Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Программное обеспечение встроенных систем (ПОВС)

ОТЧЕТ

по лабораторной работе № 3

Тема работы: Построение графиков сигналов

Выполнили

студенты: гр. 151003 Барановский Р.А.,

Матошко И.В.

Проверил: Деменковец Д.В.

Минск 2024

**Вариант 3**

**lib.h**

#ifndef LIB\_DECLARATION

#define LIB\_DECLARATION

#include "stm32f1xx\_hal.h"

#include "stm32f1xx\_hal\_adc.h"

#include "math.h"

#include "stdio.h"

#include "string.h"

#define CORRECTION\_TIME 500

#define PARAMETERS\_COUNT 4

#define SAMPLING\_RATE 100

#define PI acos(-1)

#define LOW\_BORDER 1

#define HIGH\_BORDER 9

typedef struct Pin

{

GPIO\_TypeDef\* port;

uint16\_t line;

} Pin;

extern volatile int previousUnPressedTime;

extern volatile uint16\_t adc;

extern volatile int activeParameter;

extern volatile int8\_t parameters[4];

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin);

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim);

#endif // LIB\_DECLARATION

**lib.c**

#include "lib.h"

#include "main.h"

#include "display.h"

volatile int previousUnPressedTime = 0;

volatile uint16\_t adc = 0;

volatile int activeParameter = 0;

volatile int8\_t parameters[4] = {1,1,1,1};

volatile int8\_t\* A = &parameters[0];

volatile int8\_t\* T = &parameters[1];

volatile int8\_t\* F = &parameters[2];

volatile int8\_t\* FI0 = &parameters[3];

uint8\_t data = 0;

Buffer displayBuffer = 0;

uint8\_t n = 0;

Pin selectPinByActiveParameter(int activeParameter);

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)

{

if(GPIO\_Pin == GPIO\_PIN\_1 && (HAL\_GetTick() - previousUnPressedTime) > CORRECTION\_TIME)

{

Pin previousLed = selectPinByActiveParameter(activeParameter);

HAL\_GPIO\_WritePin(previousLed.port, previousLed.line, GPIO\_PIN\_SET);

activeParameter = (activeParameter + 1) % PARAMETERS\_COUNT;

Pin newLed = selectPinByActiveParameter(activeParameter);

HAL\_GPIO\_WritePin(newLed.port, newLed.line, GPIO\_PIN\_RESET);

pushNumberIntoBuffer(&displayBuffer, parameters[activeParameter]);

printNumberOnDisplay(&displayBuffer);

previousUnPressedTime = HAL\_GetTick();

}

else if (GPIO\_Pin == GPIO\_PIN\_4 && (HAL\_GetTick() - previousUnPressedTime) > CORRECTION\_TIME)

{

if (parameters[activeParameter] > LOW\_BORDER)

{

parameters[activeParameter]--;

}

pushNumberIntoBuffer(&displayBuffer, parameters[activeParameter]);

printNumberOnDisplay(&displayBuffer);

previousUnPressedTime = HAL\_GetTick();

}

else if (GPIO\_Pin == GPIO\_PIN\_0 && (HAL\_GetTick() - previousUnPressedTime) > CORRECTION\_TIME)

{

if (parameters[activeParameter] < HIGH\_BORDER)

{

parameters[activeParameter]++;

}

pushNumberIntoBuffer(&displayBuffer, parameters[activeParameter]);

printNumberOnDisplay(&displayBuffer);

previousUnPressedTime = HAL\_GetTick();

}

}

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)

{

HAL\_ADC\_Start(&hadc1);

HAL\_ADC\_PollForConversion(&hadc1, 100);

adc = HAL\_ADC\_GetValue(&hadc1);

HAL\_ADC\_Stop(&hadc1);

n = (n + 1) % (SAMPLING\_RATE + 1);

double signal1 = (4 \* (double)\*A / \*T) \* fabs(fmod((double)n / SAMPLING\_RATE, (double)\*T) - (double)\*T / 2) - \*A;

double signal2 = (2 \* \*A / PI) \* asin(sin(2 \* PI \* \*F \* ((double)n / SAMPLING\_RATE) + \*FI0));

char firstSignalStr[50];

char secondSignalStr[15];

char adcStr[15];

snprintf((char\*)firstSignalStr, sizeof(firstSignalStr), "%f,", signal1);

snprintf((char\*)secondSignalStr, sizeof(secondSignalStr), "%f,",signal2);

snprintf(adcStr, sizeof(adcStr), "%d\n", (uint16\_t)adc);

strcat(firstSignalStr, secondSignalStr);

strcat(firstSignalStr, adcStr);

HAL\_UART\_Transmit\_DMA(&huart2, (uint8\_t\*)firstSignalStr, strlen(firstSignalStr));

}

Pin createPin(GPIO\_TypeDef\* port, uint16\_t line)

{

return (Pin){ port, line };

}

Pin selectPinByActiveParameter(int activeParameter)

{

switch (activeParameter)

{

case 0:

return createPin(GPIOA, GPIO\_PIN\_5);

case 1:

return createPin(GPIOA, GPIO\_PIN\_6);

case 2:

return createPin(GPIOA, GPIO\_PIN\_7);

case 3:

return createPin(GPIOB, GPIO\_PIN\_6);

default:

return createPin(GPIOA, GPIO\_PIN\_5);

}

}