```python
#!/usr/bin/python

import socket
import sys
import datetime
import time
import os
import re
import pickle
import json

#Need to import Temperature Sensor and Distance Sensor Data eventually
#Probably Motor as well if possible
#Will also need to incoroporate some other libraries to consider
#other items such as joystick, etc.

UDP_IP = '127.0.0.1'
global UDP_PORT
UDP_PORT = 2345
BUFF_SIZE = 1024

def distanceData():
  distance = board.getDistance()
  print_distance(distance)
  #Delay time < 0.6s
  time.sleep(0.3)

def temperatureData():
  temperature = sensor.get_temperature()
  print("The temp is %s celcius" % temperature)
  time.sleep(0.3)

def print_distance(dis):
  if board.last_operate_status == board.STA_OK:
    print("Distance %d mm" %dis)
  elif board.last_operate_status == board.STA_ERR_CHECKSUM:
    print("ERROR")
  elif board.last_operate_status == board.STA_ERR_SERIAL:
    print("Serial open failed!")
  elif board.last_operate_status == board.STA_ERR_CHECK_OUT_LIMIT:
    print("Above the upper limit: %d" %dis)
  elif board.last_operate_status == board.STA_ERR_CHECK_LOW_LIMIT:
```

```
43          print("Below the lower limit: %d" %dis)
44        elif board.last_operate_status == board.STA_ERR_DATA:
45          print("No data!")
46
47      def send_response(response, sock, destination) :
48          msg = bytes(response, 'utf-8')
49          sock.sendto(msg, destination);
50
51      def proc_request(cmd, sock, requester) :
52          #convert the cmd to a string
53          #sensorData = sock
54          cmd = bytes.decode(cmd, 'utf-8')
55          now = datetime.datetime.now()
56          #sensorDistance = {'distance' : distanceData()}
57          print(now, "Processing: " + cmd)
58          cmd = cmd.split()
59          if cmd[0] == "test":
60              print("This is a test print to let you know that the server was established")
61              send_response("Test sent", sock, requester)
62          elif cmd[0] == "run":
63              while True:
64                  print("WALL-C Activated")
65                  #sensorDistance = {'distance' : distanceData()}
66                  sock.sendto(json.dumps(distanceData()).encode('utf-8'), requester)
67                  sock.sendto(json.dumps(temperatureData()).encode('utf-8'), requester)
68          elif cmd[0] == "exit":
69              send_response("Server Exited", sock, requester)
70          else:
71              send_response("Data Not Sent", sock, requester)
72              #sensorTemp = {'temperature' : temperatureData()}
73              #sock.sendto(json.dumps(distanceData()).encode('utf-8'), (UDP_IP, UDP_PORT))
74              #sock.sendto(json.dumps(sensorTemp).encode('utf-8'), requester)
75
76      from DFRobot_RaspberryPi_A02YYUW import DFRobot_A02_Distance as Board
77      from w1thermsensor import W1ThermSensor
78
79      sys.path.append(os.path.dirname(os.path.dirname(os.path.realpath(__file__))))
80
81      sensor = W1ThermSensor()
82      board = Board()
83
84      if __name__ == '__main__':
```

```
85      if len(sys.argv) == 2:
86          UDP_PORT = int(sys.argv[1])
87
88      print ("UDP target IP:", UDP_IP)
89      print ("UDP targer Port:", UDP_PORT)
90
91      dis_min = 0;
92      dis_max = 4500;
93
94      board.set_dis_range(dis_min, dis_max)
95
96      sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #Internet UDP
97      sock.bind((UDP_IP, UDP_PORT))
98
99      while True:
100         data, addr = sock.recvfrom(BUFF_SIZE)
101         proc_request(data, sock, addr)
102
```