МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №1

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 о принципах устройства стенда SDK-1.1M и программировании микроконтроллеров.

2. Изучить устройство интерфейсов ввода-вывода общего назначения (GPIO) в микроконтроллерах и приемы использования данных интерфейсов.

Задание

Разработать и реализовать драйверы управления светодиодными индикаторами и чтения состояния кнопки стенда SDK-1.1M (расположены на боковой панели стенда). Обработка нажатия кнопки в программе должна включать программную защиту от дребезга. Функции и другие компоненты драйверов должны быть универсальными, т. е. пригодными для использования в любом из вариантов задания и не должны содержать прикладной логики программы. Функции драйверов должны быть неблокирующими, то есть не должны содержать ожиданий события (например, нажатия кнопки). Также, в драйверах не должно быть пауз с активным ожиданием (функция HAL_Delay() и собственные варианты с аналогичной функциональностью).

При необходимости параллельного выполнения разных процессов (например, опроса кнопки и ожидания перед переключением светодиодов) следует организовать псевдо параллельную работу процессов в стиле кооперативной многозадачности. Это возможно сделать без существенной потери точности соблюдения необходимых интервалов времени между действиями, поскольку действия выполняются очень быстро по сравнению с промежутками ожидания между ними. Каждый процесс (который может быть выражен всего в нескольких строках кода) должен использовать не активное ожидание (HAL_Delay()), а считывать текущее значение миллисекундного счетчика (HAL_GetTick()), проверяя, прошло ли необходимое количество времени для выполнения следующего действия. После проверки и (при необходимости) выполнения действия управление должно передаваться следующему процессу, чтобы он тоже мог провести такую проверку. Контакты подключения кнопки и светодиодов должны быть настроены в режиме GPIO.

Использование прерываний и дополнительных таймеров (кроме SysTick) не допускается.

Написать программу с использованием разработанных драйверов в соответствии с вариантом задания.

Вариант 5

Реализовать «кодовый замок». После ввода единственно верной последовательности не менее чем из восьми коротких и длинных нажатий должен загореться зеленый светодиод, обозначающий «открытие» замка. Светодиод горит некоторое время, потом гаснет, и система вновь переходит в режим ввода.

Каждый неправильно введенный элемент последовательности должен сопровождаться миганием красного светодиода и сбросом в «начало», каждый правильный – миганием

желтого. После трёх неправильных вводов начинает мигать красный светодиод, и через некоторое время возвращается в режим ввода.

Если за некоторое ограниченное время код не введен до конца, происходит сброс в «начало».

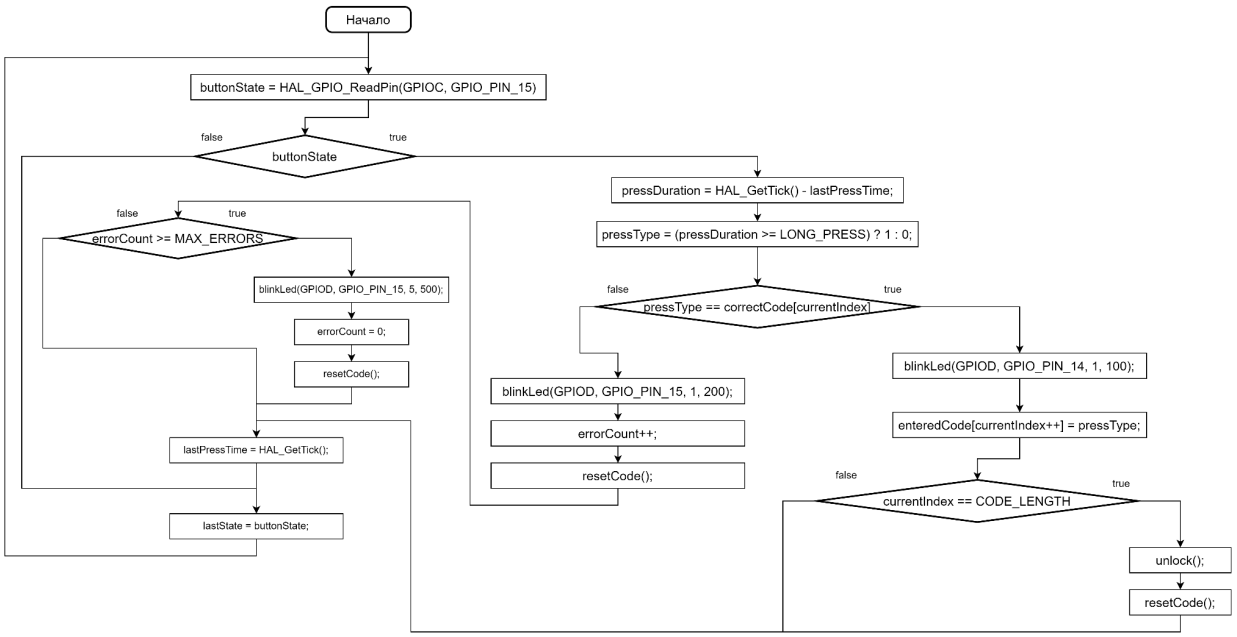
Выполнение

Листинг разработанной программы с комментариями

```
/* Private define
-----------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define SHORT_PRESS 100  // Миллисекунд для короткого нажатия
#define LONG_PRESS 500   // Миллисекунд для длинного нажатия
#define TIMEOUT 10000    // Таймаут на ввод кода (мс)
#define MAX_ERRORS 3     // Количество ошибок перед блокировкой
#define CODE_LENGTH 8    // Длина кода
/* USER CODE END PD */

/* Private variables
---------------------------------------------------------*/

/* USER CODE BEGIN PV */
uint8_t correctCode[CODE_LENGTH] = {1, 0, 1, 0, 1, 0, 1, 0};  // Пример
правильного кода
uint8_t enteredCode[CODE_LENGTH];  // Введённый пользователем код
uint8_t currentIndex = 0;          // Индекс для отслеживания ввода
uint8_t errorCount = 0;            // Счётчик ошибок
unsigned long lastPressTime = 0;   // Время последнего нажатия
uint8_t buttonState, lastState, buttonOn;
/* USER CODE END PV */

/* Private function prototypes
-----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);

/* Private user code
---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

void My_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin : PC15 */
  GPIO_InitStruct.Pin = GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : PD13 PD14 */
  GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

}

void resetCode() {
  currentIndex = 0;
  for (int i = 0; i < CODE_LENGTH; i++) {
        enteredCode[i] = 0;
```

```
  }
}

void unlock() {
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);  // Зелёный
светодиод
  HAL_Delay(5000);  // Замок открыт 5 секунд
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
}

void blinkLed(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, uint8_t times,
uint16_t delay) {
  for (uint8_t i = 0; i < times; i++) {
    HAL_GPIO_WritePin(GPIOx, GPIO_Pin, GPIO_PIN_SET);
    HAL_Delay(delay);
    HAL_GPIO_WritePin(GPIOx, GPIO_Pin, GPIO_PIN_RESET);
    HAL_Delay(delay);
  }
}

void buttonIn(){
  if(HAL_GetTick() - lastPressTime > 50) {  // debounce
    buttonState = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_15) == GPIO_PIN_RESET;

    if (buttonState && !lastState) {
      unsigned long pressDuration = HAL_GetTick() - lastPressTime;
      uint8_t pressType = (pressDuration >= LONG_PRESS) ? 1 : 0;

      if (pressType == correctCode[currentIndex]) {
        blinkLed(GPIOD, GPIO_PIN_14, 1, 100);  // Жёлтый светодиод на
верный ввод
        enteredCode[currentIndex++] = pressType;

        if (currentIndex == CODE_LENGTH) {
          unlock();  // Открываем замок
          resetCode();
        }
      } else {
        blinkLed(GPIOD, GPIO_PIN_15, 1, 200);  // Красный светодиод на
ошибку
        errorCount++;
        resetCode();
      if (errorCount >= MAX_ERRORS) {
                    blinkLed(GPIOD, GPIO_PIN_15, 5, 500);  // Красный
мигает 5 раз на блокировку
                    errorCount = 0;
                    resetCode();
                  }
              }
              lastPressTime = HAL_GetTick();
            }
            lastState = buttonState;
        }
      }
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
      HAL_Init();
      SystemClock_Config();
      MX_GPIO_Init();
      My_GPIO_Init();

      lastPressTime = HAL_GetTick();

      while (1)
      {
      buttonIn();  // Обрабатываем нажатия кнопки
      }
}
```

Описание работы программы



Вывод

В ходе работы мы изучили и использовали интерфейс GPIO для обмена данными с платой и реализации кодового замка на основе светодиодов, реализовали обработку нажатия кнопки и защиты от дребезга.