Лабораторная работа № 3.1

Создание системы кодирования и декодирования азбуки Морзе

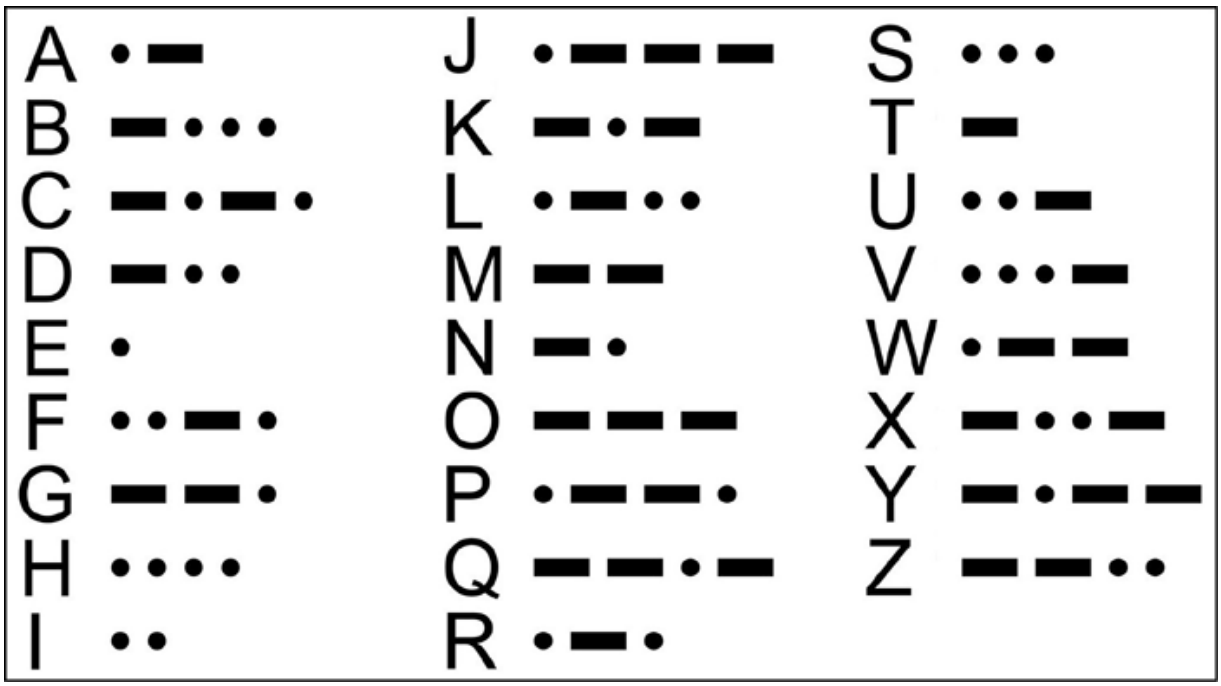
Теоретическая часть:

Азбука Морзе (код Морзе) – это система связи для кодирования любого символа в двух различных длительностях сигналов, называемых точками и тире. Азбука Морзе разработана Сэмюэлем Ф. Б. И в дальнейшем стала использоваться в телеграфии для передачи секретной информации. Больше всего она использовалась во время Второй мировой войны. Код Морзе можно формировать с помощью постукивания, мигающего света или письма. Код Морзе доступен в двух вариантах: оригинальный и международный код. В международной версии азбуки Морзе оригинальная версия модифицируется путем удаления пробелов и создания тире определенной длины.

Код Морзе может быть использован для кодирования алфавитов и цифр. Он, в основном, используется в радиосвязи и океанской связи, а также является частью подготовки солдат. Его очевидным достоинством по сравнению со всеми другими современными методами передачи информации является то, что прием информации при его использовании можно осуществлять при очень слабом уровне сигнала, ниже уровня шумов – в этих условиях не может работать ни одна современная система связи (широкополосные сигналы не в счет, приводить в сравнение их здесь некорректно).

Язык всегда был барьером для азбуки Морзе, так как трудно выполнить код для диакритических символов на другом языке. Есть несколько известных слов, которые считаются важной особенностью азбуки Морзе, например, "SOS". SOS (расшифровывается как Save Our Souls) – означает спасение наших душ. Он был создан как универсальный сигнал бедствия.

На следующем рисунке показана азбука Морзе для английского алфавита от A до Z.



В этой статье мы рассмотрим создание генератора кода Морзе на основе платы Arduino Uno, который будет принимать символы из окна монитора последовательной связи и конвертировать их в код Морзе с помощью зуммера. Статья переведена с иностранного сайта, поэтому представленный генератор кода Морзе работает только с английским алфавитом (впрочем, как и все другие аналогичные статьи на эту тему, которые мне удалось найти в рунете). Но, поняв принцип этого проекта, вы легко можете переписать его программу для работы с русским алфавитом – если вы сделаете это, будем вам признательны если поделитесь кодом этой программы с другими посетителями нашего сайта (могу включить код вашей программы в текст данной статьи).

Django Framework

Django - это фреймворк для создания веб-приложений с помощью языка программирования Python.

Django был создан в 2005 году, когда веб-разработчики из газеты Lawrence Journal-World стали использовать Python в качестве языка для создания веб-сайтов. А в 2008 году вышел публичный первый релиз фреймворка. На сегодняшний день он продолжает развиваться. Так, текущей версией фреймворка на момент написания этой статьи является версия 4.2, которая вышла в апреле 2023 года. Каждый новый релиз фреймворка выходит в средним каждые 8 месяцев. Кроме того, постоянно выходят обновления и исправления в безопасности.

Django довольно популярен. Он используется на многих сайтах, в том числе таких, как Pinterest, PBS, Instagram, BitBucket, Washington Times, Mozilla и многих других.

Фреймворк является бесплатным. Он развивается как open source, его исходный код открыт, его можно найти репозитории на **githube**.

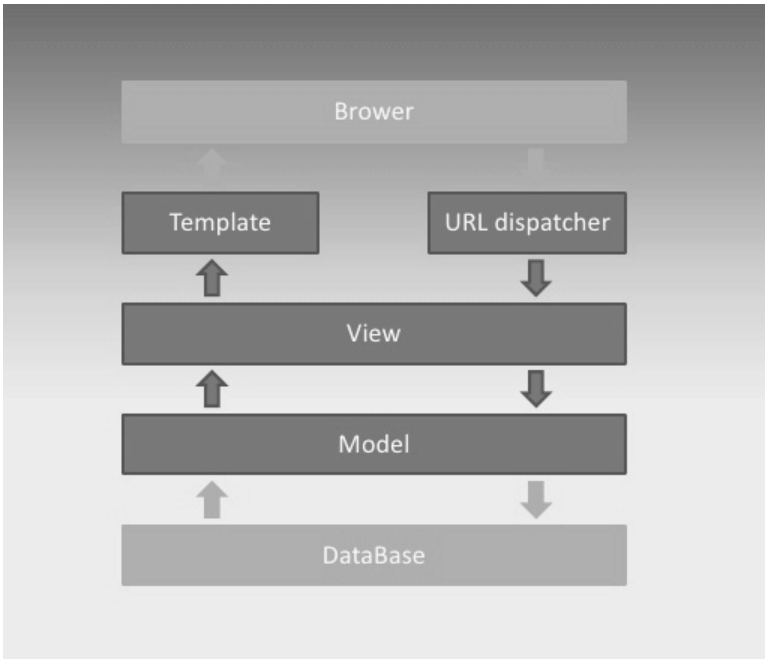
На Django можно создавать широкий диапазон веб-приложений: от небольших персональных сайтов до высоконагруженных сложных веб-сервисов.

Django по умолчанию предлагает готовую функциональность для ряда распространенных задач, например, систему аутентификации, генерацию карт сайта и т.д., благодаря чему нам можно не изобретать велосипед и достаточно взять уже готовые компоненты.

В Django большое внимание уделяется безопасности, благодаря чему фреймворк помогает разработчикам избежать многих распространенных проблем в системе безопасности, например, sql-инъекций.

Фреймворк Django реализует архитектурный паттерн Model-View-Template или сокращенно MVT, который по факту является модификацией распростаненного в веб-программировании паттерна MVC (Model-View-Controller).

Схематично мы можем представить архитектуру MVT в Django следующим образом:



Основные элементы паттерна:

- URL dispatcher: при получении запроса на основании запрошенного адреса URL определяет, какой ресурс должен обрабатывать данный запрос.
- View: получает запрос, обрабатывает его и отправляет в ответ пользователю некоторый ответ. Если для обработки запроса необходимо обращение к модели и базе данных, то View взаимодействует с ними. Для создания ответа может применять Template или шаблоны. В архитектуре MVC этому компоненту соответствуют контроллеры (но не представления).
- Model: описывает данные, используемые в приложении. Отдельные классы, как правило, соответствуют таблицам в базе данных.
- Template: представляет логику представления в виде сгенерированной разметки html. В MVC этому компоненту соответствует View, то есть представления.

Когда к приложению приходит запрос, то URL dispatcher определяет, с каким ресурсом сопоставляется данный запрос и передает этот запрос выбранному ресурсу. Ресурс фактически представляет функцию или View, который получает запрос и определенным образом обрабатывает его. В процессе обработки View может обращаться к моделям и базе данных, получать из нее данные, или, наоборот, сохранять в нее данные. Результат обработки запроса отправляется обратно, и этот результат пользователь видит в своем браузере. Как правило, результат обработки запроса представляет сгенерированный html-код, для генерации которого применяются шаблоны (Template).

Пакетный менеджер pip

Пакеты Django размещаются в центральном репозитории для большинства пакетов Python - Package Index (PyPI). И для установки из этого репозитория нам потребуется пакетный менеджер pip. Менеджер pip позволяет загружать пакеты и управлять ими. Обычно при установке python также устанавливается и менеджер pip. В этом случае мы можем проверить версию менеджера, выполнив в командной строке/терминале команду pip -V (V - с заглавной буквы):

```
C:\Users\eugen>pip -V
```

```
pip 22.0.4 from
C:\Users\eugen\AppData\Local\Programs\Python\Python310\lib\site-
packages\pip (python 3.10)

C:\Users\eugen>
```

Если pip не установлен, то мы увидим ошибку типа:

```
"pip" не является внутренней или внешней командой, исполняемой программой
или пакетным файлом
```

В этом случае нам надо установить pip. Для этого можно выполнить в командной строке/консоли следующую команду:

```
python -m ensurepip --upgrade
```

Если pip ранее уже был установлен, то можно его обновить с помощью команды

```
pip install --upgrade pip
```

Установка виртуальной среды

Виртуальная среда или venv не является неотъемлимой частью разработки на Django. Однако ее рекомендуется использовать, так как она позволяет создать множество виртуальных сред Python на одной операционной системе. Благодаря виртуальной среде приложение может запускаться независимо от других приложений на Python.

В принципе можно запускать приложения на Django и без виртуальной среды. В этом случае все пакеты Django устанавливаются глобально. Однако что если после создания первого приложения выйдет новая версия Django? Если мы захотим использовать для второго проекта новую версию Django, то из-за глобальной установки пакетов придется обновлять первый проект, который использует старую версию. Это потребует некоторой дополнительной работы по обновлению, так как не всегда соблюдается обратная совместимость между пакетами. Если мы решим использовать для второго проекта старую версию, то мы лишиемся потенциальных преимуществ новой версии. И использование

виртуальной среды как раз позволяет разграничить пакеты для каждого проекта.

Для работы с виртуальной средой в python применяется встроенный модуль venv

Итак, создадим вируальную среду. Вначале определим каталог для проектов django. Например, пусть это будет каталог C:\django. Прежде всего перейдем в терминале/командной строке в этот каталог с помощью команды cd.

```
cd C:\django
```

Затем для создания виртуальной среды выполним следующую команду:

```
python -m venv venv
```

Модулю venv передается название среды, которая в данном случае будет называться ".venv". Для наименования виртуальных сред нет каких-то определенных условностей. Пример консольного вывода:

```
C:\Users\eugen>cd C:\djangoПереход к папке будущей виртуальной среды

C:\django>python -m venv .venv Создание виртуальной среды

C:\django>
```

После этого в текущей папке (C:\django) будет создан подкаталог ".venv".

Активация виртуальной среды

Для использования виртуальную среду надо активировать. И каждый раз, когда мы будем работать с проектом Django, связанную с ним виртуальную среду надо активировать. Например, активируем выше созданную среду, которая располагается в текущем каталоге в папке .venv. Процесс активации немного отличается в зависимости от операционной системы и от того, какие инструменты применяются. Так, в Windows можно использовать командную строку и PowerShell, но между ними есть отличия.

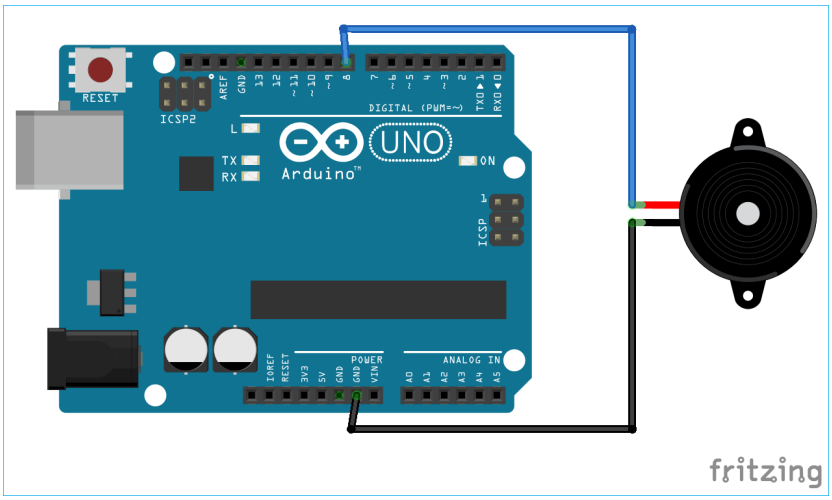
Активация в Windows в коммандной строке

Если наша ОС - Windows, то в папке .venv/Scripts/ мы можем найти файл activate.bat), который активирует виртуальную среду. Так, в Windows активация виртуальной среды в коммандной строке будет выглядеть таким образом:

```
.venv\Scripts\activate.bat
```

Практическая часть:

Схема будущего проекта:



Соответствующий код для подключения и отображения данных

```
char stringToMorseCode[] = "";
int audio8 = 8;       // к контакту 8 подключен зуммер
int note = 1200;       // music note/pitch (используемый тон)
int dotLen = 100;      // length of the morse code 'dot'
(длительность звучания точки)
int dashLen = dotLen * 3;    // length of the morse code 'dash'
(длительность звучания тире)
void setup() {
  Serial.begin(9600);
}
void loop()
{
  char inChar = 0;
  char inData[100] = "";         // data length of 6 characters
  String variable = "";
  String variable1 = "";
  int index1 = 0;

  if ( Serial.available() > 0 ) {                        // если в
последовательном порту есть принятые данные
```

```
    while (Serial.available() > 0 && index1 < 100)     // read
till 6th character
    {
      delay(100);
      inChar = Serial.read();      // начинаем их
последовательно считывать и сохранять в символьный массив (по
одному символу)
      inData[index1] = inChar;
      index1++;
      inData[index1] = '\0';         // добавляем 0 в конец
    }
    variable.toUpperCase();       // преобразуем нижний регистр
в верхний
    for (byte  i = 0 ; i < 100 ; i++) {
      variable.concat(String(inData[i]));    // объединяем
строки
    }
    delay(20);
  }
  String  stringToMorseCode = String(variable);
  for (int i = 0; i < sizeof(stringToMorseCode) - 1; i++)
  {
  char tmpChar = stringToMorseCode[i];
  tmpChar = toLowerCase(tmpChar);
  GetChar(tmpChar);
  }
}
void MorseDot()
{
  tone(audio8, note, dotLen); // start playing a tone
  delay(dotLen);              // hold in this position
}
void MorseDash()
{
  tone(audio8, note, dashLen);  // start playing a tone
  delay(dashLen);               // hold in this position
}
void GetChar(char tmpChar)
{
  switch (tmpChar) {
    case 'a':
    MorseDot();
    delay(100);
    MorseDash();
    delay(100);
    break;
    case 'b':
    MorseDash();
    delay(100);
    MorseDot();
    delay(100);
    MorseDot();
    delay(100);
```

```
        MorseDot();
        delay(100);
        break;
        case 'c':
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 'd':
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 'e':
        MorseDot();
        delay(100);
        break;
        case 'f':
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 'g':
        MorseDash();
        delay(100);
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 'h':
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        break;
```

```
        case 'i':
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 'j':
        MorseDot();
        delay(100);
        MorseDash();
        delay(100);
        MorseDash();
        delay(100);
        MorseDash();
        delay(100);
        break;
        case 'k':
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        MorseDash();
        delay(100);
        break;
        case 'l':
        MorseDot();
        delay(100);
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 'm':
        MorseDash();
        delay(100);
        MorseDash();
        delay(100);
        break;
        case 'n':
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 'o':
        MorseDash();
        delay(100);
        MorseDash();
        delay(100);
        MorseDash();
        delay(100);
```

```
        break;
        case 'p':
        MorseDot();
        delay(100);
        MorseDash();
        delay(100);
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 'q':
        MorseDash();
        delay(100);
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        MorseDash();
        delay(100);
        break;
        case 'r':
        MorseDot();
        delay(100);
        MorseDash();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 's':
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        break;
        case 't':
        MorseDash();
        delay(100);
        break;
        case 'u':
        MorseDot();
        delay(100);
        MorseDot();
        delay(100);
        MorseDash();
        delay(100);
        break;
        case 'v':
        MorseDot();
        delay(100);
        MorseDot();
```

```
    delay(100);
    MorseDot();
    delay(100);
    MorseDash();
    delay(100);
    break;
    case 'w':
    MorseDot();
    delay(100);
    MorseDash();
    delay(100);
    MorseDash();
    delay(100);
    break;
    case 'x':
    MorseDash();
    delay(100);
    MorseDot();
    delay(100);
    MorseDot();
    delay(100);
    MorseDash();
    delay(100);
    break;
    case 'y':
    MorseDash();
    delay(100);
    MorseDot();
    delay(100);
    MorseDash();
    delay(100);
    MorseDash();
    delay(100);
    break;
    case 'z':
    MorseDash();
    delay(100);
    MorseDash();
    delay(100);
    MorseDot();
    delay(100);
    MorseDot();
    delay(100);
    break;
    default:
       break;
  }
}
```

Подключение проекта к Django приложению:

В примере рассмотрен простой проект подключения светодиода к плате Arduino Uno к 13 управляющему пину.

Далее подключаем плату к компьютеру и запускаем сервер.

Сначала необходимо создать виртуальное окружение, затем необходимо в него зайти:

```
python -m venv venv
venv\Scripts\activate.bat
```

Затем необходимо установить Django и библиотеку для работы с Ардуино:

```
pip install Django
pip install pyserial
```

И создать ваш проект:

```
django-admin startproject arduino_control
```

Затем необходимо перейти в папку созданного вами проекта и попробовать запустить сервер:

```
cd arduino_control
python manage.py startapp led_control
python manage.py runserver
```

В файле arduino_control/settings.py добавьте 'led_control' в список INSTALLED_APPS:

```
INSTALLED_APPS = [
    ...
    'led_control',
]
```

В файле led_control/views.py добавьте следующий код:

```
from django.shortcuts import render
import serial
def index(request):
    return render(request, 'led_control/index.html')
def control_led(request):
    port = 'COM3'  # используйте свой COM-порт здесь
    baudrate = 9600
    action = request.POST.get('action')
    if action == 'on':
        send_command(port, baudrate, '1')
    elif action == 'off':
        send_command(port, baudrate, '0')
    return render(request, 'led_control/index.html')
```

```
def send_command(port, baudrate, command):
    arduino = serial.Serial(port, baudrate, timeout=1)
    arduino.write(command.encode())
    arduino.close()
```

В папке led_control создайте папку templates и в ней создайте файл index.html со следующим содержимым:

```
<!DOCTYPE html>
<html>
 <head>
    <title>LED Control</title>
 </head>
 <body>
    <h1>Управление светодиодом</h1>
    <form method="POST" action="{% url 'led_control' %}">
      {% csrf_token %}
      <button type="submit" name="action" value="on">Включить</button>
      <button type="submit" name="action" value="off">Выключить</button>
    </form>
 </body>
</html>
```

В файле led_control/urls.py добавьте следующий код:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('led_control/', views.control_led, name='led_control'),
]
```

В файле arduino_control/urls.py добавьте следующий код:

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('led_control.urls')),
]
```

Запустите Django сервер:

```
python manage.py runserver
```

Усложняем задачу:

1) Произвести ввод данных в проект Азбука морзе не из консоли, а через web интерфейс джанго приложения
2) Сделать логирование запросов азбуки морзе в специальную таблицу базу данных Django
3) Подключить светодиодную матрицу, которая будет дублировать сообщение азбуки морзе в световом формате
4) Подключить к схеме LCD дисплей, который будет выводить изначальное сообщение, которые было введено

   ***

5) Добавить в схему датчик света и датчик звука, которые будут распознавать сигналы азбуки морзе и переводить их в тектовое содержание. Сделать отдельную страницу Django приложения под данный модуль.