## Лабораторная работа 6: Работа с АЦП

### Цель

* Изучить принцип работы аналого-цифрового преобразования (АЦП).
* Освоить работу с АЦП на ESP32.
* Научиться подключать аналоговые датчики к ESP32 и считывать с них данные.
* Интегрировать АЦП в проект для реализации полезных функций.

## Теоретическая часть

### Аналого-цифровое преобразование (АЦП)

АЦП - это электронное устройство, которое преобразует аналоговый сигнал (непрерывный по времени и амплитуде) в цифровой (дискретный по времени и амплитуде).

**Принцип работы:**

1. **Дискретизация:** Аналоговый сигнал измеряется через равные промежутки времени, тем самым разбивая его на отдельные точки.
2. **Квантование:** Каждая точка аналогового сигнала квантуется, то есть округляется до ближайшего значения из набора разрешенных значений.
3. **Кодирование:** Квантованные значения кодируются в двоичном формате, представляя собой цифровое представление аналогового сигнала.

**Характеристики АЦП:**

* **Разрядность:** Количество бит в выходном двоичном коде. Определяет точность преобразования.
* **Частота дискретизации:** Частота, с которой измеряется аналоговый сигнал. Определяет максимальную частоту аналогового сигнала, который можно преобразовать.
* **Диапазон входных напряжений:** Минимальное и максимальное напряжения, которые может принимать АЦП.

### Встроенный АЦП на ESP32

ESP32 имеет встроенный 12-битный АЦП с 16 каналами. Это позволяет подключать к ESP32 множество аналоговых датчиков и устройств.

**Характеристики АЦП ESP32:**

* Разрядность: 12 бит (4096 возможных значений)
* Частота дискретизации: До 133 кГц
* Диапазон входных напряжений: 0 В - 3.3 В

### Подключение аналогового датчика

**Пример:** Подключение потенциометра к ESP32.

1. Подключите один вывод потенциометра к аналоговому входу ESP32 (например, A0).
2. Подключите другой вывод потенциометра к 3.3V ESP32.
3. Подключите третий вывод потенциометра к GND ESP32.

### Написание программы для работы с АЦП

const int analogPin = A0; // Аналоговый вход, к которому подключен датчик  
  
void setup() {  
 Serial.begin(115200); // Инициализация UART  
}  
  
void loop() {  
 // Считывание значения с АЦП  
 int analogValue = analogRead(analogPin);  
  
 // Преобразование аналогового значения в цифровое  
 float voltage = analogValue \* (3.3 / 4096.0); // 3.3V - диапазон входных напряжений, 4096 - максимальное значение АЦП  
  
 // Вывод значения на Serial  
 Serial.print("Аналоговое значение: ");  
 Serial.println(analogValue);  
  
 Serial.print("Напряжение: ");  
 Serial.println(voltage, 2); // Отображение с двумя знаками после запятой  
  
 delay(100);  
}

**Описание:**

* В коде определена константа analogPin, которая указывает на аналоговый вход, к которому подключен датчик.
* В функции setup() инициализируется UART для вывода данных на Serial.
* В цикле loop() считывается значение с АЦП с помощью функции analogRead().
* Значение преобразуется в напряжение с помощью voltage = analogValue \* (3.3 / 4096.0).
* Считанные значения выводятся на Serial.

### Интеграция АЦП в проект

**Пример:** Автоматическое регулирование яркости светодиода.

**Схема подключения:**

* Светодиод подключен к GPIO пину ESP32 (например, GPIO 13).
* Аналоговый датчик (например, фоторезистор) подключен к аналоговому входу ESP32 (например, A0).

**Код:**

const int analogPin = A0; // Аналоговый вход, к которому подключен датчик  
const int ledPin = 13; // GPIO пин, к которому подключен светодиод  
  
void setup() {  
 Serial.begin(115200);  
 pinMode(ledPin, OUTPUT); // Настройка GPIO пина как выходного  
}  
  
void loop() {  
 // Считывание значения с АЦП  
 int analogValue = analogRead(analogPin);  
  
 // Преобразование аналогового значения в цифровое  
 float voltage = analogValue \* (3.3 / 4096.0);  
  
 // Определение уровня яркости светодиода based on the voltage  
 int brightness = map(voltage, 0, 3.3, 0, 255);  
  
 // Установка яркости светодиода  
 analogWrite(ledPin, brightness);  
  
 // Вывод значений на Serial  
 Serial.print("Аналоговое значение: ");  
 Serial.println(analogValue);  
  
 Serial.print("Напряжение: ");  
 Serial.println(voltage, 2);  
  
 Serial.print("Яркость: ");  
 Serial.println(brightness);  
  
 delay(100);  
}

**Описание:**

* В коде определены константы analogPin и ledPin, которые указывают на аналоговый вход и GPIO пин, к которым подключены датчик и светодиод соответственно.
* В функции setup() инициализируется UART и настраивается GPIO пин светодиода как выходной.
* В цикле loop() считывается значение с АЦП, преобразуется в напряжение и используется для определения уровня яркости светодиода с помощью функции map().
* Функция analogWrite() устанавливает яркость светодиода.
* Считанные значения и уровень яркости выводятся на Serial.

**Другие примеры использования АЦП:**

* **Термостат:** Считывание данных с датчика температуры и управление нагревательным элементом.
* **Система сигнализации:** Считывание данных с датчика движения и включение сигнала тревоги.
* **Умный дом:** Считывание данных с датчиков температуры, влажности, освещенности и управление устройствами (кондиционером, освещением).

## Задачи:

**Задания:** \* Создайте систему автоматического полива растений, которая будет использовать датчик влажности почвы для определения необходимости полива. \* Создайте метеостанцию, которая будет измерять температуру, влажность и атмосферное давление.

## Привер кода для лаборатоной работы:

### 1. Автоматический полив растений

**Описание:**

1. **Датчик влажности почвы:** Подключите датчик влажности почвы к аналоговому входу ESP32 (например, A0).
2. **Насос для полива:** Подключите насос к реле, которое управляется GPIO-пином ESP32 (например, GPIO12).
3. **Резервуар с водой:** Подключите резервуар с водой к насосу.

**Код:**

const int analogPin = A0; // Аналоговый вход, к которому подключен датчик влажности  
const int relayPin = 12; // GPIO пин, к которому подключено реле  
  
int thresholdValue = 300; // Пороговое значение влажности  
  
void setup() {  
 Serial.begin(115200);  
 pinMode(relayPin, OUTPUT); // Настройка GPIO пина как выходного  
}  
  
void loop() {  
 // Считывание значения с датчика влажности  
 int sensorValue = analogRead(analogPin);  
  
 // Преобразование аналогового значения в процентное содержание влажности  
 float humidity = sensorValue \* (100.0 / 4096.0);  
  
 // Вывод информации на Serial  
 Serial.print("Значение датчика: ");  
 Serial.println(sensorValue);  
 Serial.print("Влажность: ");  
 Serial.println(humidity, 2); // Отображение с двумя знаками после запятой  
  
 // Проверка необходимости полива  
 if (humidity < thresholdValue) {  
 // Включить полив  
 digitalWrite(relayPin, HIGH);  
 delay(5000); // Время полива (в миллисекундах)  
 digitalWrite(relayPin, LOW);  
 Serial.println("Полив включен!");  
 } else {  
 // Полив не требуется  
 Serial.println("Полив не требуется.");  
 }  
  
 delay(10000); // Интервал между измерениями (в миллисекундах)  
}

**Описание работы:**

* В коде определены константы analogPin и relayPin, которые указывают на аналоговый вход и GPIO пин, к которым подключены датчик влажности и реле соответственно.
* Также установлена переменная thresholdValue, которая хранит пороговое значение влажности, при котором необходимо включить полив.
* В функции setup() инициализируется UART и настраивается GPIO пин реле как выходной.
* В цикле loop() считывается значение с датчика влажности, преобразуется в процентное содержание влажности и выводится на Serial.
* Проверяется, опустилась ли влажность ниже порогового значения.
* Если да, то включается реле на 5 секунд (время полива можно изменить) и выводится сообщение о включении полива.
* Если влажность достаточная, то полив не требуется, и выводится соответствующее сообщение.
* Цикл повторяется с интервалом 10 секунд (интервал можно изменить).

**Вариации:**

* Можно добавить возможность ручного управления поливом с помощью кнопки или веб-интерфейса.
* Можно использовать несколько датчиков влажности для разных растений.
* Можно добавить датчик температуры и влажности воздуха для более точной оценки необходимости полива.

### 2. Метеостанция

**Схема подключения:**

Схема подключения метеостанции: [неправильный URL удален]

**Описание:**

1. **Датчик температуры и влажности:** Подключите датчик температуры и влажности к I2C-шине ESP32.
2. **Датчик атмосферного давления:** Подключите датчик атмосферного давления к аналоговому входу ESP32 (например, A0).
3. **Дисплей:** Подключите дисплей к ESP32 по SPI или I2C.

**Код:**

#include <Wire.h> // Библиотека для I2C  
#include <BME280.h> // Библиотека для датчика BME280  
  
// Адреса датчиков  
const int bme280Address = 0x76; // Адрес датчика BME28  
  
// Определение объекта датчика BME280  
BME280 bme280(bme280Address);  
  
// Определение объекта дисплея  
// ... (зависит от используемого дисплея)  
  
void setup() {  
 Serial.begin(115200); // Инициализация UART  
 bme280.begin(); // Инициализация датчика BME280  
 // ... (инициализация дисплея)  
}  
  
void loop() {  
 // Считывание данных с датчика BME280  
 float temperature = bme280.readTemperature();  
 float humidity = bme280.readHumidity();  
 float pressure = bme280.readPressure();  
  
 // Отображение данных на дисплее  
 // ... (зависит от используемого дисплея)  
  
 // Вывод информации на Serial  
 Serial.print("Температура: ");  
 Serial.print(temperature, 1); // Отображение с одним знаком после запятой  
 Serial.println(" °C");  
  
 Serial.print("Влажность: ");  
 Serial.print(humidity, 1);  
 Serial.println(" %");  
  
 Serial.print("Давление: ");  
 Serial.print(pressure, 2); // Отображение с двумя знаками после запятой  
 Serial.println(" hPa");  
  
 delay(5000); // Интервал между обновлениями (в миллисекундах)  
}

**Описание работы:**

* В коде включены библиотеки Wire и BME280 для работы с I2C и датчиком BME280 соответственно.
* Определены константы bme280Address и pressureAddress, которые указывают адреса датчиков BME280 и BMP180.
* В функции setup() инициализируется UART, датчик BME280 и дисплей.
* В цикле loop() считываются данные с датчика BME280 (температура, влажность, давление).
* Считанные данные отображаются на дисплее (формат и способ отображения зависят от используемого дисплея).
* Данные также выводятся на Serial.
* Цикл повторяется с интервалом 5 секунд (интервал можно изменить).

**Вариации:**

* Можно добавить отображение истории измерений (например, за последние 24 часа).
* Можно отображать предупреждения о резких изменениях погоды (например, падение давления).
* Можно добавить возможность подключения к GPS-модулю для определения местоположения.
* Можно добавить функцию отправки данных на сервер для хранения и анализа.

**Схема подключения**

## Подключение сервопривода :

+-------------------------------------------------+  
| |  
| ESP32 |  
| |  
| 3.3V <--> Сервопривод (VCC) |  
| GPIO X <--> Сервопривод (сигнальный провод) |  
| GND <--> Сервопривод (GND) |  
| |  
+-------------------------------------------------+