ESP13 shield (ESP 2866) WFI

**From:** Zhao, Haijun   
**Sent:** Saturday, May 14, 2016 7:49 PM  
**To:** Zhao, Haijun <haijunz@qti.qualcomm.com>  
**Subject:** ESP13 shield (ESP 2866) WFI

Arduion uno R3        ESP13 (UART1)

 TX                                TX

  RX                               RX

 5v                                  5v

GND                              G

<https://fineshang.gitbooks.io/esp8266-based-serial-wifi-shield-for-arduino-user/content/chapter4.html>

void setup() {  
  // put your setup code here, to run once:  
Serial.begin(9600);  
}  
  
void loop() {  
delay(1000);  
Serial.println("ESP8266 WiFi."); //output the serial data  
}

root@haijunz-ThinkPad-T420:/home/haijunz/tty# more server.py  
import socket  
  
import sys  
  
from thread import \*  
  
   
  
HOST = ''   # Symbolic name meaning all available interfaces  
  
PORT = 6533 # Arbitrary non-privileged port  
  
   
  
s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
print 'Socket created'  
  
   
  
#Bind socket to local host and port  
  
try:  
  
    s.bind((HOST, PORT))  
  
except socket.error as msg:  
  
    print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]  
  
    sys.exit()  
  
       
  
print 'Socket bind complete'  
  
   
  
#Start listening on socket  
  
s.listen(4)  
  
print 'Socket now listening'  
  
   
  
#Function for handling connections. This will be used to create threads  
  
def clientthread(conn):  
  
    #print 'receive#####clientthrea......d'  
  
    #infinite loop so that function do not terminate and thread do not end.  
  
    while True:  
  
           
  
        #Receiving from client  
   
    #print '====(1)========'  
  
        data = conn.recv(1024)  
  
        print '====(2)========'+data   
  
  
        if not data:   
            print '====not data========'  
            break  
  
    #came out of loop  
  
    conn.close()  
  
   
  
#now keep talking with the client  
  
while 1:  
  
    #wait to accept a connection - blocking call  
  
    conn, addr = s.accept()  
  
    print 'Connected with ' + addr[0] + ':' + str(addr[1])  
  
       
  
    #start new thread takes 1st argument as a function name to be run, second is the tuple of arguments to the function.  
  
    start\_new\_thread(clientthread ,(conn,))  
   
  
s.close()

