

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №4
по дисциплине
“Проектирование вычислительных систем”

Вариант №5

Студент:

Чернова Анна Ивановна

Миху Вадим Дмитриевич

Группа Р34301

Преподаватель:

Пинкевич Василий Юрьевич

г. Санкт-Петербург

2024

Цель работы

1. Получить базовые знания об интерфейсе I2C и особенностях передачи данных по данному интерфейсу.
2. Получить базовые знания об устройстве и принципах работы контроллера интерфейса I2C в микроконтроллерах и получить навыки его программирования.

Задание

Разработать программу, которая использует интерфейс I2C для считывания нажатий кнопок клавиатуры стенда SDK-1.1. Подсистема опроса клавиатуры должна удовлетворять следующим требованиям:

- реализуется защита отдребезга;
- нажатие кнопки фиксируется сразу после того, как было обнаружено, что кнопка нажата (с учетом защиты отдребезга), а не в момент отпускания кнопки; если необходимо, долгое нажатие может фиксироваться отдельно;
- кнопка, которая удерживается дольше, чем один цикл опроса, не считается повторно нажатой до тех пор, пока не будет отпущена (нет переповторов);
- распознается и корректно обрабатывается множественное нажатие (при нажатии более чем одной кнопки считается, что ни одна кнопка не нажата, если это не противоречит требованиям к программе);
- всем кнопкам назначаются коды от 1 до 12 (порядок на усмотрение исполнителей). Программа должна иметь два режима работы, переключение между которыми производится по нажатию кнопки на боковой панели стенда:
- режим тестирования клавиатуры;
- прикладной режим.

Уведомление о смене режима выводится в UART.

В режиме тестирования клавиатуры программа выводит в UART коды нажатых кнопок.

В прикладном режиме программа обрабатывает нажатия кнопок и выполняет действия в соответствии с вариантом задания.

Вариант 5

Реализовать музыкальную ритм-игру. С помощью звукоизлучателя воспроизводится последовательно, состоящая из звуков разной частоты («мелодия»). Каждый звук сопровождается зажиганием светодиода определенного цвета и с определенной яркостью (регулируется коэффициентом заполнения). Должно существовать взаимно однозначное соответствие между частотой звука и цветом/яркостью светодиода. Во время каждого

звука/импульса светодиода игрок должен ввести символ, соответствующий текущей частоте звука или цвету/яркости. Чем больше звуков будет «угадано» правильно и на большей скорости игры, тем больше очков заработает игрок (система начисления очков – на усмотрение исполнителей).

Всего необходимо предусмотреть девять видов импульсов: зеленый, желтый и красный на 20 %, 50 % и 100 % яркости. К ним следует подобрать звуки произвольных частот, легко отличимых одна от другой на слух. Предусмотреть одну стандартную последовательность импульсов длительностью не менее 20-ти элементов (простейший вариант – циклический перебор девяти импульсов). Когда последовательность заканчивается или досрочно останавливается игроком, в UART выводится количество набранных очков и «трассировка» нажатий, где отмечены правильные и неправильные нажатия. Отсутствие нажатия в течение импульса должно считаться неправильным нажатием.

Выполнение

Листинг разработанной программы с комментариями

```
#include "main.h"
#include "i2c.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"
#include "matrix_keyboard.h"

#define CODE_LENGTH 9
#define TIMEOUT 10000
#define MAX_ATTEMPTS 3
#define GREEN_LED_Pin GPIO_PIN_13
#define YELLOW_LED_Pin GPIO_PIN_14
#define RED_LED_Pin GPIO_PIN_15
#define BUTTON_Pin GPIO_PIN_15

char LOCK_CODE[CODE_LENGTH] = "123456789";
char input_buffer[CODE_LENGTH];
volatile uint8_t current_pos = 0;
volatile uint8_t attempt_count = 0;
volatile uint8_t mode = 0; // 0 = Unlock Mode, 1 = Change Code Mode, 2 =
Await Confirmation
volatile uint8_t uart_mode = 0; // 0 = Polling, 1 = Interrupt
volatile uint8_t show_led = 1;
volatile uint8_t show_sound = 1;
uint8_t received_data;
char buffer = '@';
char new_code[CODE_LENGTH];
volatile uint8_t new_code_pos = 0;

char uart_buffer[UART_BUFFER_SIZE];
volatile uint8_t write_index = 0;
volatile uint8_t read_index = 0;

void SystemClock_Config(void);
void UART_Send_Message(const char *msg);
void UART_Send_Char(char ch);
void Change_Code_Logic(char input);
void Confirm_New_Code(char input);
void Process_Input(char input);
void System_Reset(void);

char parse_matrix_button(uint16_t* keyboard);

int main(void)
{
```

```

HAL_Init();

SystemClock_Config();

MX_GPIO_Init();
MX_USART6_UART_Init();
MX_TIM6_Init();
MX_TIM4_Init();
MX_TIM1_Init();
MX_I2C1_Init();
keyboard_init();

HAL_TIM_Base_Start_IT(&htim6);
UART_Send_Message("System Ready\r\n");

HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4);

char input;
uint16_t keyboard_state;

while (1)
{
    keyboard_get_state(&keyboard_state);
    char c = parse_matrix_button(&keyboard_state);
    if (c != 0) {
        Process_Input(c);
    }

    if (HAL_UART_Receive(&huart6, (uint8_t *)&input, 1, 100) ==
HAL_OK) {
        Process_Input(input);
    }

}

}

char parse_matrix_button(uint16_t* keyboard) {
    if (keyboard_is_set(keyboard, BUTTON_1)) return '1';
    if (keyboard_is_set(keyboard, BUTTON_2)) return '4';
    if (keyboard_is_set(keyboard, BUTTON_3)) return '7';
    if (keyboard_is_set(keyboard, BUTTON_4)) return 'L';
    if (keyboard_is_set(keyboard, BUTTON_5)) return '2';
    if (keyboard_is_set(keyboard, BUTTON_6)) return '5';
    if (keyboard_is_set(keyboard, BUTTON_7)) return '8';
    if (keyboard_is_set(keyboard, BUTTON_8)) return '0';
    if (keyboard_is_set(keyboard, BUTTON_9)) return '3';
    if (keyboard_is_set(keyboard, BUTTON_10)) return '6';
    if (keyboard_is_set(keyboard, BUTTON_11)) return '9';
    if (keyboard_is_set(keyboard, BUTTON_12)) return 'S';
    return 0;
}

void UART_Send_Message(const char *msg) {
    HAL_UART_Transmit(&huart6, (uint8_t *)msg, strlen(msg),
HAL_MAX_DELAY);
}

void UART_Send_Char(char ch) {
    if (ch == '\r') {UART_Send_Message("\r\n");}
    HAL_UART_Transmit(&huart6, (uint8_t *)&ch, 1, HAL_MAX_DELAY);
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {

    if (htim->Instance == TIM6) {
        int code = LOCK_CODE[current_pos] - '0';

        if (show_led) {
            int pulse = 0;
            switch (code) {

```

```

        case 1: pulse = 25;    htim4.Instance->CCR2 =
pulse; htim4.Instance->CCR3 = 0; htim4.Instance->CCR4 = 0; break;
        case 2: pulse = 50;    htim4.Instance->CCR2 =
pulse; htim4.Instance->CCR3 = 0; htim4.Instance->CCR4 = 0; break;
        case 3: pulse = 100;   htim4.Instance->CCR2 =
pulse; htim4.Instance->CCR3 = 0; htim4.Instance->CCR4 = 0; break;
        case 4: pulse = 25;    htim4.Instance->CCR3 =
pulse; htim4.Instance->CCR2 = 0; htim4.Instance->CCR4 = 0; break;
        case 5: pulse = 50;    htim4.Instance->CCR3 =
pulse; htim4.Instance->CCR2 = 0; htim4.Instance->CCR4 = 0; break;
        case 6: pulse = 100;   htim4.Instance->CCR3 =
pulse; htim4.Instance->CCR2 = 0; htim4.Instance->CCR4 = 0; break;
        case 7: pulse = 25;    htim4.Instance->CCR4 =
pulse; htim4.Instance->CCR2 = 0; htim4.Instance->CCR3 = 0; break;
        case 8: pulse = 50;    htim4.Instance->CCR4 =
pulse; htim4.Instance->CCR2 = 0; htim4.Instance->CCR3 = 0; break;
        case 9: pulse = 100;   htim4.Instance->CCR4 =
pulse; htim4.Instance->CCR2 = 0; htim4.Instance->CCR3 = 0; break;
    }
    } else {
        htim4.Instance->CCR2 = 0;
        htim4.Instance->CCR3 = 0;
        htim4.Instance->CCR4 = 0;
    }

    if (show_sound) {
        int code = LOCK_CODE[current_pos] - '0';
        switch (code) {
            case 1: htim1.Instance->PSC = 59; break;
            case 2: htim1.Instance->PSC = 54; break;
            case 3: htim1.Instance->PSC = 49; break;
            case 4: htim1.Instance->PSC = 44; break;
            case 5: htim1.Instance->PSC = 39; break;
            case 6: htim1.Instance->PSC = 34; break;
            case 7: htim1.Instance->PSC = 29; break;
            case 8: htim1.Instance->PSC = 24; break;
            case 9: htim1.Instance->PSC = 19; break;
        }

        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    } else {
        HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1);
    }
}

void Process_Input(char input) {
    UART_Send_Char(input);
    if (input == 'L') {
        show_led = !show_led;
        return;
    }
    if (input == 'S') {
        show_sound = !show_sound;
        return;
    }
    if (input == '_') {
        mode = 1;
    }
    if (mode == 0) {
        if (input == LOCK_CODE[current_pos]) {
            UART_Send_Message("..Correct..");
            current_pos++;
            if (current_pos == CODE_LENGTH) {
                UART_Send_Message("\r\nUnlocked!\r\n");
                System_Reset();
            }
        } else {
            UART_Send_Message("..Incorrect..");
            attempt_count++;
            if (attempt_count >= MAX_ATTEMPTS) {
                UART_Send_Message("\r\nTry again(\r\n");
                System_Reset();
            }
        }
    }
    } else if (mode == 1) {

```

```

        Change_Code_Logic(input);
    } else if (mode == 2) {
        Confirm_New_Code(input);
    }
}

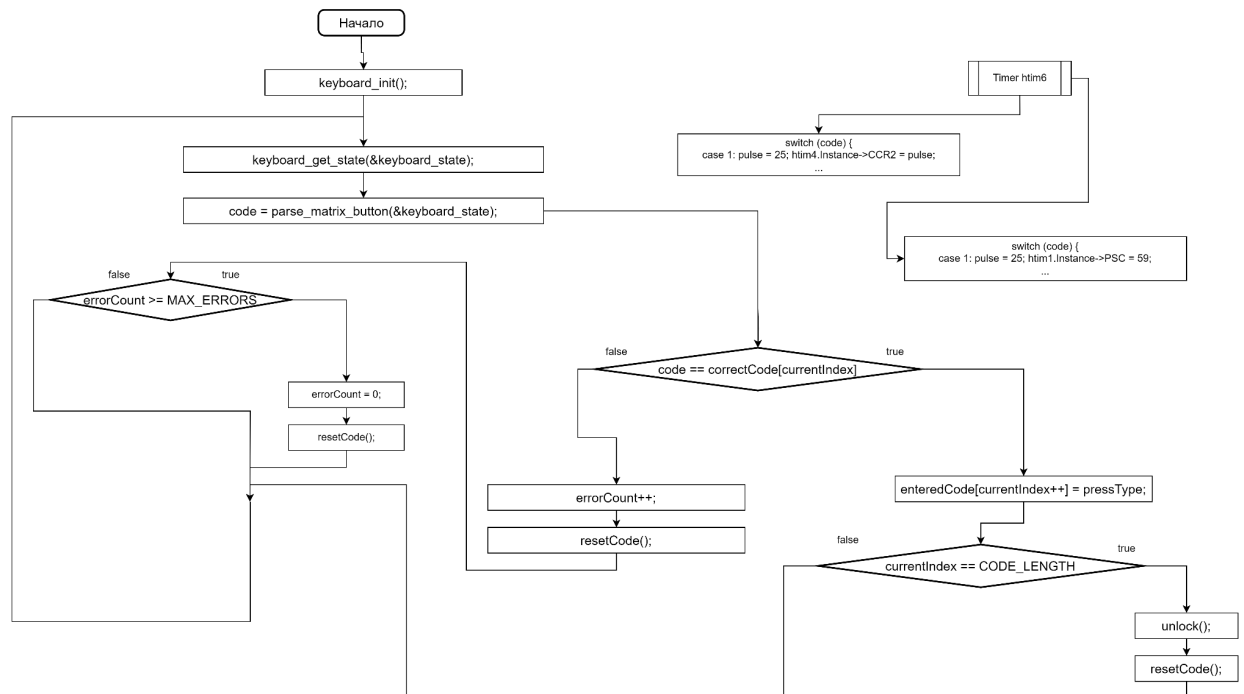
void Confirm_New_Code(char input) {
    if (input == 'y' || input == 'Y') {
        strncpy(LOCK_CODE, new_code, CODE_LENGTH);
        UART_Send_Message("\r\nNew code set successfully!\r\n");
        System_Reset();
    } else if (input == 'n' || input == 'N') {
        UART_Send_Message("New code discarded.\r\n");
        System_Reset();
    }
}

void Change_Code_Logic(char input) {
    if (input != ' ') {
        new_code[new_code_pos] = input;
        new_code_pos++;
    }
    if (new_code_pos >= CODE_LENGTH) {
        UART_Send_Message("\r\nSet as new code? (y/n)\r\n");
        mode = 2;
    }
}

void System_Reset(void) {
    current_pos = 0;
    new_code_pos = 0;
    attempt_count = 0;
    mode = 0;
    memset(input_buffer, 0, CODE_LENGTH);
}

```

Описание работы программы



Вывод

В ходе работы мы изучили и использовали интерфейсы для опроса матричной клавиатуры