上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

工程实践与科技创新

SCIENCE AND TECHNOLOGY INNOVATION

智能家居系统（Ⅳ-C）

SMART-HOME SYSTEM (IV-C)

课程指导书

GUIDE BOOK

小组成员：朱耀明 庞博 包伟铭 姬厚祥

小组编号：Ⅳ C - 01

指导教师：张 士 文

学　　院：电子信息与电气工程学院

目录

1 在树莓派上搭建系统

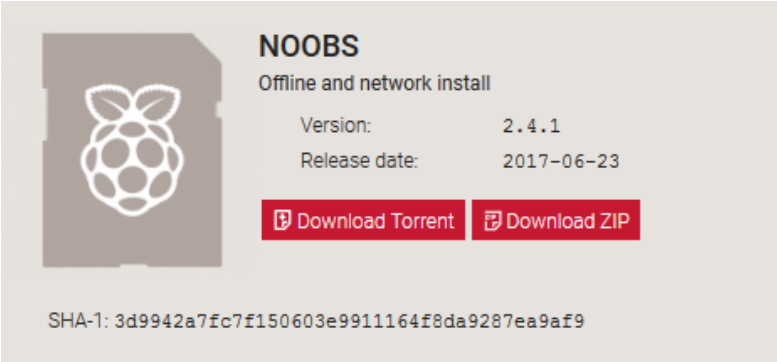
实验室分配的树莓派是 Raspberry Pi 三代 B 型，目前已有较多使用。

拿到树莓派后，首先需要完成的工作是在其上搭建系统。

在树莓派上可用的操作系统大多数是 Linux 系的，常见的有：Android 4.0（Ice Cream Sandwich），Arch Linux ARM，Debian Squeeze，Chromium OS，FreeBSD 等。此外，微软为树莓派二代定制了专用的操作系统——Windows 10 for Raspberry Pi 2。 可以说树莓派可用的操作系统也是花样繁多了。其中使用最广泛的是基于 Debian，为树莓派优化的 Raspbian 系统。本实验指导书也会以此为例，指导安装。

树莓派官网推荐初学者使用 NOOBS(New Out Of the Box Software)来进行系统的安装。使用此安装需要的硬件有：TF 卡、网线、显示器，以及电源插线。

首先要将 TF 卡格式化为 FAT32 格式。之后在 http://www.raspberrypi.org/downloads/ 网页下下载相应的 NOOBS。



之后，将 NOOBS 解压后的所有文件拷贝到 TF 卡的根目录下，组装树莓派并上电。第一次在树莓派上搭建系统时推荐使用独立显示器。

启动树莓派，可见 NOOBS 会在网上下载可用操作系统的列表。之后选择 Raspbian 选择 Install 并选择覆盖 TF 卡上数据。完成安装后，树莓派会自动重启，在新启动的页面下选择 Expand System 让树莓派系统使用全部的 TF 卡空间。设置用户与密码后再次重启，便完成了系统的安装。

- 该部分一些推荐的资料网站：

  极客学院

  树莓派官网

2 树莓派的 GPIO 控制

树莓派是一个集成了 GPIO 端口的智能系统，GPIO 这样的硬件编程总是给我们留下了不好的印象。但是这些在树莓派上有一个很好的解决方案，就是使用 Python 的 GPIO 包，这个包允许我们像使用 Arduino 一样进行单片机的编程。

下面我就简单的讲解一下 Python GPIO 包的使用。

2.1 GPIO 初步

- 首先安装 Python，同时需要安装一些开发工具

```
Sudo apt-get install python-dev
```

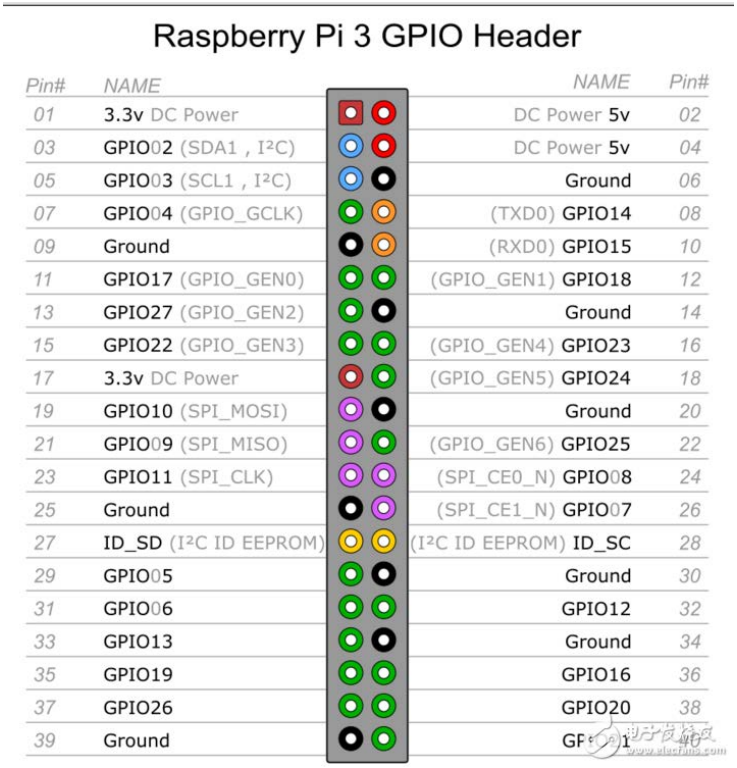
- 接下来安装 GPIO 包：

```
$ wget http://raspberry-gpio-python.googlecode.com/files/RPi.GPIO-0.5.3a.tar.gz
$ tar xvzf RPi.GPIO-0.5.3a.tar.gz
$ cd RPi.GPIO-0.5.3a
$ sudo python setup.py install
```

- GPIO 编程的方法：
  下面我们以一个控制发光二极管的例子来讲解，假设发光二极管接在 11 号管脚上
  - 导入 GPIO 包：

```
import RPi.GPIO as GPIO

 # BOARD 编号方式，基于插座引脚编号
```

  - 定义 GPIO 管脚标号模式：GPIO 的管脚有两种编号模式一种是 GPIO.BOARD，另一种是 GPIO.BCM，一般我们使用 BOARD 模式，这个模式和下图的管脚号对应：




```
GPIO.setmode(GPIO.BOARD)
```

- 定义管脚的模式（输入 or 输出）

```
GPIO.setup(11, GPIO.OUT)
```

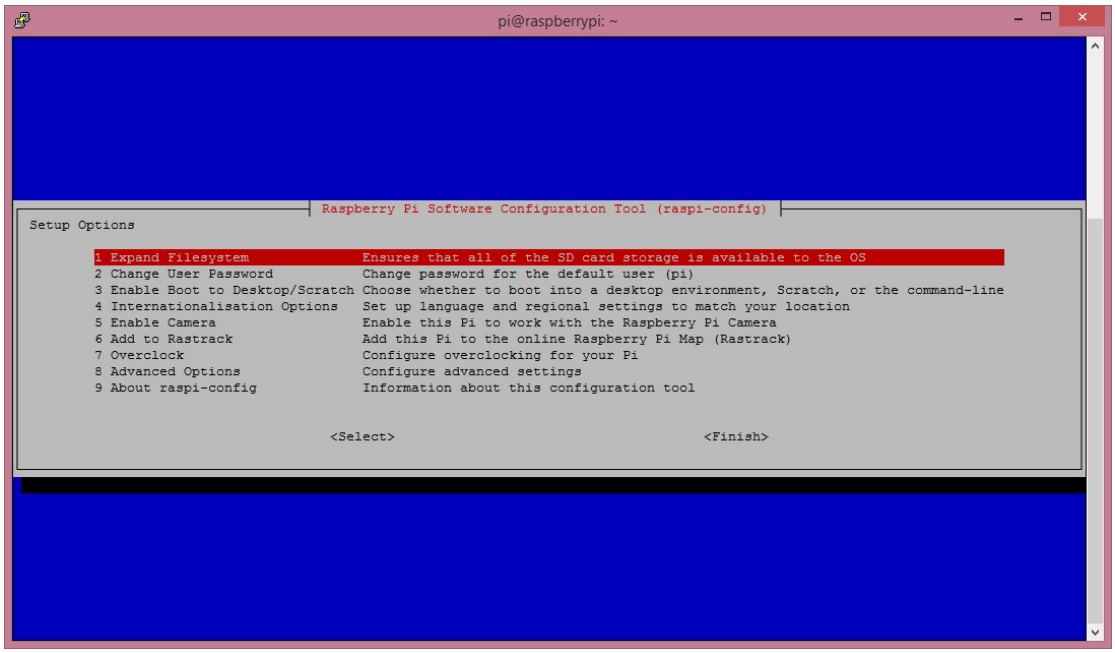
- 对输出值进行控制：

```
  while True:
    GPIO.output(11, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(11, GPIO.LOW)
    time.sleep(1)
```

2.2 特殊传感器的控制

当然不是每一个接在 GPIO 上的设备都可以向上边这样简单的控制，还会存在着一些特殊的控制方式：比如传感器会自动的将数据写入到一个指令路径的文件中，通过访问指定的路径文件可以获取传感器的信息，这样的传感器就包括温度传感器。

- 首先我们要通过设置打开 GPIO。
  在终端中输入：sudo raspi-config



在 Interfacing Option（或是 Advanced Options）下找到 1-Wire，并打开

- 在连接好传感器之后到 /sys/bus/w1/devices 下查看自己的传感器，会有一个对应的文件夹。
- 之后可以直接打开/sys/bus/w1/devices/28-031561d43aff/w1_slave 这个文件，查看数据，28-031561d43aff 就是设备编号。

- 一个通过 PWM 控制舵机的例子

```
import RPi.GPIO as GPIO
import time
import signal
import atexit

atexit.register(GPIO.cleanup)

servopin = 38
GPIO.setmode(GPIO.BOARD)
GPIO.setup(servopin, GPIO.OUT, initial=False)
p = GPIO.PWM(servopin,50) #50HZ
p.start(0)
time.sleep(2)

while(True):
  #for i in range(0,181,10):
  i = 181
  p.ChangeDutyCycle(2.5 + 10 * i / 180)
  time.sleep(1)
  p.ChangeDutyCycle(0)
  time.sleep(0.2)
```

```
#for i in range(181,0,-10):
i = 0
p.ChangeDutyCycle(2.5 + 10 * i / 180)
time.sleep(1)
p.ChangeDutyCycle(0)
time.sleep(0.2)
```

- 该部分一些推荐的资料网站：
  Google
  极客工坊
  极客学院

3 移动控制终端

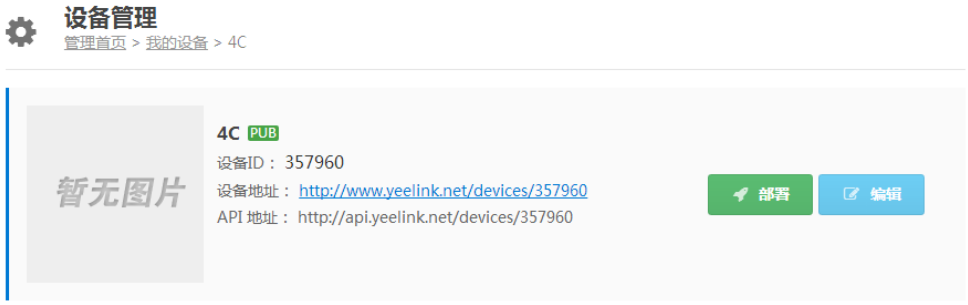
3.1 Yeelink 简介

- YeeLink 是开放的数字家庭物联网数据服务平台，接入 Yeelink 平台后，用户可以通过应用进行远程管理和控制，
- 平台提供统一的物联网数据服务接口，家庭设备可讲采集数据通过接口上传，以通过数据模型形式存储，可预设规则执行触发动作，实现特定事件监测和预警。平台还提供可定制的数据可视化界面，以图表形式呈现动态变化的家庭物联网数据。
- YeeLink 支持多平台接入，跨平台的应用接入，全面支持桌面浏览器及 Android，iOS 平台，用户随时随地都能了解自己的设备动态。

3.2 平台使用

- YeeLink 需要用户首先注册一个个人账号，对于每个账户会有独立的 API Key。

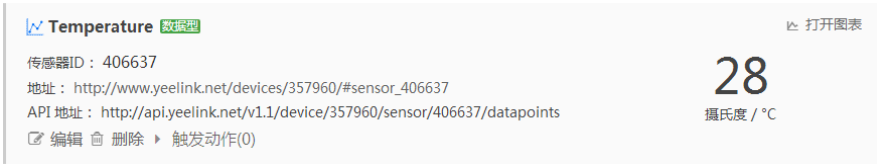
- 账户下面可以新建多个设备，一个设备下可以添加多个传感器。可以看到设备有自己的 ID，设备地址以及 API 地址。设备地址是直接可以通过浏览器打开的，API 地址用于在后面调用处理器数据以及收发指令使用。

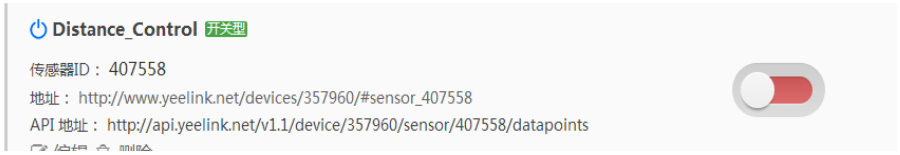


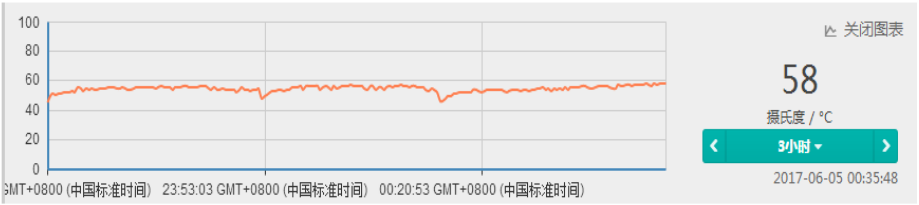
- 一个设备下可以添加多个传感器，每个传感器有自己的 ID，地址以及 API 地址。相似地，地址是可以直接打开的，API 地址用于程序中的调用。传感器有多个类型 :数据型，开关型，GPS 型，范型，微博型和图片型。我们主要用到的是数据型（采集数据）以及开关型（控制的基本操作）。数据型返回到平台上的是实时的一些参数数据，这些数据

采集和硬件有关。开关型主要是返回一个 1 或者 0 到硬件端。

- 对于数据型传感器，平台会自动做出数据的图像，显示当前信息。







3.3 API 接口使用

对于每个传感器我们需要设备（传感器群）的 header，还需要每个传感器对应的 apiurl。这里我们虽然是说用到的硬件传感器，其实是用到的数据型设备（datapoints）。一个 device 表示一组传感器的集合. 一个 datapoint 是由 key 和 value 组成的键值对.

```
apiheaders={'U-ApiKey':'14b2cab18ae1bbe626804b620fa733dd','content-type': 'application/json'}

cputemp_apiurl="http://api.yeelink.net/v1.1/device/357960/sensor/406644/datapoints"
cpuused_apiurl="http://api.yeelink.net/v1.1/device/357960/sensor/406645/datapoints"
memeryused_apiurl="http://api.yeelink.net/v1.1/device/357960/sensor/406646/datapoints"
temp_apiurl="http://api.yeelink.net/v1.1/device/357960/sensor/406637/datapoints"
humi_apiurl="http://api.yeelink.net/v1.1/device/357960/sensor/407284/datapoints"
light1url = 'http://api.yeelink.net/v1.1/device/357960/sensor/407593/datapoints'
```

对该 URL 的一个 HTTP POST 请求会为指定的传感器创建一个新的数据点，使用此 API 来为传感器存储历史数据.
URL：http://api.yeelink.net/v1.0/device/<device_id>/sensor/<sensor_id>/datapoints
注意目前限定相邻数据上传间隔须大于等于 10s，过于频繁的请求会收到 406 Response.请求的参数是 device_id 和 sensor_id,两者都是 int 型的数据。数据格式为 JSON，一个 datapoint 是由 key 和 value 组成的键值对. 数值型传感器中 key 为 timestamp，value 为数值。key 为唯一索引; timestamp 为 ISO 8601 标准时间格式(默认时区为中国标准时间 CST)，例如：2012-03-15T16:13:14. 访问授权：需要在 HTTP Header 中增加 API Key 来授权写入操作。

3.4 查看数据点

对该 URL 的请求返回指定 key 的 datapoint，若未指定 key，则返回该 sensor 的最新数据.

参数名	必需	类型	说明
device_id	true	string	设备ID
sensor_id	true	string	传感器ID
key	true	string	数值和GPS型为数据的时间戳，泛型为自定义字符串

使用 GET 返回值是请求的传感器信息，返回值数据格式是 JSON。开关型其实是数据型的一个返回值只有 0 和 1 的特例，所以使用方法与数据型是差不多的。下面是我们在真正使用时进行的调用。

```
r1 = requests.get(light1url,headers=apiheaders)
r2 = requests.get(light2url,headers=apiheaders)
r3 = requests.get(light3url,headers=apiheaders)
off = requests.get(lightoff,headers=apiheaders)
#print(r1.text)
led1 = r1.json()
led2 = r2.json()
led3 = r3.json()
offf = off.json()
```

4 第一个 Django 项目

在智能家居系统中，为了实现网页控制树莓派的功能，我们需要在树莓派上搭建网页，实现用户-网页-树莓派的交互。目前常见的网页开发框架种类繁多，其中 Django 是由 Python 开发的一个免费的开源网站框架，可以用于快速搭建高性能，网站。本指导书会以 Django 为例谈一谈网站开发的简单流程。

4.1 Django 安装

首先是关于 Django 的安装，本次项目使用的是 1.7.1 版本。树莓派操作系统为 Fedora：

```
$ pip install django==1.7.1
```

也可以在官方的源码包列表中进行下载，之后解压安装

```
$ tar -xvf django-1.8.13.tar.gz
$ cd django-1.8.13
$ sudo python3 setup.py install
```

之后可以在 Python 交互环境下检测是否已正确安装

```
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.get_version()
'1.7.1'
```

关于开发环境，笔者推荐使用 PyCharm 专业版作为 IDE 开发。使用 SJTU 邮箱可以免费申请 JetBrain 公司的专业版账号，享受 PyCharm 带来的便捷。

4.2 创建新 Django 项目

接着在树莓派本地创建一个文件夹用于本次的开发工作。在该目录下，直接使用一下命令便可以创建一个 Django 项目：

```
$django/ $ django-admin startproject SmartHome
$django/ $ cd HelloWorld
$HelloWorld/ $ ls
SmartHome manage.py
```

在其中可以新建一个 app，之后将其作为主页，命名为 HomePage

```
$HelloWorld/ $ python3 manage.py startapp HomePage
$HelloWorld/ $ ls
Hello    HomePage manage.py
```

使用 Linux 的 tree 命令可以查看当前的目录树：

```
.
├── HomePage
│   ├── admin.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── manage.py
└── SmartHome
    ├── __init__.py
    ├── __init__.pyc
    ├── settings.py
    ├── settings.pyc
    ├── urls.py
    └── wsgi.py
```

它们的作用分别是：

urls.py
链接入口，关联到对应的 views.py 中的一个函数（或者乘坐 generic 类），访问的链接就对应一个函数。

views.py
处理用户发出的请求，从 urls.py 中对应而来，通过渲染 templates 中的网页可以为用户显示页面内容，比如登录后的用户名，用户请求的数据，通过其输出到页面。

models.py
与数据库操作相关，存入或读取数据时使用。当不使用数据库的时候，也可以当做一般的类封装文件，存储各种类的定义。

forms.py
表单，用户在浏览器上输入提交，对数据的验证工作以及输入框的生成等工作，都依托于此。

admin.py
后台文件，可以用少量的代码就拥有一个强大的后台。

settings.py
Django 的设置、配置文件，比如 DEBUG 的开关，静态文件的位置等等。

除了这些，还有以上目录中未提及的：

templates 目录

views.py 中的函数渲染 templates 中的 html 模板，得到动态内容的网页，可以用缓存来提高渲染速度。

此外，对于不同版本下的 Django 协同开发，可以使用 virtualenv 分离各个开发环境。关于 virtualenv 工具的使用超出了本指导书的范围，有兴趣的同学可以自行上网查阅。

4.3 定向跳转设置

接下来介绍开发智能家居网页的一个关键 Django 要素：视图与定向跳转

我们将之前新建的 HomePage 这个 app 添加至 settings.py 中的 INSTALL_APPS 中，Django 就可以自动的找到 app 中的模板文件（app-name/templates/all_files）。

```
# Application definition

INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'HomePage',
)
```

接下来的操作是定义视图函数。在 HomePage 目录中，在 learn 目录下先增加一个 templates 目录，用来放置模板文件，该目录下创建 home.html 文件作为网页。

在 Django 的默认配置中，模板文件放在对应 app 的 templates 的目录中。所以我们新建的这个目录中存放的模板文件，Django 会自动在此查询文件。

```
<!DOCTYPE html>
<html>
<head>
    <title>HomePage</title>
</head>
<body>
This is HomePage.
</body>
</html>
```

接下来编辑 views.py：

```
#coding: utf-8
from django.shortcuts import render
from django.http import HttpResponse


def index(request):
    return HttpResponse(u"This is index page!")


def home(request):
    return render(request, 'home.html')
```

第三行引入 HttpResponse ，它是用来向网页返回内容的，就像 Python 中的 print 函数一样，只不过 HttpResponse 是把内容显示到网页上。）

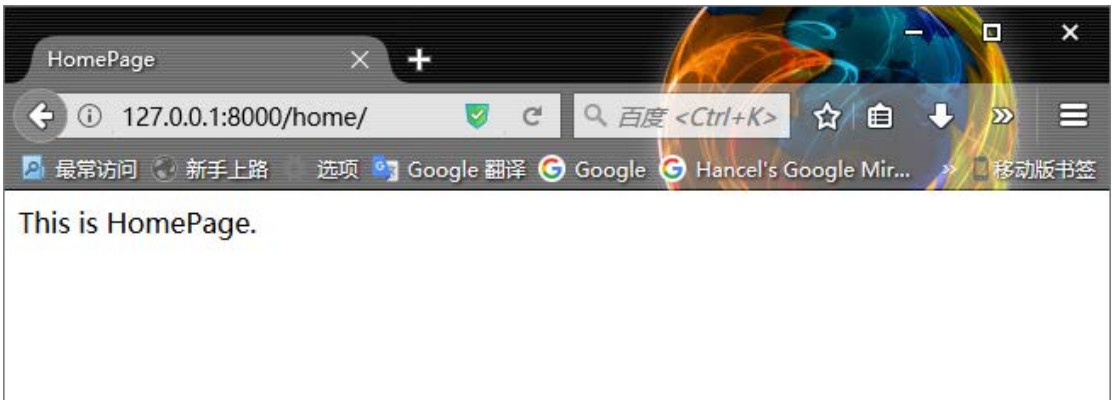
最后我们定义了一个 index() 函数，第一个参数必须是 request，这个与请求有关，request 变量里包含了 get 或 post 方式（这是一种 RESTful API 的设计风格，在资源传递方式中还有 put、delete 另两种方式，只是之前提到的两个最常用）传递而来的参数。我们可以对这些参数做出定制处理，然后向用户层返回待展示的数据。。

最后定义视图相关的 URL，即定向跳转配置。编辑文件 urls.py：

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
from HomePage import views as homePage_view

urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'SmartHome.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    url(r'^admin/', include(admin.site.urls)),
    url(r'^home/', homePage_view.home),
    url(r'^index/', homePage_view.index),

)
```

完成后，在项目目录启动 Django 服务器查看效果

```
$ python manage.py runserver 8000
```



看见类似上图的网页，可认为基础框架开发完成。

4.4 前后端交互

最后谈一谈前后端交互的话题:
当前端需要向后端传递数据时，在前端用 get 或 post 方法提交一些表单数据。
以下的 HTML 与 Python 可实现简单的网站加法。定向跳转配置等请读者自行完成。
前端：

```
<!DOCTYPE html>
<html>
<body>
<p>Please input 2 numbers:</p>

<form action="/add/" method="get">
    a: <input type="text" name="a"> <br>
    b: <input type="text" name="b"> <br>
    <input type="submit" value="submit">
</form>

</body>
</html>
```

后端：

```
from django.http import HttpResponse
from django.shortcuts import render

def index(request):
    return render(request, 'index.html')

def add(request):
    a = request.GET.get('a', None)
    b = request.GET.get('b', None)
```

```
    a = int(a)
    b = int(b)
    return HttpResponse(str(a+b))
```

对于后端向前端传递数据，比较常见的方法之一是直接在视图函数（views.py 中的函数）中将 JSON 对象 和网页其它内容一起传递到 Django 模板

下面的代码可以实现将后端数据显示到前端弹窗上

后端：

```
from __future__ import unicode_literals
from django.shortcuts import render


def home(request):
    List = ['a', 'b']
    return render(request, 'home.html', {'List': List})
```

前端：

```
<script type="text/javascript">
    var List = {{ List }};
    alert(List);
</script>
```

需要注意两点：

1. 视图函数中的字典或列表要用 json.dumps()处理。
2. 在模板上要加 safe 过滤器。

- 该部分一些推荐的资料网站：

  实验楼

  自强学堂

5 简单的 HTML5 前端网页

5.1 HTML5 前端网页简介

HTML5 的设计目的是为了在移动设备上支持多媒体。新的语法特征被引进以支持这一点，如 video、audio 和 canvas 标记。该标准与 2014 年正式发布，将是未来的趋势。对 canvas 的支持可以使浏览器脱离 flash 和 silverlight 显示图片。HTML5 加 CSS3 可实现响应式网页（即网页的布局排版非固定，灵活适配不同屏幕大小）。

HTML5 沿袭了 HTML 标准的文件结构，以 HTML 标签形式嵌套组织实体，网页主体在<html></html>标签中，常规二级结构包括：<head></head>声明网页的一些元数据，指定网页外部元素链接等，和<body></body>网页主体显示部分。

一个简单的 HTML 网页示例如下：

```
<html>
<head>
<meta charset="utf-8">
<title>D01-Smart Home System</title>
<meta name="description" content="《工程实践与科技创新 IV-C》项目网站
">
<!-- Stylesheet
================================================== -->
<link rel="stylesheet" type="text/css"
href="../static/css/style.css">
<link rel="stylesheet" type="text/css"
href="../static/css/responsive.css">

</head>
<body>
<!-- Home Page
==========================================-->
<p>《工程实践与科技创新 IV-C》项目网站</p>

<script type="text/javascript"
src="../static/js/jquery.1.11.1.js"></script>
<script type="text/javascript"
src="../static/js/bootstrap.js"></script>
<script type="text/javascript"
src="../static/js/SmoothScroll.js"></script>
<script type="text/javascript"
src="../static/js/main.js"></script>
</body>
</html>
```

网页编辑工具成熟的商业产品有很多，但对于报告网页此类网站建议直接使用文本编辑器。初学建议从比较简单，结构相对清晰的网页模板开始。学习时边查询 W3School 相关教程，边动手码字，效果最好。

5.2 Bootstrap 使用简介及示例

Bootstrap 是 Twitter 推出的一个用于前端开发的开源工具包。它由 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作开发，是目前最受欢迎的前端框架。

Bootstrap 提供了许多丰富的元素，具体可参考相关资料。

Bootstrap 文档（包括 CSS，JavaScript 文档）可从官网：http://getbootstrap.com/下载。官网同时提供了官方模板和收录的用户优秀模板作品。

下面简单介绍使用 Bootstrap 分栏：

通过 Bootstrap 可以很方便地进行分栏，实现如屏幕过小时，正常四栏会按顺序变为 2x2 显示的效果。Bootstrap 默认将一个屏幕分为 12 栏，在 div 标签中通过制定 class 名方式设置。如要默认分为三栏，则每个 div 都应归为 col-md-4 类下。

```
<section class="main">
    <div class="row">
        <div class="col-md-4">
            <hr>
            <h2 class="text-center">LED 1</h2>
            <div class="switch demo3">
                <input id="cb1" type="checkbox">
                <label><i></i></label>
            </div>
            <hr>
        </div>
        <div class="col-md-4">
            <hr>
            <h2 class="text-center">LED 2</h2>
            <div class="switch demo3">
                <input id="cb2" type="checkbox">
                <label><i></i></label>
            </div>
            <hr>
        </div>
        <div class="col-md-4">
            <hr>
            <h2 class="text-center">LED 3</h2>
            <div class="switch demo3">
                <input id="cb3" type="checkbox">
                <label><i></i></label>
            </div>
            <hr>
        </div>
    </div>
</section>
```

结合相关 CSS 文件可实现如下效果：

5.3 Django 下前端页面链接

使用 Django 建站后，网站的 URL 解析由 Django 完成，即非服务器源文件的相对路径。

Django 中每一个 View 对应一个文件夹，该文件夹中包括 Python 文件及 migrations 文件夹，以及存放静态网页模板的 templates 文件夹。

Templates 文件夹中存放，与 templates 文件夹同一级可新建名为 static 的文件夹，该文件夹内可以放置非嵌入的网页静态元素，如 CSS，JavaScript 脚本，静态图片等元素源文件。（但 static 整个工程中最好仅有一个，为防冲突，Django 会全局索引。）

Templates 文件夹下对 static 文件中的静态 css 的加载示例如下：

```
<!-- Stylesheet
================================================== -->
<link rel="stylesheet" type="text/css"  href="../static/css/style.css">
<link rel="stylesheet" type="text/css" href="../static/css/responsive.css">
```

页面链接时，超链接指向 urls.py 中 `urlpatterns` 指定的路径即可，非实际路径，下图时本项目前端网页导航条超链接的一个示例。

```
      <ul class="nav navbar-nav navbar-right">
        <li><a href="/control/" class="page-scroll">Control Panel</a></li>
        <li><a href="/log/" class="page-scroll">Log</a></li>
        <li><a href="/video/" class="page-scroll">Video Monitor</a></li>
      </ul>
```

- 该部分一些推荐的资料网站：

  W3School

  Bootstrap 中文网：http://www.bootcss.com/

  Bootstrap 教程：http://www.runoob.com/bootstrap/bootstrap-tutorial.html