Федеральное государственное бюджетное образовательное учреждение
высшего образования «Воронежский государственный университет»

Отчет по лабораторной работе № 2
По курсу «Введение в интернет вещей»

«Порты ввода-вывода общего назначения»

Выполнила:
Меркутова Кристина Денисовна
3 группа

Воронеж 2025

Цель работы:

Изучить работу с GPIO портами на ESP32, научиться управлять выводами как в режиме вывода, так и в режиме чтения, освоить создание простых управляющих устройств.

Ход выполнения работы:

1. Ответьте на вопросы в теоретической части максимально подробно.

2. Соберите схему практической части и напишите программу для микроконтроллера. Убедитесь, что код выполняется верно, используйте симулятор wokwi или лабораторный стенд. Скриншоты или фото устройства и листинг программы микроконтроллера обязательны.

1. Теоретическая часть. Ответьте на вопросы

GPIO — это программируемые цифровые интерфейсы микроконтроллеров и микропроцессоров, позволяющие взаимодействовать с внешними устройствами на уровне отдельных битов (логических сигналов 0/1).

Назначение GPIO портов

GPIO используются для:

- Подключения кнопок, переключателей, датчиков (вход).
- Управления светодиодами, реле, транзисторами, маломощными нагрузками (выход).
- Реализации простых цифровых протоколов.
- Передачи сигналов управления между микроконтроллерами и другими ИС.
- Генерации или захвата импульсов

Аппаратная реализация GPIO

На уровне кристалла микроконтроллера каждый GPIO-пин подключён к:

- Выходному буферу (обычно триггер + драйвер MOSFET), который может быть:

        - Отключён (высокий импеданс — Hi-Z),
        - Установлен в «0» (подключён к земле),
        - Установлен в «1» (подключён к питанию через транзистор).
    - Входному буферу (триггер Шмитта или обычный буфер), который считывает уровень напряжения на пине и передаёт его в регистры процессора.
    - Мультиплексору, позволяющему выбрать между:
        - Режимом GPIO (прямое управление),
        - Альтернативными функциями (например, UART, SPI, PWM и т.д.).

Также часто присутствуют:

- Подтяжка (pull-up) и подтяжка к земле (pull-down) — внутренние резисторы (~20–50 кОм).
- Защитные диоды от статики и перенапряжений.
- Возможность генерировать прерывания по изменению уровня или по уровню.

Типы режимов работы GPIO

a) Вход

- Пин считывает внешний сигнал.
- Может быть:
    - Floating input — без подтяжки (чувствителен к помехам).
    - Input with pull-up — внутренний резистор подтягивает к VCC.
    - Input with pull-down — внутренний резистор подтягивает к GND.

b) Выход

- Пин управляет внешней цепью.
- Типы:
    - Push-pull — активно выдаёт 0 или 1.

    - Open-drain — может только «затягивать» к земле или быть в Hi-Z; требует внешней подтяжки для лог. 1.
- Может иметь разные уровни drive strength (сила тока, которую может отдать пин — обычно 2–20 мА).

c) Альтернативные функции

- Пин используется не как GPIO, а как часть встроенного периферийного модуля:
    - UART (TX/RX),
    - SPI (SCK, MOSI, MISO),
    - I²C (SDA, SCL),
    - PWM (таймерные выходы),
    - ADC (аналоговый вход — в этом случае цифровой буфер часто отключается).
- Выбирается через регистры конфигурации (например, в STM32 — через AFRL/AFRH).

Ограничения и рекомендации по подключению внешних устройств

Ограничения:

- Максимальный ток на пин: обычно 4–20 мА.
- Суммарный ток на порт или весь чип: может быть ограничен (например, 100 мА на весь MCU).
- Напряжение: GPIO обычно работают в диапазоне 1.8–3.3 В или 5 В. Превышение — повреждение!
- Ёмкостная нагрузка: слишком большая ёмкость (длинные провода, входы ИС) замедляет фронт сигнала.
- Частота переключения: ограничена внутренней задержкой и нагрузкой.

Рекомендации:

- Используйте токовые ограничители (резисторы) при подключении светодиодов.

- Для управления мощными нагрузками (реле, моторы) — транзисторы или драйверы.
- При работе с 5 В устройствами и 3.3 В MCU — уровневые преобразователи.
- Используйте внешние подтягивающие резисторы, если внутренние недостаточны (например, для I²C).
- Избегайте «висящих» входов — всегда подключайте к GND/VCC через резистор или используйте внутреннюю подтяжку.

Дребезг контактов

Механические контакты (кнопки, реле) при замыкании/размыкании не переходят мгновенно в новое состояние. Вместо этого они многократно отскакивают, вызывая серию коротких импульсов (от микросекунд до миллисекунд). Это приводит к ложным срабатываниям.

Как бороться?

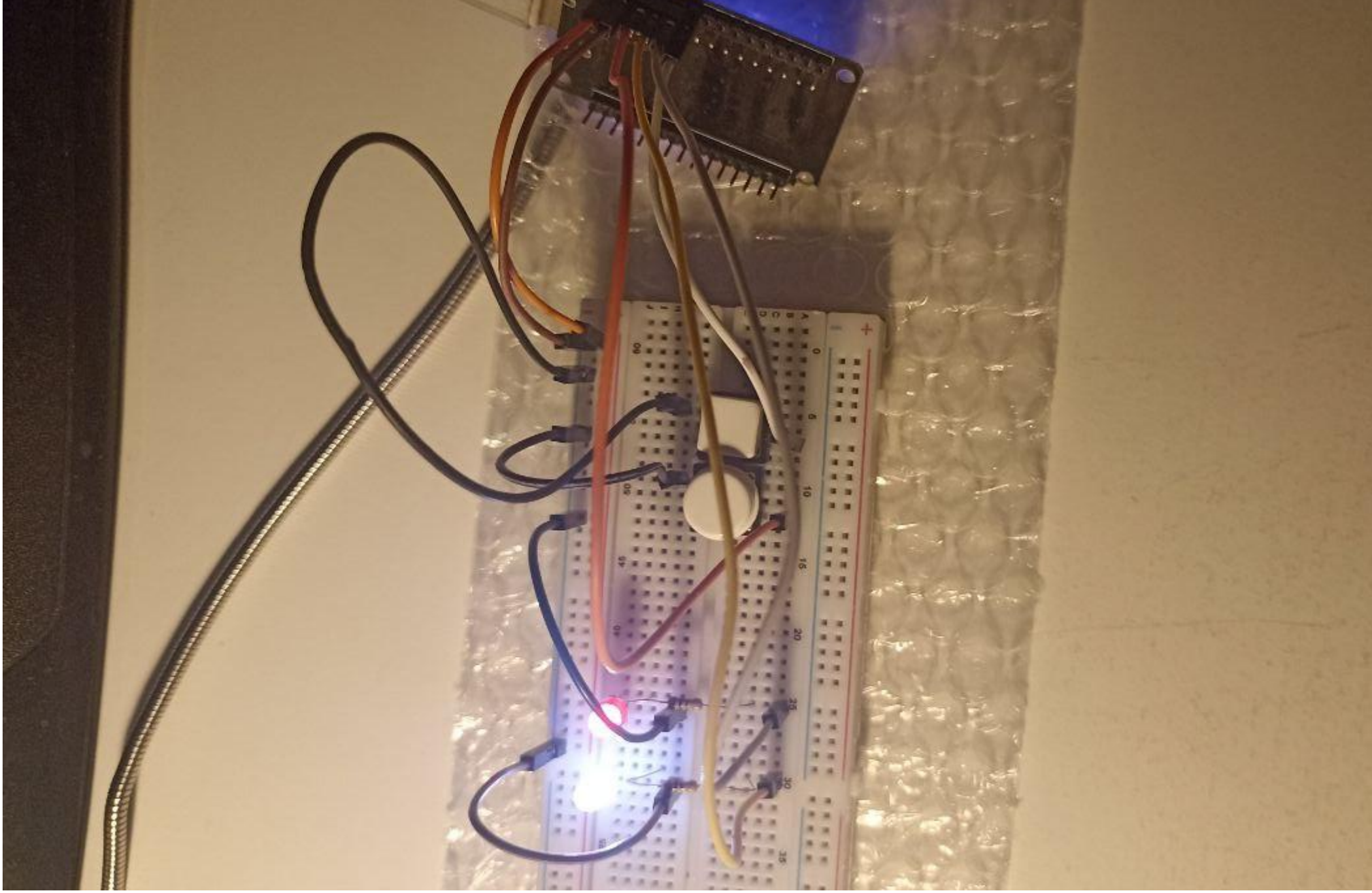
Аппаратные методы:

- RC-фильтр (резистор + конденсатор) — сглаживает скачки.
- Шмитт-триггер — устраняет неоднозначность при медленных фронтах.
- Дедикated IC — специализированные микросхемы подавления дребезга.

Программные методы:

- Дебаунсинг по времени:
    - При изменении состояния — ждать 10–50 мс и проверить снова.

2. Практическая часть:



```c
#include <stdio.h>
#include <stdbool.h>
#include <inttypes.h>
#include "driver/gpio.h"
#include "esp_timer.h"
#include "esp_log.h"


#define LED1_GPIO      GPIO_NUM_2
#define LED2_GPIO      GPIO_NUM_4
#define BUTTON1_GPIO   GPIO_NUM_15
#define BUTTON2_GPIO   GPIO_NUM_16

#define MIN_BLINK_PERIOD_MS   200
#define MAX_BLINK_PERIOD_MS   3000
#define DEFAULT_BLINK_PERIOD_MS 1000
#define DEBOUNCE_DELAY_MS     50

static const char *TAG = "BLINK_LED_DEMO";

static uint32_t blink_period_ms = DEFAULT_BLINK_PERIOD_MS;
static bool led_state = false;
static uint64_t last_blink_time = 0;
static uint64_t last_button1_press = 0;
static uint64_t last_button2_press = 0;


static void init_gpio(void) {
    gpio_set_direction(LED1_GPIO, GPIO_MODE_OUTPUT);
    gpio_set_direction(LED2_GPIO, GPIO_MODE_OUTPUT);

    gpio_set_direction(BUTTON1_GPIO, GPIO_MODE_INPUT);
    gpio_set_direction(BUTTON2_GPIO, GPIO_MODE_INPUT);

    gpio_set_pull_mode(BUTTON1_GPIO, GPIO_PULLUP_ONLY);
    gpio_set_pull_mode(BUTTON2_GPIO, GPIO_PULLUP_ONLY);

    gpio_set_level(LED1_GPIO, 0);
    gpio_set_level(LED2_GPIO, 0);
}

static uint64_t get_time_ms(void) {
    return esp_timer_get_time() / 1000;
}

// Проверка нажатия кнопки с защитой от дребезга
static bool is_button_pressed(gpio_num_t button_pin, uint64_t *last_press_time) {
    uint64_t current_time = get_time_ms();

    if (gpio_get_level(button_pin) == 0) {
        // Проверка дребезга - кнопка должна быть нажата достаточно долго
        if (current_time - *last_press_time > DEBOUNCE_DELAY_MS) {

            uint32_t debounce_check = 0;
            for (int i = 0; i < 10; i++) {
                if (gpio_get_level(button_pin) == 0) debounce_check++;
                esp_rom_delay_us(100);
            }

            if (debounce_check >= 8) {
                *last_press_time = current_time;
                return true;
            }
        }
    }
    return false;
}

static void update_leds(void) {
    uint64_t current_time = get_time_ms();

    if (current_time - last_blink_time >= blink_period_ms) {
        last_blink_time = current_time;
        led_state = !led_state;

        gpio_set_level(LED1_GPIO, led_state);
        gpio_set_level(LED2_GPIO, led_state);

        ESP_LOGI(TAG, "LED %s (период: %" PRIu32 " мс)",
                 led_state ? "ВКЛ" : "ВЫКЛ", blink_period_ms);
    }
}


static void handle_buttons(void) {

    if (is_button_pressed(BUTTON1_GPIO, &last_button1_press)) {
        if (blink_period_ms < MAX_BLINK_PERIOD_MS) {
            blink_period_ms += 100;
            ESP_LOGI(TAG, "Период увеличен до %" PRIu32 " мс", blink_period_ms);
        } else {
            ESP_LOGI(TAG, "Достигнут максимальный период (%d мс)", MAX_BLINK_PERIOD_MS);
        }
    }

    if (is_button_pressed(BUTTON2_GPIO, &last_button2_press)) {
        if (blink_period_ms > MIN_BLINK_PERIOD_MS) {
            blink_period_ms -= 100;
            ESP_LOGI(TAG, "Период уменьшен до %" PRIu32 " мс", blink_period_ms);
        } else {
            ESP_LOGI(TAG, "Достигнут минимальный период (%d мс)", MIN_BLINK_PERIOD_MS);
        }
    }
}

void app_main(void) {
    ESP_LOGI(TAG, "Инициализация системы...");

    init_gpio();

    last_blink_time = get_time_ms();
    last_button1_press = get_time_ms();
    last_button2_press = get_time_ms();

    ESP_LOGI(TAG, "Система запущена!");
    ESP_LOGI(TAG, "Стартовый период мигания: %" PRIu32 " мс", blink_period_ms);
    ESP_LOGI(TAG, "Кнопка 1 (GPIO%d): увеличить период", BUTTON1_GPIO);
    ESP_LOGI(TAG, "Кнопка 2 (GPIO%d): уменьшить период", BUTTON2_GPIO);
    ESP_LOGI(TAG, "Диапазон периода: %d-%d мс", MIN_BLINK_PERIOD_MS, MAX_BLINK_PERIOD_MS);

    while (1) {
        update_leds();

        handle_buttons();

        esp_rom_delay_us(1000);
    }
}
```