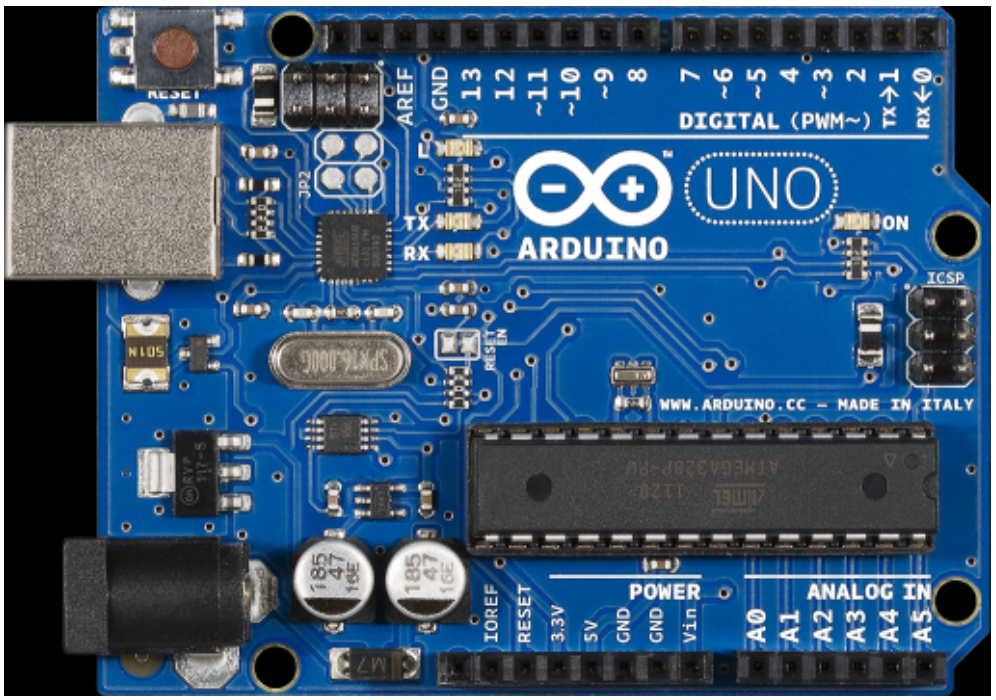
使用Python和Arduino、ESP8266等进行物联网项目开发

目录

1 Arduino

1.1 Arduino UNO特性



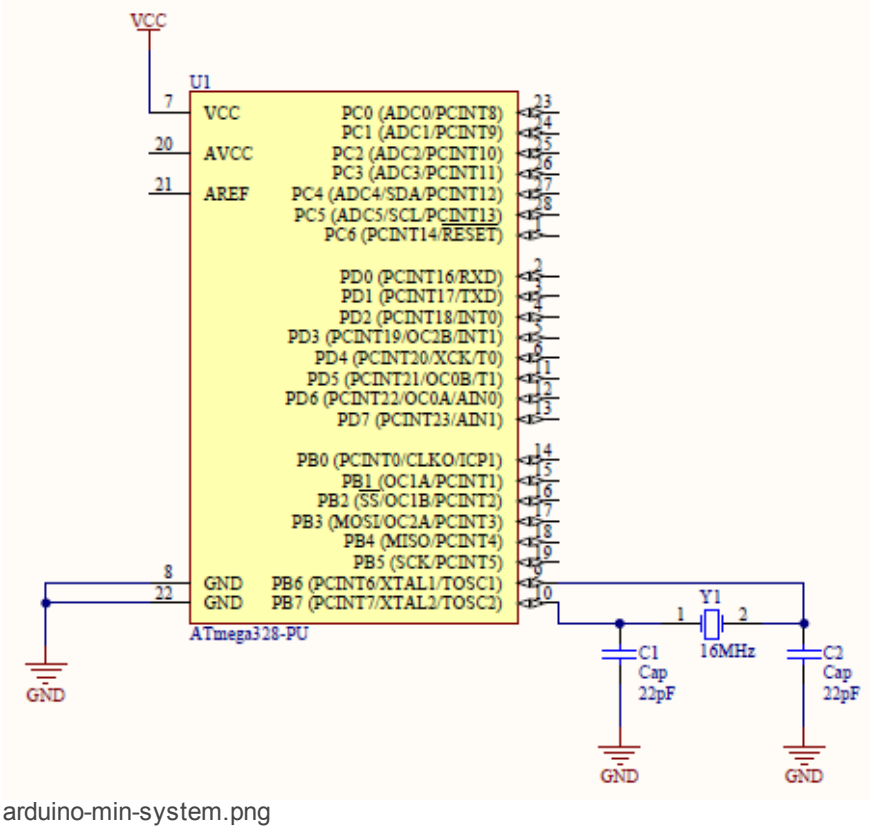
Arduino UNO是基于ATmega328P的单片机开发板。该开发板由14路数字输入/输出引脚（其中6路可以用作PWM输出）、 6路模拟输入、1个16MHz的石英晶体振荡器、一个USB接口、1个电源接头、1个ICSP数据头以及1个复位按钮组成。

UNO包含了单片机运行所需的所有要素，只需用USB连接线将其连接到计算机，或利用AC-DC适配器或电池供电后即可启动。

UNO的特色在于将ATmega16U2编程为一个USB-to-serial转换器，以便能简单、轻松和自由地安装驱动程序。

官方网站：https://www.arduino.cc/

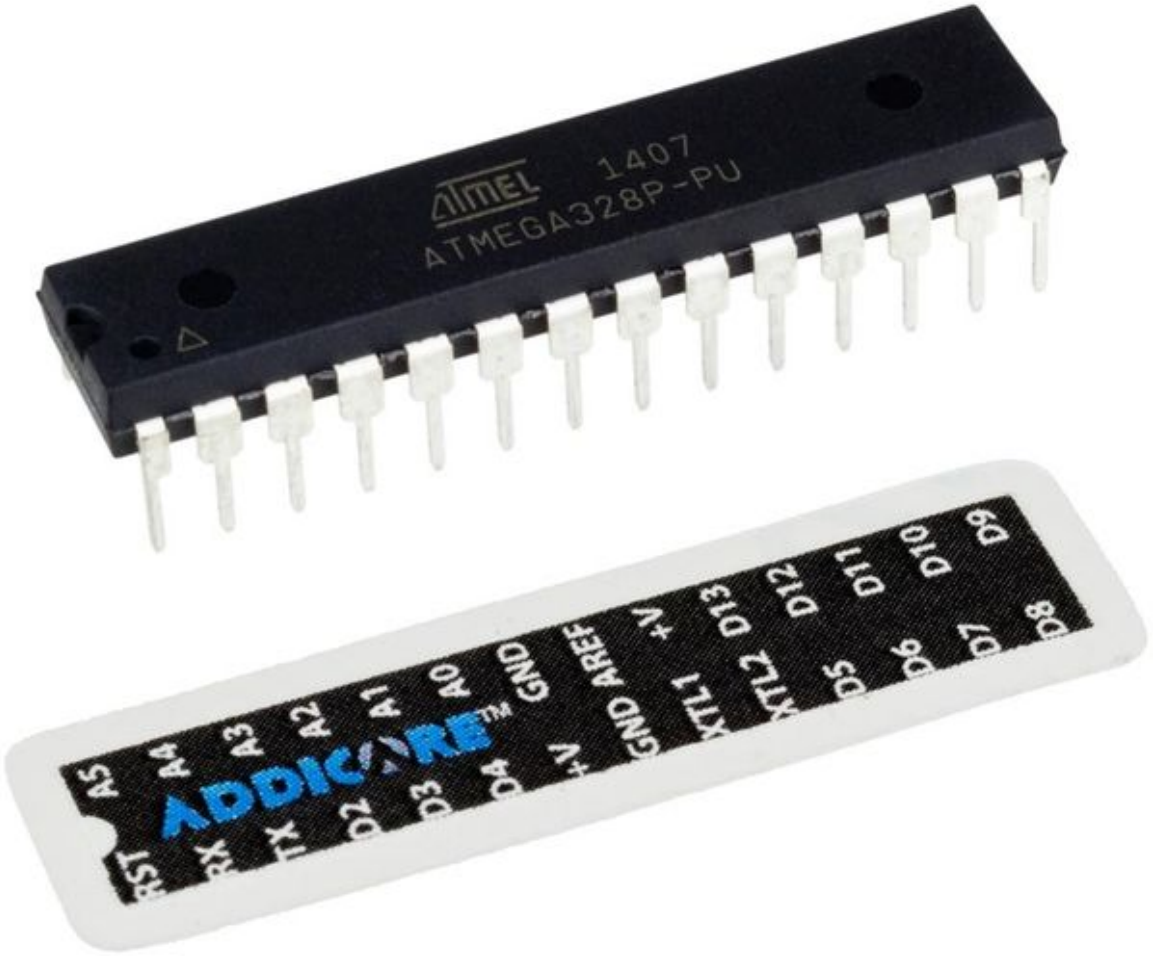
1.2 Arduino最小系统



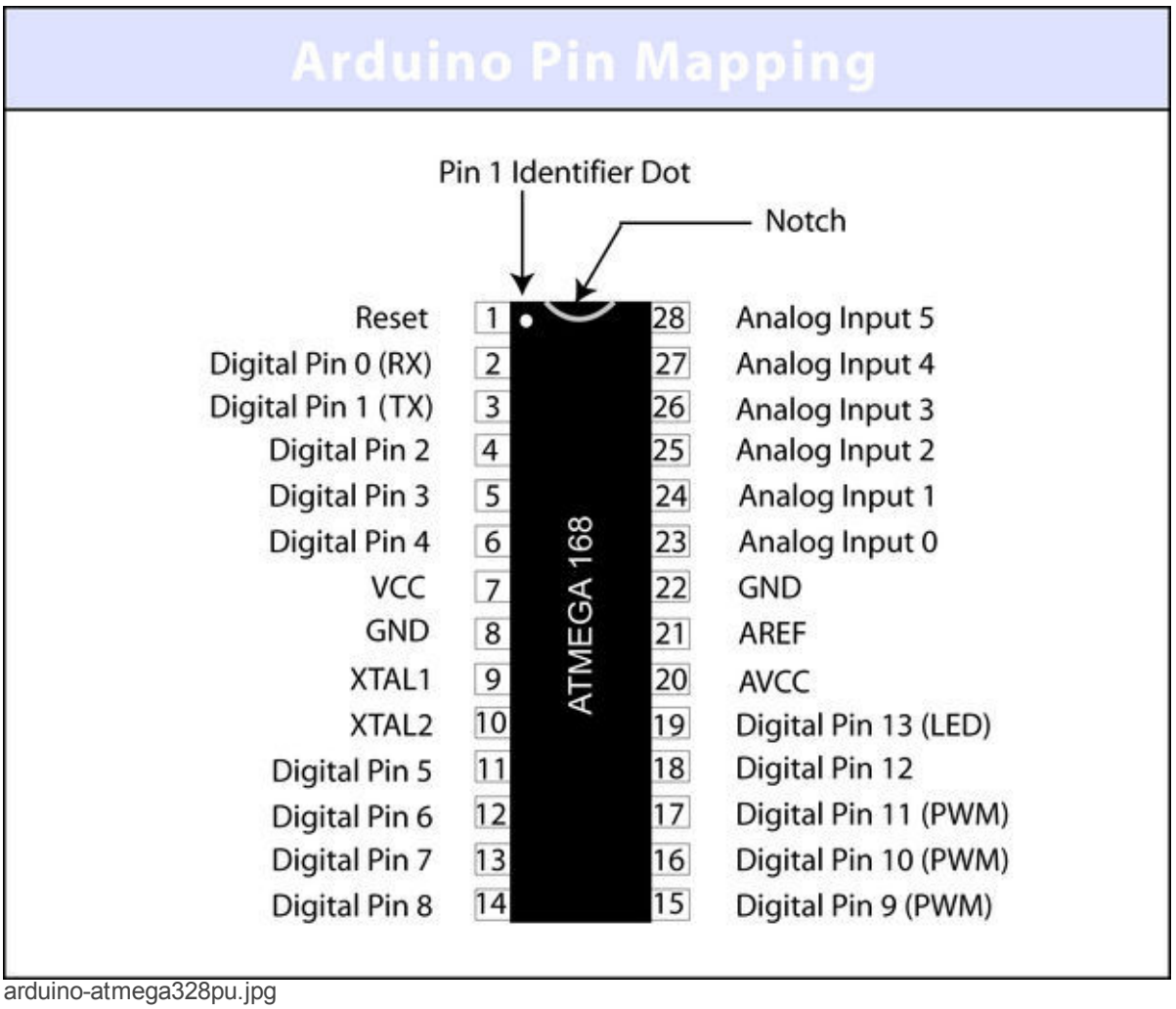
arduino-min-system.png

下面两个图说明了ATMEGA329PU芯片的引脚和Arduino UNO引脚之间的对应关

系。http://www.instructables.com/id/%E8%87%AA%E9%80%A0%E4%BD%A0%E7%9A%84Arduino-UNO%E6%9D%BF/?ALLSTEPS描述了如何自建Arduino UNO系统。



atmega328pu-pins.jpg



arduino-atmega328pu.jpg

2 ESP8266

ESP8266 是由Espressif Systems设计生产的一款高度集成的芯片，该芯片专门针对无线连接的需求而开发，是一个完整且自成系统的 WiFi 网络解决方案。它能够搭载软件应用，也能通过另一个应用处理器卸载所有的 WiFi 网络功能。

ESP8266 具备强大的片上处理和存储功能，这使其可通过 GPIO 口集成传感器及其他应用的特定设备，既缩短前期开发时间，也最大限度减少运行中系统资源的占用。
ESP8266 高度片内集成，仅需极少的外部电路，而其包括前端模块在内的整个解决方案，可将设计中 PCB 所占的空间降到最低。

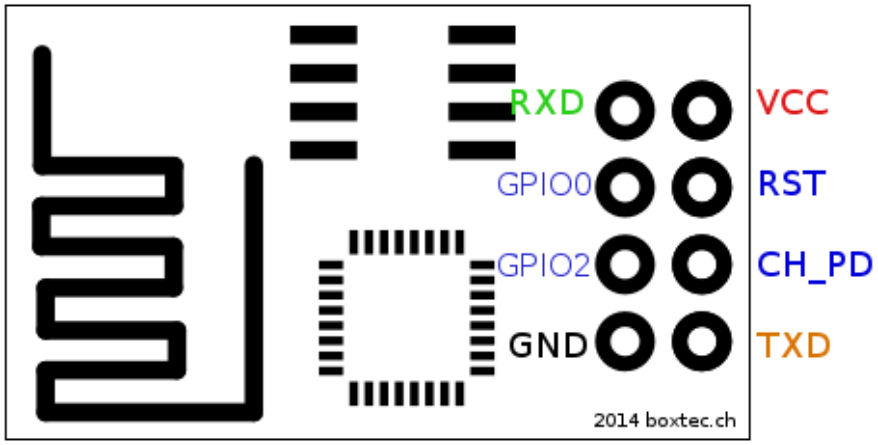
国内的安信可科技等基于该芯片生产了一系列的模块，并提供了基于开源SDK改造的firmware。

产品规格

- 802.11 b/g/n
- WiFi Direct (P2P)、soft-AP
- 集成 TCP/IP 协议栈
- 集成 TR 开关、 balun、LNA、 PA 和匹配网络
- 集成PLL、稳压器、DCXO 和电源管理单元
- 802.11b模式下 +19.5dBm 的输出功率
- 小于 10uA 的断电泄露电流
- 集成低功耗 32 位 CPU，可以兼作应用处理器
- SDIO 1.1/2.0、 SPI、UART 接口

- STBC、 1×1 MIMO、2×1 MIMO
- A-MPDU & A-MSDU 聚合 & 0.4ms 的保护间隔
- 2ms 之内唤醒并传递数据包
- 待机状态消耗功率少于 1.0mW (DTIM3)

2.1 ESP8266无线模块引脚图



ESP8266无线模块正面（元件面）视图

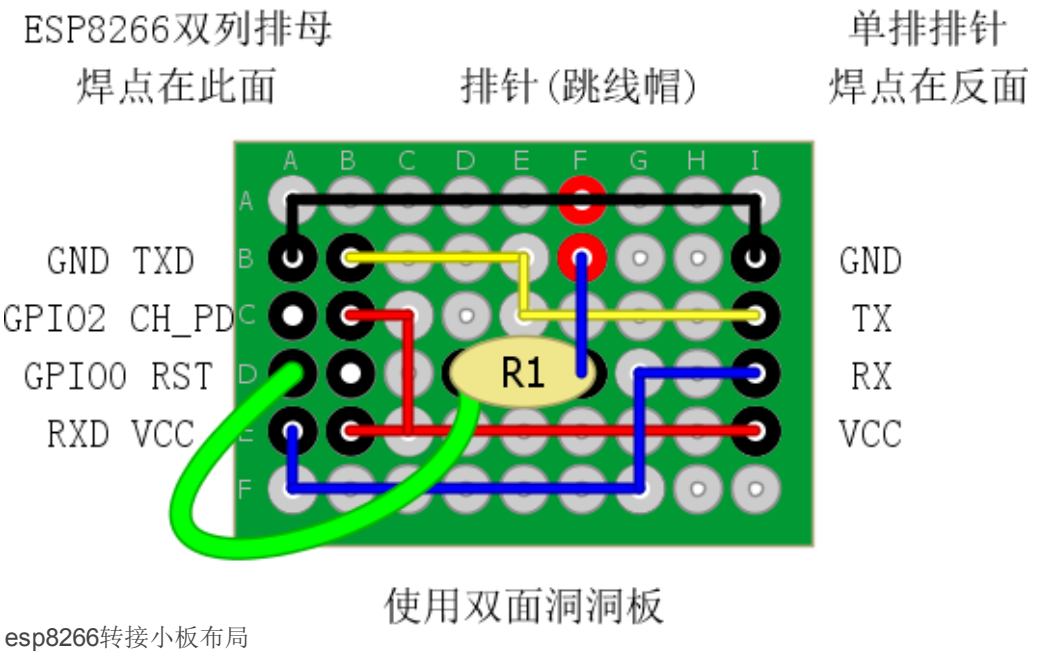
其中，ESP8266无线模块的VCC和CH_PD连接3.3V电源，需要注意的是arduino和USB-TTL模块提供的3.3V电源不能保证模块稳定工作，因此最好外接**3.3V**电源。我这里使用的是YwRobot面包板专用电源模块，该模块提供两路电源输出，可分别设置为5V和3.3V输出。

ESP8266无线模块的GND接地，要注意多个模块或电路连接时，需要实现共地。

ESP8266无线模块的RXD和TXD引脚为串口通信的接收和发送引脚，分别接到对方的TXD和RXD。

ESP8266无线模块的GPIO0引脚在刷新固件时，可通过1K电阻接到GND。正常工作时悬空即可。

为了方便在面包板上使用，可以自制一个转接用的小板，将需要使用的引脚用单排排针引出。



esp8266转接小板布局

2.2 AT指令

ESP8266提供了AT命令集，这样单片机和其他设备能通过AT指令和ESP8266进行通信。不同版本的AT指令存在一些差异，具体请查看文档。

http://wiki.iteadstudio.com/ESP8266_Serial_WIFI_Module#AT_Commands提供了AT指令的参考和文档下载。

常用的AT指令包括：

- AT
- AT+RST
- AT+GMR
- AT+CWMODE=
- AT+CWLAP
- AT+CWJAP=
- AT+CIFSR
- AT+CIPMUX=
- AT+CIPSTART
- AT+CIPCLOSE=
- AT+CIPSERVER=
- AT+CIPSEND
- AT+CIPBAUD(AT+UART)

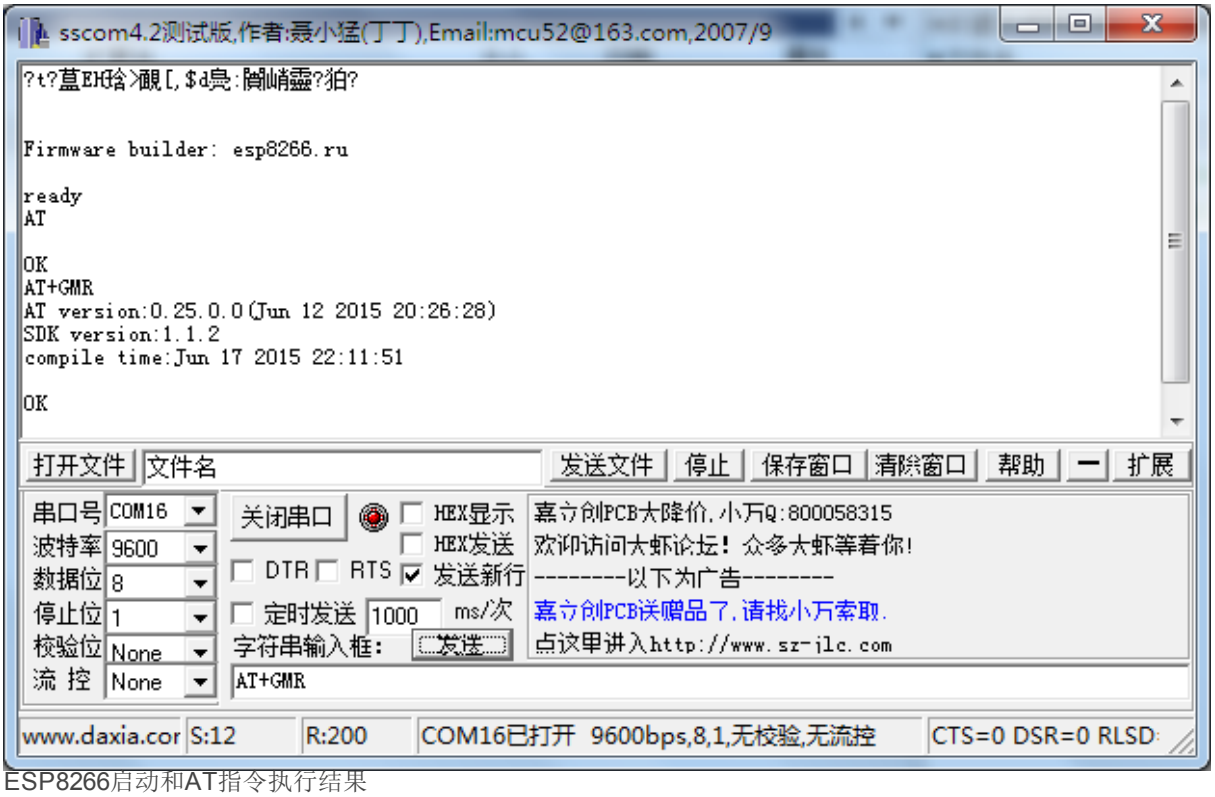
具体语法和说明参考AT指令参考手册。

2.3 使用USB转TTL连接ESP8266

USB转TTL小板非常有用,可用于手机、单片机、机顶盒等刷机。可选择CH340G方案的，淘宝上在5元左右即可买到。

连接步骤：

1. 分别将小板的Tx，Rx，GND连接到ESP8266模块的RXD，TXD和GND。
2. 将面包板专用电源的3.3V电源和地分别接到ESP8266模块的VCC（CH_PD）和GND。
3. 将USB-TTL插入到电脑USB接口。
4. 在电脑上打开串口工具，选择正确的COM口，并以115200波特率打开。
5. 按下面包板专用电源上的电源开关。
6. 你可以看到模块启动过程，启动完成之后显示ready。你可以在串口工具中输入AT指令并观察ESP8266的响应结果。



ESP8266启动和AT指令执行结果

2.4 波特率设置

ESP8266出厂默认波特率是115200，在刷新固件之后的默认波特率也是如此。也有部分固件做了修改，刷新固件之后的波特率是常用的9600。

在使用串口工具或arduino和模块通信时，必须保证双方的波特率等参数一致。

Arduino使用软串口（SoftwareSerial）时，受MCU性能限制，最高波特率为38400，一般使用9600波特率。

修改波特率可通过串口助手SSCOM等工具，以默认的115200波特率连接之后，输入如下AT指令可修改波特率：

- `AT+CIOBAUD=9600` (0.21及之前AT指令集版本)
- `AT+UART=9600,8,1,0,0` (0.22及之后AT指令集)

修改波特率之后需要重新以新波特率打开窗口，并重启模块。即可看到模块启动过程和最后的ready信息。

2.5 固件刷新

建议使用EspressIf官方发布的固件下载软件flash download tool进行固件刷新。在刷新之前先下载好合适的固件文件，注意某些固件对Flash大小有要求，通过AT指令 `AT+RST` 重启模块之后可以看到Flash容量等信息。下载的固件一般有说明文档，一定要仔细阅读。

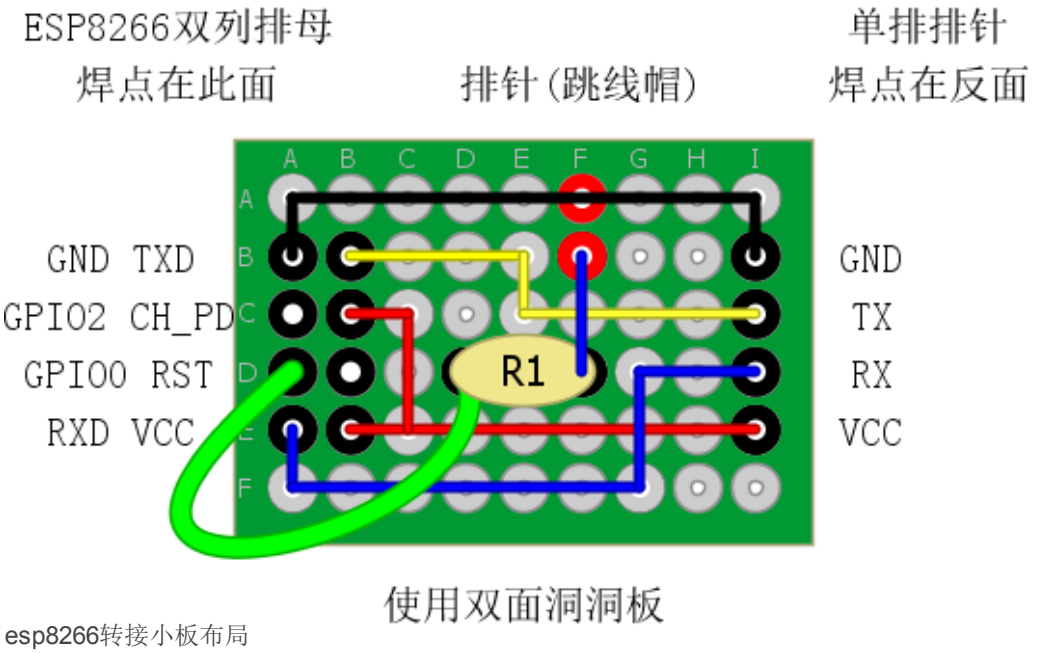
根据项目测试经验，从http://www.espruino.com/files/下载的`ESP8266_AT25-SDK112-512k.bin`这个固件比较可靠。

固件刷新步骤：

1. ESP8266无线模块的GPIO0引脚通过1K电阻接到GND。
2. 打开Flash Download Tool，选择固件文件，按照固件文档中的说明设置好参数。

3. 点击START按钮开始刷新。
4. 按下面包板专用电源上的电源给模块上电，固件下载过程随即开始。
5. 固件下载完成之后，即可通过波特率设置中的方法进行参数设置。

备注：可以在[2.1 ESP8266无线模块引脚图]中描述的专用小板的基础上，在GPIO引脚加一个电阻，并通过排针和跳线帽的方式来控制小板的工作方式。



esp8266转接小板布局

3 使用Python测试ESP8266

程序来源 https://github.com/guyz/pyesp8266

`esp8266test.py` 是一个通过USB-TTL向ESP8266发送AT指令并接收响应的python程序。

`esp8266server.py` 这个python程序将ESP8266连接到一个无线AP或无线路由器，然后启动web程序，通过esp8266提供服务。模块工作在multiple connections模式，默认的固件在第二次执行`AT+CIPCLOSE=0`命令时会失去响应，必须对固件进行刷新。

在测试之前，确保波特率和电气连接正确无误。

4 Arduino连接ESP8266

Arduino UNO只有一个内置串口，在使用USB连接线进行编程和串口监视时，串口就不够用了。因此更多时候，会使用`SoftwareSerial`连接ESP8266模块。在这种情况下，软串口的最高波特率为38400，一般设置为9600。

在连接之前，通过USB转TTL小板连接ESP8266，通过AT指令修改模块默认波特率。

5 Nokia 5110 LCD使用

其它类型的产品相比，LCD5110模块具有以下特点：

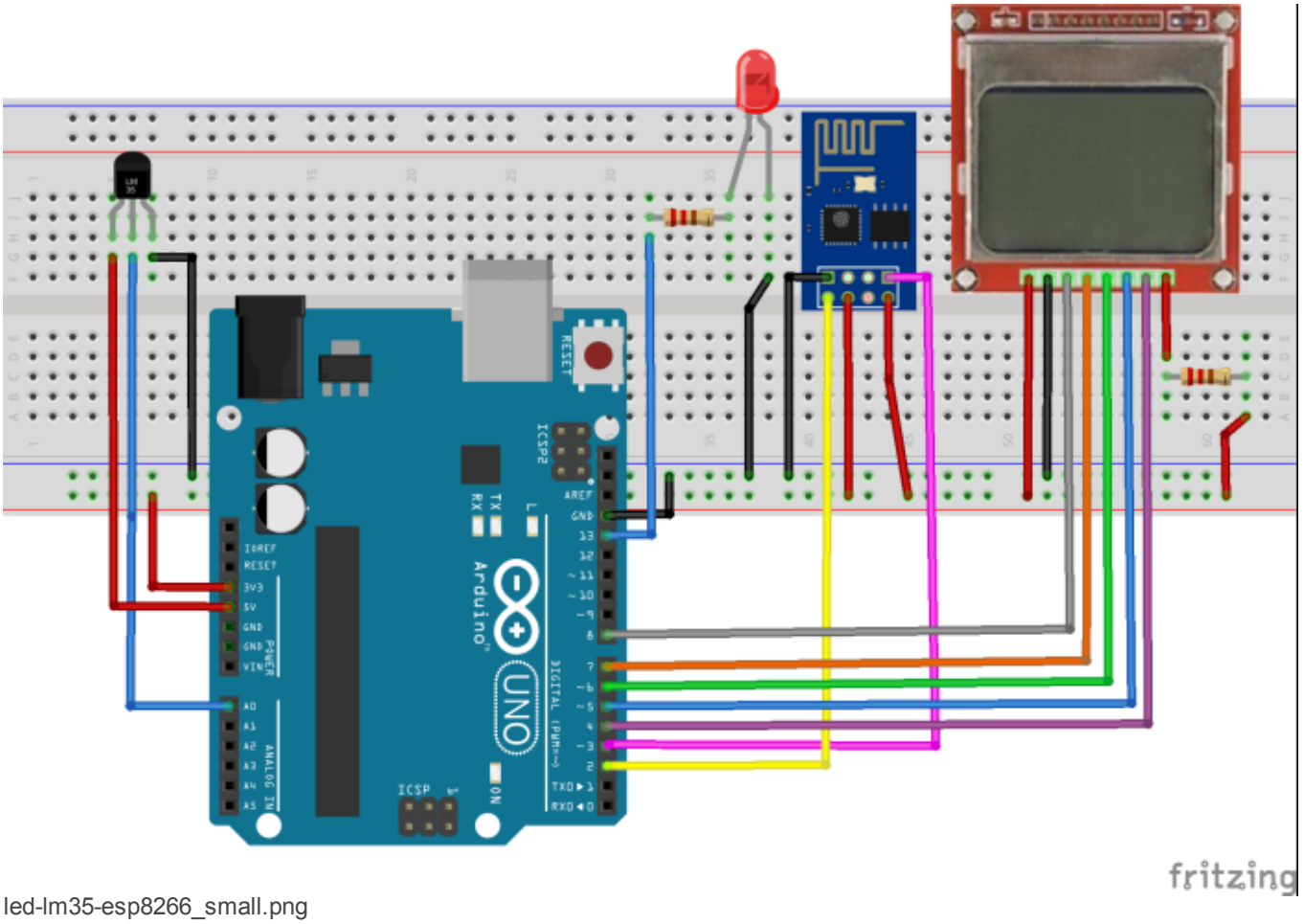
- 84x48 的点阵LCD，可以显示4 行汉字，
- 采用串行接口与主处理器进行通信，接口信号线数量大幅度减少，包括电源和地在内的信号线仅有9 条。
- 支持多种串行通信协议（如AVR 单片机的ＳＰI、MCS51 的串口模式０等），传输速率高达4Mbps，可全速写入显示数据，无等待时间。
- 可通过导电胶连接模块与印制版，而不用连接电缆，用模块上的金属钩可将模块

固定到印制板上，因而非常便于安装和更换。

- LCD 控制器 / 驱动器芯片已绑定到LCD 晶片上，模块的体积很小。
- 采用低电压供电，正常显示时的工作电流在200μA 以下，且具有掉电模式。

http://yfrobot.com/thread-2412-1-1.html这篇文章描述了该LCD的使用方法，其中后一个程序用到了https://github.com/carlosefr/pcd8544这个arduino库。下载zip文件后，将`PCD8544.h`和`PCD8544.cpp`文件拷贝到*Arduino\Libraries*下，即可使用。注意：第二段代码中的`Nokia5110.h`和`nokia5110`类名要分别改为`PCD8544.h`和`PCD8544`。

6 一个温度测量和LED控制的物联网项目



led-lm35-esp8266_small.png

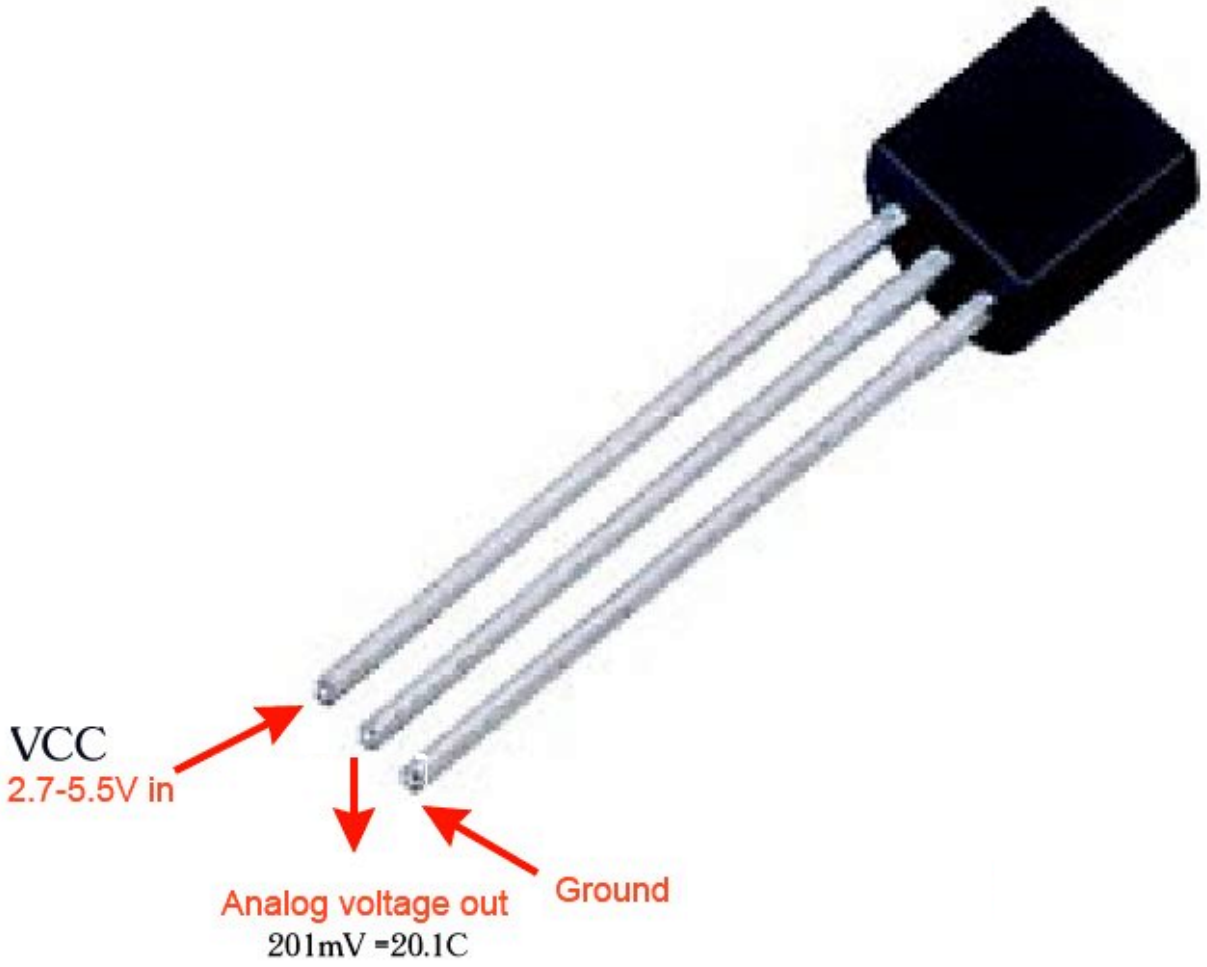
其中LCD用于显示系统启动过程

6.1 材料清单(BOM)

- Arduino UNO 一个
- ESP8266模块 一个
- Nokia 5110 LCD 一个
- LM35 温度传感器 一个
- 1千欧电阻 2个
- LED 一个
- 面包板一块
- 面包板专用电源一个

注意：图中使用了Arduino UNO的3.3V电源输出，实际不能提供足够的电流输出。

6.2 引脚连接



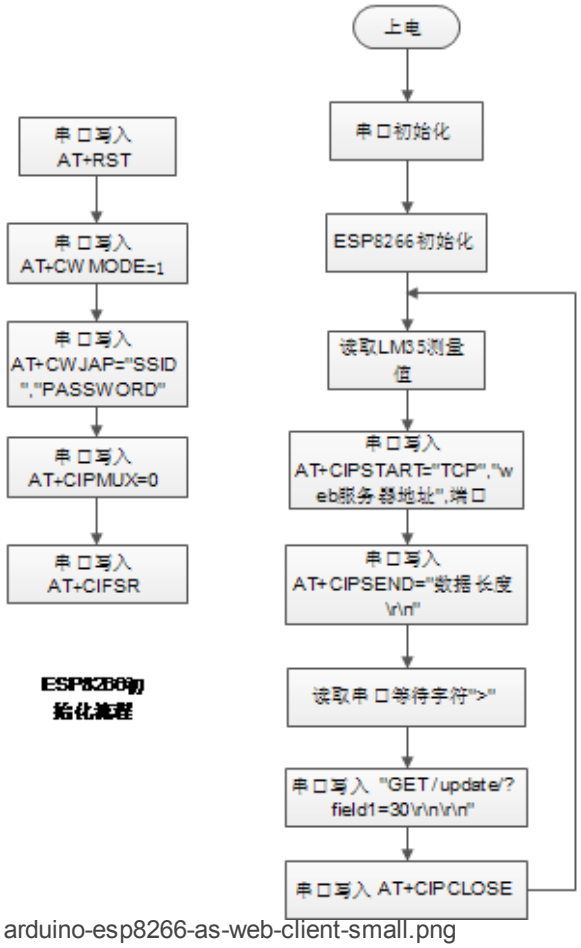
lm35.png

Nokia5110.jpg

- 13脚接电阻和LED
- A0接LM35的输出端
- 2脚接ESP8266的TX
- 3脚接ESP8266的RX
- 4脚接LCD的CLK
- 5脚接LCD的DIN
- 6脚接LCD的DC
- 7脚接LCD的RST
- 8脚接LCD的CE
- LCD的BL引脚通过1K电阻接3.3V电源
- LM35的VCC接+5V
- 其他红色连线接3.3V电源，黑色连线接地

6.3 工作方式一

ESP8266工作为workstation模式，Arduino作为客户端，定时测量温度，通过REST API将数据发送给PC上的Web服务器。



arduino-esp8266-as-web-client-small.png

Web服务器采用Python Web框架Django开发，处理update操作的方法定义在`views.py`中，代码如下：

```
def update(request):
    a=request.GET["field1"]
    currTime=time.strftime("%Y-%m-%d %H:%M:%S",time.localtime(time.time()))
    cx = sqlite3.connect("d:/myprog/Arduino/lm35-esp8266-client/test.db")
    t=(float(a),currTime)
    cx.execute("INSERT INTO temp(tmpr,ts) VALUES (?,?)",t)
    cx.commit()
    cx.close()
    return HttpResponse("done")
```

在`urls.py`中，配置如下：

```
urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'^view/', 'webtest.views.view', name='view'),
]
```

Django Web默认监听在127.0.0.1:8000，为了让其他设备能访问web，需要使用命令`python manage.py runserver 0.0.0.0:8000`修改。

Arduino程序如下：

```
#include <SoftwareSerial.h>
#include <stdlib.h>
#include <PCD8544.h>

#define DEBUG true
```

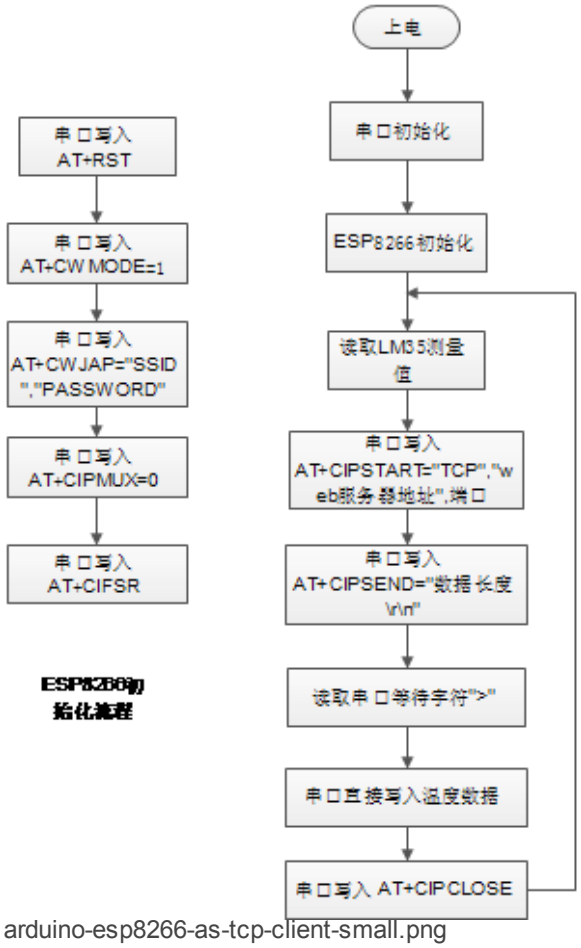
```
#define PIN_CE   8 //Pin 3 on LCD
#define PIN_RST 7 //Pin 4 on LCD
#define PIN_DC    6 //Pin 5 on LCD
#define PIN_DIN  5 //Pin 6 on LCD
#define PIN_CLK  4 //Pin 7 on LCD

int ledPin = 13;  // LED
int lm35Pin = 0;  // LM35 analog input

static const byte LCD_WIDTH = 84;
static const byte LCD_HEIGHT = 48;

static PCD8544 lcd(4, 5, 6, 7, 8);

SoftwareSerial ser(2, 3); // RX, TX

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);  // enable debug serial
  ser.begin(9600); // enable software serial

  // reset ESP8266
  sendData("AT+RST\r\n",5000,DEBUG); // reset module
  sendData("AT+CWMODE=1\r\n",2000,DEBUG); // configure as station
  sendData("AT+CWJAP=\"SSID\",\"PASSWORD\"\r\n",15000,DEBUG);
  sendData("AT+CIPMUX=0\r\n",2000,DEBUG); //single connection mode
  res = sendData("AT+CIFSR\r\n", 5000, DEBUG); // get ip address
  displayIPAddr(res); //display IP Address
}

//把ESP8266的IP地址显示在LCD上
void displayIPAddr(String s) {
  int p1 = s.indexOf("+CIFSR:STAIP,\"") + 14;
  char ip[16];
  int idx = 0;
  while (s[p1 + idx] != '"') {
    ip[idx] = s[p1 + idx];
    idx++;
  }
  ip[idx] = 0;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(ip);
}

String sendData(String command, const int timeout, boolean debug)
{
    String response = "";
    ser.print(command); // send the read character to the esp8266
    long int time = millis();
    while( (time+timeout) > millis())
    {
      while(ser.available())
      {
        char c = ser.read(); // read the next character.
        response+=c;
      }
```

```
    }
    if(debug)
    {
      Serial.print(response);
      lcd.clear();
      lcd.setCursor(0, 0);
      if (response.indexOf("OK") != -1)
          lcd.print(command + ": OK");
      else
          lcd.print(command + ": Failure");
    }
    return response;
}


void loop() {

  // read the value from LM35.read 10 values for averaging.
  int val = 0;
  for(int i = 0; i < 10; i++) {
      val += analogRead(lm35Pin);
      delay(500);
  }

  // 转换成温度数据
  float temp = val*50.0f/1023.0f;

  // convert to string
  char buf[16];
  String strTemp = dtostrf(temp, 4, 1, buf);

  Serial.println(strTemp);

  // 和Web服务器建立连接
  String cmd = "AT+CIPSTART=\"TCP\",\"192.168.199.174\",8000\r\n";
  sendData(cmd,5000,DEBUG);

  // 准备 GET string
  String getStr = "GET /update/?field1="+String(strTemp)+"\r\n\r\n";

  //发送数据长度
  cmd = "AT+CIPSEND=" + String(getStr.length()) + "\r\n";
  ser.print(cmd);

  if(ser.find(">")){
    //发送数据
    ser.print(getStr);
  }
  //关闭连接
  sendData("AT+CIPCLOSE\r\n",3000,DEBUG);

  delay(15000);
}
```

6.4 工作方式二

ESP8266工作为workstation模式，Arduino作为客户端，定时测量温度，通过TCP将数

据发送给PC上的TCP Server。



arduino-esp8266-as-tcp-client-small.png

Arduino程序的差别主要在于发送数据部分，仅列出不同的部分：

```
void loop() {

  // read the value from LM35.read 10 values for averaging.
  int val = 0;
  for(int i = 0; i < 10; i++) {
      val += analogRead(lm35Pin);
      delay(500);
  }

  // 转换成温度数据
  float temp = val*50.0f/1023.0f;

  // convert to string
  char buf[16];
  String strTemp = dtostrf(temp, 4, 1, buf);

  Serial.println(strTemp);

  // 和Web服务器建立连接
  String cmd = "AT+CIPSTART=\"TCP\",\"192.168.199.174\",9999\r\n";
  sendData(cmd,5000,DEBUG);

  // 准备要发送的数据
  String getStr = String(strTemp)+"\r\n\r\n";

  //发送数据长度
```

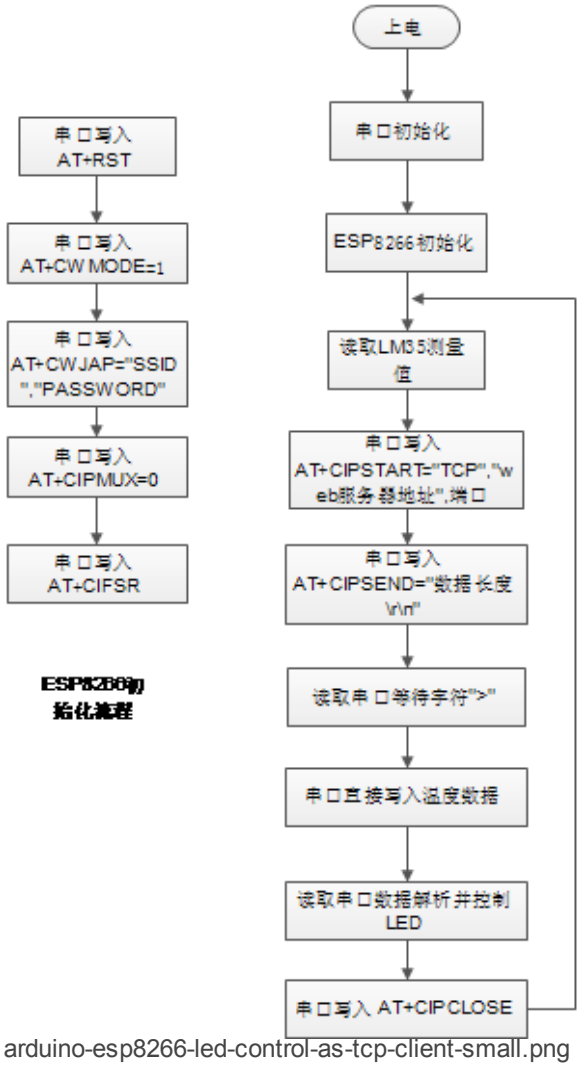
```
  cmd = "AT+CIPSEND=" + String(getStr.length()) + "\r\n";
  ser.print(cmd);

  if(ser.find(">")){
    //发送数据
    ser.print(getStr);
  }
  //关闭连接
  sendData("AT+CIPCLOSE\r\n",3000,DEBUG);

  delay(15000);
}
```

Python语言编写的TCP Server代码如下：

```
import SocketServer
import datetime

class MyTCPHandler(SocketServer.BaseRequestHandler):
    def handle(self):
        # self.request is the TCP socket connected to the client
        self.data = self.request.recv(1024).strip()
        print "ClientIP:", self.client_address[0]
        print "Rx: ", self.data
        s = str(datetime.datetime.now())
        print "Tx: ", s
        self.request.sendall( s )

if __name__ == "__main__":
    HOST, PORT = "192.168.199.174", 9999

    # Create the server, binding to localhost on port 9999
    server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)
    print
    print "Started: %s   Port: %d" % ( HOST, PORT )
    print "Server: ", server
    print

    # Activate the server; this will keep running until you
    # interrupt the program with Ctrl-C
    server.serve_forever()
```

6.5 工作方式三

ESP8266工作为workstation模式，Arduino作为客户端，定时测量温度，通过TCP将数据发送给PC上的TCP Server，并解析服务器回传的数据控制LED的亮灭。



arduino-esp8266-led-control-as-tcp-client-small.png

Arduino程序的loop部分

```
void loop() {

  // read the value from LM35.read 10 values for averaging.
  int val = 0;
  for(int i = 0; i < 10; i++) {
      val += analogRead(lm35Pin);
      delay(500);
  }

  // 转换成温度数据
  float temp = val*50.0f/1023.0f;

  // convert to string
  char buf[16];
  String strTemp = dtostrf(temp, 4, 1, buf);

  Serial.println(strTemp);

  // 和Web服务器建立连接
  String cmd = "AT+CIPSTART=\"TCP\",\"192.168.199.174\",9800\r\n";
  sendData(cmd,5000,DEBUG);

  // 准备要发送的数据
  String getStr = String(strTemp)+"\r\n\r\n";
```

```
  //发送数据长度
  cmd = "AT+CIPSEND=" + String(getStr.length()) + "\r\n";
  ser.print(cmd);

  if(ser.find(">")){
    String s = sendData(getStr, 1000, DEBUG);
    if (s.indexOf("#LED:ON#") != -1)
      toggleLed(1);
    else if (s.indexOf("#LED:OFF#") != -1)
      toggleLed(0);
    else
      Serial.println("bad command");
  }

  delay(15000);
}
```

Python程序如下： import SocketServer import sqlite3

```
class MyTCPHandler(SocketServer.BaseRequestHandler):
    def handle(self):
        # self.request is the TCP socket connected to the client
        self.data = self.request.recv(1024).strip()
        print "ClientIP:", self.client_address[0]
        print "Rx: ", self.data
        cx = sqlite3.connect("d:/myprog/Arduino/lm35-esp8266-client/test.db")
        cu = cx.cursor()
        cu.execute("SELECT status FROM led")
        row=cu.fetchone()
        x=row[0]
        cx.close()
        s="#LED:OFF#"
        if x==0:
            s="#LED:OFF#"
            print "Tx: ", s
        else:
            s="#LED:ON#"
            print "Tx: ", s
        self.request.sendall( s )

if __name__ == "__main__":
    HOST, PORT = "192.168.199.174", 9800

    # Create the server, binding to localhost on port 9800
    server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)
    print
    print "Started: %s   Port: %d" % ( HOST, PORT )
    print "Server: ", server
    print

    # Activate the server; this will keep running until you
    # interrupt the program with Ctrl-C
    server.serve_forever()
```

其中，LED的亮灭由数据库表led中的status决定。在Django应用中添加led控制器和一个led.html模板文件。在`views.py`中添加：

```
def led(request):
    s=request.GET["led"]
    cx = sqlite3.connect("d:/myprog/Arduino/lm35-esp8266-client/test.db")
    cx.execute("UPDATE led set status="+s)
    cx.commit()
    cx.close()
    return render(request,'led.html')
```

修改`urls.py`为：

```
urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'^update/', 'webtest.views.update', name='update'),
    url(r'^led/', 'webtest.views.led', name='led')
]
```

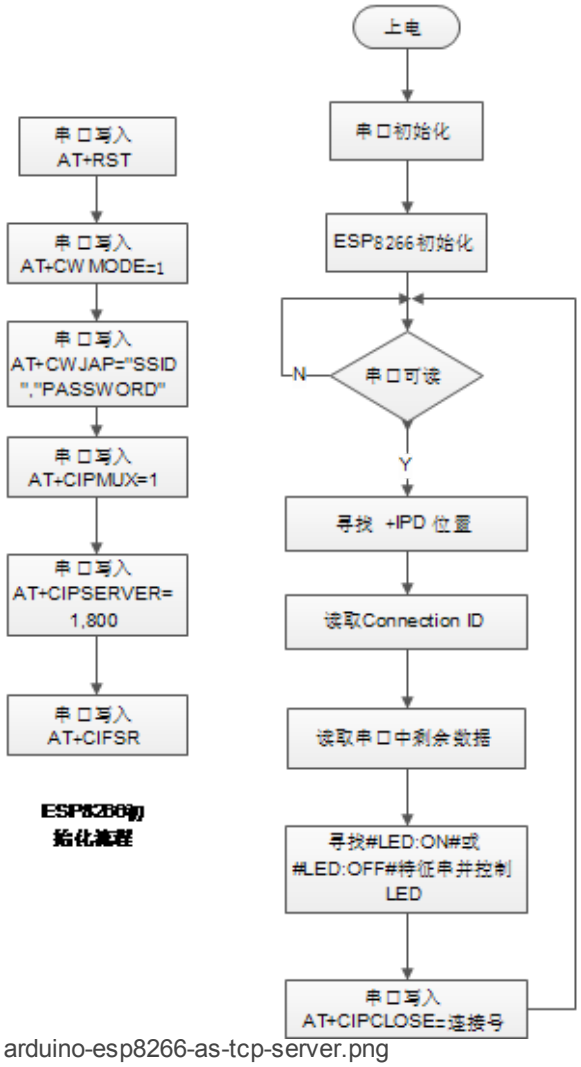
`led.html`内容如下：

```
<!DOCTYPE html>
{% load staticfiles %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
    <script src="{% static "js/jquery-1.9.1.min.js" %}"></script>
</head>
<body>
    <button id="ledon" class="led">Turn On</button> <!-- button for pin 11 -->
    <button id="ledoff" class="led">Turn Off</button> <!-- button for pin 12 -->

    <script type="text/javascript">
        $(document).ready(function(){
            $(".led").click(function(){
                var p = $(this).attr('id');
                if(p=="ledon")
                    $.get("/led/?led=1"); // execute get request
                else
                    $.get("/led/?led=0"); // execute get request
            });
        });
    </script>
</body>
</html>
```

其中，Django中静态文件的存放和位置请参考文后参考资料。

6.6 工作方式四

ESP8266工作为workstation模式，Arduino作为服务器，PC机上的客户端程序通过TCP向Arduino发送命令控制LED。



arduino-esp8266-as-tcp-server.png

ESP8266工作在Multiple connections模式，这种情况下必须得到连接客户端的id，在发送关闭连接的AT命令时格式为：`AT+CIPCLOSE=id`

Arduino程序在对ESP8266初始化时修改如下：

```
sendData("AT+RST\r\n", 5000, DEBUG); // reset module
sendData("AT+CWMODE=1\r\n", 2000, DEBUG); // configure as station
sendData("AT+CWJAP=\"HiWiFi_Free\",\"openusing\"\r\n", 15000, DEBUG); //connect to AP
sendData("AT+CIPMUX=1\r\n", 1000, DEBUG); // configure for multiple connections
sendData("AT+CIPSERVER=1,800\r\n", 1000, DEBUG); // turn on server on port 800
res = sendData("AT+CIFSR\r\n", 5000, DEBUG); // get ip address
displayIPAddr(res);
```

loop循环的内容如下：

```
void loop()
{
  if (esp8266.available()) // check if the esp is sending a message
  {
    if (esp8266.find("+IPD,"))
    {
      delay(1000);
      //获取connection id
      int connectionId = esp8266.read() - 48;

      String response = "";
      while (esp8266.available())
```

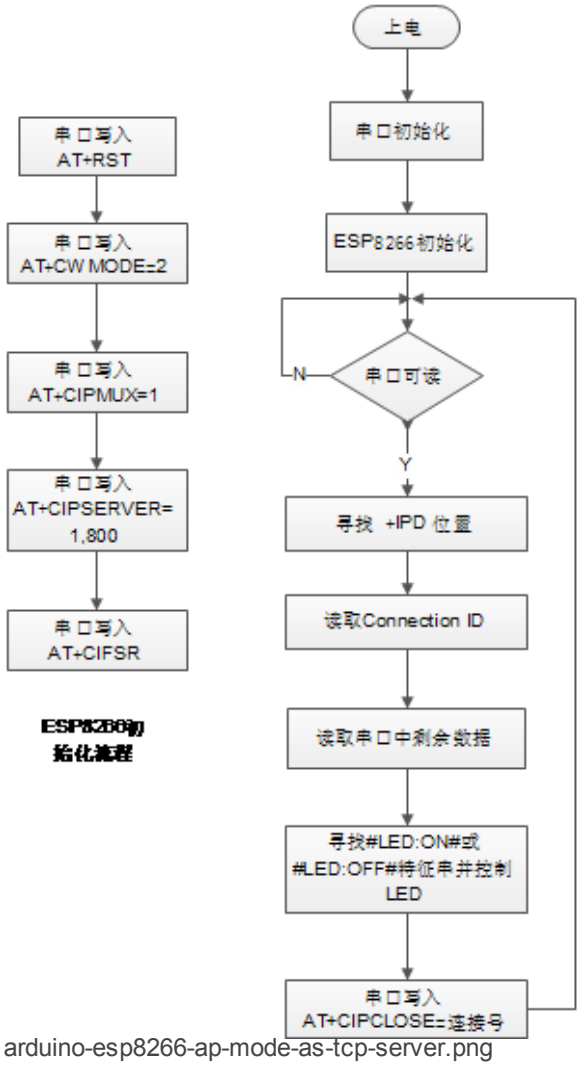
```
      {
        char c = esp8266.read();
        response += c;
      }

      if (response.indexOf("#LED:ON#") != -1)
        digitalWrite(13, HIGH);
      else if (response.indexOf("#LED:OFF#") != -1)
        digitalWrite(13, LOW);
      else
        Serial.println("Bad command");

      String closeCommand = "AT+CIPCLOSE=";
      closeCommand += connectionId;
      closeCommand += "\r\n";

      sendData(closeCommand, 1000, DEBUG); // close connection
    }
  }
  delay(1000);
}
```

PC端使用Python编写的TCP客户端程序如下：

```
import socket

HOST, PORT = "192.168.199.197", 800

# Create a socket (SOCK_STREAM means a TCP socket)
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    data=raw_input("command:")
    # Connect to server and send data

    sock.connect((HOST, PORT))
    sock.sendall(data + "\n")

    # Receive data from the server and shut down
    received = sock.recv(1024)
finally:
    sock.close()

print "Sent:     {}".format(data)
print "Received: {}".format(received)
```

6.7 工作方式五

ESP8266工作为AP模式，Arduino作为服务器，PC机上的客户端程序通过TCP向Arduino发送命令。



arduino-esp8266-ap-mode-as-tcp-server.png

Arduino程序在对ESP8266初始化时修改如下：

```
sendData("AT+RST\r\n", 5000, DEBUG); // reset module
sendData("AT+CWMODE=2\r\n", 2000, DEBUG); // configure as station
sendData("AT+CIPMUX=1\r\n", 1000, DEBUG); // configure for multiple connections
sendData("AT+CIPSERVER=1,800\r\n", 1000, DEBUG); // turn on server on port 800
res = sendData("AT+CIFSR\r\n", 5000, DEBUG); // get ip address
displayIPAddr(res);
```

PC必须连接到ESP8266建立的AP上获取IP地址，ESP8266默认的IP地址是192.168.4.1。将工作方式四的python程序中连接的服务器地址改为192.168.4.1即可进行测试。

当多个设备需要控制时，控制设备需要连接到不到的AP进行控制，这样是非常复杂的。因此AP模式不适合使用。

6.8 小结

通过案例，可以了解到：

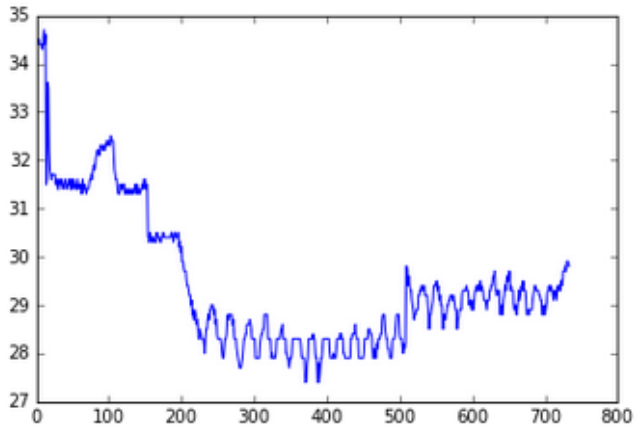
- 作为客户端工作时，不管是TCP还是Web客户端，均通过`AT+CIPSTART`指令建立连接，然后通过`AT+CIPSEND`向服务器发送数据，然后获取服务器返回的响应信息进行后续处理，最后通过`AT+CIPCLOSE`关闭连接。
- 作为服务器工作时，在特定端口进行监听之后，循环读取串行口数据并分析处理，最后通过`AT+CIPCLOSE`关闭连接。

7 使用python进行简单数据分析

建议安装整合了NumPy,SciPy等的Anaconda安装包。在命令行上输入：`ipython notebook`启动Ipython，在浏览器界面上新建Python输入代码并运行。

```
import sqlite3
from pylab import *
%pylab inline
cx = sqlite3.connect("d:/myprog/Arduino/lm35-esp8266-client/test.db")
cu = cx.cursor()
cu.execute("select tmpr from temp order by ts asc")
rows=cu.fetchall()
for r in rows:
    print r[0],
t=list(rows)
plot(t)
show()
```

运行后在浏览器中显示下图：



tmpr-fig.png

也可在Django项目中添加绘图模块，在`views.py`中添加：

```
def view(request):
    cx = sqlite3.connect("d:/myprog/Arduino/lm35-esp8266-client/test.db")
    cu = cx.cursor()
    cu.execute("select tmpr from temp order by ts asc")
    rows=cu.fetchall()
    t=list(rows)
    plot(t)
    savefig("d:/myprog/python/webtest/webtest/static/images/result.png")
    return render(request,'data.html')
```

在模板文件夹中添加`data.html`，内容如下：

```
<!DOCTYPE html>
{% load staticfiles %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
    <img src="{% static "images/result.png" %}"/>
</body>
```

```
</html>
```

修改`urls.py`如下：

```
urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'^update/', 'webtest.views.update', name='update'),
    url(r'^view/', 'webtest.views.view', name='view'),
    url(r'^led/', 'webtest.views.led', name='led')
]
```

运行Django项目后，在浏览器中打开*http://localhost:8000/view/*即可看到绘制的曲线。

参考文献

- Django基础教程
- 自造你的Arduino-UNO板
- Serial-to-WiFi Tutorial using ESP8266
- Arduino WiFi Control with ESP8266 Module
- Easy ESP8266 WiFi Debugging with Python
- ESP8266 Arduino LED Control (Control The Digital Pins Via WiFi, Send Data From Webpage to Arduino)
- ESP8266 Serial WIFI Module
- diy-layout-creator
- Nokia 5110液晶显示模块的使用
- IPython Notebook: 交互计算新时代
- http://ipython.org/index.html
- 用Python进行SQLite数据库操作