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

Отчет по лабораторной работе № 5
По курсу «Введение в интернет вещей»

«шина связи 1-ware»

Выполнила:
Меркутова Кристина Денисовна
3 группа

Воронеж 2025

Цель работы:

Освоить принципы функционирования и программной реализации однопроводной шины данных 1-Wire на микроконтроллере ESP32, включая инициализацию шины, обнаружение устройств, обмен данными с датчиками (например, DS18B20).

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:

1. Какова основная особенность физической реализации шины 1-Wire, отличающая её от других последовательных интерфейсов (например, I²C или SPI)?
   (Подсказка: количество проводов + питание.)
   Ответ: ключевое отличие — это совмещение в одном проводе функций передачи данных и питания подключаемых устройств, что позволяет создавать крайне простые и дешёвые сети датчиков с минимальным количеством соединений.
2. Опишите этапы инициализации шины 1-Wire: какие сигналы посылает мастер и как на них реагируют ведомые устройства?
   Ответ:
   Этап 1: Сигнал "Сброс" от Мастера
   Мастер на короткое время переводит шину в состояние низкого логического уровня.
   Цель: Этот продолжительный низкий уровень выполняет две функции:
   1. Сброс всех ведомых устройств: Каждое устройство на шине воспринимает этот длинный низкий уровень как команду "стоп". Все устройства внутренне сбрасывают свои логические

состояния и готовятся к приему команд. Это аналогично аппаратному сбросу микропроцессора.

2. Обнаружение конфликтов: если бы мастер просто слушал шину, он не смог бы отличить тишину от передачи '1' всеми устройствами. Сигнал сброса создает четкую точку отсчета.

Этап 2: Ответный сигнал "Присутствие" от ведомых устройств

- Действие мастера: после завершения импульса сброса мастер переводит свой выход в высокоимпедансное состояние (логическая '1', "отпускает" шину). Физически шина подтягивается резистором к напряжению питания (+5В).
- Действие ведомых устройств: каждое ведомое устройство, обнаружившее сигнал сброса, должно ответить. Чтобы подтвердить свое присутствие на шине, устройство само активно подтягивает шину к низкому уровню на короткое время.
- Как это выглядит на шине: когда мастер "отпускает" шину, напряжение на ней начинает расти. Но ведомые устройства почти сразу же "перехватывают" инициативу и снова притягивают шину к '0'. В результате на осциллограмме видно, что после длинного низкого уровня от мастера шина поднимается, но почти сразу же снова опускается на короткий импульс, а затем окончательно поднимается до высокого уровня.

Этап 3: Интерпретация результата мастером

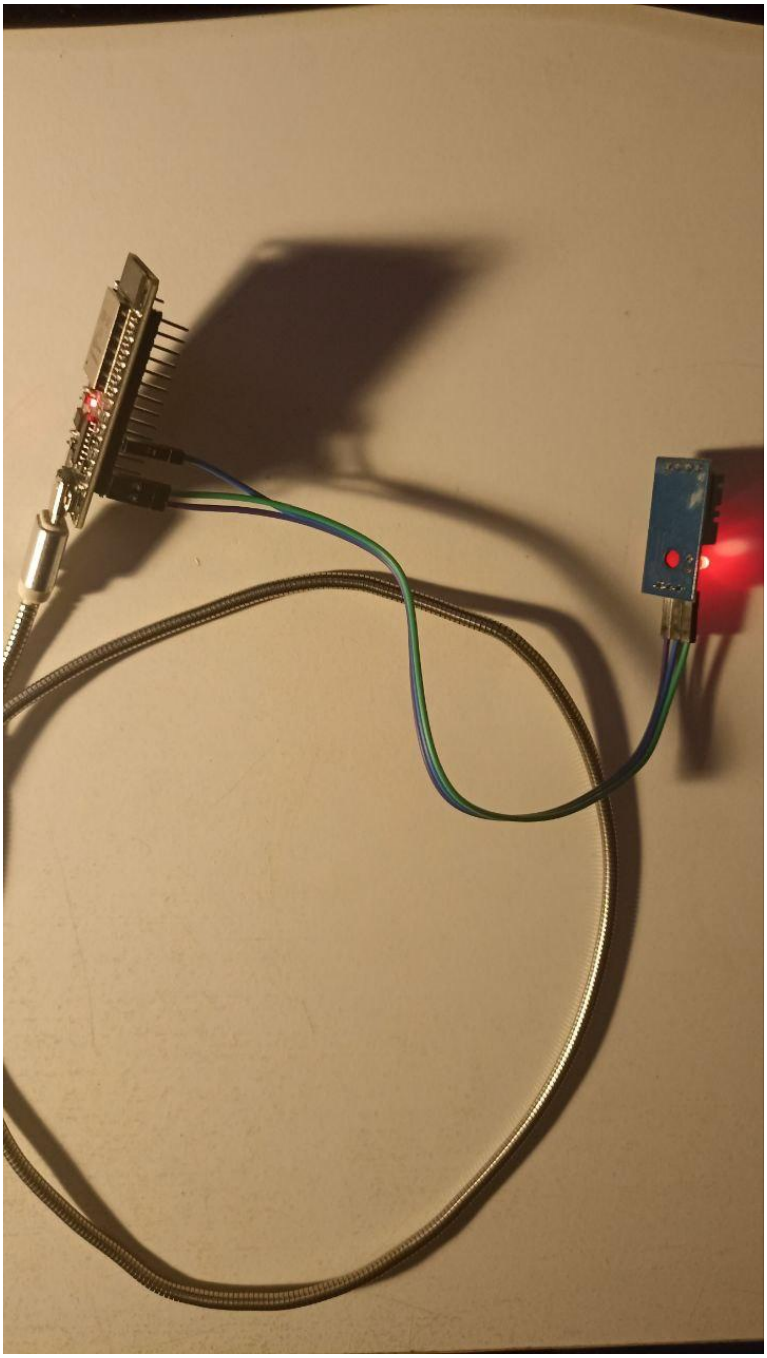
- Ситуация А: Устройства есть. Мастер, переключившись в режим чтения, наблюдает этот короткий низкий уровень (импульс присутствия). Это означает, что на шине есть одно или несколько ведомых устройств, готовых к работе.
- Ситуация Б: Устройств нет. Если мастер, отпустив шину, видит, что она сразу же поднялась до высокого уровня и осталась там (импульс присутствия отсутствует), это означает, что на шине нет ни одного отвечающего устройства.

3. Почему датчик DS18B20 требует задержки после запуска преобразования температуры? Как можно организовать «неразрушающее» (non-blocking) ожидание результата?

   Ответ: запуск преобразования температуры — это асинхронная операция, поэтому требуется задержка. Организовать «неразрушающее» (non-blocking) ожидание результата можно используя таймер или использование паразитного питания и команды "Read Power Supply".

2. Практическая часть:

1. Соберите схему, используя модуль DS18B20, обязательно используйте резистор 4.7 кОМ. Напишите код для получения данных температуры с датчика используя esp-idf, библиотеку для работы с датчиком по вашему выбору. Возможна реализация как в wokwi так и на макетной плате.



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_log.h"
#include "driver/gpio.h"
#include "esp_system.h"
#include "dht.h"

static const dht_sensor_type_t sensor_type = DHT_TYPE_AM2301;
static const gpio_num_t dht_gpio = 4;

static const char *TAG = "AM2302_Example";

void configure_dht_sensor(void) {
    gpio_config_t io_conf = {
        .pin_bit_mask = (1ULL << dht_gpio),
        .mode = GPIO_MODE_INPUT_OUTPUT_OD,
        .pull_up_en = GPIO_PULLUP_ENABLE,
        .pull_down_en = GPIO_PULLDOWN_DISABLE,
        .intr_type = GPIO_INTR_DISABLE
    };
    gpio_config(&io_conf);
    gpio_set_level(dht_gpio, 1);
    vTaskDelay(pdMS_TO_TICKS(100));
}

void dht_task(void *pvParameters) {
    int16_t temperature = 0;
    int16_t humidity = 0;
    int error_count = 0;

    configure_dht_sensor();

    ESP_LOGI(TAG, "Запуск чтения с датчика AM2302...");

    while (1) {
        esp_err_t result = dht_read_data(sensor_type, dht_gpio, &humidity, &temperature);

        if (result == ESP_OK) {
            error_count = 0;
            ESP_LOGI(TAG, "Влажность: %d.%d%%", humidity / 10, abs(humidity % 10));
            ESP_LOGI(TAG, "Температура: %d.%d°C", temperature / 10, abs(temperature % 10));
        } else {
            error_count++;
            ESP_LOGE(TAG, "Ошибка чтения датчика (попытка %d): %s", error_count, esp_err_to_name(result));

            if (error_count >= 3) {
                ESP_LOGW(TAG, "Перенастройка датчика...");
                configure_dht_sensor();
                error_count = 0;
            }
        }

        vTaskDelay(pdMS_TO_TICKS(5000));
    }
}

void app_main(void) {
    xTaskCreate(dht_task, "dht_task", configMINIMAL_STACK_SIZE * 4, NULL, 5, NULL);
}
```