Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

09.03.01 Информатика и вычислительная техника

код и наименование направления подготовки

**ОТЧЕТ**

по преддипломной практике

по направлению 09.03.01 «Информатика и вычислительная техника»,

направленность (профиль) – «Электронно-вычислительные машины, комплексы, системы и сети», квалификация – бакалавр,

программа академического бакалавриата,

форма обучения – очная, год начала подготовки (по учебному плану) – 2015

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. ИВ-522  «25» мая 2019 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Старышкин Н.А./ |
| Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_» |  |  |
| Руководитель практики от университета  Старший преподаватель Кафедры ВС  «25» мая 2019 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Гонцова А.В./ |

Новосибирск 2019

**ПЛАН-ГРАФИК ПРОВЕДЕНИЯ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ**

Тип практики: преддипломная практика

Способ проведения практики: стационарная

Форма проведения практики: дискретно по периодам проведения практики

Тема ВКР: Разработка аудиоплеера с Wi-Fi модулем

Содержание практики

|  |  |
| --- | --- |
| Наименование видов деятельности | Дата  (начало – окончание) |
| Постановка задачи на практику, определение конкретной индивидуальной темы, формирование плана работ. Водный инструктаж по технике безопасности (охране труда, пожарной безопасности) | 11.02.18-17.02.18 |
| Работа с библиотечными фондами, сбор и анализ материалов по теме практики | 19.02.18-03.03.18 |
| Выполнение работ в соответствии с составленным планом | 05.03.18-14.04.18 |
| Анализ полученных результатов и произведенной работы, составление отчета по практике | 07.05.19-25.05.18 |

Согласовано:

|  |  |  |
| --- | --- | --- |
| Руководитель практики от университета  Старший преподаватель Кафедры ВС | \_\_\_\_\_\_\_\_\_\_\_\_ | /Гонцова А.В./ |

**ЗАДАНИЕ НА ПРЕДДИПЛОМНУЮ ПРАКТИКУ**

1. Цель – разработка аудиоплеера, который будет брать музыку через wi-fi из сети интернет.
2. Технические требования:

* Реализовать устройство на плате NodeMcu v3 на базе контроллера ESP12E.
* Устройство должно получать аудио-поток через URL, обрабатывать программными средствами, переводить в аналоговый сигнал и выводить в соответствующий порт.
* Реализовать переключение аудио-потоков посредством потенциометра.
* На дисплей должно выводиться тематическое название потока.
* Реализовать интерфейс для ввода данных к точке wi-fi.

**ВВЕДЕНИЕ**

В эпоху распространения интернета и стремительного развития цифровых технологий, радиоприемники все больше уходят в прошлое. Однако радиостанции по-прежнему остаются. Это происходит потому, что радиовещание переходит в интернет.

Все большую популярность набирают интернет-радио приемники. Такой приемник я попытался реализовать. Для этого я должен изучить контроллер ESP8266, взаимодействие с его помощью с wi-fi, получение и обработку аудио-потока.

**ОСНОВНАЯ ЧАСТЬ**

Цель данной преддипломной практики: разработать мп3-плеер с модулем wi-fi.

Для начала необходимо выбирать микроконтроллер, на базе которого будет собрано готовое стационарное устройство. Был выбран NodeMcu v3 на базе контроллера ESP12E, поскольку он имеет возможность работы с сетью интернет через wi-fi посредством AT команд, а также его можно программировать.

Контроллер будет обрабатывать аудио-поток по url онлайн радиостанции, декодировать его и отправлять в пин вывода аудио-сигнала.

Дисплей будет отображать текущую интернет радиостанцию.

Переключение интернет радиостанций будет осуществляться посредством потенциометра и кнопки для подтверждения.

Перед обработкой аудио необходимо подключиться к сети интернет средствами wi-fi модуля.

Как это реализовано:

1. Когда мы запускаем контроллер, он устанавливается в режим станции и пытается подключиться к ранее сохраненной точке wi-fi.
2. Если подключиться не удалось, модуль переходит в режим точки доступа и запускает DNS и WebServer (ip по умолчанию 192.168.4.1).
3. При подключении к данной точке wi-fi, пользователь, с помощью Captive Portal, перенаправляется на web станицу конфигураций, на которой необходимо ввести SSID и PASS точки, к которой мы подключаемся.
4. Теперь, когда у нас есть данные для подключения к точке доступа, переходим обратно в режим станции и подключаемся к точке доступа с данными из предыдущего пункта, если они неверные, переходим к пункту (2).

Обработку аудио-потока можно разделить на следующие этапы:

1. Получить аудио-поток:

Для получения аудио-потока нам нужно взаимодействовать с программным протоколом Icecast или Shoutcast. Взаимодействие клиент-сервер происходит способом, аналогичным тому, как взаимодействуют браузер и веб-сервер - по протоколу HTTP. Однако Icecast и Shoutcast имеют дополнительные заголовки. Клиент подключается к серверу и, в добавок к обычному HTTP-заголовку, отправляет ему дополнительное поле:

icy-metadata:val\r\n

Этот тэг указывает на то, что если val=1, то клиент может обрабатывать

названия песен (метаданные), передаваемые в потоке, и, таким образом,

сервер будет посылать дополнительную информацию о названии. Если val=0,то метаданные передаваться не будут.

Затем сервер отправляет ответ:

ICY 200 OK\r\n (означает, что сервер принял запрос)

icy-notice1:<BR>This stream requires <a

href="<http://www.winamp.com/>">Winamp</a><BR> (избыточное замечание)

icy-notice2:SHOUTcast Distributed Network Audio Server/posix v1.x.x<BR>

(сообщает клиенту, какой это сервер и его версию)

icy-name:Unnamed Server\r\n (имя сервера)

icy-genre:Unknown Genre\r\n (жанр сервера)

icy-url:[http://www.shoutcast.com](http://www.shoutcast.com/)\r\n (homepage сервера)

icy-pub:1\r\n (публичный или непубличный сервер)

icy-br:56\r\n (битрейт сервера)

icy-metaint:8192\r\n (см. далее)

\r\n (конец заголовка)

С этого момента сервер начинает посылать аудио-данные.

Итак, процедура со стороны клиента выглядит следующим образом:

1. Получаем и разбиваем адрес трансляции на имя хоста и порт.
2. В случае работы с Shoutcast-сервером создаем TCP-сокет и соединяем его с сервером. В случае icecast-сервера создаем два сокета: один (TCP) для получения mp3-потока, другой (UDP) для передачи пользовательских датаграмм и метаданных.
3. Отправляем в сокет сообщение вида (в случае Icecast обмен сообщениями идет через UDP-сокет):

GET / HTTP/1.0

*в случае Shoutcast-сервера или*

GET / HTTP/1.0

Host: \*\*\*\*.\*\*\*\*.\*\*\*\*.\*\*\*

x-audiocast-udpport: 6000

Icy-MetaData: 0

Accept: \*/\*

*в случае Icecast-сервера.*

1. Получаем из сокета сообщение вида:

HTTP/1.0 200 OK

Server: Icecast/VERSION

Content-Type: audio/mpeg

x-audiocast-name: Great Songs

x-audiocast-genre: Jazz

x-audiocast-url: http://icecast.serv.dom/

x-audiocast-streamid:

x-audiocast-public: 0

x-audiocast-bitrate: 24

x-audiocast-description: served by Icecast

*для Icecast-сервера или,*

ICY 200 OK

icy-notice1:<BR>This stream requires <a

href="/go.html?url=http://www.winamp.com/">Winamp</a><BR>

icy-notice2:SHOUTcast Distributed Network Audio Server/posix v1.0b<BR>

icy-name:whatever

icy-genre:whatever

icy-url:whatever

icy-pub:1

icy-br:128

*в случае, Shoutcast-сервера*

1. И далее читаем из сокета в буфер данные: в случае icecast будет только mp3-поток - <data>; в случае shoutcast mp3-поток может прерываться метаданными - <data><songtitle><data>.

Mp3-данные передаются в виде так называемых фреймов (frame, или кадр), в которых хранятся аудиоданные внутри mp3-файла.

1. Поместить аудио-поток в буфер оперативной памяти методом двойной буферизации и направить в декодер.

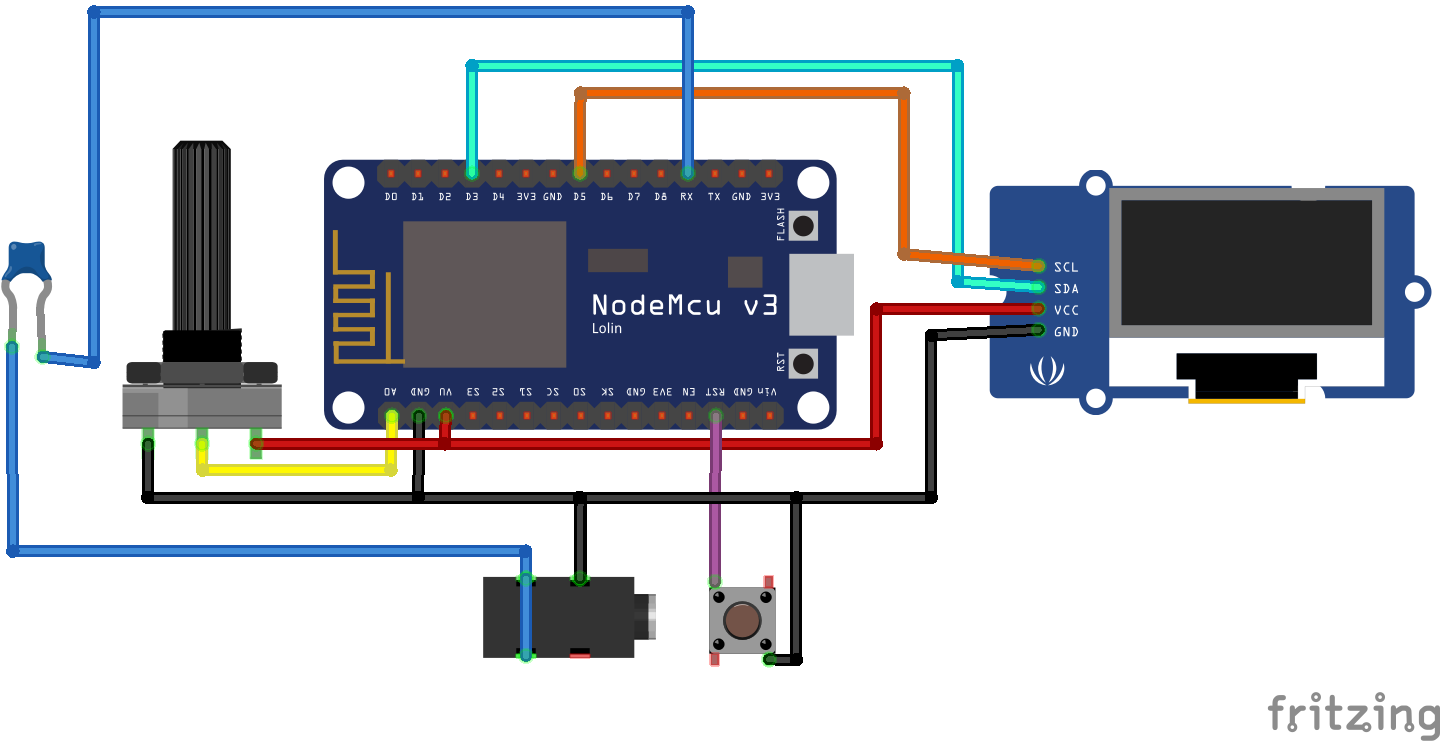
Принцип двойной буферизации:

Вместо одного буфера используются два буфера, один из которых предназначен для заполнения, а второй - для извлечения. Процесс работает, заполняя первый буфер и передавая указатель на него в задачу или процедуру извлечения. Этот заполненный буфер затем просто используется программным обеспечением для извлечения данных. Пока это происходит, второй буфер заполняется так, что когда первый буфер очищается, второй буфер будет заполнен следующим набором данных.

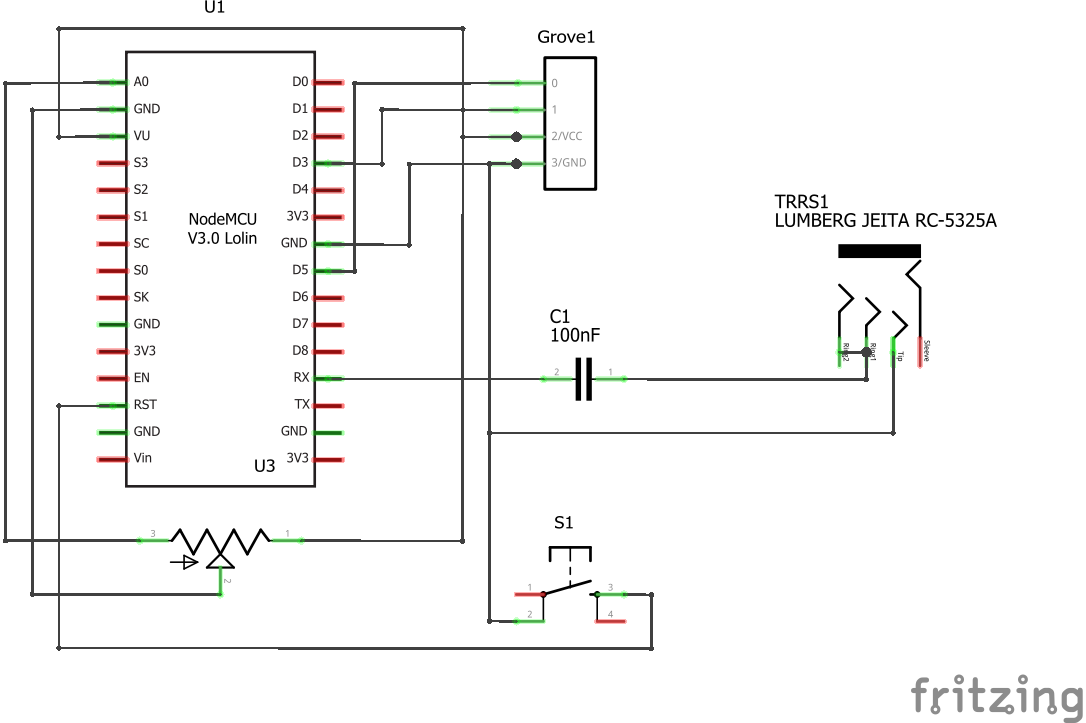
1. Декодировать аудио-поток с помощью программного или аппаратного ЦАП.

В своей реализации я выбрал программный ЦАП. Он преобразует цифровой аудио-поток в аналоговый сигнал, который можно вывести на аудио-гнездо.

Схема устройства:



Интегральная схема:



**ОПИСАНИЕ ОБОРУДОВАНИЯ**

* 1. Контроллер NodeMcu v3 Lua

NodeMcu – платформа на основе ESP8266 для создания различных устройств интернета вещей (IoT). Модуль умеет отправлять и получать информацию в локальную сеть либо в интернет при помощи Wi-Fi.

Технические характеристики модуля:

* Поддерживает Wi-Fi протокол 802.11 b/g/n;
* Поддерживаемые режимы Wi-Fi – точка доступа, клиент;
* Входное напряжение 3,7В – 20 В;
* Рабочее напряжение 3В-3,6В;
* Максимальный ток 220мА;
* Встроенный стек TCP/IP;
* Диапазон рабочих температур от -40С до 125С;
* 80 МГц, 32-битный процессор;
* Время пробуждения и отправки пакетов 22мс;
* Встроенные TR переключатель и PLL;
* Наличие усилителей мощности, регуляторов, систем управления питанием.

Питание модуля NodeMcu

Подавать питание на модуль можно несколькими способами:

* Подавать 5-18 В через контакт Vin;
* 5В через USB-разъем или контакт VUSB;
* 3,3В через вывод 3V.

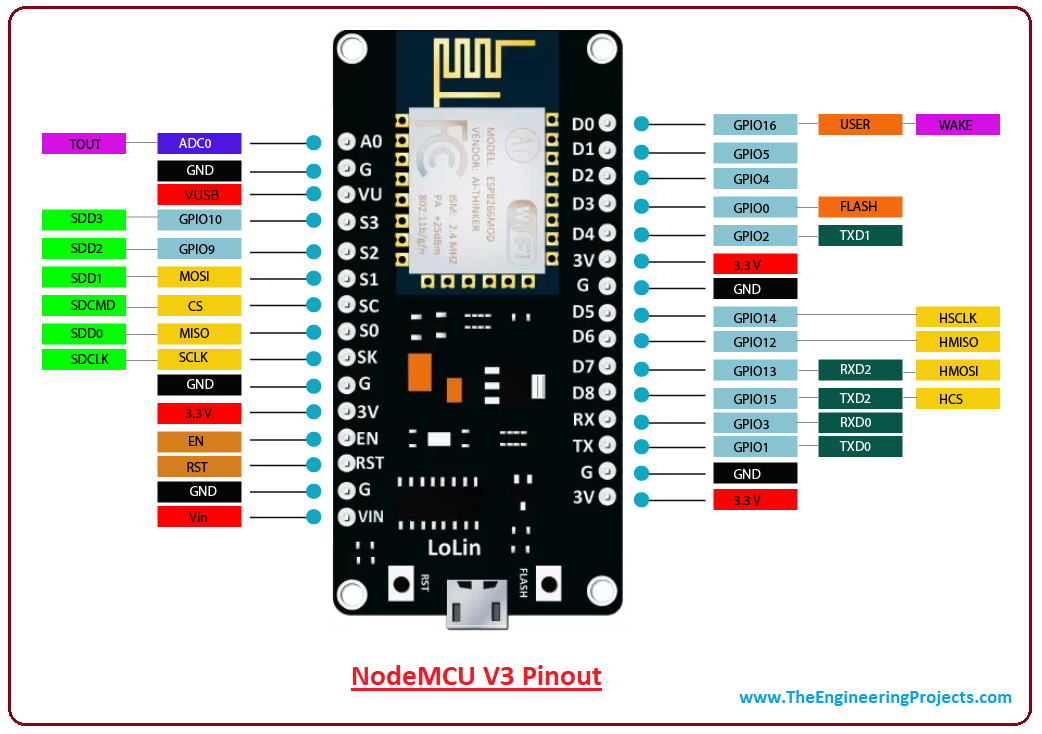
Преимущества NodeMcu v3

* Наличие интерфейса UART-USB с разъемом micro USB позволяет легко подключить плату к компьютеру.
* Наличие флэш-памяти на 4 Мбайт.
* Возможность обновлять прошивку через USB.
* Возможность создавать скрипты на LUA и сохранять их в файловой системе.

## Распиновка NodeMcu v3

Модуль V3 имеет 11 контактов ввода-вывода общего назначения. Помимо этого некоторые из выводов обладают дополнительными функциями:

* D1-D10 – выводы с широтно-импульсной модуляцией;
* D1, D2– выводы для интерфейса I²C/TWI;
* D5–D8 – выводы для интерфейса SPI;
* D9, D10 – UART;
* A0 – вход с АЦП.



**Описание модуля ESP12E**

WiFi модуль ESP-12E разработан компанией Ai-thinker и построен на базе процессора с ядром ESP8266, отличительной особенностью которого является наличие радиоинтерфейса WiFi. Ядро ESP8266 интегрировано в Tensilica L106 – 32-битный микроконтроллер с ультранизким энергопотреблением. Поддержка тактовых частот 80 и 160 МГц, поддержка RTOS, встроенные Wi-Fi MAC/BB/RF/PA/LNA, микрополосковая антенна на плате модуля.

Модуль поддерживает стандарт IEEE802.11 b/g/n, полный стек TCP/IP протоколов. Пользователи могут использовать модули либо в качестве дополнения для подключения какого-либо устройства к сети, либо в качестве отдельного сетевого контроллера.

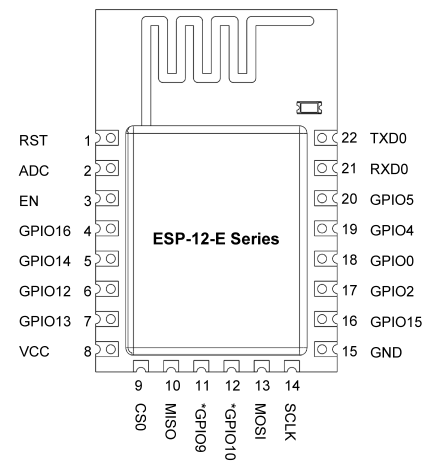
Характеристики (кратко):

* 802.11 b/g/n;
* встроенный 32-битный MCU с низким энергопотреблением;
* встроенный 10-битный АЦП;
* встроенный стек протоколов TCP/IP;
* встроенный РЧ коммутатор, РЧ трансформатор сопротивлений, LNA, усилитель мощности;
* встроенные блоки ФАПЧ и управления мощностью;
* Wi-Fi 2,4 ГГц, поддержка WPA/WPA2;
* SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO;
* STBC, 1x1 MIMO, 2x1 MIMO;
* потребление в режиме глубокого сна < 10 мкА, ток утечки в выключенном режиме < 5 мкА;
* просыпание и передача пакетов через < 2 мс;
* выходная мощность +20 дБм в режиме 802.11b;
* диапазон рабочик температур: -40 – 125 ºC.

## Технические характеристики

| **Категории** | **Параметры** | **Значения** | |
| --- | --- | --- | --- |
| WiFi параметры | WiFi протоколы | 802.11 b/g/n | |
| Частотный диапазон | 2,4ГГц–2,5ГГц (2400М–2483,5М) | |
| Характеристики аппаратной части | Периферийные шины | UART/HSPI/I2C/I2S/инфракрасный интерфейс удаленного управления | |
| Рабочее напряжение | 3,0 – 3,6 В | |
| Рабочий ток | около 80 мА | |
| Диапазон рабочих температур | -40 – 125 ºC | |
| Размеры | 16мм x 24мм x 3мм | |
| Характеристики программного обеспечения | WiFi режим | станция, программная точка доступа, программная точка доступа + станция | |
| Безопасность | WPA/WPA2 | |
| Шифрование | WEP/TKIP/AES | |
| Обновление прошивки | загрузка через UART / OTA (через сеть) / загрузка и запись прошивки через хост | |
| Разработка ПО | поддержка Cloud Server Development / SDK для разработки пользовательских прошивок | |
| Сетевые протоколы | IPv4, TCP/UDP/HTTP/FTP | |
| Пользовательская настройка | Набор AT команд, Cloud Server, приложение Android/iOS |

Назначение выводов



| **Номер вывода** | **Название вывода** | **Назначение** |
| --- | --- | --- |
| 1 | RST | Сброс модуля |
| 2 | ADC | Вход аналого-цифрового преобразователя. Входное напряжение в диапазоне 0–1 вольт; результат 0–1024. |
| 3 | EN | Вывод включения чипа. Активный уровень – высокий. |
| 4 | IO16 | GPIO16; может использоваться для пробудить чипсет из режима глубокого сна. |
| 5 | IO14 | GPIO14; HSPI\_CLK |
| 6 | IO12 | GPIO12; HSPI\_MISO |
| 7 | IO13 | GPIO13; HSPI\_MOSI; UART0\_CTS |
| 8 | VCC | Вход питания 3,3В. |
| 9 | CS0 | Выбор чипа |
| 10 | MISO | Вход ведущего, выход ведомого. |
| 11 | IO9 | GPIO9 |
| 12 | IO10 | GPIO10 |
| 13 | MOSI | Выход ведущего, вход ведомого. |
| 14 | SCLK | Последовательный тактовый сигнал. |
| 15 | GND | Корпус. |
| 16 | IO15 | GPIO15; MTDO; HSPICS; UART0\_RTS |
| 17 | IO2 | GPIO2; UART1\_TXD |
| 18 | IO0 | GPIO0 |
| 19 | IO4 | GPIO4 |
| 20 | IO5 | GPIO5 |
| 21 | RXD | UART0\_RXD; GPIO3 |
| 22 | TXD | UART0\_TXD; GPIO1 |

### Описание функционала

#### MCU

ESP8266EX встроен в 32-битный микроконтроллер (MCU) Tensilica L106, который отличается ультранизким энергопотреблением 16-битной RISC архитектурой. Тактовая частота CPU составляет 80 МГц. Также она может достичь максимального значения 160 МГц.

ESP8266EX часто интегрируется с внешними датчиками и другими специальными устройствами через его выводы GPIO; код для подобных приложений поставляется в примерах в SDK.

#### Организация памяти

##### Внутренние SRAM и ROM.

Однокристальная WiFi система ESP8266EX встраивается вместе с контроллером памяти; включая SRAM и ROM. MCU может обращаться к памяти через интерфейсы iBus, dBus и AHB.

Согласно текущей версии SDK, пользователю доступно следующее пространство памяти:

* размер RAM < 36 Кбайт, то есть, когда ESP8266EX работает в режиме клиентской станции и подключен к роутеру, программируемое пространство, доступное пользователю в куче и секции data, составляет около 36 Кбайт;
* в однокристальной системе нет программируемой памяти ROM; пользовательская программа должна храниться во внешней SPI флеш-памяти.

##### Внешняя SPI флеш-память

В модуль для хранения пользовательских программ установлена внешняя SPI флеш-память размером 4 Мбайта. Если необходимо больший объем, то необходимо установить флеш-память большего размера. Теоретически, поддерживается емкость до 16 Мбайт.

Предполагаемая емкость SPI флеш-памяти:

* OTA выключено: минимальный размер флеш-памяти, который будет поддерживаться, составляет 512 Кбайт;
* OTA включено: минимальный размер флеш-памяти, который будет поддерживаться, составляет 1 Мбайт;

Поддерживается несколько SPI режимов, в том числе, Standard SPI, Dual SPI и Quad SPI.

Перед загрузкой во флеш-память выбирайте правильный SPI режим, иначе прошивки/программы, которые вы загрузили, могут работать неправильно.

* 1. **Описание OLED дисплея - 128x64 (0,96") IIC I2C**

Графический OLED-дисплей имеет черную поверхность и белый цвет графики, хорошее качество изображения, широкие углы обзора. Отсутствие подсветки делает его тоньше и легче.  
  
OLED дисплей с размером экрана 0,96″ и разрешением 128\*64 точек для подключения к платам семейства Arduino. Подключение:  I2C.

В OLED (Organic Light-Emitting Diode) дисплеях используется технология в которой светодиоды сами излучают свет без дополнительного подсвечивания как например в LCD дисплеях. Дисплей OLED состоит из тонкой многослойной органической пленки, помещенной между анодом и катодом. OLED обладает высоким потенциалом применения практически для всех типов дисплеев и рассматривается в качестве конечной технологии для следующего поколения плоских дисплеев. Качество отображения информации, дешевизна и идеальные углы обзора OLED дисплея делают его лидером среди дисплеев.

Особенности OLED L2C дисплея:

Не требуется подсветка дисплея.

Высокое разрешение: 128 \* 64 пикселей.

Угол обзора: больше 160 градусов.

Полностью совместим с Arduino, контроллерами 51 серии, MSP430 серии, STM32 / 2, КСО IC и т.д.

Ультра-низкое энергопотребление: при полном свечении экрана 0.08W.

Рабочее напряжение: 3V ~ 5 В, постоянного тока.

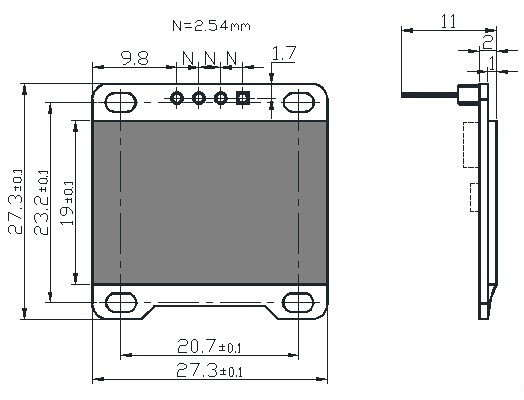
Рабочая температура: -30°C ~ +70°C.

Драйвер IC: SSD1306.

Размер платы: 27.8мм х 27.3мм х 4.3мм.

Размер дисплея: 27мм х 19.5мм (0,96″ дюйма).

Интерфейс I2C IIC



* 1. Описание потенциометра

Потенциометр- регулируемый делитель электрического напряжения.

Потенциометр чаще всего используется в двух целях:

* В качестве элементов управления
* В качестве подстроечных элементов

Потенциометры как правило поворачиваются на один оборот, но бывают и многооборотистые, а так же и линейные.

Подключив потенциометр к плате, появляется возможность отслеживать текущее состояние потенциометра и производить какие либо действия в зависимости от текущего положения.

В нашем случаем мы подключаем потенциометр к пину А0.

| **Arduino** | **Потенциометр** | **Цвет** |
| --- | --- | --- |
| GND | 1 ножка (левая) | Черный |
| A0 | 2 ножка (центральная) | Желтый |
| 5V | 3 ножка (Правая) | Красный |

Максимальное сопротивление 100 кОм.

**ЗАКЛЮЧЕНИЕ**

В результате проделанной работы был создан учебный прототип мп3-плеера с возможностью слушать интернет радиостанции через модуль wi-fi. Изучил взаимодействие контроллера ESP12E с wi-fi и с непрерывными аудио-файлами.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. ESP12E Wi-fi module Datasheet. URL: <https://www.elecrow.com/download/ESP-12F.pdf> (дата обращения: 15.05.19)
2. Introduction to NodeMCU V3. URL: <https://www.theengineeringprojects.com/2018/10/introduction-to-nodemcu-v3.html> (дата обращения: 15.05.19)
3. Протоколы сетевых радиотрансляций Icecast/Shoutcast. URL: <https://www.opennet.ru/base/sys/radio_protocols.txt.html> (дата обращения: 15.05.19)
4. Double buffering. URL: <https://www.brainkart.com/article/Double-buffering_7738/> (дата обращения: 15.05.19)

**ПРИЛОЖЕНИЯ**

Листинг программы:

Wifi-player.ino

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include "ESP8266WebServer.h"

#include "DNSServer.h"

#include "WiFiManager.h" // https://github.com/tzapu/WiFiManager

#include "AudioFileSourceICYStream.h"

#include "AudioFileSourceBuffer.h"

#include "AudioGeneratorMP3.h"

#include "AudioOutputI2SNoDAC.h"

// For a connection via I2C using Wire include

#include <Wire.h> // Only needed for Arduino 1.6.5 and earlier

#include "SSD1306Wire.h" // legacy include: `#include "SSD1306.h"`

// Initialize the OLED display using Wire library

//GPIO0 = D3

//GPIO14 = D5

SSD1306Wire display(0x3c, 0, 14);

// To run, set your ESP8266 build to 160MHz, update the SSID info, and upload.

int sensorPin = A0; // The port to which the potentiometer is connected

int sensorValue = 0; // Variable to store readings

AudioGeneratorMP3 \*mp3;

AudioFileSourceICYStream \*file;

AudioFileSourceBuffer \*buff;

AudioOutputI2SNoDAC \*out;

// Called when a metadata event occurs (i.e. an ID3 tag, an ICY block, etc.

void MDCallback(void \*cbData, const char \*type, bool isUnicode, const char \*string)

{

const char \*ptr = reinterpret\_cast<const char \*>(cbData);

(void) isUnicode; // Punt this ball for now

// Note that the type and string may be in PROGMEM, so copy them to RAM for printf

char s1[32], s2[64];

strncpy\_P(s1, type, sizeof(s1));

s1[sizeof(s1)-1]=0;

strncpy\_P(s2, string, sizeof(s2));

s2[sizeof(s2)-1]=0;

Serial.printf("METADATA(%s) '%s' = '%s'\n", ptr, s1, s2);

Serial.flush();

}

// Called when there's a warning or error (like a buffer underflow or decode hiccup)

void StatusCallback(void \*cbData, int code, const char \*string)

{

const char \*ptr = reinterpret\_cast<const char \*>(cbData);

// Note that the string may be in PROGMEM, so copy it to RAM for printf

char s1[64];

strncpy\_P(s1, string, sizeof(s1));

s1[sizeof(s1)-1]=0;

Serial.printf("STATUS(%s) '%d' = '%s'\n", ptr, code, s1);

Serial.flush();

}

//int counter = 1;

/\*void drawProgressBarDemo() {

int progress = (counter / 5) % 100;

// draw the progress bar

display.drawProgressBar(0, 32, 120, 10, progress);

// draw the percentage as String

display.setTextAlignment(TEXT\_ALIGN\_CENTER);

display.drawString(64, 15, String(progress) + "%");

}\*/

void setup()

{

system\_update\_cpu\_freq(SYS\_CPU\_160MHZ);

Serial.begin(115200);

delay(1000);

WiFiManager wifiManager;

wifiManager.autoConnect("radio");

Serial.println("Connected");

// Initialising the UI will init the display too.

display.init();

display.flipScreenVertically();

display.setFont(ArialMT\_Plain\_10);

String URL;

sensorValue = analogRead(sensorPin);

if (sensorValue < 250){

URL = "http://jazz.streamr.ru/jazz-64.mp3";

} else if (sensorValue < 500)

URL = "http://kpradio.hostingradio.ru:8000/russia.radiokp128.mp3";

else if (sensorValue < 750)

URL = "http://retroserver.streamr.ru:8043/retro64";

else

URL = "http://online.radiorecord.ru:8102/chil\_64";

file = new AudioFileSourceICYStream(URL.c\_str());

file->RegisterMetadataCB(MDCallback, (void\*)"ICY");

buff = new AudioFileSourceBuffer(file, 4096);

buff->RegisterStatusCB(StatusCallback, (void\*)"buffer");

out = new AudioOutputI2SNoDAC();

mp3 = new AudioGeneratorMP3();

mp3->RegisterStatusCB(StatusCallback, (void\*)"mp3");

mp3->begin(buff, out);

}

void loop()

{

//static int lastms = 0;

if (mp3->isRunning()) {

/\*if (millis()-lastms > 1000) {

lastms = millis();

Serial.printf("Running for %d ms...\n", lastms);

Serial.flush();

}\*/

if (!mp3->loop()) mp3->stop();

} else {

//Serial.printf("MP3 done\n");

delay(1000);

}

display.clear();

display.setTextAlignment(TEXT\_ALIGN\_LEFT);

display.setFont(ArialMT\_Plain\_24);

sensorValue = analogRead(sensorPin);

if (sensorValue < 250){

display.drawString(0, 20, "1-fm jazz");

} else if (sensorValue < 500){

display.drawString(0, 20, "2-fm news");

}else if (sensorValue < 750){

display.drawString(0, 20, "3-fm retro");

}else{

display.drawString(0, 20, "4-fm chillout");

}

// write the buffer to the display

display.display();

}