednotlivých častiach.

3.1 MAIN

V "maine" sa nachádza samostatná iniciálizácia častí projektu(obr.2), bez ktorých by nebolo možné vykonať všetky funkcie, ktoré sú potrebné. Najprv sme inicializovali zbernicu SPI, DMA,GPIO, úvodné menu, ADC prevodník a nakoniec NVIC - radič prerušení. V "maine" sa nachádza aj funkcia, ktorá slúži na kontrolu zvolenej položky menu, ktorú si zvolí používateľ . Podľa tejto kontroly sa následne spúšťajú jednotlivé submenu.

```
int main(void)
{
        initSPI2();  //inicializujem zbernicu SPI
        dma_init();      //inicializujem DMA
        initGPIO();  //Inicializujem GPIO
        initCD_Pin();  //
        initCS_Pin();
        initRES_Pin();
        initMenu();  //inicializujem uvodne menu
        adc_init(); //inicializujem ADC
        nvic_init();  //inicializujem NVIC
        while (1)
        {
                pohybMenu(klavesnica);  //funkcia
                //Delay(50);
```

Obrázok 2 Kód main-u

3.2 TEST 1.

Je to modul, ktorý slúži na prácu s displejom. Obsahuje viacero funkcií, ktoré budú rozpísané v jednotlivých bodoch.

3.2.1 Inicializácia

Táto časť je veľmi dôležitá, pretože tu sme inicializovali všetky súčasti projektu, bez ktorých by nebola možná realizácia.

Inicializácia menu

Slúži na inicializáciu menu, kde sa zobrazí hlavné menu a vypíšu sa jednotlivé položky.

Inicializácia GPIO

V tejto časti sme inicializovali periférie GPIO. V našom projekte sme používali tri porty GPIOA,GPIOB,GPIOC. Pre všetky porty sme zapli hodiny. Nainicializovali sme pin 1 na port C, kde sme jeho režim zvolili ako analógový a nepotrebovali sme ani žiaden pull up alebo pull down. (obr.č.3)

```
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB, ENABLE);
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1 ;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL ;
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

Obrázok 3 Inicializácia GPIOC

Na port A sme inicializovali piny 0, 1,4. Taktiež sme zvolili také nastavenia ako pri prvom príklade. Na port B sme nainicializovali pin 0 s rovnakými nastaveniami.

Inicializácia Direct Memory Access (DMA)

V tejto časti sme inicializovali DMA s potrebnými nastaveniami. Nastavovali sme to na kanál 1, ktorý môžeme vidieť na obrázku č. 4.

```
dma_init(){  //inicializujem DMA1
   DMA_InitTypeDef       DMA_InitStruct;
   RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
   /* DMA1 Stream0 channel0 configuration **********************************
   DMA_InitStruct.DMA_PeripheralBaseAddr = (uint32_t)&ADC1->DR;//ADC1's dat
   DMA_InitStruct.DMA_MemoryBaseAddr = (uint32_t)&ADC1ConvertedValue[0];
   DMA_InitStruct.DMA_DIR = DMA_DIR_PeripheralSRC;
   DMA_InitStruct.DMA_BufferSize = 5;
   DMA_InitStruct.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
   DMA_InitStruct.DMA_MemoryInc = DMA_MemoryInc_Enable;
   DMA_InitStruct.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
   DMA_InitStruct.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;//Stores
   DMA_InitStruct.DMA_Mode = DMA_Mode_Circular;
   DMA_InitStruct.DMA_Priority = DMA_Priority_High;
   DMA_Init(DMA1_Channel1, &DMA_InitStruct);
   DMA_Cmd(DMA1_Channel1, ENABLE); //Enable the DMA1 - Channel1
```

Obrázok 4 DMA

Inicializácia ADC prevodníka

Inicializácia ADC prevodníka je zobrazená na obrázku 5. Môžeme vidieť ako sme nastavili jeho jednotlivé súčasti. Rozlíšenie sme zvolili 12 b, konverzný mód sme zvolili continuous. Výsledok z ADC prevodníka sme zvolili tak, aby sa zobrazilo napravo. Taktiež sme tu nastavovali kanály, kde sme čas vzorkovania zvolili 9 cyklov.

```
ADC_InitTypeDef ADC_InitStructure;
//while(RCC_GetFlagStatus(RCC_FLAG_HSIRDY) == RESET);
RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
ADC_StructInit(&ADC_InitStructure);
ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
ADC_InitStructure.ADC_ScanConvMode = ENABLE;//The scan is configured in multipl
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
ADC_InitStructure.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_None;
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
ADC_InitStructure.ADC_NbrOfConversion = 5;
ADC_InitStructure.ADC_ExternalTrigConv = 0;
ADC_Init(ADC1, &ADC_InitStructure);
ADC_RegularChannelConfig(ADC1, ADC_Channel_0, 1, ADC_SampleTime_9Cycles);//PC0
ADC_RegularChannelConfig(ADC1, ADC_Channel_1, 2, ADC_SampleTime_9Cycles);
ADC_RegularChannelConfig(ADC1, ADC_Channel_4, 3, ADC_SampleTime_9Cycles);
ADC_RegularChannelConfig(ADC1, ADC_Channel_8, 4, ADC_SampleTime_9Cycles);
ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 5, ADC_SampleTime_9Cycles);
```

Obrázok 5 ADC

Inicializácia NVIC

Ako posledný krok je inicializácia radiča prerušení pre ADC prevodník. Nastavenie môžeme vidieť na obrázku 6.

```
void nvic_init(){                          // inicializacia preruseni
        //NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
        NVIC_InitTypeDef NVIC_InitStructure;
        //NVIC_InitStructure.NVIC_IRQChannel = ADC1_IRQn;
        NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel1_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_Init(&NVIC_InitStructure);
    DMA_ITConfig(DMA1_Channel1, DMA_IT_TC, ENABLE);
    DMA_Cmd(DMA1_Channel1, ENABLE);
    ADC_DMACmd(ADC1, ENABLE);
```

Obrázok 6 NVIC

3.2.2 Pohyb Menu

Táto funkcia slúži na pohyb v menu a na výber operácie, ktorá sa má vykonať na danom zariadení. Na základe nasledúcich podmienok (obr. č. 7) sa zavolá prislušná funkcia implementujúca danú operáciu.

```
else if ((klavesnica >= 2800) && (klavesnica <= 2940)){  //vstup do submenu
        if ((aktualneA == 0 && defaultA == 12) || aktualneA == 12){
                otvorInfo();
        }
        else if (aktualneA == 32){
                otvorRevers();
        }
        else if (aktualneA == 52){
                otvorExpo();
        }
        else if (aktualneA == 72){
                otvorMix();
        }
        else if (aktualneA == 92){
                otvorEPA();
        }
}
```

Obrázok 7 Operácie

Pohyb menu je implementovaný prostredníctvom funkcií PosunSipkyDole a PosunSipkyHore.

3.2.2 Zhrnutie

Na začiatku je potrebná inicializácia všetkých pinov, ADC prevodníka, NVIC - radiča prerušení a DMA. Po inicializácii nasleduje funkcia, ktorá vytvára menu. Defaultné nastavenie je že šípka sa nachádza na prvej položke v menu. Podľa stlačenej klávesy sa šípka prekreslí na novú pozíciu. Pri stlačení plusu sa nám na klávesnici otvorí submenu. Opustenie submenu sa vykona stlačením mínusu. Na to aby bol premazaný displej slúži funkcia lcdClearDisplay. Na vykreslenie šípky je funkcia lcdPinyTrojuholník. Funkcia pohyb menu sleduje aké tlačidlo bolo stlačené. Je veľmi dôležité poznamenať, že pracujeme s hodnotami z ADC prevodníka, ktorú získame prostredníctvom DMA a je postupne ukladaná do poľa. Pole sa naplní vždy pri prerušení.

3.3 SUBMENU

V tejto časti sa budeme venovať jednotlivým funkciám, ktoré má podľa zadania náš RC vysielač vykonávať.

3. 3.1 Normalizuj

Táto funkcia slúži na normalizáciu hodnôt, ktoré dostaneme z ADC prevodníka. V praxi to znamená, že dané hodnoty interpolujeme do intervalu <-1,1>

```
float normalizuj(float hodnota,float hodnotaMIN,float hodnotaMAX){ //normalizovanie hodnot od -1 do 1
        NORM =(((hodnota-hodnotaMIN)/(hodnotaMAX-hodnotaMIN))*(1-(-1)))+(-1);
        return NORM;
}
```

Obrázok 8 Normalizácia

3.3.2 FloatToString

Táto funkcia slúži na vypísanie formátu desatinného čísla v správnom tvare. Vstupom je float číslo a výstypom je string. Funkcia rozdelí desatinné číslo na dve čísla typu int, jedno pred desatinnou čiarkou a jedno za. Následne ich formátovaním uloží do stringu. Je prítomné aj ošetrenie ak je číslo za desatinnou čiarkou menšie ako 10 resp. začína 0.

```
char *FloatToString(float NORM){
        NORMint=(int)NORM;                              // zaokruhli vstupne cislo
        NORMdiff=NORM-(float)NORMint;   // cislo za desatinnou ciarkou
        NORMint2=(int)(NORMdiff*100);   // cislo za desatiinnou ciarkou vynasobene 100 (posunute o 2 miesta vlavo)
        if (NORM < 0){  // ak je cislo zaporne prida minusy
                static char str[6];
                if(NORMint2 < 10) sprintf(str,"-%d.0%d", NORMint, abs(NORMint2));       // ak je cislo za desatinnou ciarkou mensie ako .10
                else sprintf(str,"-%d.%d", NORMint, abs(NORMint2));
                        return str;
                }
        else{
                static char str[5];
                if(NORMint2 < 10) sprintf(str,"%d.0%d", NORMint, abs(NORMint2));
                else sprintf(str,"%d.%d", NORMint, NORMint2);
                        return str;
                }
```

Obrázok 9 FloatToString

3.3.3 FloatToStringReverz

Táto funkcia je v podstate rovnaká ako funkcia FloaToString, ale rozdiel spočíva v reverze výsledných hodnôt. V praxi to znamená, že napr. pri výslednej hodnote 1, táto funkcia vypíše -1.

```
char *FloatToStringReverz(float NORM){  //to iste ako floattostring, opacne znamienka
        NORMint=(int)NORM;
        NORMdiff=NORM-(float)NORMint;
        NORMint2=(int)(NORMdiff*100);
        if (NORM < 0){
                static char str[6];
                if(NORMint2 < 10) sprintf(str,"%d.%d", NORMint, abs(NORMint2));
                else sprintf(str,"%d.%d", NORMint, abs(NORMint2));
                        return str;
                }
        else{
                static char str[5];
                if(NORMint2 < 10) sprintf(str,"-%d.0%d", NORMint, abs(NORMint2));
                else sprintf(str,"-%d.%d", NORMint, NORMint2);
                        return str;
                }

}
```

Obrázok 10 Reverzné hodnoty

3.3.4 Otvor_info

Táto funkcia slúži na spustenie informácii v menu. Keď užívateľ stlačí tlačidlo Info zobrazia sa hodnoty z ovládačov. Tieto jednotlivé hodnoty reprezentujú klapku, výškovku, plyn a smerovku. Nemôžeme zabudnúť , že jednotlivé hodnoty sú normalizované.

```
void otvorInfo(){       //funkcia, ktora po vyvolani zobrazi jednotlive kanaly, hodnoty su normalizovane, rozsah -1 az 1
        subMenu = 1;
        char str[5];
        lcdClearDisplay(decodeRgbValue(255, 255, 255));                 //"vycisti" display
        while(subMenu==1){
                if((klavesnica >= 3200) && (klavesnica <= 3440)){       //-
                                break;
                        }
                else{
                lcdPutS("HODNOTY KANALOV",23, 17, 0x0000, 0xFFFF);      //zobrazi na LCD

                kridielkoNORM = normalizuj((float)kridielko,kridielkoMIN,kridielkoMAX); //normalizovanie hodnoty z analogu
                sprintf(str,"kridielko: %s", FloatToString(kridielkoNORM));     //prevod cisla na char
                lcdPutS(str, 23, 37, 0x0000, 0xFFFF);   //vypis na displej

                vyskovkaNORM = normalizuj((float)vyskovka,vyskovkaMIN,vyskovkaMAX);
                sprintf(str,"Vyskovka: %s", FloatToString(vyskovkaNORM));
                lcdPutS(str, 23, 47, 0x0000, 0xFFFF);

                plynNORM = normalizuj((float)plyn,plynMIN,plynMAX);
                sprintf(str,"Plyn: %s", FloatToString(plynNORM));
                lcdPutS(str, 23, 57, 0x0000, 0xFFFF);

                smerovkaNORM = normalizuj((float)smerovka,smerovkaMIN,smerovkaMAX);
                sprintf(str,"Smerovka: %s", FloatToString(smerovkaNORM));
                lcdPutS(str, 23, 67, 0x0000, 0xFFFF);
                }
        }
        return;
}
```

Obrázok 11 Informácie

3.3.5 Otvor Revers

Hlavná časť tejto funkcie je nastavenie reverzného ovládania výškoviek, smerovky,plynu alebo klapiek. Pomocou funkcie FloatToStringReverz pre každú súčasť nášho zariadenia uskutočnujeme reverz analógových hodnôt.

```
void otvorRevers(){     //funkcia na revers hodnot analogov
        subMenu = 1;
        char str[5];
        lcdClearDisplay(decodeRgbValue(255, 255, 255));
        while(subMenu==1){
                if((klavesnica >= 3200) && (klavesnica <= 3440)){       //-
                                break;
                        }
                else{
        lcdPutS("REVERS",23, 17, 0x0000, 0xFFFF);

        kridielkoNORM = normalizuj((float)kridielko,kridielkoMIN,kridielkoMAX);
        sprintf(str,"kridielko: %s", FloatToStringReverz(kridielkoNORM));
        lcdPutS(str, 23, 37, 0x0000, 0xFFFF);

        vyskovkaNORM = normalizuj((float)vyskovka,vyskovkaMIN,vyskovkaMAX);
        sprintf(str,"Vyskovka: %s", FloatToStringReverz(vyskovkaNORM));
        lcdPutS(str, 23, 47, 0x0000, 0xFFFF);

        plynNORM = normalizuj((float)plyn,plynMIN,plynMAX);
        sprintf(str,"Plyn: %s", FloatToStringReverz(plynNORM));
        lcdPutS(str, 23, 57, 0x0000, 0xFFFF);

        smerovkaNORM = normalizuj((float)smerovka,smerovkaMIN,smerovkaMAX);
        sprintf(str,"Smerovka: %s", FloatToStringReverz(smerovkaNORM));
        lcdPutS(str, 23, 67, 0x0000, 0xFFFF);
                }
        }
}
```

Obrázok 12 Revers

3.3.6 Otvor Expo

Funkcia expo slúži na exponenciálne ovládanie kanálov.

```
void otvorExpo(){
        subMenu = 1;
        char str[5];
        lcdClearDisplay(decodeRgbValue(255, 255, 255));
        while(subMenu==1){
                if((klavesnica >= 3200) && (klavesnica <= 3440)){       //-
                                break;
                        }
                else{
        lcdPutS("Aktivne EXPO",23, 17, 0x0000, 0xFFFF);

        kridielkoNORM = normalizuj((float)kridielko,kridielkoMIN,kridielkoMAX);
        if(kridielkoNORM<0){
                kridielkoNORM = exp2(kridielkoNORM*(-1))-1;             //vypocet exponencialnej funkcii 2^x; x je ciselny vstup do funkcie
                kridielkoNORM = kridielkoNORM*(-1);
        }
        else{kridielkoNORM = exp2(kridielkoNORM)-1;}
        sprintf(str,"kridielko: %s", FloatToString(kridielkoNORM));
        lcdPutS(str, 23, 37, 0x0000, 0xFFFF);

        vyskovkaNORM = normalizuj((float)vyskovka,vyskovkaMIN,vyskovkaMAX);
        if(vyskovkaNORM<0){
                vyskovkaNORM = exp2(vyskovkaNORM*(-1))-1;
                vyskovkaNORM = vyskovkaNORM*(-1);
        }
        else{vyskovkaNORM = exp2(vyskovkaNORM)-1;
        kridielkoNORM = kridielkoNORM*(-1);}
        sprintf(str,"Vyskovka: %s", FloatToString(vyskovkaNORM));
        lcdPutS(str, 23, 47, 0x0000, 0xFFFF);

        plynNORM = normalizuj((float)plyn,plynMIN,plynMAX);
        if(plynNORM<0){
                plynNORM = exp2(plynNORM*(-1))-1;
                plynNORM = plynNORM*(-1);
        }
        else{plynNORM = exp2(plynNORM)-1;}
        sprintf(str,"Plyn: %s", FloatToString(plynNORM));
        lcdPutS(str, 23, 57, 0x0000, 0xFFFF);

        smerovkaNORM = normalizuj((float)smerovka,smerovkaMIN,smerovkaMAX);
        if(smerovkaNORM<0){
                smerovkaNORM = exp2(smerovkaNORM*(-1))-1;
                smerovkaNORM = smerovkaNORM*(-1);
        }
        else{smerovkaNORM = exp2(smerovkaNORM)-1;}
        sprintf(str,"Smerovka: %s", FloatToString(smerovkaNORM));
        lcdPutS(str, 23, 67, 0x0000, 0xFFFF);
                        }
        }
}
```

Obrázok 13 Exponenciálne ovládanie

3.3.7 Otvor mix

Táto funkcia primárne slúži na prepínanie medzi klasickým ovládaním a mix ovládaním. Rozdiel spočíva v tom, že mix ponúka ovládať samokrídlo s hodnotami, ktoré sú už nastavené pre tento účel.

```
void otvorMix(){        //funkcia na mixovanie kanalov, vahy: vyskovka 50%, klapky 50%
        subMenu = 1;
        float vahaVyskovky = 0.5;
        float vahaKlapky = 0.5;
        char str[5];
        lcdClearDisplay(decodeRgbValue(255, 255, 255));
        while(subMenu==1){
                if((klavesnica >= 3200) && (klavesnica <= 3440)){       //-
                                break;
                        }
                else{
        lcdPutS("MIX 50%",23, 17, 0x0000, 0xFFFF);

        kridielkoNORM = normalizuj((float)kridielko,kridielkoMIN,kridielkoMAX);
        vyskovkaNORM = normalizuj((float)vyskovka,vyskovkaMIN,vyskovkaMAX);

        if(vyskovkaNORM>0){
                if(kridielkoNORM>0){
                        MIX = kridielkoNORM*vahaKlapky + vyskovkaNORM*vahaVyskovky;
                        sprintf(str,"Servo: %s", FloatToString(MIX));
                        lcdPutS(str, 23, 37, 0x0000, 0xFFFF);
                }
                else{
                MIX = vyskovkaNORM*vahaVyskovky + kridielkoNORM*vahaKlapky;
                sprintf(str,"Servo: %s", FloatToString(MIX));
                lcdPutS(str, 23, 37, 0x0000, 0xFFFF);
                }
        else{
                if(kridielkoNORM>0){
                        MIX = kridielkoNORM*vahaKlapky + vyskovkaNORM*vahaVyskovky;
                        sprintf(str,"Servo: %s", FloatToString(MIX));
                        lcdPutS(str, 23, 37, 0x0000, 0xFFFF);
                }
                else{
                MIX = kridielkoNORM*vahaKlapky + vyskovkaNORM*vahaVyskovky;
                sprintf(str,"Servo: %s", FloatToString(MIX));
                lcdPutS(str, 23, 37, 0x0000, 0xFFFF);
                }
        }
        plynNORM = normalizuj((float)plyn,plynMIN,plynMAX);
        sprintf(str,"kridielko: %s", FloatToString(kridielkoNORM));
        lcdPutS(str, 23, 47, 0x0000, 0xFFFF);
        sprintf(str,"Vyskovka: %s", FloatToString(vyskovkaNORM));
        lcdPutS(str, 23, 57, 0x0000, 0xFFFF);
        sprintf(str,"Plyn: %s", FloatToString(plynNORM));
        lcdPutS(str, 23, 67, 0x0000, 0xFFFF);
                        }
        }
```

Obrázok 14 Mix

3.3.8 EPA

Trimovanie je zabezpečené pomocou funkcie otvorEPA. Pre názornosť a nedostatok ovládacích prvkov je trimovanie pripravené na plyn. Samotné trimovanie pripočítava „offset“, čím zmení stred intervalu a tým aj aktuálnu výchylku.

```
void otvorEPA(){
        subMenu = 1;
        char str[5];
        lcdClearDisplay(decodeRgbValue(255, 255, 255));
        while(subMenu==1){
                if((klavesnica >= 3200) && (klavesnica <= 3440)){       //-
                                break;
                        }
                else{
        lcdPutS("TRIMOVANIE",23, 17, 0x0000, 0xFFFF);   //zobrazi na LCD

        if((klavesnica >= 1800) && (klavesnica <= 2100)){       //hore
                trim += 0.01;
        }
        if((klavesnica >= 3450) && (klavesnica <= 3600)){       //dole
                trim -= 0.01;
        }

        plynNORM = normalizuj((float)plyn,plynMIN,plynMAX) + trim;
        sprintf(str,"Plyn: %s", FloatToString(plynNORM));
        lcdPutS(str, 23, 57, 0x0000, 0xFFFF);
        sprintf(str,"Trim: %s", FloatToString(trim));
        lcdPutS(str, 23, 67, 0x0000, 0xFFFF);
                        }
        }
}
```

Obrázok 15 Epa

4 Link na GitHub

https://github.com/tomaskolek/test11/tree/master/src