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

Отчет по лабораторной работе № 4
По курсу «Введение в интернет вещей»

«Широтно-импульсная модуляция»

Выполнила:
Меркутова Кристина Денисовна
3 группа

Воронеж 2025

Цель работы:

Изучить основные понятия широтно-импульсной модуляции и ее применения с использованием микроконтроллеров ESP32. Научиться использовать ШИМ для работы с пьезоизлучателем и сервоприводом SG90.

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. Что такое ШИМ? В чём заключается принцип её работы?
   Ответ: ШИМ (Широтно-Импульсная Модуляция) — это метод управления мощностью, подаваемой на нагрузку, без изменения амплитуды напряжения питания. Вместо этого метод основан на изменении ширины (длительности) импульсов постоянной амплитуды и частоты. Основная идея — это быстрое включение и выключение питания нагрузки. Сигнал ШИМ представляет собой последовательность прямоугольных импульсов. Ключевой параметр — это отношение времени, когда сигнал включен (высокий уровень), к общему периоду импульса.
2. Какие основные параметры характеризуют ШИМ-сигнал? Дайте определения: скважность, частота, коэффициент заполнения.
   Ответ: Частота - количество полных циклов (импульс + пауза) сигнала за одну секунду. Измеряется в Герцах (Гц). Коэффициент заполнения - отношение длительности импульса (времени, когда сигнал высокий) к периоду сигнала. Обычно выражается в процентах (%). Скважность - величина, обратная коэффициенту заполнения. Показывает, во сколько раз период сигнала больше длительности импульса. Безразмерная величина.

3. Как связан коэффициент заполнения с эффективным (средним) значением ШИМ-сигнала?
   Ответ: Эффективное (среднее) значение напряжения ШИМ-сигнала прямо пропорционально коэффициенту заполнения.
4. Почему ШИМ эффективна для управления мощностью в нагрузке?
   Ответ: ШИМ эффективна по двум основным причинам: минимальные потери мощности на ключевом элементе**,** генерировать цифровой сигнал с помощью микроконтроллеров и цифровых схем проще и дешевле, чем создавать точные и мощные аналоговые схемы
5. В каких устройствах и системах применяется ШИМ?
   Ответ: системы управления двигателями, светотехника, источники питания, аудиотехника, сервоприводы.
6. Чем отличается аналоговое управление от цифрового с использованием ШИМ?
   Ответ: ключевое различие — это принцип действия: аналоговый метод плавно меняет уровень сигнала, а цифровой ШИМ-метод — его временные характеристики, работая в эффективном ключевом режиме
7. Как влияет частота ШИМ на работу светодиода и двигателя постоянного тока?
   Ответ:
   для светодиода:
   - Слишком низкая частота (<50-100 Гц): Глаз человека начинает замечать мерцание. Это вызывает дискомфорт и усталость. Светодиод будет явственно "моргать".
   - Оптимальная частота (> 200 Гц): Мерцание становится неразличимым для глаза из-за инерционности зрительного восприятия. Мы видим ровный свет с плавно регулируемой яркостью.
   - Слишком высокая частота: не оказывает негативного влияния на восприятие, но может быть ограничена быстродействием управляющей электроники.

   для двигателя постоянного тока:
   - Слишком низкая частота (<20 Гц - 1 кГц, зависит от двигателя): Двигатель может работать рывками, слышен характерный "писк" или гул на частоте ШИМ. КПД двигателя снижается, возможен перегрев.
   - Оптимальная частота (обычно 5–20 кГц): Двигатель работает плавно и бесшумно, такому вращению способствует

механическая инерция ротора. Шум и пульсации тока минимизированы.
    - Слишком высокая частота: увеличиваются потери на переключение в силовом ключе (транзисторе), что снижает общий КПД системы и может привести к его перегреву.

8. Какие типы ШИМ (симметричная, асимметричная) вы знаете и в чём их различие?
   Ответ: это различие относится в основном к аппаратной реализации генераторов ШИМ в микроконтроллерах и специализированных схемах (таймерах).
   Асимметричная (быстрорегулируемая) ШИМ (Fast PWM)
    - Принцип: Счетчик постоянно увеличивает свое значение от 0 до максимума, затем сбрасывается в 0. Сигнал на выходе устанавливается в высокий уровень при сбросе и сбрасывается в низкий, когда значение счетчика совпадает с заданным регистром сравнения.
    - Особенности:
        - Импульс формируется только один раз за период.
        - Изменение регистра сравнения мгновенно влияет на скважность следующего же импульса (отсюда название "быстрорегулируемая").
        - Недостаток: может приводить к несимметричным импульсам, что нежелательно для некоторых применений (например, аудиоусилителей).

   Симметричная (корректно-фазная) ШИМ (Phase Correct PWM)
    - Принцип: Счетчик работает "вверх-вниз": увеличивается от 0 до максимума, а затем уменьшается обратно до 0. Сигнал переключается, когда значение счетчика совпадает с регистром сравнения (и при прямом, и при обратном счете).
    - Особенности:
        - Импульс формируется дважды за период (центрирован относительно середины периода).
        - Частота сигнала в два раза ниже, чем у асимметричной ШИМ при той же тактовой частоте счетчика.
        - Преимущество: обеспечивает абсолютно симметричный сигнал, что критически важно для мостовых схем

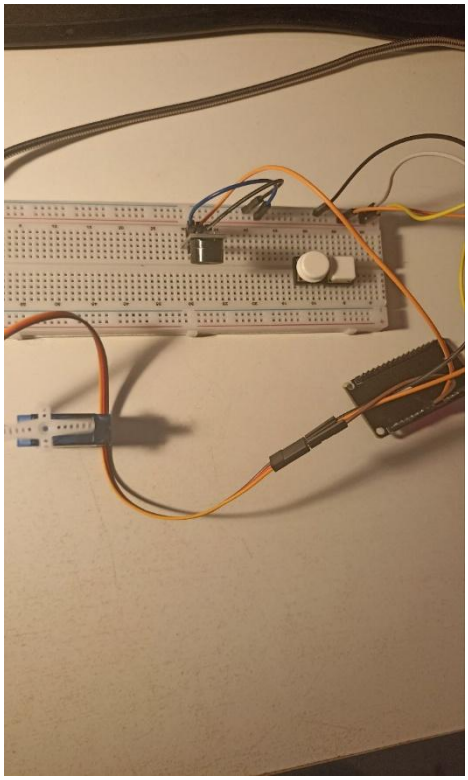
управления двигателями (меньше вибраций и акустического шума) и для аудиоприменений.

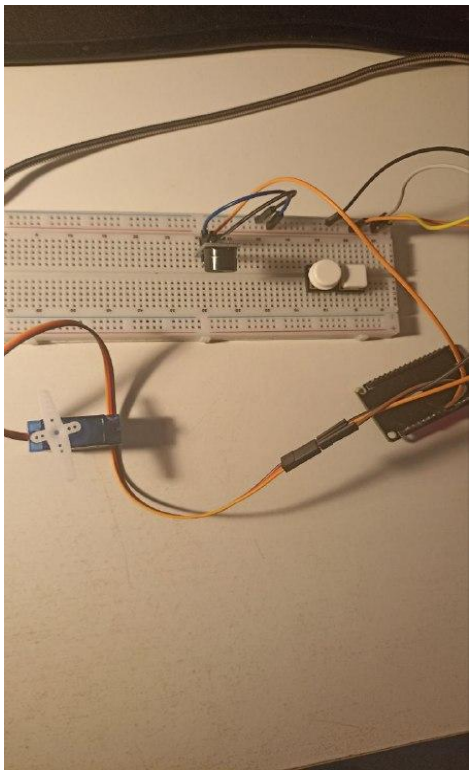
Главное различие: В симметричной ШИМ импульс центрирован, что делает сигнал более "чистым" и уменьшает паразитные гармоники, но она работает на вдвое меньшей частоте. Асимметричная ШИМ проще и позволяет работать на более высоких частотах.

2. Практическая часть:

1. Соберите схему, использующую пьезоизлучатель (звуковой модуль). Подключите модуль к контроллеру. Напишите код, управляющий звуком динамика, создавая ШИМ разной частоты и скважности.

2. Соберите схему, использующую Сервопривод SG -90, подайте питание с вывода USB - 5 вольт на питание сервопривода. Изучите документацию к сервоприводу и определите частоту импульсов для ШИМ и скважность, ее связь с углом поворота сервопривода. Напишите код, управляющий сервоприводом. Поворотом привода от 0 до 180 градусов и обратно.





Код и для пьезодинамик и для сервоприворда:

```c
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/mcpwm.h"
#include "driver/ledc.h"

#define SERVO_PIN 18
#define BUZZER_PIN 5

#define BUZZER_TIMER       LEDC_TIMER_0
#define BUZZER_MODE        LEDC_LOW_SPEED_MODE
#define BUZZER_CHANNEL     LEDC_CHANNEL_0
#define BUZZER_DUTY_RES    LEDC_TIMER_13_BIT

void buzzer_init() {
    ledc_timer_config_t ledc_timer = {
        .speed_mode       = BUZZER_MODE,
        .timer_num        = BUZZER_TIMER,
        .duty_resolution  = BUZZER_DUTY_RES,
        .freq_hz          = 1000,
        .clk_cfg          = LEDC_AUTO_CLK
    };
    ledc_timer_config(&ledc_timer);

    ledc_channel_config_t ledc_channel = {
        .speed_mode     = BUZZER_MODE,
        .channel        = BUZZER_CHANNEL,
        .timer_sel      = BUZZER_TIMER,
        .intr_type      = LEDC_INTR_DISABLE,
        .gpio_num       = BUZZER_PIN,
        .duty           = 0,
        .hpoint         = 0
    };
    ledc_channel_config(&ledc_channel);
}

void buzzer_play_tone(int frequency, int volume) {
    ledc_set_freq(BUZZER_MODE, BUZZER_TIMER, frequency);
    ledc_set_duty(BUZZER_MODE, BUZZER_CHANNEL, volume);
    ledc_update_duty(BUZZER_MODE, BUZZER_CHANNEL);
}

void buzzer_stop() {
    ledc_set_duty(BUZZER_MODE, BUZZER_CHANNEL, 0);
    ledc_update_duty(BUZZER_MODE, BUZZER_CHANNEL);
}

void servo_init() {
    mcpwm_gpio_init(MCPWM_UNIT_0, MCPWM0A, SERVO_PIN);

    mcpwm_config_t pwm_config;
    pwm_config.frequency = 50;
    pwm_config.cmpr_a = 0;
    pwm_config.counter_mode = MCPWM_UP_COUNTER;
    pwm_config.duty_mode = MCPWM_DUTY_MODE_0;

    mcpwm_init(MCPWM_UNIT_0, MCPWM_TIMER_0, &pwm_config);
}

void servo_write_angle(int angle) {
    if (angle < 0) angle = 0;
    if (angle > 180) angle = 180;

    uint32_t pulse_us = 500 + (angle * 2000 / 180);

    mcpwm_set_duty_in_us(MCPWM_UNIT_0, MCPWM_TIMER_0, MCPWM_OPR_A, pulse_us);
}

void app_main() {
    servo_init();
    buzzer_init();

    buzzer_play_tone(600, 4000);
    vTaskDelay(pdMS_TO_TICKS(200));
    buzzer_stop();
    vTaskDelay(pdMS_TO_TICKS(500));

    while (1) {
        servo_write_angle(0);
        buzzer_play_tone(400, 4000);
        vTaskDelay(pdMS_TO_TICKS(500));
        buzzer_stop();
        vTaskDelay(pdMS_TO_TICKS(500));

        servo_write_angle(90);
        buzzer_play_tone(600, 4000);
        vTaskDelay(pdMS_TO_TICKS(500));
        buzzer_stop();
        vTaskDelay(pdMS_TO_TICKS(500));

        servo_write_angle(180);
        buzzer_play_tone(800, 4000);
        vTaskDelay(pdMS_TO_TICKS(500));
        buzzer_stop();
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}
```