Университет ИТМО
Факультет программной инженерии и компьютерной техники

Дисциплина:
Проектирование вычислительных систем
Лабораторная работа № 1

Выполнили: Тищенко Б.В., Васькин А.А.
Группа: P34112

Санкт-Петербург
2021

Вариант 5

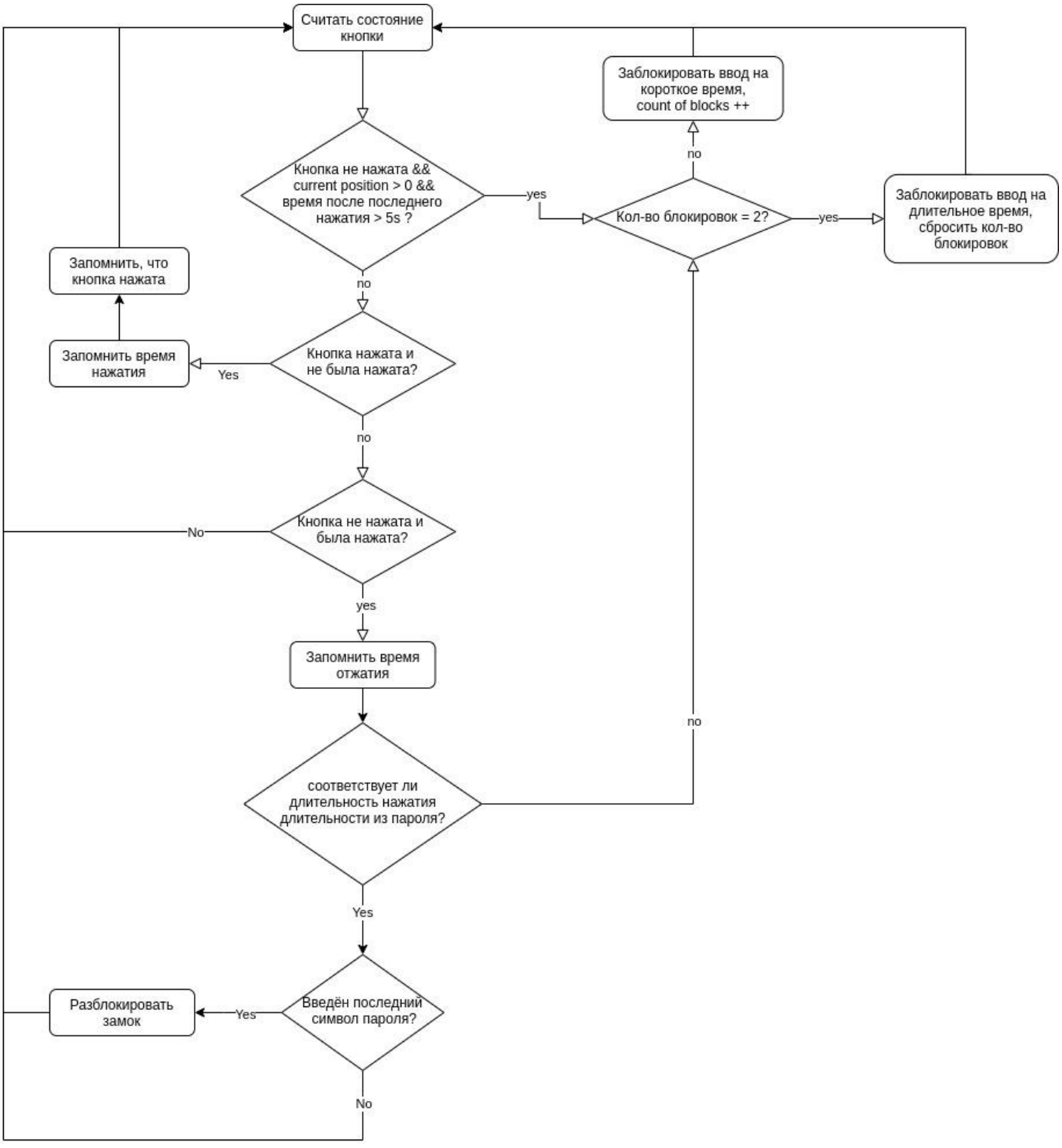
Реализовать «кодовый замок». После ввода единственно верной последовательности из не менее чем восьми коротких и длинных нажатий должен загореться зеленый светодиод, обозначающий «открытие» замка. Светодиод горит некоторое время, потом гаснет, и система вновь переходит в «режим ввода». Каждый неправильно 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 код не введен до конца за некоторое ограниченное время, происходит сброс в «начало».

Выполнение

Описание программы

Мы написали тонкий слой абстракции в `gpioutil.h` (исходный код ниже) и простыми инструкциями вида `wait(100), toggle(PIN_RED)` описали логику мигания лампочки.
Пароль представлен массивом целых чисел длины восемь. В бесконечном цикле происходит считывание состояния кнопки, замеряется время нажатия. Короткие нажатия кодируются числом ноль, длинные -- единицей. После каждого успешного ввода указатель на текущий символ пароля сдвигается вперед. Как только все символы введены успешно, происходит мигание зеленой лампочки.

Блок-схема


Считать состояние кнопки
Кнопка не нажата && current position > 0 && время после последнего нажатия > 5s ?
yes
Кол-во блокировок = 2?
yes
Заблокировать ввод на длительное время, сбросить кол-во блокировок
no
Заблокировать ввод на короткое время, count of blocks ++
no
Запомнить, что кнопка нажата
Запомнить время нажатия
Yes
Кнопка нажата и не была нажата?
no
Кнопка не нажата и была нажата?
No
yes
Запомнить время отжатия
соответствует ли длительность нажатия длительности из пароля?
no
Yes
Разблокировать замок
Yes
Введён последний символ пароля?
No

Инструментарий

1. STM32CubeIDE
2. SDK 1.1M (427)
3. Колпачок от ручки

Исходный код

```
// gpioutil.h
#define PIN_RED GPIO_PIN_15
#define PIN_YELLOW GPIO_PIN_14
#define PIN_GREEN GPIO_PIN_13

void toggle(uint16_t pin);
void blink(uint16_t pin, uint32_t
delay, uint16_t times);
void wait(uint32_t time);
```

```
//gpioutil.c
#include "gpioutil.h"

void toggle(uint16_t pin) {
      HAL_GPIO_TogglePin(GPIOD, pin);
}

void wait (uint32_t time) {
      HAL_Delay(time);
}

void blink (uint16_t pin, uint32_t delay,
uint16_t times) {
      for (uint16_t i = 0; i < times; i++) {
      toggle(pin);
      wait(delay);
      toggle(pin);
      wait(delay);
      }
}
```

```
//main.c: функции
void block(
  uint16_t * current_position,
  uint16_t * is_pushed_down,
  uint16_t * block_times
) {
  * current_position = 0;
  * is_pushed_down = 0;
  ( * block_times) ++;
  if ( * block_times < 3) {
      blink(PIN_RED, 250, 3);
  } else {
      * block_times = 0;
      blink(PIN_RED, 350, 10);
  }
}

void checkBtn(
  uint16_t btn_input,
  uint16_t * current_position,
```

```
//main.c: цикл
uint16_t password[8];
setupPassword(password);
uint16_t btn_input;

uint16_t current_position = 0;
uint16_t block_times = 0;
uint16_t is_pressed = 0;
uint32_t press_timestamp;
uint32_t release_timestamp;
uint32_t short_press_time_ms = 100;
uint32_t long_press_time_ms = 1000;

GPIO_PinState btn_state;
while (1) {
  btn_state = HAL_GPIO_ReadPin(GPIOC,
GPIO_PIN_15);
  if (btn_state == GPIO_PIN_SET &&
current_position > 0 &&
```

```
  uint16_t * password,
  uint16_t * block_times,
  uint16_t * is_pushed_down
) {
  if (btn_input == password[ *
current_position]) {
      ( * current_position) ++;
      if ( * current_position == 8) {
      * current_position = 0;
      * block_times = 0;
      blink(PIN_GREEN, 350, 10);
      } else {
        blink(PIN_YELLOW, 250, 3);
      }
  } else {
      block(current_position,
is_pushed_down, block_times);
  }
}

uint16_t checkTime(
  uint32_t push_down_timestamp,
  uint32_t pull_up_timestamp,
  uint32_t long_push_time_ms
) {
  return (pull_up_timestamp -
push_down_timestamp ⩾
long_push_time_ms) ? 1 : 0;
}
```

```
      checkTime(release_timestamp,
HAL_GetTick(), 5000) == 1) {
      block( & current_position, & is_pressed, &
block_times);
  }
  if (is_pressed && btn_state == GPIO_PIN_SET) {
      release_timestamp = HAL_GetTick();
      btn_input = checkTime(press_timestamp,
release_timestamp, long_press_time_ms);
      is_pressed = 0;
      checkBtn(btn_input, & current_position,
password, & block_times, & is_pressed);
  }
  if (!is_pressed && btn_state ==
GPIO_PIN_RESET) {
      press_timestamp = HAL_GetTick();
      wait(short_press_time_ms); // debounce
      btn_state = HAL_GPIO_ReadPin(GPIOC,
GPIO_PIN_15); // debounce
      is_pressed = btn_state == GPIO_PIN_RESET;
  }
}
```

Выводы

Очень важно не путать кнопку отладки с кнопкой запуска. Было трудно подобрать значения времени для определения коротких и длинных нажатий, возвращения в исходное состояние. Даже для такой простой программы блок-схему строить довольно утомительно.