SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Vnorené riadiace systémy

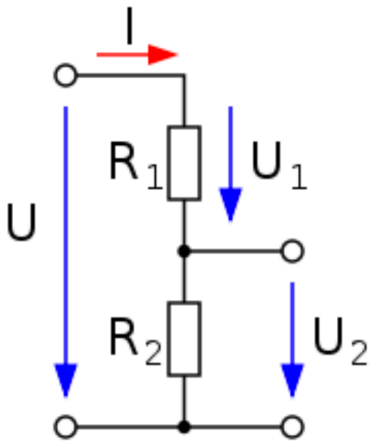
Cvičenie 4

Andrej Halán
Patrik Buranský

Link na repozitár : https://github.com/Patndrikej/vrs_cv4_final

Cieľom cvičenia bolo vytvoriť program, ktorý reagoval na stlačenie rôznych tlačidiel blikaním LED diódy v rôznych frekvenciách. Frekvencia blikania sa mení podľa zmeny stlačenia tlačidiel.

Prvou úlohou bolo zapojiť klávesnicu a rezistor podľa schémy :



Ako analógový vstup sme použili vstup na Arduino shielde A0.

Ako vstup na cvičenie sme použili vstupnú funkciu adc_init(void), kde sme upravili nasledovné riadky kde sme zmenili nastavenia na vstup 0, podľa zapojenia našej schémy :

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 ;

ADC_RegularChannelConfig(ADC1, ADC_Channe1_0, 1,
ADC_SampleTime_16Cycles);
```

Na inicializáciu periférie pre LED sme vytvorili funkciu init_LED(), ktorá nastavila periférium na GPIO_PIN_5

V druhej úlohe sme realizovali samotný program.

Frekvencia LED diódy sa mení v závislosti od stlčeného tlačidla, pričom boli použité 4 tlačidlá + 2 krát kombinácia dvoch tlačidiel. Na začiatku sme si merali

hodnotu po stlačení jednotlivého tlačidla a kombináciu dvoch tlačidiel. Vo while cykle kontrolujeme 6 hodnôt, ktorá sa mení frekvencia blikania LED diódy. Ďalej používame pomocnú premennú ktorú využívame na dĺžku pauzy medzi zapnutím a vypnutím LED diódy.

Zdrojový kód programu:

Hlavná funkcionalita

```
void adc_init(void);
void init_LED();
void delay_for_led(int);

int main(void){

  int i = 0;
  int value = 20; // rozsah odchýlky pre stlačenie tlačidla
  int v = 50000;  // pomocná premenná pre nastavovanie Delay
  int AD_value; // pomocná premenná na uchovanie stavu o stlačení tlačidla

  adc_init();  // nastavenie periférie
  init_led();  // nastavenie periférie LED

  while (1){

  ADC_SoftwareStartConv(ADC1);
          while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC)){}
          AD_value = ADC_GetConversionValue(ADC1);

        /* Nastavenie dĺžky frekvencie LED diódy podľa stlačeného tlačidla */

          if(( AD_value < 3655 + value) && ( AD_value > 3655 - value )) {
                v = 50000;

          }else if(( AD_value < 3459 + value ) && ( AD_value > 3459 - value )) {
                v = 125000;

          }else if(( AD_value < 2914 + value  ) && ( AD_value > 2914 - value )) {
                v = 200000;
```

```
        }else if(( AD_value < 2007 + value  ) && ( AD_value > 2007 - value )) {
                v = 275000;

        }else if(( AD_value < 3240 + value ) && ( AD_value > 3240 - value )) {
                v = 350000;

        }else if(( AD_value < 1700 + value  ) && ( AD_value > 1700 - value )) {
                v = 425000;
        }

        GPIO_SetBits(GPIOA, GPIO_Pin_5); // zapnutie LED diódy
        delay_for_led(v); // Nastavenie dĺžky pauzy medzi zapnutím a vypnutím

        GPIO_ResetBits(GPIOA, GPIO_Pin_5); // vypnutie LED diódy
        delay_for_led(v); // Nastavenie dĺžky pauzy medzi vypnutím a zapnutím

      i++;
  }

  return 0;
}

void delay_for_led(int value){
        for(int j=0;j<=value;j++){}
}

void init_led(){
        RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);

        GPIO_InitTypeDef gpioInitStruc;

        gpioInitStruc.GPIO_Mode = GPIO_Mode_OUT;
        gpioInitStruc.GPIO_OType = GPIO_OType_PP;
        gpioInitStruc.GPIO_Pin = GPIO_Pin_5;
        gpioInitStruc.GPIO_Speed = GPIO_Speed_400KHz;

        GPIO_Init(GPIOA, &gpioInitStruc);
}

void adc_init(void) {
```

```
        GPIO_InitTypeDef GPIO_InitStructure;
        ADC_InitTypeDef  ADC_InitStructure;
        int ADC_Channe1_0;

        /* Enable GPIO clock */
        RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);

  /* Configure ADCx Channel 2 as analog input */
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 ;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL ;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        /* Enable the HSI oscillator */
        RCC_HSICmd(ENABLE);

        /* Check that HSI oscillator is ready */
        while(RCC_GetFlagStatus(RCC_FLAG_HSIRDY) == RESET);

  /* Enable ADC clock */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

  /* Initialize ADC structure */
        ADC_StructInit(&ADC_InitStructure);

  /* ADC1 configuration */
        ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
        ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
        ADC_InitStructure.ADC_ExternalTrigConvEdge =
ADC_ExternalTrigConvEdge_None;
        ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
        ADC_InitStructure.ADC_NbrOfConversion = 1;

        ADC_Init(ADC1, &ADC_InitStructure);

        /* ADCx regular channel8 configuration */
        ADC_RegularChannelConfig(ADC1, ADC_Channe1_0, 1,
ADC_SampleTime_16Cycles);

  /* Enable the ADC */
```

```
        ADC_Cmd(ADC1, ENABLE);

    /* Wait until the ADC1 is ready */
        while(ADC_GetFlagStatus(ADC1, ADC_FLAG_ADONS) == RESET){}

        /* Start ADC Software Conversion */
        ADC_SoftwareStartConv(ADC1);
}
```