# 多人体实时姿态监控系统详细设计说明书

[多人体实时姿态监控系统详细设计说明书 1](#_Toc24099148)

[1．引言 1](#_Toc24099149)

[1.1编写目的 1](#_Toc24099150)

[2．总体设计 2](#_Toc24099151)

[2.1需求概述 2](#_Toc24099152)

[2.2软件结构 2](#_Toc24099153)

[3．程序描述 2](#_Toc24099154)

[3.1功能 3](#_Toc24099155)

[3.2程序代码 3](#_Toc24099156)

[3.2.1 ReceiveData.py 3](#_Toc24099157)

[3.2.2 Get\_Data.py 4](#_Toc24099158)

[3.2.3 Web\_Manage.py 6](#_Toc24099159)

[3.2.4 Merge.html 8](#_Toc24099160)

[4.启动方式及界面展示 11](#_Toc24099161)

[4.1启动方式 11](#_Toc24099162)

[4.2界面展示 12](#_Toc24099163)

# 1．引言

## 1.1编写目的

本文档介绍了多人体姿态监控系统的软件开发细节，实现了服务器端接收多人体模块数据帧，通过socket传到服务器运行的web后端对数据帧进行解析后得到姿态数据，并且进行姿态判别后，查询数据库比对人名成功后显示到前端页面，前端页面能够根据网关号和设备号区分不同属网关下的不同个体。并能够对时间，人名，设备号进行一定的设置。如果出现异常 能够实时报警。软件部分分为数据接受模块，WEB前后端模块。

# 2．总体设计

## 2.1需求概述

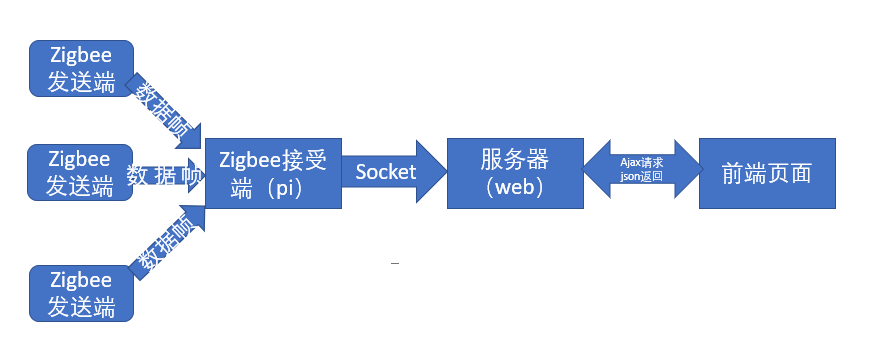
1.数据接收模块能够接受由不同网关发来的消息，并能够对数据进行解析得到具体的姿态数据值。通过socket发送给服务器正在运行的web后端。

2.web后端能够对socket接受的数据帧进行解析，并通过实时姿态数据判别姿态。

3.前端能够设置检测时间，能够设置人名和设备号以创建新人员，并且能删除人员。

## 2.2软件结构

树莓派上的ReceiveData.py通过zigbee串口模块接收各个IMU传感器数据，之后以tcp网络通信的方式传到web后端，web后端广播监听设备所对应的端口号的数据，并对接受到的数据帧进行解析，得到最终的姿态数据。前端增加/删除人员根据请求访问后端数据库，若前端添加新人员，后端有所对应设备号的设备，则在前端进行检测。



数据传输模型

# 3．程序描述

ReceiveData.py: 负责通过串口读取zigbee的数据并通过socket转发给服务器。

Web\_manage.py: 一方面负责接受socket传来的数据，另一方面负责与前端交互，相应前端请求，负责数据库增删改查。

Get\_Data.py: 负责接受socket传来的数据帧并解析得到姿态数据，并将姿态数据压栈。

Merge.html: 负责前端的展示，可以实现增加/删除人员，设置时间。

## 3.1功能

1.接收数据模块能够实现接受串口数据，并通过socket 将数据发送给服务器上运行的web后端。

2.web后端能够实现对收到的数据帧进行解析，对姿态进行判别。并能够对前端请求的增加/删除人员的请求做出响应。

3.web前端能够显示当前的人数，并且能够设置检测时间段，有显著标志进行增加/删除人员。

## 3.2程序代码

### 3.2.1 ReceiveData.py

负责通过串口读取zigbee的数据并通过socket转发给服务器

#!/usr/bin/python3

# -\*- coding: UTF-8 -\*-

# 引入必要的module

import serial

import socket

import pickle

import time

def main():

# read serial

x = serial.Serial('COM8', 115200) #设置串口号，波特率

r = []

num = 0

num1 = 0

length = 0

j = 0

s = socket.socket()

host='192.168.43.178' # 服务器的ip

port = 10040 # 所对应的端口

s.connect((host, port))

while True:

str\_read = x.read()

i = ['%02x' % b for b in str\_read]

print(i)

if i[0] == 'aa' and num == 0:

num = num + 1

continue

elif i[0] == 'aa' and num == 1: # 数据帧头为 AA AA

num = num + 1

continue

if num >= 2 and num < 15: # 数据帧的长度为14

r.extend(i)

num = num + 1

if len(r) == 12:

print('ii', r)

r.extend(["s1"])

a = pickle.dumps(r)

s.send(a) # 发送到服务端

if num == 14:

num = 0

num1 = 0

r = []

s.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()

### 3.2.2 Get\_Data.py

import socket

import threading

import pickle

from collections import deque

class Getdata():

def \_\_init\_\_(self):

self.receivedata = 0

self.sensor\_data\_1 = 0

self.sensor\_data\_2 = 0

self.web\_sensors\_data\_1\_stack = deque(maxlen = 3)

self.web\_sensors\_data\_2\_stack = deque(maxlen = 3)

def tcp\_server(self):

s = socket.socket()

host = '0.0.0.0'

port = 10040

s.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

s.bind((host, port))

s.listen(5)

while True:

sock, addr = s.accept()

print('tcp client addr: ', addr)

t = threading.Thread(target=self.tcplink, args=(sock, addr))

t.start()

def tcp\_link(self, sock, addr):

print('Accept new connection from %s:%s...' % addr)

global predictions\_data

# set predictions\_data = 4, represent connect

predictions\_data = 4

global flag

flag = 0

while True:

client\_data = sock.recv(1024)

if not client\_data:

print('disconnect')

predictions\_data = 5

break

try:

self.receivedata = pickle.loads(client\_data)

#print(self.receivedata)

except:

print('pickle.loads error')

continue

self.sensor\_data\_1, self.sensor\_data\_2 = self.slice\_data(self.receivedata)

self.web\_sensors\_data\_1\_stack.appendleft(self.sensor\_data\_1)

#if len(self.sensor\_data\_2)!=0:

self.web\_sensors\_data\_2\_stack.appendleft(self.sensor\_data\_2)

sock.close()

print('Connection from %s:%s closed.' % addr)

def slice\_data(self, data):

sensor\_data\_1=[]

sensor\_data\_2=[]

if data[1] == "00":

roll = int(data[6] + data[7], 16)/100

pitch = 0 - int(data[8] + data[9], 16)/100

sensor\_data\_1.append("0")

sensor\_data\_1.append(roll)

sensor\_data\_1.append(pitch)

if pitch >= -100:

sensor\_data\_1.append("Innormal")

else:

sensor\_data\_1.append("Normal")

elif data[1] == "01":

roll = int(data[6] + data[7], 16)/100

pitch = 0 - int(data[8] + data[9], 16)/100

sensor\_data\_2.append("1")

sensor\_data\_2.append(roll)

sensor\_data\_2.append(pitch)

if pitch >= -180:

sensor\_data\_2.append("Innormal")

else:

sensor\_data\_2.append("Normal")

return sensor\_data\_1, sensor\_data\_2

def run(self):

thread1 = threading.Thread(target=self.tcp\_server, args=())

thread1.start()

get\_data = Getdata()

### 3.2.3 Web\_Manage.py

一方面负责接受socket传来的数据，另一方面负责与前端交互，相应前端请求，负责数据库增删改查。

# -\*- coding:utf-8 -\*-

import json

import re

import numpy

from flask import Flask, request, render\_template, jsonify

import sqlite3

from getdata import get\_data

'''conn = sqlite3.connect('test.db')

print ("Opened database successfully")

#读取数据库

cursor = conn.execute("SELECT id, name from PATIENT")

personList = []

count = 0

for row in cursor:

pp=[str(count),str(row[0])+row[1]]

print ("ID = ", row[0])

print ("NAME = ", row[1])

count=count+1

personList.append(pp)

personList.append(['number',count])

personList=dict(personList)

print(personList)'''

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return render\_template('merge.html')

@app.route('/api/', methods=['GET', 'POST'])

def api():#读取数据库并且将病人信息数据传递到前端

if request.method == 'POST':

data = request.form['data']

nam=data[:-1]#最后一位是id

idd=int(data[-1])

print(nam,idd)

conn = sqlite3.connect('test.db')

print("Opened database successfully")

conn.execute("INSERT INTO PATIENT (ID,NAME) \

VALUES ({},'{}')".format(int(idd),nam));

conn.commit()

#获取前端的病人姓名与id信息，将其记录到数据库中

else:

try:

conn = sqlite3.connect('test.db')#数据库第一列是id，第二列为姓名

print("Opened database successfully")

cursor = conn.execute("SELECT id, name from PATIENT")

data\_pack = []#用于将id与姓名打包成键值，并按顺序赋予数字键名

count = 0#数字键名

for row in cursor:

item = [str(count), str(row[0]) + row[1]]

count = count + 1

data\_pack.append(item)

data\_pack.append(['number', count])

data\_pack = dict(data\_pack)

data= str(data\_pack)

except IndexError:

pass

return jsonify({"data":json.loads(data.replace("'", '"'))})

#获取前端传递过来的病人id，并按id删除病人数据

@app.route('/api2/', methods=['GET', 'POST'])

def api2():

if request.method == 'POST':

data = request.form['data']

iddd=int(data[-1])

conn = sqlite3.connect('test.db')

conn.execute("DELETE from PATIENT where ID={};".format(iddd))

conn.commit()

if re.match('[0-9a-fA-F]\*$', data):

socket\_server.response\_data.appendleft(data.encode())

return jsonify({'result': 1})

else:

return jsonify({'result': 0})

@app.route('/api3/', methods=['GET', 'POST'])

def api3():

data0 = {"device": "s1", "id":"1", "roll": 0.0, "pitch":0.0, "status":"Normal"}

if request.method == "GET":

id = request.form['data']

print("id is ", id)

if id == "0":

data0 = {"id":"0", "status":"Normal"}

elif id == "1":

data0 = {"id":"1", "status":"Normal"}

print(data0)

data0 = str(data0)

return jsonify({"data":json.loads(data0.replace("'", '"'))})

elif request.method == "POST":

data1 = get\_data.web\_sensors\_data\_1\_stack.pop()

data2 = get\_data.web\_sensors\_data\_2\_stack.pop() #从数据库中拿出病人的数据

print(data1)

print(data2)

if len(data1)!= 0:

data0["id"] = data1[0]

data0["roll"] = data1[1]

data0["pitch"] = data1[2]

data0["status"] = data1[3]

if len(data2) != 0:

data0["id"] = data2[0]

data0["roll"] = data2[1]

data0["pitch"] = data2[2]

data0["status"] = data2[3]

print(data0)

data0 = str(data0)

return jsonify({"data":json.loads(data0.replace("'", '"'))})

if \_\_name\_\_ == '\_\_main\_\_':

get\_data.run()

app.run(host='0.0.0.0', debug=False, use\_reloader=False)

### 3.2.4 Merge.html

负责前端的展示，可以实现增加/删除人员，设置时间。

#### （1）更新函数

function update(){

$.ajax({

type: "GET",

url: "/api/",

success:function (res){

console.log(res.data)

var id="";//用于临时存放id与姓名

var i=0;

var num = res.data['number'];

number=num;

for(;i<num;i++){

id=res.data[String(i)].slice(0,1);

$.ajax({

type: "POST",

url: "/api3/",

data: {'data': id},

success:function (res) {

console.log(res.data)

new\_status = res.data["status"];//传过来的状态信息

console.log(new\_status)

updateID = res.data["id"];//设备号ID

eleID = "People"+updateID;//确认需要修改的病人信息id

var personStatus = document.getElementById(eleID);

console.log(personStatus.innerHTML);

personStatus.innerHTML = new\_status;

}

});

}

}

});

}

//边框闪烁

function update1(){

$.ajax({

type: "POST",

url: "/api3/",

success:function (res) {

console.log(res.data)

id = res.data["id"]

pitch = res.data["pitch"]

status = res.data["status"]

eleID = "People"+id;//确认需要修改的病人信息id

console.log(eleID)

var personStatus = document.getElementById(eleID);

if (display\_status == true)

personStatus.innerHTML = "<p>" + String(status);

else

personStatus.innerHTML = "<p>" + String(pitch);

console.log(personStatus.innerHTML);

}

});

}

function update2(){

display\_status = !display\_status;

}

#### （2）判断函数

function check() {

var j=0;

for(;j<number;j++) {

var status\_info = document.getElementsByClassName("people\_pose")[j + 1].textContent; # 获得状态信息

console.log(status\_info);

if (status\_info == "Innormal") {

var mm = document.getElementsByClassName("card")[j + 1];

blink(mm); //异常边框闪烁

plysond();//放预警声音信号

}

if(status\_info == "Normal"){

var mm = document.getElementsByClassName("card")[j + 1];

mm.className="card border-success mb-3";

}//恢复正常显示

}

}

#### （3）增加及删除功能函数

function addDiv() {

var str = "请输入姓名";

var str2 = "请输入ID";

var name = prompt(str,"Z.chao");

if(name==null) return;

var ID = prompt(str2,"1");

if(ID==null) return;

var info = name+ID;

console.log(info);

$.ajax({

type: 'POST',

url: '/api/',

data: {'data': info},

success: function (res) {

alert("添加成功！");

}

})

window.location.reload();

}

function remove(element) {

var del\_id = element.id;

$.ajax({

type: 'POST',

url: '/api2/',

data: {'data': del\_id},

success: function (res) {

alert("成功！");

}

})

window.location.reload();

}

# 4.启动方式及界面展示

### 4.1启动方式

（1）启动硬件，打开模块电池开关即可，并将zigbee接收端的串口插到树莓派上，树莓派启动ReceiveData.py发送给服务器。

（2）在服务端启动Get\_Data.py,并在/Server/web目录下启动web后端服务

python Get\_Data.py

python web\_manage.py

（3）访问web前端页面

服务器ip：port（在浏览器中键入）

### 4.2界面展示

