Методические указания

НТО ЦСС 2023-2024

Структура задачи

Проект предполагает разработку распределенной системы сбора и обработки аудиоинформации. Система состоит из двух микрофонов, подключенных к двум разным микроконтроллерам (МК). В качестве устройства получения звука используется электретный микрофон, однако для получения и записи звука с помощью такого микрофона необходимо устройство, усиливающее электрический сигнал, а именно - усилитель. В качестве микроконтроллеров используется платформа ESP-32. Оба МК данной системы являются компонентами беспроводной сети, сервером которой выступает одноплатный компьютер Raspberry Pi. Схематично проект представлен на рисунке 1.

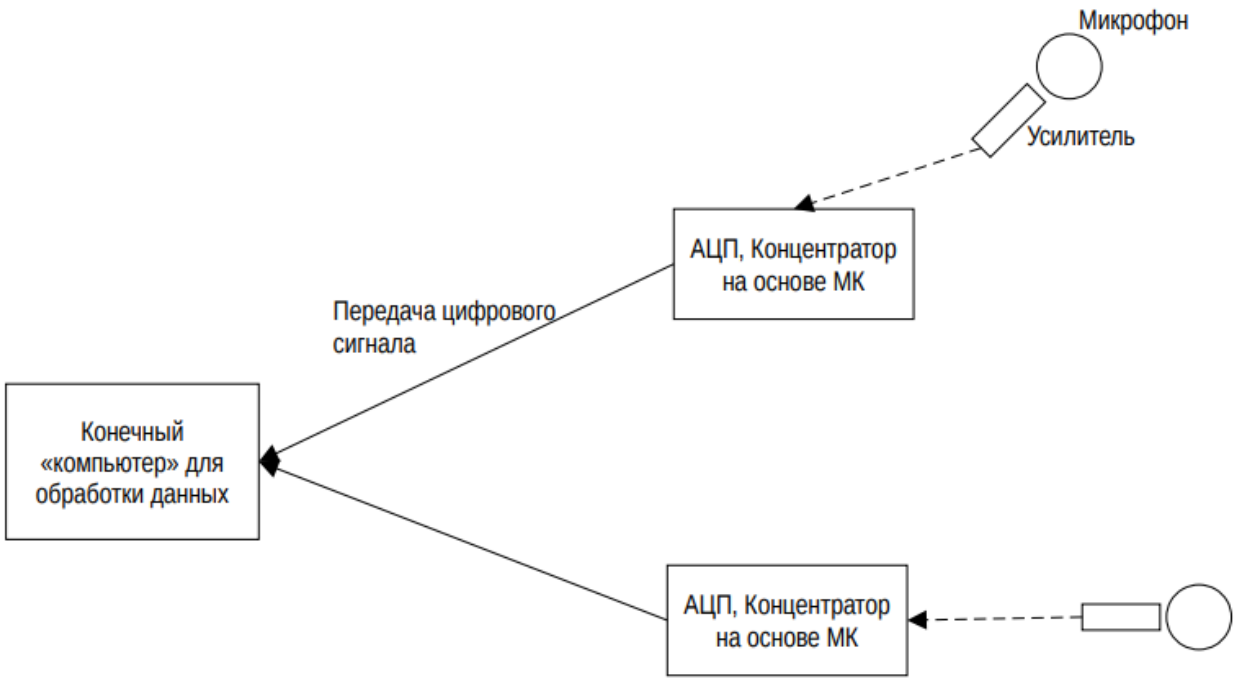

Рисунок 1 – структурная схема проекта

Общие сведения

Под термином «сигнал» мы традиционно понимаем некоторый физический процесс (например, изменяющееся во времени напряжение), отображающий некоторую информацию. Математически сигнал описывается функцией *f(t)*

определенного вида. Характеризовать виды сигналов будем с использованием следующих понятий.

Аналоговый сигнал – описывается непрерывной (или кусочно-непрерывной) функцией *f(t)*.

Дискретный сигнал – это функция дискретного аргумента $y = y(nT) = y(n) = \{y_n\}$ с областью определения $D = \{nT \mid n \in Z\}$. Значения $y(nT)$ называются отсчетами сигнала, а величина *T*, представляющая собой расстояние по оси аргумента между соседними отсчетами, называется шагом дискретизации.

Цифровым сигналом называется дискретный сигнал, который может принимать значения из конечного множества чисел, $\forall n\, y(n) \in \{d_1, \ldots, d_M\}$. Возможные значения d_j *(j=1,...,M)* называются **уровнями сигнала**.

Сопоставление функции непрерывной переменной функции дискретной переменной называется **дискретизацией**.

Так как дискретная функция имеет непрерывную ОДЗ, записать значение в память цифровой аппаратуры не представляется возможным, так как для этого потребовалось бы бесконечно много памяти. В связи с этим, требуется разбиение диапазона оценки входной функции на уровни – кванты. Такой процесс называется **квантованием** уровней. [1]

Техническая система, которая может преобразовывать входной аналоговый сигнал в соответствующий ему цифровой, называется **аналогово-цифровым преобразователем (АЦП)**.

Усилитель

В данном проекте усилитель необходим ввиду того, что электретный микрофон при воздействии звука вызывает колебания очень малого напряжения (порядка десятков мВ), которого недостаточно для того, чтобы контроллер мог детектировать и записать сигнал.

Схема усилителя выполнена на основе операционного усилителя (ОУ) LM358P [2], включенного в инвертирующем виде (Рисунок 2). Более подробно о схемах на базе ОУ в [3-4].

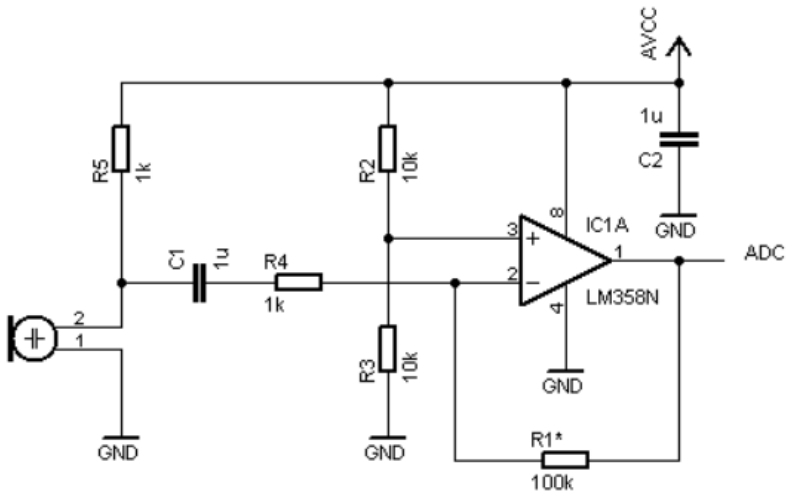

Рисунок 2 – схема инвертирующего усилителя на базе ОУ

AVCC – питание ОУ (не ниже 5 В, см. [2]). ADC – вывод на АЦП микроконтроллера. На данной схеме усиление происходит в 100 раз, согласно формуле коэффициента усиления для инвертирующего усилителя на ОУ (Рисунок 3).

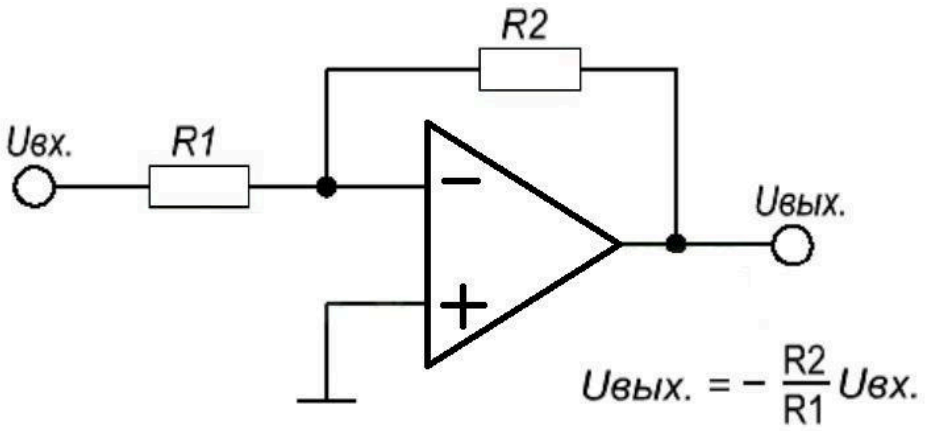

Рисунок 3 – расчет коэффициента усиления для инвертирующего усилителя на базе ОУ

Помимо этого, может понадобиться подключить к выходу усилителя (вывод ADC) фильтр нижних частот (ФНЧ), ограничивающий частоты выше частоты пропускания (Рисунок 4).

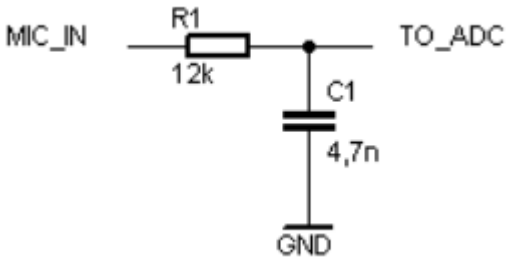

Рисунок 4 – схема ФНЧ

MIC_IN – вывод, подключенный к выходу усилителя, TO_ADC – вывод, подключаемый к АЦП микроконтроллера.

Микроконтроллер

В ESP-32 есть два многоканальных 12-битных АЦП. Вам достаточно выбрать один канал и настроить на него АЦП. Чтение значения на входе может быть выполнено функцией adc1_get_raw(). Перед вызовом этой функции, необходимо сконфигурировать АЦП. С примером конфигурации можно ознакомиться на официальном сайте espressif [5]. В среде Arduino IDE конфигурация не требуется.

В ESP-32 драйвер АЦП предоставляет девять функций:

1. analogRead(pin): получить значение АЦП для указанного вывода.
2. analogReadResolution(bits): установите разрешение вывода аналогового чтения. По умолчанию 12-битный, но возможные значения от 9 до 12.
3. analogSetWidth(bits): устанавливает биты выборки и разрешение чтения. По умолчанию 12-битный, но диапазон от 9 до 12.
4. analogSetClockDiv(clockDiv): установите делитель для частоты АЦП.
5. analogSetAttenuation(аттенюация): установите затухание для всех каналов. По умолчанию 11 дБ, но возможные значения: 0 дБ, 2,5 дБ, 6 дБ и 11 дБ.
6. analogSetPinAttenuation(pin, затухание): установите затухание для определенного вывода.
7. adcAttachPin(pin): прикрепить вывод к АЦП (выполняется автоматически в AnalogRead).

8. analogSetVRefPin(pin): установите вывод, который будет использоваться для калибровки АЦП, если ESP32 еще не откалиброван. Возможные контакты: 25, 26 или 27.
9. analogReadMilliVolts(pin): получить значение в милливольтах для вывода.

Помните, что АЦП 12 - битный, поэтому желательно обнулить старшие биты результата вызова функции (там может содержаться мусор). Для этого необходимо наложить битовую маску: res &= 0x0FFF.

Передача данных

Для организации передачи данных между ESP32 и Raspberry Pi в первую очередь необходимо установить связь между ними на физическом уровне. Для этого следует настроить точку доступа на Raspberry Pi. Один из способов - использование пакета `hostapd` [13]. Пакет `hostapd` предоставляется в стандартных репозиториях ОС raspbian.

Далее, после установления физической связи между концентратором и конечным компьютером, необходимо определить протокол, по которому будут передаваться данные.

Может показаться, что поскольку передача будет потоковая, вам больше подойдёт UDP, однако лучше воздержаться от использования этого протокола, поскольку вам придётся позаботиться о порядке пришедших пакетов и их размере.

За основу проекта лучше взять пример tcp-клиента в esp-idf. В этом случае, у вас уже будут реализована функция подключения к wifi (example_connect) [12]. Настройка данных wifi-сети выполняется через менюконфиг (вызывается командой idf.py menuconfig). В подменю Example Connection Configuration. Кроме того, в подменю Example Configuration можно указать ip-адрес и порт сервера (однако возможно это сделать из файла с исходниками, в то время как настройки wifi в исходниках будет указать сложнее).

Для разработки tcp сервера на конечном компьютере возможно использование модуля socket [14] в языке python.

Также допустимо использование готовых протоколов, таких как http.

Рекомендуется предусмотреть разбиение потоковой передачи данных на пакеты (chunk) например по 30с, для удобства обработки их модулем постобработки.

Постобработка

Получив и распаковав пакет, содержащий звуковую информацию, следующая задача – проанализировать полученные данные и вычислить амплитудный спектр полученного сигнала, вычислить среднее арифметическое значение АЦП, медианное, минимальное и максимальное по всей выборке, а также оценить шум (в дБ).

Спектр сигнала можно вычислить из полученного цифрового сигнала, применив формулу дискретного преобразование Фурье.

Французский математик Жан Батист Жозеф Фурье доказал, что любую функцию, удовлетворяющую некоторым условиям: (непрерывность во времени, периодичность, удовлетворение условиям Дирихле) можно разложить в ряд – т.е. сумму бесконечного числа слагаемых коэффициентов, упорядоченных в определённой последовательности, который в дальнейшем получил его имя — **ряд Фурье**.

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos cos \frac{n\pi t}{T} + b_n \sin sin \frac{n\pi t}{T}) \quad (1)$$

Для определения коэффициентов ряда Фурье справедливы следующие выражения:

$$a_n = \frac{1}{T} \int_{-T}^{T} f(t) \bullet \cos cos \frac{n\pi t}{T} dt \quad (2)$$

$$b_n = \frac{1}{T} \int_{-T}^{T} f(t) \bullet \sin sin \frac{n\pi t}{T} dt \quad (3)$$

Если раскладываемая функция является чётной (*f(-t) = f(t)*), то ряд Фурье состоит только из косинусов, т. е. все коэффициенты при синусах равны 0. Если раскладываемая функция является нечётной (*f(-t) ≠ f(t)*), то ряд Фурье состоит только из синусов, т. е. все коэффициенты при косинусах равны 0. В общем случае, коэффициенты при синусах и косинусах не равны 0.

Таким образом, разложение в ряд Фурье позволяет разложить непрерывную функцию в сумму других непрерывных функций. В инженерной практике разложение периодических функций в ряд Фурье широко используется в задачах теории цепей и в цифровой обработке сигналов. Однако, эволюционным продолжением этой идеи стало **преобразование Фурье**, применяемое к непрерывным функциям - операция, сопоставляющая одной функции вещественной переменной другую (вообще говоря, комплекснозначную) функцию вещественной переменной. Эта новая функция описывает коэффициенты («амплитуды») при разложении исходной функции на элементарные составляющие — гармонические колебания с разными частотами.

$$S(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \qquad (4)$$

где ω – частота гармонического колебания (в рад/с). Иначе: можно выразить через частоту $\nu = \frac{1}{T}$ [Гц] : $\omega = 2\pi \cdot \nu$

Функция $S(\omega)$ – носит название частотного спектра или спектральной плотности, или спектральной характеристики функции (сигнала) *f(t)*. Функция спектра непрерывной функции является также непрерывной.

Обратное преобразование позволяет получить из спектра сигнала обратно функцию времени:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) e^{i\omega t} d\omega \qquad (5)$$

Таким образом, при выполнении определенных условий сигнал можно описать как во временной области, т.е. через функцию *f(t)*, так и в частотной – через ее спектр $S(\omega)$.

В технике часто применяется термины «амплитудного спектра», являющегося модулем функции $S(\omega)$ и «фазового спектра», являющегося аргументом комплексозначной функции $S(\omega)$.

Помимо этого в вычислительной технике получило больше применение **дискретное преобразование Фурье**, в силу того, что обработка сигнала, зачастую, ведется уже с обработанными цифровыми данными – дискретными и квантованными величинами.

Прямое преобразование:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi i}{N} kn} = \sum_{n=0}^{N-1} x_n (\cos cos \frac{2\pi nk}{N} - i \sin sin \frac{2\pi nk}{N}) \quad (6)$$

Обратное преобразование:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} = \frac{1}{N} \sum_{n=0}^{N-1} X_k (\cos cos\, 2\frac{\pi nk}{N} + i \sin sin \frac{2\pi nk}{N}) \quad (7)$$

Переход от экспоненциального вида записи к тригонометрическому происходит с помощью формулы Эйлера:

$$e^{ix} = cos(x) + i\, sin(x) \quad (8)$$

Обозначения:

- *N* - количество отсчетов сигнала, измеренных за период *T*, а также количество компонент разложения;
- x_n, *n = 0, ... , N - 1* - измеренные значения сигнала (в дискретных временных точках с номерами *n = 0, ... , N - 1),* которые являются входными данными для прямого преобразования и выходными для обратного;
- X_k, *k = 0, ... , N - 1* - комплексных амплитуд синусоидальных сигналов, слагающих исходный сигнал; являются выходными данными для прямого преобразования и входными для обратного; поскольку амплитуды комплексные, то по ним можно вычислить одновременно и амплитуду, и фазу;
- *k* – индекс частоты. Частота *k*-го сигнала равна $\frac{k}{T}$, где *T* - период времени, в течение которого брались входные данные.

Из последнего видно, что преобразование раскладывает сигнал на синусоидальные составляющие (которые называются гармониками) с частотами от *1* колебания за период до *N - 1* колебаний за период (плюс константа). Поскольку частота дискретизации сама по себе равна *N* отсчётов за период, то высокочастотные составляющие не могут быть корректно отображены — возникает т.н. муаровый эффект. Это приводит к тому, что вторая половина из *N* комплексных амплитуд, фактически, является зеркальным отображением первой и не несёт дополнительной информации.

Таким образом, чтобы рассчитать амплитудный спектр, необходимо, воспользовавшись прямым дискретным преобразованием Фурье, взять модуль (абсолютное значение) от комплексосодержащей функции (6).

Помимо этого, подробнее о преобразовании Фурье можно прочитать в [6-10].

Вычисление шума в дБ подробно описано в [11].

Список источников

1. Умняшкин С.В. Основы теории цифровой обработки сигналов: Учебное пособие. Изд. 5-е, , испр. и доп., М.: ТЕХНОСФЕРА, 2019. - 550 с.
2. Сайт Chipdip, информационный лист на прибор LM358P URL: https://static.chipdip.ru/lib/632/DOC012632389.pdf (дата обращения 28.03.2024)
3. Хоровиц П., Хилл У. Искусство схемотехники: Пер. с англ. – Изд. 2-е. – М.: Издательство БИНОМ. – 2016. – 704 с.
4. Джонс М.Х. Электроника – практический курс. Изд. 3-е, , испр. М.: ТЕХНОСФЕРА, 2021. - 512 с.
5. Сайт Espressif, информационный лист об API-функции АЦП для СнК ESP32 URL: https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/api-reference/peripherals/adc.html (дата обращения 28.03.2024)
6. Простыми словами о преобразовании Фурье / Хабр URL: https://habr.com/ru/articles/196374/ (дата обращения 28.03.2024)
7. Спектральный анализ сигналов / Хабр URL: https://habr.com/ru/articles/253447/ (дата обращения 28.03.2024)
8. Оппенгейм А., Шафер Р. Цифровая обработка сигналов, изд. 3-е, испр. и доп. М.: ТЕХНОСФЕРА, 2019. - 1048 с.
9. Дискретное преобразование Фурье / Электронная библиотека ИТМО URL: https://de.ifmo.ru/bk_netra/page.php?tutindex=25&index=10&layer=1 (дата обращения 28.03.2024)
10. Преобразование Фурье / simenergy.ru URL:http://simenergy.ru/mathematical-analysis/digital-processing/fourier-transform (дата обращения 28.03.2024)
11. Измерение уровня звука (шума) в децибелах с помощью Arduino и микрофона // Мир контроллеров URL: https://microkontroller.ru/arduino-projects/izmerenie-urovnya-zvuka-shuma-

v-deczibelah-s-pomoshhyu-arduino-i-mikrofona/ (дата обращения 28.03.2024)

12. Сайт Espressif с документацией на TCP/IP: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/lwip.html (дата обращения 21.03.2024)

13. Настройка локальной Wi-Fi точки доступа в Linux: https://www.opennet.ru/tips/3080_wifi_ap_wpa_wlan_linux_hostapd_dnsmasq.shtml (дата обращения 23.03.2024)

14. socket — Low-level networking interface https://docs.python.org/3/library/socket.html (дата обращения 23.03.2024)