## Лабораторная работа 2: Работа с цифровыми входами и выходами

**Цель:**

* **Изучение** принципов работы цифровых входов и выходов ESP32.
* **Развитие** навыков подключения и программирования ESP32 для управления внешними устройствами.
* **Анализ** методов дебаунсинга кнопок и выбор оптимального решения. **Оборудование:**
* Микроконтроллер ESP32
* Плата разработки (например, ESP32 DevKitC)
* Кнопка
* Светодиод
* Резисторы (по необходимости)
* Соединительные провода
* Компьютер
* USB-кабель

## Теоретическая часть:

### Цифровые входы и выходы ESP32:

**ESP32 обладает множеством цифровых входов и выходов (GPIO)**, которые можно использовать для **разнообразных задач**:

* **Считывание информации с датчиков:**
  + Датчики температуры, влажности, света, давления и т.д.
* **Управление исполнительными устройствами:**
  + Реле, сервоприводы, светодиоды, зуммеры и т.д.
* **Взаимодействие с пользователем:**
  + Кнопки, джойстики, сенсорные панели и т.д.

**Каждый цифровой вход/выход может работать в двух режимах:**

* **Входной режим (INPUT):**
  + Считывает состояние подключенного устройства (кнопка, датчик).
  + Возвращает значение **LOW (0 В)** или **HIGH (3,3 В)**.
* **Выходной режим (OUTPUT):**
  + Управляет подключенным устройством (светодиод, реле).
  + Может устанавливать на пине напряжение **LOW (0 В)** или **HIGH (3,3 В)**.

**Важно!** Не все GPIO-пины ESP32 могут работать как входы/выходы. \* Перед использованием **проверьте документацию** на конкретную модель ESP32.

### Подключение кнопки:

* **Кнопка** - это **простой электронный компонент**, который замыкает или размыкает цепь при нажатии.
* **Подключение кнопки к ESP32:**
  + Один вывод кнопки соединяется с **землей (GND)**.
  + Другой вывод кнопки соединяется с **выбранным цифровым входом ESP32**.
  + **Варианты подключения:**
    - **С подтягивающим резистором:**
      * Резистор подтягивает вывод кнопки к **высокому уровню (3,3 В)**, когда она не нажата.
      * Это **устраняет дребезжание** кнопки и **обеспечивает четкое срабатывание**.
    - **Без подтягивающего резистора:**
      * Внутренний подтягивающий резистор ESP32 может быть использован.
      * **Не рекомендуется**, так как может быть **нестабильным**.

### Подключение светодиода:

* **Светодиод** - это **электронный компонент**, который **испускает свет** при подаче на него напряжения.
* **Подключение светодиода к ESP32:**
  + Анод светодиода соединяется с **выбранным цифровым выходом ESP32**.
  + Катод светодиода соединяется с **землей (GND)**.
  + **Важно!** Включите в цепь **токоограничивающий резистор**, чтобы защитить светодиод от перегорания.

### Дебаунсинг кнопки:

**Дебаунсинг** - это процесс **устранения дребезжания** кнопки, которое может привести к **ложным срабатываниям**.

**Существует два основных способа дебаунсинга:**

**Программный дебаунсинг:**

* Реализуется программно **с помощью задержки и проверки** состояния кнопки.
* **Преимущества:**
  + Простой в реализации
  + Не требует дополнительных компонентов
* **Недостатки:**
  + Может **увеличить время реакции** системы
  + **Загружает процессор**

**Аппаратный дебаунсинг:**

* Реализуется с помощью **аппаратных компонентов**: конденсаторов и резисторов.
* **Преимущества:**
  + **Быстрое и точное** срабатывание
  + **Не загружает процессор**
* **Недостатки:**
  + **Требует дополнительных компонентов**
  + **Сложнее в реализации**

**Варианты реализации аппаратного дебаунсинга:**

* **RC-цепочка:**
  + Состоит из резистора и конденсатора, подключенных к кнопке.
  + **Простой и эффективный** метод.
* **Диод Шоттки:**
  + Используется для **подавления коротких импульсов** дребезжания.
* **Интегральные микросхемы:**
  + **Специализированные микросхемы** для дебаунсинга кнопок.
  + **Обеспечивают наилучшую точность и надежность**.

## Задачи:

**1. Настройка среды разработки:**

* Установите Arduino IDE или Espressif IDF (в зависимости от ваших предпочтений).
* Подключите ESP32 к компьютеру с помощью USB-кабеля.
* Выберите соответствующую плату ESP32 в Arduino IDE или Espressif IDF.

**2. Подключение кнопки и светодиода:**

* Соедините кнопку с ESP32, как описано выше (с подтягивающим резистором или без него).
* Соедините светодиод с ESP32, как описано выше, с токоограничивающим резистором.

**3. Написание программы:**

* **Вариант 1: Программный дебаунсинг:**
  + Напишите программу, которая считывает состояние кнопки с задержкой, чтобы устранить дребезжание.
  + При нажатии кнопки включите светодиод, а при отпускании - выключите.
* **Вариант 2: Аппаратный дебаунсинг (с RC-цепочкой):**
  + Напишите программу, которая считывает состояние кнопки, игнорируя короткие импульсы дребезжания.
  + При нажатии кнопки включите светодиод, а при отпускании - выключите.

**4. Тестирование программы:**

* Загрузите программу на ESP32.
* Нажмите кнопку и убедитесь, что светодиод мигает **четко и надежно**.
* Попробуйте разные варианты нажатия кнопки (быстрое, медленное, многократное) и убедитесь, что программа **корректно** реагирует.

**5. Эксперименты:**

* Сравните работу программ с программным и аппаратным дебаунсингом.
* Измените значения резисторов и конденсаторов в RC-цепочке (если используется) и **наблюдайте за влиянием** на работу программы.
* **Попробуйте другие методы дебаунсинга**, если есть возможность.

**6. Отчет:**

* Опишите **схему подключения** кнопки и светодиода к ESP32.
* **Предоставьте код** программ с программным и аппаратным дебаунсингом.
* **Опишите результаты** тестирования и экспериментов.

**7. Дополнительные задачи (по желанию):**

* **Реализуйте кнопку “стоп”**, которая позволяет **прервать** мигание светодиода.
* **Используйте несколько кнопок** для управления **разными светодиодами**.

## Примеры кода для лабораторной работы:

**1. Программный дебаунсинг:**

const int buttonPin = 13; // Определение пина кнопки  
const int ledPin = 2; // Определение пина светодиода  
  
bool buttonState = false; // Переменная для хранения состояния кнопки  
unsigned long lastButtonPressTime = 0; // Переменная для хранения времени последнего нажатия кнопки  
const unsigned long debounceDelay = 50; // Задержка для дебаунсинга (в миллисекундах)  
  
void setup() {  
 pinMode(buttonPin, INPUT); // Настройка пина кнопки как входного  
 pinMode(ledPin, OUTPUT); // Настройка пина светодиода как выходного  
}  
  
void loop() {  
 bool currentButtonState = digitalRead(buttonPin); // Чтение текущего состояния кнопки  
  
 if (currentButtonState != buttonState) { // Проверка изменения состояния кнопки  
 lastButtonPressTime = millis(); // Запись времени последнего нажатия кнопки  
 buttonState = currentButtonState; // Обновление переменной состояния кнопки  
 }  
  
 if (millis() - lastButtonPressTime > debounceDelay && buttonState) { // Проверка дебаунсинга  
 digitalWrite(ledPin, HIGH); // Включение светодиода  
 } else {  
 digitalWrite(ledPin, LOW); // Выключение светодиода  
 }  
}

**2. Аппаратный дебаунсинг (с RC-цепочкой):**

const int buttonPin = 13; // Определение пина кнопки  
const int ledPin = 2; // Определение пина светодиода  
const int resistor = 10000; // Номинал резистора (в омах)  
const int capacitor = 100; // Номинал конденсатора (в микрофарадах)  
  
void setup() {  
 pinMode(buttonPin, INPUT); // Настройка пина кнопки как входного  
 pinMode(ledPin, OUTPUT); // Настройка пина светодиода как выходного  
}  
  
void loop() {  
 bool buttonState = digitalRead(buttonPin); // Чтение текущего состояния кнопки  
 digitalWrite(ledPin, buttonState); // Управление светодиодом напрямую  
}

**Объяснение кода:**

**Программный дебаунсинг:**

* **buttonPin и ledPin:** Определяют пины кнопки и светодиода.
* **buttonState:** Переменная для хранения текущего состояния кнопки.
* **lastButtonPressTime:** Переменная для хранения времени последнего нажатия кнопки.
* **debounceDelay:** Задержка для дебаунсинга (в миллисекундах).
* **setup():** Настраивает пины кнопки и светодиода как входной и выходной соответственно.
* **loop():**
  + Читает текущее состояние кнопки (currentButtonState).
  + Проверяет изменение состояния кнопки.
    - Если состояние изменилось, обновляет lastButtonPressTime и buttonState.
  + Проверяет дебаунсинг:
    - Если прошло достаточно времени после последнего нажатия кнопки (millis() - lastButtonPressTime > debounceDelay), и кнопка нажата (buttonState), включает светодиод (digitalWrite(ledPin, HIGH)).
    - В противном случае выключает светодиод (digitalWrite(ledPin, LOW)).

**Аппаратный дебаунсинг (с RC-цепочкой):**

* **resistor и capacitor:** Определяют номиналы резистора и конденсатора в RC-цепочке.
* **setup():** Настраивает пины кнопки и светодиода как входной и выходной соответственно.
* **loop():**
  + Читает текущее состояние кнопки (buttonState).
  + Управляет светодиодом напрямую (digitalWrite(ledPin, buttonState)).

## Примеры кода для доп.заданий:

**1. Кнопка “стоп”:**

const int buttonPin = 13; // Определение пина кнопки  
const int ledPin = 2; // Определение пина светодиода  
const int stopButtonPin = 12; // Определение пина кнопки "стоп"  
  
bool buttonState = false; // Переменная для хранения состояния кнопки  
unsigned long lastButtonPressTime = 0; // Переменная для хранения времени последнего нажатия кнопки  
const unsigned long debounceDelay = 50; // Задержка для дебаунсинга (в миллисекундах)  
bool ledOn = false; // Переменная для управления состоянием светодиода  
  
void setup() {  
 pinMode(buttonPin, INPUT); // Настройка пина кнопки как входного  
 pinMode(ledPin, OUTPUT); // Настройка пина светодиода как выходного  
 pinMode(stopButtonPin, INPUT); // Настройка пина кнопки "стоп" как входного  
}  
  
void loop() {  
 bool currentButtonState = digitalRead(buttonPin); // Чтение текущего состояния кнопки  
 bool stopButtonState = digitalRead(stopButtonPin); // Чтение состояния кнопки "стоп"  
  
 if (currentButtonState != buttonState) { // Проверка изменения состояния кнопки  
 lastButtonPressTime = millis(); // Запись времени последнего нажатия кнопки  
 buttonState = currentButtonState; // Обновление переменной состояния кнопки  
 }  
  
 if (millis() - lastButtonPressTime > debounceDelay && buttonState && !stopButtonState) { // Проверка дебаунсинга  
 ledOn = !ledOn; // Переключение состояния светодиода  
 digitalWrite(ledPin, ledOn); // Управление светодиодом  
 }  
}

**Объяснение:**

* **stopButtonPin:** Определяет пин кнопки “стоп”.
* **ledOn:** Переменная для управления состоянием светодиода.
* **stopButtonState:** Переменная для хранения состояния кнопки “стоп”.
* **В loop():**
  + Читается состояние кнопки “стоп” (stopButtonState).
  + Условие millis() - lastButtonPressTime > debounceDelay && buttonState && !stopButtonState
    - Проверяет дебаунсинг кнопки
    - Проверяет, нажата ли кнопка
    - Проверяет, не нажата ли кнопка “стоп”
  + Если все условия true, состояние светодиода переключается (ledOn = !ledOn) и светодиод управляется (digitalWrite(ledPin, ledOn))

**2. Несколько кнопок для разных светодиодов:**

const int button1Pin = 12; // Кнопка "вверх"  
const int button2Pin = 13; // Кнопка "вниз"  
const int button3Pin = 2; // Кнопка выбора 1  
const int button4Pin = 3; // Кнопка выбора 2  
const int led1Pin = 4; // Светодиод 1  
const int led2Pin = 5; // Светодиод 2  
const int led3Pin = 16; // Светодиод 3  
const int led4Pin = 17; // Светодиод 4  
  
int selectedLED = 1; // Выбранный светодиод (1-4)  
int brightness[4] = {0, 0, 0, 0}; // Яркость каждого светодиода (0-255)  
  
void setup() {  
 pinMode(button1Pin, INPUT);  
 pinMode(button2Pin, INPUT);  
 pinMode(button3Pin, INPUT);  
 pinMode(button4Pin, INPUT);  
 pinMode(led1Pin, OUTPUT);  
 pinMode(led2Pin, OUTPUT);  
 pinMode(led3Pin, OUTPUT);  
 pinMode(led4Pin, OUTPUT);  
  
 // Восстановление яркости из памяти (EEPROM)  
 for (int i = 0; i < 4; i++) {  
 brightness[i] = EEPROM.read(i);  
 }  
}  
  
void loop() {  
 // Чтение состояния кнопок  
 bool button1State = digitalRead(button1Pin);  
 bool button2State = digitalRead(button2Pin);  
 bool button3State = digitalRead(button3Pin);  
 bool button4State = digitalRead(button4Pin);  
  
 // Управление яркостью  
 if (button1State && selectedLED > 0) {  
 brightness[selectedLED - 1] = min(brightness[selectedLED - 1] + 10, 255);  
 }  
 if (button2State && selectedLED > 0) {  
 brightness[selectedLED - 1] = max(brightness[selectedLED - 1] - 10, 0);  
 }  
  
 // Выбор светодиода  
 if (button3State) {  
 selectedLED = (selectedLED + 1) % 4 + 1;  
 }  
 if (button4State) {  
 selectedLED = (selectedLED - 1 + 4) % 4 + 1;  
 }  
  
 // Сохранение яркости в памяти (EEPROM)  
 for (int i = 0; i < 4; i++) {  
 EEPROM.write(i, brightness[i]);  
 }  
  
 // Установка яркости светодиодов  
 analogWrite(led1Pin, brightness[0]);  
 analogWrite(led2Pin, brightness[1]);  
 analogWrite(led3Pin, brightness[2]);  
 analogWrite(led4Pin, brightness[3]);  
}

**Объяснение:**

* **brightness[4]:** Массив для хранения яркости каждого светодиода.
* **selectedLED:** Переменная для хранения номера выбранного светодиода.
* **EEPROM:** Используется для сохранения яркости в памяти even after power off.
* **setup():**
  + Загружает сохраненную яркость из EEPROM.
* **loop():**
  + Сохраняет яркость в EEPROM перед выходом из цикла.

**Схема подключения:**

## Схема подключения кнопки :

+-------------------------------------------------+  
| |  
| ESP32 |  
| |  
| GPIO 13 <--> Кнопка (один вывод) |  
| GPIO 13 <--> Резистор 10 кОм <--> GND |  
| 3.3V <--> Кнопка (другой вывод) |  
| |  
+-------------------------------------------------+

## Схема подключения светодиода :

+-------------------------------------------------+  
| |  
| ESP32 |  
| |  
| GPIO 2 <--> Светодиод (анод) |  
| GPIO 2 <--> Резистор 220 Ом <--> GND |  
| GND <--> Светодиод (катод) |  
| |  
+-------------------------------------------------+