МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине

“Проектирование вычислительных систем”

Вариант №5

Студент:

Чернова Анна Ивановна

Миху Вадим Дмитриевич

Группа P34301

Преподаватель:

Пинкевич Василий Юрьевич

г. Санкт-Петербург

2024

Цель работы

1. Получить базовые знания об устройстве и режимах работы таймеров микроконтроллерах.
2. Получить навыки использования таймеров и прерываний от таймеров.
3. Получить навыки использования аппаратных каналов ввода-вывода таймеров.

Задание

Разработать программу, которая использует таймеры для управления яркостью светодиодов и излучателем звука (по прерыванию или с использованием аппаратных каналов). Блокирующее ожидание (функция HAL_Delay()) в программе использоваться не должно.

Стенд должен поддерживать связь с компьютером по UART и выполнять указанные действия в качестве реакции на нажатие кнопок на клавиатуре компьютера. В данной лабораторной работе каждая нажатая кнопка (символ, отправленный с компьютера на стенд) обрабатываются отдельно, ожидание ввода полной строки не требуется.

Для работы с UART на стенде можно использован один из двух вариантов драйвера (по прерыванию и по опросу) на выбор исполнителя. Поддержка двух вариантов не требуется. Частота синхросигнала процессорного ядра и сигнала ШИМ для управления яркостью светодиодов (если используется) должны соответствовать указанным в варианте 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"


#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
#define UART_BUFFER_SIZE 64

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 recieved_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);
/* USER CODE BEGIN PFP */
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);

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();

  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;

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

  }
```

```
}


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') {
```

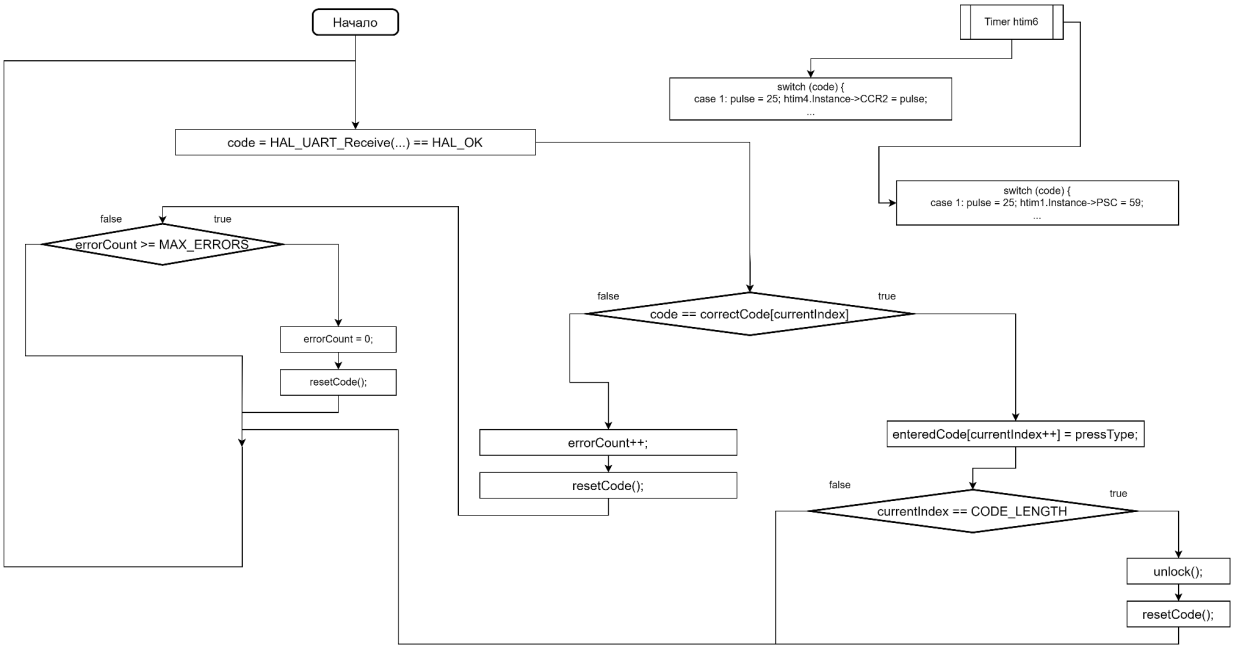
```
            show_sound = !show_sound;
            return;
        }
        if (input == '_') {
            mode = 1;
        }
        if (mode == 0) {
                if (input == LOCK_CODE[current_pos]) {
                    UART_Send_Message("..Correct..");
                    current_pos++;
                    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 таймеры для управления яркостью светодиодов и звуком динамика.