**大连理工大学本科毕业设计（论文）**

**基于CSI数据的实时可视化系统**

**The Real-time Visualization System Based on CSI**

学 院（系）： 软件学院

专 业： 网络工程

学 生 姓 名： 张人杰

学 号： 201492027

指 导 教 师： 王雷

评 阅 教 师： 覃振权

完 成 日 期： 2018.6.15

大连理工大学

Dalian University of Technology

摘 要

无线信号室内定位现正在变得越来越受到关注。除了红外线，声呐，超宽带等方式，无线电波，特别是Wi-Fi信号在该领域的应用受到了普适计算和无线定位领域的双重关注。在这其中除了传统的RSSI, TDOA, AOA等方法外，近年来提出的利用CSI信息来分析多径传输特性，进而得到亚米级的定位精度的方法尤其受到人们的关注。在该项研究火热的现在，在实验过程中时而出现测试实验样本是否有效，需要了解环境信道状态等情况，而由于目前CSI采集所需的设备还是设置需要时间，试验场地的准备需要成本，把数据存储下来再显示会拉长实验-验证结论-修改这个循环的周期长度。因此一款能够实时显示CSI数据的应用对于这种场景是必要的。

我们基于Intel 5300网卡的基础上，借鉴CSI-Tool工程的实现，完成了matlab和Python双平台的CSI实时可视化系统的开发。该系统已经在我们的室内定位相关的论文和研究中经常使用，运行良好。本文主要是关于该系统实现的背景，设计与实现细节相关的。

**关键词：无线信号室内定位，信道状态信息，实时可视化系统**

**The Real-time Visualization System Based on CSI**

Abstract

Wireless Signal Indoor Localization has been focused on more and more. Except for infrared ray, sonar, ultra-wide band and other methods, wireless radio, especially Wi-Fi Signal’s applications in this field has drawn double attention from both Pervasive Computing and Wireless Localization. Within this area, except for traditional methods like RSSI, TDOA, AOA and so on, a new method put forward in recent decades, which makes use of CSI ( Channel State Information) to analyze multipath effects and so that it can reach a sub-meter level resolution, has caught people’s eyes much. When this research is hot now, there are often some scenes such as testing the validity of experimental samples, getting to know about the channel state of the current environment and so on. However, by now it still need time to set the devices in charge of collecting CSI data, and the preparation of experimental environment still cost much, and as a result the circle of experimenting-confirming the assumption-modifying processes will be longer if we choose to save the data and plot them. So it is necessary for the scenes above to develop an application to visualize CSI data in real-time.

We finish the development of the Real-time Visualization System Based on CSI-Tool and Intel 5300 NIC. This system has been adopted many times in our researches and papers on wireless localization, and it actually works well. This paper is talking about what’s behind it and how we design and implement it in detail.

**Key Words：Wireless Signal Indoor Localization；Channel State Information (CSI)；Real-time Visualization System**

目 录

[摘 要 I](#_Toc517247109)

[Abstract 2](#_Toc517247110)

[1 引言 4](#_Toc517247111)

[2 背景与相关工作 5](#_Toc517247112)

[2.1 背景 5](#_Toc517247113)

[2.1.1 RSSI 6](#_Toc517247114)

[2.1.2 TOA 和TDOA 8](#_Toc517247115)

[2.1.3 AOA 9](#_Toc517247116)

[2.1.4 CSI 10](#_Toc517247117)

[2.2 相关工作 12](#_Toc517247118)

[2.2.1 相关工具 12](#_Toc517247119)

[2.2.2 相似应用考察 13](#_Toc517247120)

[3 项目设计 14](#_Toc517247121)

[3.1 需求分析 14](#_Toc517247122)

[3.2 总体设计 15](#_Toc517247123)

[4 实现与效果 16](#_Toc517247124)

[4.1 发送端模块（包括CSI-Tool自带驱动部分） 16](#_Toc517247125)

[4.2 接收端数据解析模块 19](#_Toc517247126)

[4.2.1 matlab端 21](#_Toc517247127)

[4.2.2 Python端 25](#_Toc517247128)

[4.3 接收端绘图模块 32](#_Toc517247129)

[4.3.1 matlab端 32](#_Toc517247130)

[4.3.2 Python端 35](#_Toc517247131)

[4.4 系统效果 38](#_Toc517247132)

[5 总结及不足之处 40](#_Toc517247133)

[致 谢 42](#_Toc517247134)

[参考文献 43](#_Toc517247135)

1 引言

无线通讯早在几年前就开始普及：蓝牙，Zigbee以及Wi-Fi都在现代有着广泛的用途。尤其是Wi-Fi（Wireless Fidelity, IEEE 802.11b），如果你曾用手机无线接入过互联网的话你一定对这个词不陌生。换而言之，Wi-Fi信号的普及性有目共睹。

然而，Wi-Fi信号不仅可以传输信息，其信号的变化本身就可以作为一种反映发送端或接收端物理姿态变化的依据。近年来，国内外很多研究者都在试图使用Wi-Fi信号来进行室内或室外定位，姿态识别，穿墙透视[1]等工作。这可能会让人想到声纳，红外感应以及雷达等传统定位设备，但是需要承认，这些传统方法大多需要较为不菲的接受或发送设备以及分析仪器，相比之下，Wi-Fi的设备要求和普通用户的基本通讯要求一致，移动设备的大量普及更是大大降低了设备的成本，于是用Wi-Fi特别是普通的2.4GWi-Fi进行定位吸引了工业界和军方的注意。

近年来很多不同的而有效方案得到业界认可，各自也有着各自的适用场景和局限性，其中很多方案都利用了正交频分复用(OFDM)下的Wi-Fi子载波信息进行分析，这些信息被囊括进一个包，叫做信道状态信息(CSI, Channel State Information)。因为要利用Wi-Fi信号来做定位等利用信号变化来进行分析的工作，普通被解调过的Wi-Fi信号大部分情况并不是需要的，而是接收端直接接受到的原始信号强度相关信息。无论是接收端或发送端是固定还是移动的，被检测对象是携带发送端还是不携带设备(device-free)，无线信号的非内容相关信息都几乎是必须的。

通过CSI进行分析的方案多种多样，有rssi(Received Signal Strength Indicator，接受信号强度指数)，aoa(到达角度差)，tdoa(到达时间差)等。但实际上，CSI信号的稳定性可能会因为环境和设备的限制而受到很大影响，所以实验环境的细微偶发状况导致实验数据不够理想作废的情况屡屡发生。此外，对设备进行了改进，为了验证采到的CSI的情况变化而不停的存数据，分析数据，作废再重复的过程是很花时间的。再次，如果希望能够看到实时的实验环境变化对CSI信号的影响，将实验环境的变化记录下再和存下来分析后的CSI数据对照，一花时间二精度怎么也没有能够实时显示CSI和其相关参数的系统来的直观精确。以上的问题都催生着一种能够尽量实时显示接收端CSI数据和相关参数的工具的产生。基于这样原因，本文主要将介绍一种能够实时显示CSI数据及相关参数的工具型应用，主要运行在pc端，语言主要是C，Matlab(脚本型语言)和Python（编程类语言），没有操作系统要求。

本文除本节外，将这样组织：

第1节将介绍CSI相关的背景知识，以及CSI实时绘图的现有相关工作。

第2节将介绍我们项目的需求分析，总体设计，详细设计及实现相关细节。

第3节描述一下我们项目的运行环境以及运行效果。

第4节将对我们的工作进行总结，并且对我们工作的不足之处和未来计划做个大致的阐述。

2 背景与相关工作

2.1 背景

本文主要介绍了一种用于Wi-Fi的CSI信号实时显示的研究用工具，为了能够更好的解释这个工具的用途，有必要对室内定位相关，CSI是什么和CSI的用途等做一个大致的介绍。

室内定位是一个相对比较新的方向，尽管早在上世纪90年代就有陆续研究投入使用，但是获得商品化还是在近十年。在室外，人们可以可以用GPS定位来查找位置，导航等，这些地理位置信息的精度一般是米级别的，但是在室内，GPS是访问不到的，并且室内的定位精度一般要求很高，大到房间位置小到室内物品，这就要求新的手段与方法，因此室内定位的需求是迫切的而实际存在的。近20年来，无线室内定位的相关研究在各个方向取得突破，利用声音信号和超声波，红外线信号，RFID（射频识别），蓝牙，蜂窝，Zigbee, Wi-Fi, UWB（超宽带技术）等测量载体，运用各种算法来实现定位。这些不同的方法都是利用了发射信号的空间特征，分析室内物体或人的运动或位置来进行定位的。

Wi-Fi（无线局域网，IEEE 802.11b）已经融入了人类日常社会生活的一部分，在公共场合和家庭中广泛接受并应用，并且只需要手机，笔记本电脑等设备就可以接受或发射信号。与此同时，随着移动计算和普适计算的发展与繁荣，能够运行在移动端或小型嵌入式系统的室内定位系统自然受到青睐，离这个目标最近的之一自然就是Wi-Fi室内定位。所谓普适计算（Pervasive Computing），就是指1999年IBM第一次提出的[2]，即随时随地都可以用设备得到自己想要的信息的概念。轻量型计算是普适计算的核心之一，因为需要复杂的计算功能尽可能在硬件资源有限的个人设备上实现，也有着尽量降低应用对于硬件的要求的意义。Wi-Fi用于室内定位相对其他有几种方法来说成本低，且一旦有需要便可以很容易的在普通设备上投入使用。利用Wi-Fi进行室内定位的方法有几种，下面将简要介绍和本文相关的无线室内定位研究相关的几种方法。

2.1.1 RSSI

RSSI(Received Signal Strength Indicator)即接收信号强度指数，反映的是信号强度，这一信息可以直接从接收信号通过计算可得[3]。RSSI的理论计算方法如下：

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

其中是瞬时值，I,Q分别表示同向调制，正交调制的信号功率。若一秒采得M个RSSI瞬时值，则1秒的RSSI平均值是：

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

但实际信号是时域采样所得的数据，因此实际上可以利用采样信息直接求平均值。

事实上，RSSI作为信号强度，和另外一个常用的dBm（分贝毫瓦）单位是可以相互转换的，dBm的计算公式为：

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

RSSI并非是写在802.11帧里的字段，而是可以由厂商自定义，这也使得RSSI的数值可能在不同厂商设备中表达的意义不相等，因此很多厂商也提供了其他的信号强度指标。此外，RSSI还可以有收到的CSI数据进行计算所得。

RSSI除了作为信号强度的指标之外，在无线信号室内定位领域也是非常重要的指标。有两种常用的定位方法：几何位置定位和指纹定位。

几何位置定位可以利用RSSI来做。RSSI反映的是信号强度，通过所谓的路径衰减模型可以反映信号源的距离。如果在不知道信号方向的情况下可以用三角测量等方法确定位置。即再使用两个AP，通过其RSSI算出距离，再根据已知的三个AP之间距离，用相应公式（如正弦定理和余弦定理）来计算信号源的相对位置。但是利用RSSI来进行这样的测量存在问题，主要是关于多径效应。

多径效应（multipath effect）是指电磁波（包括无线电波）在空间中以多条不相同的路径进行传播后，各条路径下电磁波不同时到达接收端，而是相互有相位差的情况下，多条路径上在接收端这一点各分量进行干涉等信号干扰现象，导致原来要传送的信息损坏，信号失真或有错误[4]。举例来说，如果Wi-Fi信号沿着两条不同的路径传播，这两条路径一条是视距(Line-of-Sight)即直线传播，另一条经地面反射一次传播，而这两条路径正好相差为波长的一半，那么在接收端，以不同路径传输的两个信号必然发生相消干涉，这会导致信号衰减到几乎收不到，信息也必然失真严重。多径效应对于数字通信，雷达监测等应用有着十分严重的影响。多径效应是无线信号衰减的主要原因。各条路径的频率响应会按照时间的变化而变化，因此接收端各条路径的信号之间的叠加状态也随时间变化而变化，而他们之间的相位叠加状态对于不同频率的影响也是不相同的，因此最后的干涉效果也会根据频率不同而相异，这种现象叫做频率选择性衰落。无线信号信道中，信号传播除了直射波和地面反射波之外，在传播过程中还会有各种经过障碍物所引起的散射波而导致多径效应。对于室内环境来说，狭小的环境本身就使得地面反射波和直接传输波路径差异相对较大，墙壁反射也增加了多径的复杂性，更不必说在有遮挡物，家具，挂饰和行走的人等障碍情况下的真实室内环境了。多径传输对于数字无线信号通讯系统中，多径效应会导致码间串扰(inter-symbol-interference，ISI)，从而影响信号质量，OFDM等技术都能用于减小多径效应导致的干扰。

对于Wi-Fi来说，多径效应主要反映在信道总特征较差，码间串扰严重。对于现在常用的OFDM技术，防止码间串扰问题的常用做法是增加一个保护间隔（GI），但是这又会造成波间干扰。对于室内来说，即使是毫无障碍物的小房间，墙壁影响下的多径效应也十分严重，更别提如果应用在实际场景，家具，移动的人，墙壁等等因素会对RSSI的测量造成很大的不确定性和精度丧失。因此为了应对高精度的场景，指纹（fingerprinting）定位方法出现了。

指纹定位方法需要较为繁琐的准备和测量工作，他基本上采用的是一种模式匹配的方法。基本的思路是通过收集室内所有可能会出现的点或位置上信号源发出信号的特征，形成一个较为庞大的指纹数据库。在实际使用这套系统进行定位时，通过收集信号源的信号特征，与数据库中每条数据，即每个位置上的信号特征进行匹配，距离最小的可以认为对应的就是信号源所在的位置。这一方案避开了分析复杂的多径传输情况，直接通过匹配来完成定位，一般能做到亚米级别的精度[5]。

该方法为了使指纹数据库达到要求的信度，一般需要对同一位置进行多次测量，为了使得匹配结果有效，需要使得不同位置的数据足够不同，相同位置的数据匹配度足够高。该方法尽管避开了多径传输问题，但是缺点也很显著。首先，该方法只能对同一场景有效，场景发生变化（如家具位置变动，人数增加等）都需要重新构建数据库，这使得该方法并不能构建一次数据库就可以具有全场景的通用性；然后，在使用这个方法前需要对每个位置都进行数据库录入，还需要进行校验，此外数据库越大，匹配需要的时间就越长，这大大增加了准备和方案实施所需要的精力，成本与时间。近年来，为了解决这一问题，有用众包等方案的研究出现。在提高指纹定位准确率方面的问题上，近年来开始火热的机器学习以及深度学习方法为匹配精度提供了方案，但基于深度学习的方法也相应地进一步增加了指纹定位方法的准备时间与实施成本。

2.1.2 TOA 和TDOA

TOA(Time of Arrival) [6]是一种在无线通讯领域曾广泛使用的测距方法，和RSSI不同的是，TOA用接收端接受信号的时间和发射端发送的时间的差值来计算距离。该方法原理简单，但是若想成功实施需要很多条件。

首先在室内想要使用这一方法，米级的距离除以无线电波的速度光速，数据是非常小的，这就需要精度非常高的测量手段，像是测量信号的带宽的要求很高。目前符合要求的就有超宽带技术（Ultra-Wide Bandwidth (UWB)）。UWB是一种新型的无线通信技术。它通过对具有很陡上升和下降时间的冲激脉冲进行直接调制，使信号具有GHz量级的带宽。这一技术解决了困扰传统无线技术多年的有关传播方面的重大难题，它具有对信道衰落不敏感、发射信号功率谱密度低、低截获能力、系统复杂度低、能提供数厘米的定位精度等优点。同时，它的时域精度非常高，可以计算短距离以光速运行的信号时间差。

其次，TOA方法也面临着在对多径效应严重的场景精度受到影响的问题，在比较糟糕的环境，比如说较为嘈杂或障碍较多的室内，在非视距(Non-Line-of-Sight)情况下，即有障碍物遮挡的情况下，信号传输的路径变得不确定而可能不同，使得精度下降。此外还有来自发射端和接受端之间时钟同步些许不一致导致的同步误差等。解决方法有多AP，接收设备多天线等。

为了满足这两个条件，TOA对硬件设备的要求比较高，对测量环境也有一定要求，这对于普适计算的室内定位来说并不是十分理想。

在TOA方法的基础上，提出了TDOA[6]（Time Difference of Arrival，到达时间差）的方法。不同于TOA，TDOA通过信号到达两个或多个AP的时间差来确定信号源的位置，而放弃了直接计算到达时间来计算距离。这种方法相对于TOA对于时钟同步的要求大大降低。具体操作方法可以采用三个AP，对其中任意不同的两组（每组是两个AP）的到达时间计算差值，然后分别以两个AP为焦点，差值长为实轴长做双曲线，分别作的两条双曲线交点即为信号源位置。TDOA相对于TOA而言，计算时由于同步误差，多径误差等参数对于距离不远的多个AP而言几乎相等，可以相互抵消，但是相应的由于距离相近的相邻AP的信噪比太小，又会导致较大的测量误差。两个AP的TDOA的获取一般有两种办法：

其一是用从信号源到两个AP的TOA的差值得到的。这种方法直观易懂，但是这就对AP间的是中通的要求比较高。然而这种方法在相邻AP相距较近，或者信道传输状态（多径状态）相似时可减少多径误差。

其二是将两个AP收到的信号进行相关运算（具体来说有chan和taylor算法等），来得到TDOA的值。相比上一种方法，该方法对于时钟同步的要求就没有了，并且也能达到较高的精度。在实际过程中，对网络和同步要求低，定位精度高的相关TDOA有很大的实践意义，曾是研究的热点，现在也在研究领域广泛应用着。

那么，TDOA减省了TOA所需的一些必须通信步骤，精度也比TDOA要高一些。目前在超宽带技术中得到应用得到青睐。

2.1.3 AOA

AOA(Angle of Arrival)也是无线网络定位中常用的一种算法。AOA即到达角度，即信号到达接收端时的角度。如果都是视距情况，那么如果有三个及以上位置的接收端的话可以直接计算出发射端所在位置，因为三个位置的距离和所成角度都是，用相关算法（如正弦，余弦定理）可以进行定位。

关于关键的如何测量距离的问题，一般用天线阵列或定向天线来计算角度，而计算角度的众多算法中轻量级而又有名的就是MUSIC算法。MUSIC(Multiple Signal Classification)算法是最著名的空间谱估计算法之一，其思想是计算接收信号的协方差矩阵(Rx)进行特征分解值分解，分离出信号子空间和噪声子空间，再利用信号方向向量与噪声子空间的正交性来构造空间谱函数，进行谱峰搜索，从而得到阵列接受的信号的空间信号数目和入射方向和信号源强度。算法原理基本如下[7][8]：

（1）无论定向天线阵列的排列怎样，信号入射的路径形成的角度个数有多少，阵列接收信号后输出的信号的DOA(Direction of Arrival, DOA)模型为：

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

Y是在天线阵列输出，X是发送端发送的空间信号，A是阵列的方向矩阵，而N是环境白噪声

（2）计算Y的协方差矩阵：

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

信号观测空间可以分解为信号子空间和噪声子空间，这两个空间是独立的，也就是不相关的。在这里协方差矩阵也可以视为分解为了两个不相关的矩阵，为信号部分

（3）对协方差矩阵进行特征值分解，有

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

前项为大特征值对应的特征向量构成的信号子空间，后项为小特征值对应的特征向量构成的噪声子空间。

（4） 前面说过，信号子空间和噪声子空间应该是相互独立，也就是正交的。这样的话信号子空间每个特征向量和噪声子空间应该也是正交的，这样就有：

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

那么我们就可以根据这个性质，在接受到的空间谱中找到我们需要的信号及所对应角度。实际情况下，子空间的维长度不可能是无限的，以上公式只是一个理想情况，所以我们需要追求的应该：

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

(5)直接能在谱中得到，那么怎么得到呢？数据的协方差矩阵的极大似然估计为：

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

对做特征分解就可以得到。

(6)因此最终入射信号及对应角度的计算问题变成了一个最优化问题。将以上两公式连立带入即为：

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

(7)MUSIC空间谱估计公式应该是

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

然而，该方法也是很收到多径效应等因素影响的，对于非视距和多径效应严重的室内，一般的设备精度效果不佳，解决方法有用多天线(4根及以上)来提高测量精度。此外，定向天线和天线阵列，无论哪一条都对于普适计算适用的设备来说硬件要求太高了，这限制了该方法的实际应用范围。

2.1.4 CSI

CSI(Channel State Information)，即信道状态信息，它描述了信号在每条传输路径上的衰弱因子，如信号散射（Scattering），环境衰弱（fading，multipath fading or shadowing fading），距离衰减（power decay of distance）等信息[5][9]。CSI的主要用途是通过提供当前通信系统的信道条件，从而改变系统的传播策略，在多天线系统中为高可靠性高速率的通信提供了保障。近年来有提出可以利用来提高一般室内定位方法精度的手段之一。它其实是OFDM方法下的子载波，在每个子载波下分别有不同的幅度与相位信息。这一方法相对于其他方法而言，有效的降低了多径效应的影响。具体我们需要从OFDM说起。

OFDM(Orthogonal Frequency Division Multiplexing)即正交频分复用技术，由所谓的MCM(Multi-Carrier Modulation)发展而来。OFDM一般是将信道分成若干正交子信道，将告诉的数据信号转换成并行的低速子数据流，分别调制到每个子信道上传输，这充分利用了信号的带宽。在接收端可以采用相应的调制办法进行解调。调制和解调分别利用了IFFT（逆快速傅里叶变换）和FFT（快速傅里叶变换）来实现，复杂度较低，并且将数据分解为正交子信号再解调，大大减少了子信道间的互相干扰（ISI），而由于每个子信道的带宽要远小于信道的带宽，所以每个子信道都可以认为是平坦性衰落，从而减少码间串扰，可以适应信道环境较差的一些环境，因为这些优势而得到广泛应用。

OFDM调制的这些子载波可以被用于计算CSI。至于为什么，需要先理解一下CSI是如何计算出来的，以及CIR和CFR的概念。

众所周知，室内传输的无线信号经受着很严重的多径效应带来的衰减。如果我们对于多径效应带来的损耗建模，即把它当成一个线性时域滤波器，这就是CIR(Channel Impulse Response，信道冲激响应)。

CIR 的计算可由如下得到：

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

其中和分别是第i条路径上波的振幅，相位和时延。N是多径效应中的路径总数。是狄拉克函数。每个冲击信号都代表了多径效应中多径延迟的一部分。

在频域，信号的传输也会引发频域选择性衰减的信号损耗现象，这种现象也可以用CFR(Channel Frequency Response，信道频域响应)来表示。在无限的频带上，CIR和CFR是相同的。而实际上CFR可以由CIR的傅里叶变换得来。相比RSSI，CIR和CFR就相当于是彩虹之于太阳光，前者各个多径成分分开，后者是杂糅在一起的。如果可以求得CIR或CFR，那么就有了可以分析多径传输各条路径不同的衰减条件的办法，势必也可以得到更高精度的室内定位办法。

那么在时域上，我们知道，接受到的信号是发送信号和CIR即信道冲击响应的时域卷积：

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

相应的，在频域上，接收信号的频谱应当是发送信号的频谱和CFR即信道频域响应的积[5]：

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

那么如果我们已知各条路径接受到的信号强度，相位以及时延信息，我们就可以反求出相应的CIR或CFR。自然，由多个子载波组成的OFDM，包含所有需要的信息，那么只需在解调端用相应的程序分析就可以得到相应的CIR。

近年来，很多厂商都开发了可以进行如此测量的路由器和网卡。Intel 5300网卡就是常用的收集这样的信道信息的网卡之一，它通过OFDM的各子载波信息计算出每个子载波相对应的CFR值，以CSI的格式存储。这就是CSI的来由。

利用CSI的各条子载波相应数据来代替RSSI进行几何位置定位和指纹定位，精度都得到了很大提高。

而本文所研究的就是如何用装有Intel 5300网卡的设备实时显示每次收到的CSI信息。

2.2 相关工作

2.2.1 相关工具

CSI作为一种较为新兴的概念与手段，目前没有得到较大规模的商业化，市场上CSI相关的产品较少，但也有支持比较高精度的CSI采集的商业设备，如商用软件无线电设备(Software-Based Radio)很多是在研究所和大学实验室在进行测试与研究，而这些校内研究大多是基于Intel 5300 无线网卡，而很多人使用名为CSI-Tool的CSI数据解析工具(包括我们)。但是最近我们也有研究了其他各种新的CSI采集工具，有Ateros CSI Tool(Qualcomm)，Nexmon Channel State Information Extractor(Broadcom)，还有基于ESP32，一种可携带的嵌入式设备，的CSI接收计算模块。目前我们尝试过的有Intel CSI(Intel 5300 NIC)和ESP32 Wi-Fi Module(ESP32)。本位我们主要还是基于Intel 5300 网卡，暂未提供对于其他支持CSI采集的网卡的支持。

该项目在Github上开源，可免费访问下载[10]。这可以适用于普通的Wi-Fi设备，运行在网卡的MIMO(Multi-input Multi-output)无线电设备上。通过该工具从IWL (Intell Wireless) 5300网卡采集到的数据是30个子载波的数据，每条信道包含的数据矩阵为实部和虚部都为8比特表示精度的复数，基本表示了每对天线之间信号的增益和相位情况。一般来说使用该工具，无论是指纹定位还是其他方法，都是根据该工具得到的数据直接或间接(加以计算)处理来进行定位，动作识别和其他研究实验的。我们的工具也是作为CSI-Tool的一个插件来开发的。

CSI除了在硬件驱动提供了方法之外，也提供了调用CSI数据的接口，也提供了读取CSI数据的程序。我们的需求主要是在两方面，首先是实时绘图，之后是对绘图端口的Python改写，这两项需求产生的原因和CSI-Tool的数据读取端口用matlab写成的事实密不可分，关于这一点会在后面的章节细说。

2.2.2 相似应用考察

不可否认的是CSI-Tool为CSI研究提供便利和成本低廉的工具，而对于该工具的扩展的发布与讨论以及同类产品开发也在该项目相关专题热度不减。对于和本文相关的实时化现实相关的内容，本文做了些许调查，也试图验证本文工具开发的需求。本文调查结果来源主要来源于该项目的Issues列表。

首先是较为成型的软件。来自清华大学可信网络与系统研究所的团队做出了基于JAVA的CSI实时监测系统。该系统可以说得上是基本实现了我们所想实现的功能，不仅实现了各条天线CSI数据的实时显示，还可以显示通过CSI计算的RSSI等数据，参数调节也比较人性化。该工具还开发了一套报警系统，用于检测环境中检测对象的状态变化程度是否超过阈值。除此以外，该团队还根据CSI安装步骤给出已经安装好环境配置的软件，可以代替CSI-Tool的常规安装。

图2.1 清华制作的CSI显示界面

然而可惜的是，这样的工程尽管可以直接拿来使用，但却有二次开发困难和某些机制不明的问题。因为该工具是作为程序包直接运行的，代码并不是开源的，想要在此基础上修改机制或是添加一些接口比如机器学习代码是做不到的。此外，关于实际使用的时候，其速率调节机制似乎是有极限的，无法调到很高的速度，但是我们实验室做的其他CSI研究中有时候后需要很高的数据发送接收速度，这就不符合我们的需求。基于这两点原因，我们仍旧需要开发一套自己的CSI实时显示系统。

除了这款同类产品，也有其他研究同仁开发了开源的CSI实时显示工程。在CSI-Tool的相关Issues中有人给出了基于CSI-Tool开发的在matlab平台的LiveStreamCSI基本满足了CSI实时显示的需求，具体请见参考文献[12]。

而GitHub上也有同仁开发了基于Python的CSI实时解析和绘图工具，具体请见参考文献[13][14]。

不得不说，这两个项目分别各自满足了我们的两个需求中的一个，我们在开发过程中也有借鉴过这两个工程，尽管主要借鉴的是CSI-Tool本身。导致我们最后决定自己开发的原因之一是因为对方代码实现较为复杂，不如自己开发，其二实现的效果还是和我们的构想有所距离，想要能够方便迭代开发，最好还是自己开发自己维护一套代码来的稳妥。

3 项目设计

3.1 需求分析

基于CSI-Tool的Wi-Fi数据CSI可视化可以在当前有重要意义的CSI相关研究中使用。目的是为了避免收集数据再跑数据的时间浪费，让研究者可以快速及时的看到现场的CSI变化情况，来达到判断实验数据是否失常，实验环境是否理想，或者检测实验步骤或结果是否符合预期等目标。

这里解释一下，我们在这里应该有一个CSI信号源，CSI接收端和CSI绘图端。一般来说后两者可以合并在同一个机器上，实际上CSI-Tool正是基于这样的考量才使用文件存储本地的方式分析CSI数据的。但是我们有很多场景没有办法做到CSI接收端和绘图端在一个设备上，比如没有装matlab的linux系统的接收端，或者是像ESP32那样的移动嵌入式设备。所以我们采取了CSI接收端接受CSI，再用TCP发送给绘图端电脑，绘图端进行实时绘图。由于CSI发送端和我们的应用没有直接关系，所以我们在本文中不涉及这个概念，相对的我们称CSI接收端为（TCP报文的）发送端，CSI绘图端为（TCP报文的）接收端。

总的来说，这样一套系统应该能够提供发送端（CSI接收端）和接受端（图像信息接收端）的接口，完成实时绘制发送端（CSI接收端）程序发来的CSI信息，并在绘图应该尽可能提供友好的操作界面，并能根据用户（研究人员）的意向改变显示模式，这就要求系统做到良好的操作性，可交互性。此外在实时性方面应要做到不卡顿，不延迟，使得现场数据准确地显示在界面上，这就要求系统有高效性，有效性。最后，为了能够满足系统能够二次开发，迭代开发，比如添加机器学习代码，系统应当具有多平台的可移植性，可扩展性，并且代码开源可读性高，能够省去二次开发人员的大量学习阅读过程，逻辑应当清晰合理，注释也应该便于理解。

根据以上分析，Wi-Fi数据CSI可视化系统的要求如下：

（1）系统基本需求：

① 能够让发送端从内核读取并发送CSI信息，接收端接受CSI信息并实时绘图

② 提供至少两种常用平台开发下的程序接口

（2）系统性能需求：

① 交互界面友好并适用实际情况要求

② 系统实时性目标较好，避免积累延迟卡顿现象

③ 系统开源代码可读性较好，便于理解

因此，这套系统应该是能够通过网络连接等方法接收到CSI信息，并实时解析数据，然后绘图并根据数据的刷新实时更新图像。图像可以是RSSI值，可以是从Intel 5300网卡收集的一次30个子载波的信号强度，这个根据实际情况而定。并且可以应对CSI高采样率的情况，使得数据卡顿或延迟尽量小。同时能够满足多平台的使用，比如基于原先CSI-Tool的数据解析平台matlab或者是能和当前机器学习开发火热使用的Python相结合。注释和模块设计应当便于阅读。

3.2 总体设计

根据以上分析，我们应该设计以下模块

发送端（CSI接收端）模块：包括从内核读CSI数据，发送CSI数据到接收端模块

接收端（图像数据接收端）数据解析模块：应分别实现matlab端和Python端，包括网络模块，数据读取，数据解析等模块。

接收端绘图模块：也应分别实现matlab端和Python端，包括可能的缓存处理，绘图处理和数据更新处理等。

**数据源**

**用户**

图3.1 用例图

4 实现与效果

4.1 发送端模块（包括CSI-Tool自带驱动部分）

发送端（CSI接收端）主要是在CSI-Tool自身的发送端程序[10]基础上改写的，原本的程序是C，不需要平台迁移。主要是运行log\_to\_server.c文件，用TCP/IP的形式连接到服务器（发送端）的。

这里由于知识水平限制，对CSI-Tool的驱动等机制不做过多赘述，这里需要知道从CSI信号源得到的封装好的二进制数据只要在内核中读就可以了。CSI-Tool原先的文件名为log\_to\_file.c，稍微说一下实现的思路。首先是建立一个套接字，用来读内核的数据，也就是采集到的发送端网卡收到的包的数据，这其中也包含着网卡计算的前一个数据包的CSI的数据，CSI数据单独作为一种报文写在内核里。连接建立后（也就是所谓的客户端connect过程）把数据存在文件里。需要一提的是CSI-Tool也有相配的从文件里读取二进制文件将其变为标准结构体的程序，这一点在之后的章节细说。

我们的改写思路就比较直接，我们所要做就是把存文件改成以TCP报文发到服务器。这样的话需要在除了原先读内核接收到的CSI信息而建立的套接字外，再新建一个套接字，用于和服务器通讯。

**代码4.1 发送端套接字代码节选 log\_to\_server.c**

/\* Connect with Internet server\*/

bzero(&servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(serv\_port);

Inet\_pton(AF\_INET, serv\_ip, &servaddr.sin\_addr);

sock\_cli\_fd = Socket(AF\_INET, SOCK\_STREAM, 0);

Connect(sock\_cli\_fd, &servaddr, sizeof(servaddr));

/\* Setup the socket \*/

sock\_fd = socket(PF\_NETLINK, SOCK\_DGRAM, NETLINK\_CONNECTOR);

if (sock\_fd == -1)

exit\_program\_err(-1, "socket");

/\* Initialize the address structs \*/

memset(&proc\_addr, 0, sizeof(struct sockaddr\_nl));

proc\_addr.nl\_family = AF\_NETLINK;

proc\_addr.nl\_pid = getpid(); // this process' PID

proc\_addr.nl\_groups = CN\_IDX\_IWLAGN;

memset(&kern\_addr, 0, sizeof(struct sockaddr\_nl));

kern\_addr.nl\_family = AF\_NETLINK;

kern\_addr.nl\_pid = 0; // kernel

kern\_addr.nl\_groups = CN\_IDX\_IWLAGN;

存文件的代码改成写套接字的形式，以二进制的形式传输到服务器。在这里CSI-Tool再原先从内核获取的数据的基础上，在包头添加了一段描述包长度的16比特数据，我们自然也是完全遵循其写文件的顺序发报文的。

**代码4.2 发送端文件传输代码节选 log\_to\_server.c**

/\* Poll socket forever and then send data to server\*/

while (1)

{

/\* Receive from socket with infinite timeout \*/

ret = recv(sock\_fd, buf, sizeof(buf), 0);

if (ret == -1)

exit\_program\_err(-1, "recv");

/\* Pull out the message portion and print some stats \*/

cmsg = NLMSG\_DATA(buf);

if (count % SLOW\_MSG\_CNT == 0)

printf("received %d bytes: id: %d val: %d seq: %d clen: %d\n", cmsg->len, cmsg->id.idx, cmsg->id.val, cmsg->seq, cmsg->len);

/\* Send the data to server \*/

l = (unsigned short) cmsg->len;

l2 = htons(l);

if( write(sock\_cli\_fd, &l2, sizeof(unsigned short)) < 0)

exit\_program\_err(-1, "write");

if (write(sock\_cli\_fd, cmsg->data, l) < 0)

exit\_program\_err(-1, "write");

if (count % 100 == 0)

printf("wrote %d bytes [msgcnt=%u]\n", ret, count);

++count;

}

exit\_program(0);

关于数据格式的问题将在下一章节描述。

开始

建立连接

信号源发送正常的数据包

发送端（CSI接收端）接收信号并存在内核

建立套接字准备读取内核信息

再建立套接字准备连接到接收端

从套接字读取内核CSI信息封装

以一定格式将数据封装并发送

连接存在？

结束

**是**

**否**

图4.1 发送端程序处理流程图

4.2 接收端数据解析模块

CSI-Tool自身也有read\_bf\_file.m[10]这样能够直接读存下来CSI数据文件的matlab函数。正如需求中所阐述的那样，我们先做了matlab上接收解析数据的操作，再做了与之功能相同，但是先略有差别的Python语言的解析文件。

首先先描述一下原来CSI-Tool的读取并解析文件的实现思路。基本上就是直接二进制形式读文件，再解析文件转换成matlab的结构体的过程。读文件用fread()函数进行，之前说过发送端在写常规CSI信息前会附加一个16比特无符号数表示包长度的报文，所以需要先读这个2个字节的包头，才知道后面的CSI报文有多长。在得知包长度后，我们需要读之后一个字节的数据也就是报文的前8位，转换成无符号数。这是一个标志，若值为187，那么该报文就是CSI数据，我们正是用这样的规则过滤报文的。之后我们根据解析出的包头报文长度，读取整个包。CSI-Tool提供了直接将二进制文件报文解析成包含报文各种参数的matlab结构体的程序，read\_bfee.c. 该程序是C-MEX程序，matlab可以直接调用，以C语言的形式读入读取的报文，再以matlab的结构体形式返回，关于C语言和matlab如何通讯的问题这里不再赘述。总之，解析出来的报文结构体参数和解释大致如下。

表4.1 解析出来的报文参数及其对应解释

|  |  |
| --- | --- |
| 字段 | 意义 |
| timestamp\_low | Intel 5300网卡的1MHz时钟时间戳的最低32位，大约是4300秒，或者72分钟 |
| bfee\_count: | 被网卡接收并发往用户区的CSI包的总数。有时netlink模块传输内核和用户区时会丢包，因此该条信息可以用于检测数目是否和原本发送的宝数目吻合 |
| Nrx | 网卡接收端的天线数量 |
| Ntx | 发送数据方的数据时间/空间流数目。一般来说，发送方发送一条数据流，接收方用3根天线接收（Intel 5300网卡） |
| rssi\_a, rssi\_b, rssi\_c | 这三个数值分别对应于网卡每根天线接收到信号的信号强度。该数据在处理包头时进行计算。数值单位为dB，若要转换为dBm为单位的RSSI我们必须将其和参数agc(Automatic Gain Control)结合并减去一个常数。 |
| noise | 噪声值，若噪声值<-189, 可以认为噪声值为-92dBm |
| agc | 自动增益控制参数，用于计算RSSI |
| perm | 表示从三根天线接收到的信号分别对应于哪条射频链。举例而言，[3 1 2]的数据意味着天线C的数据对应射频链A，天线A的数据对应射频链B，天线B的数据对应射频链C。 |
| rate | 发包速率。即CSI包被发送时的采样率。 |
| csi | 本身是个Ntx\*Nrx\*30的复数矩阵。分别表示每根数据流每根天线对应OFDM30个子载波的CSI数据，数值已被内部标准化处理。 |

有了这样的工具，我们可以省去自己学习报文参数的步骤。我们可以看到不仅CSI，还有三根天线（5300网卡自带三根天线）的RSSI等参数也包含在里面，这些可以为程序功能的扩展提供支撑。

最后是关于异常处理的问题，程序中设定了读到文件末尾及以后捕获异常并结束程序，由于我们是信任包头报文长度这个参数的准确性的，所以只对读报文长度这个语句做异常处理。所有收集到的结构体被存在元胞数组里作为返回值。

以上就是CSI-Tool处理的思路。在此基础上可以加以修改做成matlab端的实时显示代码，而Python端的需要对上述函数重新实现。

**是**

是否是187？

**否**

开始

建立套接字并监听发送端的端口

跳过后面数据长度-1字节的数据

读取后面数据长度-1字节的数据

连接建立？

**否**

读取前两字节，即包头的数据长度

按照数据封装格式解析数据，并以结构体的形式存储

读取是否出错？

**否**

数据再处理及计算

读取后一字节，确认包类型是否是CSI

**是**

绘图模块

**是**

图4.2 接收端总程序处理流程图（数据解析模块视角）

4.2.1 matlab端

基于以上介绍和需求所述，我们需要进行的修改主要是把读文件改成读套接字，接受文件改成网络连接，还有增加比较重要的实时显示（绘图）功能。

**代码4.3 matlab端网络处理代码节选 read\_bf\_socket.m**

%% Build a TCP Server and wait for connection

port = 8090;

t = tcpip('0.0.0.0', port, 'NetworkRole', 'server');

t.InputBufferSize = 1024;

t.Timeout = 15;

fprintf('Waiting for connection on port %d\n',port);

fopen(t);

fprintf('Accept connection from %s\n',t.RemoteHost);

**代码4.4 matlab端数据读取代码节选 read\_bf\_socket.m**

%% Process all entries in socket

% Need 3 bytes -- 2 byte size field and 1 byte code

while 1

% Read size and code from the received packets

s = warning('error', 'instrument:fread:unsuccessfulRead');

try

field\_len = fread(t, 1, 'uint16');

catch

warning(s);

disp('Timeout, please restart the client and connect again.');

break;

end

code = fread(t,1);

% If unhandled code, skip (seek over) the record and continue

if (code == 187) % get beamforming or phy data

bytes = fread(t, field\_len-1, 'uint8');

bytes = uint8(bytes);

if (length(bytes) ~= field\_len-1)

fclose(t);

return;

end

else if field\_len <= t.InputBufferSize % skip all other info

fread(t, field\_len-1, 'uint8');

continue;

else

continue;

end

end

...

end

整体而言我们的思路应该是建立套接字监听并建立连接，按字段读文件，然后解析文件变成matlab的结构体，再进行数据处理，绘图以及实时更新这样的过程。首先建立套接字并建立连接，matlab提供了封装的相当好的函数 tcpip()，一步完成从建立套接字监听再建立连接的全部过程，而读/写套接字也和linux套接字编程相似，matlab可以把套接字当做文件直接读（fread()），另外并且可以按照指定格式直接转换。

但是和linux常用的读取套接字函数recv(), read()不同，fread是真的把缓存区当做一个文件来读，新发到的报文会作为文件的最新一部分写在缓存区文件末尾，缓存区读了一部分之后读取位置会变成已读取部分的后一位。而我们所熟知的linux socket的常用读函数，都是先按照设定的缓存区的大小接收一次发送端发送的报文，无论报文大小多少都按照相同缓存区大小接收，其余超过缓存大小的全部抛去，这样处理对于一次性发送多次报文内容等情况来说就必须要做一个报文划分或是拼接的一个过程来把一次报文划分成每一单次的报文，这种操作可以说是比较浪费效率的并且实现较为复杂的，而且如果避开这种情况的话，进行网络测试可能会比较麻烦，而像matlab的fread()函数就很好，省去了上述编程处理上的很多麻烦。

而按字段读文件可以直接参照CSI-Tool的代码，既然套接字和文件的操作一样，不再赘述。那么重要的就是如何处理结构体里的参数了。我们主要是显示CSI数据，而实际上5300网卡直接采集的CSI数据是一个1\*3\*30（Ntx=1,Nrx=3,网卡采集子载波数量为30）复数数组，这样的数据是难以让人看到变化趋势的。官网上CSI-Tool作者对于试图绘图的情况给出了两种建议[15]：采用rssi\_a, rssi\_b, rssi\_c结合agc(automatic gain control, 自动增益控制)计算总共的RSSI数据，单位为dBm；或者是csi结合noise计算计算3根天线，30个子载波各自SNR（Signal-Noise ratio, 信噪比），单位是dB，CSI-Tool直接提供了get\_simo\_SNRs和get\_scaled\_csi等函数来分别实现这两种功能。为了能够显示CSI的信息。我们倾向于先将SNR信息显示，而RSSI信息在某些场景下也很重要，也可以提供显示选项。

**代码4.5 matlab端数据解析代码节选 read\_bf\_socket.m**

**…**

if (code == 187) % (tips: 187 = hex2dec('bb')) Beamforming matrix -- output a record

csi\_entry = read\_bfee(bytes);

perm = csi\_entry.perm;

Nrx = csi\_entry.Nrx;

if Nrx > 1 % No permuting needed for only 1 antenna

if sum(perm) ~= triangle(Nrx) % matrix does not contain default values

if broken\_perm == 0

broken\_perm = 1;

fprintf('WARN ONCE: Found CSI (%s) with Nrx=%d and invalid perm=[%s]\n', filename, Nrx, int2str(perm));

end

else

csi\_entry.csi(:,perm(1:Nrx),:) = csi\_entry.csi(:,1:Nrx,:);

end

end

end

index = mod(index+1, 10);

csi = get\_scaled\_csi(csi\_entry);%CSI data

关于SNR的提取，get\_scaled\_csi写的比较详细，简而言之就是把原先网卡相对量的CSI数据改成绝对量dB，具体方法主要是将RSSI测量值和CSI的数组的各项模之和比对算出量纲差，再去除noise提供的噪声值所带来的影响，进行常规转换计算出将量纲化为绝对值而非相对值的CSI数据。具体得到数值还需要对get\_scaled\_csi的返回值矩阵压缩，再计算模，再化为dB值，得到一个3x30的double数组。这个数据绘图出来就是三条曲线，每条曲线分别代表每根天线，而横坐标30个值分别代表30个子载波，纵坐标则是信噪比SNR值，单位为dB。如下为一组CSI数据转换后的绘图效果示意图。关于如何实现，matlab自己提供squeeze(), abs(), db()来分别进行多维矩阵压缩，求复数矩阵各项的模以及绝对能量值转换分贝dB值的功能。这里具体实现请详见附录代码。有关绘图的过程将在下一节细说。

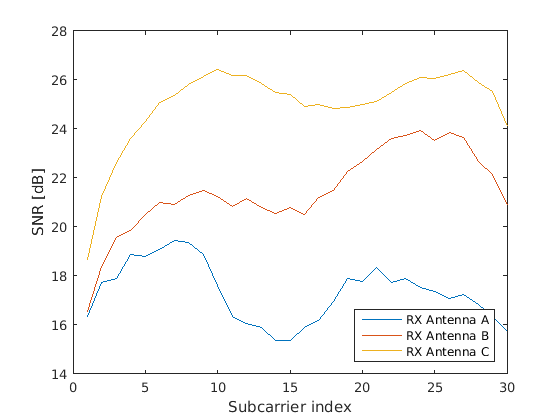


图4.3 一次CSI数据绘制的图像（来自CSI-Tool官网）

最后关于异常处理的问题，按道理我们需要在原先读文件的异常之外增加网络处理的异常，但是由于读文件和读套接字的函数fread()可以通用，将原先的异常处理按原先方式保留仍然起到了想要的效果，所以这里和CSI-Tool原版的代码基本相同。

4.2.2 Python端

之前说过，我们想要脱离matlab平台将实时绘图功能建立在一些可扩展性好的平台，比如Python，R等。这样做一方面是因为matlab是一个大体量且收费的软件（尽管如此，也出现了开源matlab代替品如Octave，但体量大是不可避免的，有兴趣建议尝试一下），另一方面matlab扩展性可能不比Python这些开源平台好，比如最近一个需求是用深度学习训练CSI来进行指纹定位，matlab不是做不了深度学习，但是如今主流的，用户化程度高的机器学习方法大多基于Python的包实现，以方便和今后可扩展性而言，在Python平台将上述内容再实现一遍是有必要的，这将会今后的研究工作省去很多麻烦。对于没有接触过Python的我而言，也正是一个绝好的学习Python的机会。

那么我们实现的思路还是按照上述过程，建立网络连接，读文件，解析文件，数据处理再绘图这样的过程，只不过上面提到的matlab函数，除了客户端，其他都是由Python实现的，也就是需要做一个重新实现的过程。Python的实现在很多方面和matlab并不相同，下面说一些改写的地方。

首先关于网络连接的设置问题。Python的网络编程更接近于linux的socket编程，基本上linux中socket,bind,listen这些函数都在Python中有同名对应，所以我们可以按照linux网络编程的基本流程设计。但是正如上一节所述，按照惯常的linux网络编程处理就需要对读套接字(在Python中基本上只有recv()函数)的时候可能需要做个数据划分或拼接的操作，那么我们当然还是想要像matlab里fread()那样，可以把缓存当做一个权限为w+的文件进行读写。幸运的是，Python里有这样把socket缓存区转换成文件的办法：makefile。该函数作为套接字socket的类函数，直接返回一个文件描述符就是可以把套接字缓存区当做文件读的，参数即时常见文件选项，只读(r)，读写(w+)等选项。这样的话我们就可以像matlab的fread()那样连续处理报文了。

**代码4.6 Python端网络处理代码节选 read\_bf\_file.py**

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

host = socket.gethostname()

port = 8098

s.bind((host, port))

s.listen(5)

print('waiting for connection on port', port)

c, addr = s.accept()

c.settimeout(15)

print('连接地址：', addr)

fd = c.makefile('rb')

之后是读文件的问题，这里实际上还有一些问题。前面提过fread可以直接把读的二进制文件转换为相应的数据类型，比如无符号数等。但是Python的数据转换要相对麻烦一些，特别是有关二进制数据的处理。有几种方法可以把二进制文件转换成相应的Python数据类型，我只介绍我们尝试成功的一种。对于Python3来说，如果把二进制文件转换成无符号整型，如果数据只有8位的话，可以直接当做整型用，比如对于b'\xFF'，可以直接当做无符号整型数127使用。但是对于多字节的数据，可以用int.from\_bytes函数进行转换，参数一般有两个，被转换的二进制文件在这里也就是被转换的缓存区文件描述符，和字节序也就是所谓的大端小端，出来的也是无符号数。

**代码4.7 Python端数据读取代码节选 read\_bf\_file.py**

p = np.zeros((30,30), dtype='double', order = 'C')

while True:

# Read size and code from the received packets

try:

fl = fd.read(2)

field\_len = int.from\_bytes(fl, 'big')

if field\_len == 0:

break

except:

print('Timeout, please restart the client and connect again.');

break

co = fd.read(1)

if not isinstance(co, int):

code = int.from\_bytes(co, 'big')

else:

code = co

# If unhandled code, skip (seek over) the record and continue

if code == 187: # get beamforming or phy data

Bytes = fd.read(field\_len-1)

if len(Bytes) != field\_len-1:

c.close()

exit()

elif field\_len <= 1024: # skip all other info

fd.read(field\_len-1)

continue;

else:

continue;

关于转换成无符号数，Python本身只提供了整型（Interger）这个类型，并没有我们试过对我们有效的有符号数转换方法，所以我们是按照有符号数和补码的定义自定义实现了有符号数转换的方法。实现了上述从二进制数据到整形数据的转换，剩下的步骤就和matlab端的别无二之了，还是按照先读包头以获取包长度，在读后面第一字节时过滤有CSI信息的报文，再解析CSI数据。

**代码4.8 Python端无符号数转有符号数 read\_bfee.py**

def usig2sig(b):

if b > 127:

return b

else:

return ~(b-128) + 1

解析文件的问题，需要对CSI-Tool原本的C-mex文件read\_bfee.c进行改写。这个mex文件形成的数据结构只能由matlab读取，我们需要将其改成Python处理，或者C语言和Python可通讯的数据结构方法。而用C语言程序和Python通讯理应比较方便，因为Python基本是基于C来实现的，所以C语言中大部分数据结构都可以直接在Python中使用。但是我们需要一种比较复杂的结构体，来存储CSI的各种信息，包括复数数组，这并不是C语言库中已包含的数据结构可以轻易实现的。解决方法之一是使用numpy中的结构数组，但是该方法我们们尝试之后，认为该方法主要还是面向数值型字段的数据结构，对于结构中一个字段包含一个数组的处理并不是非常好。所以我们决定直接用Python自带的数据结构来实现数据的解析，而CSI数组的部分采用numpy的数组进行存储。基本按照这个方式我们可复刻read\_bfee函数的功能。当然对于一些之前提过的在matlab中已经存在的库函数很多在Python中没有实现，我们需要自己实现或者在Python中找到替代的方法。

**代码4.9 Python端二进制数据解析代码 read\_bfee.py**

import numpy as np

def read\_bfee(Bytes):

timestamp\_low = int.from\_bytes(Bytes[0:4], 'little')

bfee\_count = int.from\_bytes(Bytes[4:6], 'little')

Nrx = Bytes[8]; Ntx = Bytes[9]; rssi\_a = Bytes[10];

rssi\_b = Bytes[11]; rssi\_c = Bytes[12];

noise = usig2sig(Bytes[13])

agc = Bytes[14]; antenna\_sel = Bytes[15]

len = int.from\_bytes(Bytes[16:18], 'little')

fake\_rate\_n\_flags = int.from\_bytes(Bytes[18:20], 'little')

calc\_len = int((30 \* (Nrx \* Ntx \* 8 \* 2 + 3) + 7) / 8)

index = 0; payload = Bytes[20:];

csi = np.empty((Ntx, Nrx, 30), dtype=complex, order = 'C')

perm = [((antenna\_sel) & 0x3) + 1, ((antenna\_sel >> 2) & 0x3) + 1, ((antenna\_sel >> 4) & 0x3) + 1]

if len != calc\_len:

print('Error: Wrong beamforming matrix size.')

return

for i in range(0, 30):

index += 3

remainder = index % 8;

for j in range(0, Ntx):

for k in range(0, Nrx):

re = (payload[(int)(index/8)] >> remainder) | (payload[(int)(index/8)+1] << (8-remainder))

im = (payload[(int)(index/8)+1] >> remainder) | (payload[(int)(index/8)+2] << (8-remainder))

csi[j,k,i] = complex(re, im)

index += 16

structure =[timestamp\_low, bfee\_count, Nrx, Ntx, rssi\_a, rssi\_b, rssi\_c, noise, agc, perm, fake\_rate\_n\_flags, csi]

return structure

接下来是数据处理的方法，依然采用计算SNR的方式。那么我们需要实现matlab端中的get\_scaled\_csi()，db()，get\_total\_rss()等函数，squeeze()函数在Python的numpy库中集成了类似的方法，就可以直接使用。具体实现办法其实主要是按照matlab端的代码逻辑来做的。

**代码4.10 Python端工具函数实现代码节选 get\_scaled\_csi.py**

import numpy as np

import math

import matplotlib.pyplot as plt

def dbinv(csi\_st):

ret = pow(10, csi\_st/10)

return ret

def db(csi\_st):

ret = np.log10(csi\_st) \* 10

return ret

def get\_total\_rss(csi\_st):

rssi\_mag = 0;

if csi\_st[4] != 0:

rssi\_mag = rssi\_mag + dbinv(csi\_st[4])

if csi\_st[5] != 0:

rssi\_mag = rssi\_mag + dbinv(csi\_st[5])

if csi\_st[6] != 0:

rssi\_mag = rssi\_mag + dbinv(csi\_st[6])

ret = db(rssi\_mag) - 44 - csi\_st[8]

return ret

**代码4.11 Python端计算csi绝对值代码 get\_scaled\_csi.py**

def get\_scaled\_csi(csi\_st):

csi = csi\_st[11];

csi\_sq = csi \* np.conjugate(csi).real

csi\_pwr = sum(csi\_sq.flatten())

rssi\_pwr = dbinv(get\_total\_rss(csi\_st))

scale = rssi\_pwr / (csi\_pwr / 30)

if csi\_st[7] == -127:

noise\_db = -92

else:

noise\_db = csi\_st[7]

thermal\_noise\_pwr = dbinv(noise\_db)

quant\_error\_pwr = scale \* (csi\_st[2] \* csi\_st[3])

total\_noise\_pwr = thermal\_noise\_pwr + quant\_error\_pwr;

ret = csi \* math.sqrt(scale / total\_noise\_pwr);

if csi\_st[3] == 2:

ret = ret \* math.sqrt(2);

elif csi\_st[3] == 3:

ret = ret \* math.sqrt(dbinv(4.5));

return ret

关于异常处理，基本和matlab端一致，这里不再赘述。绘图的代码将在下一章介绍。

4.3 接收端绘图模块

正如前面介绍的CSI那样，CSI并没有实现实时绘图这项功能，这是我们需要添加的关键方法之一。为了实现实时显示我们也做了诸多尝试，下面将分别从matlab端和Python端的实现介绍我们的实现。

**是**

开始

建立套接字并监听

获得3x30的存有CSI的SNR信息的结构体

在维护的plot数组中，将最老的三个数据更新为得到的新数据

数据解析模块

以30个子载波和SNR分别为x, y轴，将plot数组数据一次性绘制

连接建立？

**否**

图4.4 接收端总程序处理流程图（绘图模块视角）

4.3.1 matlab端

首先是关于我们希望得到什么样的显示结果的问题。我们得到的数据是3（3根天线）x30（30个子载波）的数据，我们倾向于二维的图像表示，因为绘图效率和速度可能会快一点。我们分别用红绿蓝三种颜色表示三根天线的数据，横坐标是30个子载波，纵坐标是SNR即信噪比。

**代码4.12 matlab端绘图代码节选 read\_bf\_socket.m**

...

%% Set plot parameters

clf;

axis([1,30,-10,30]);

t1=0;

m1=zeros(30,1);

...

p = plot(t1,m1,'MarkerSize',5);

...

xlabel('Subcarrier index');

ylabel('SNR (dB)');

...

index = mod(index+1, 10);

...

...

%This plot will show graphics about recent 10 csi packets

set(p(index\*3 + 1),'XData', [1:30], 'YData', db(abs(squeeze(csi(1,1,:)).')), 'color', 'b', 'linestyle', '-');

if Nrx > 1

set(p(index\*3 + 2),'XData', [1:30], 'YData', db(abs(squeeze(csi(1,2,:)).')), 'color', 'g', 'linestyle', '-');

end

if Nrx > 2

set(p(index\*3 + 3),'XData', [1:30], 'YData', db(abs(squeeze(csi(1,3,:)).')), 'color', 'r', 'linestyle', '-');

end

axis([1,30,-10,40]);

drawnow;

**…**

这样规划的仅仅是对于一次数据的显示，我们想要最好可以反映趋势变化。为此我们的办法是每次显示最新十次的数据，将其绘图实时更新。接下来说一下具体怎么实现的。我们绘图主要用的还是plot()，来画折线图，但实际上我们需要维护一个10x3x30的队列，每次将获得的数据存进队列，而又把最旧的一个数据踢出队列，然后对其绘图。队列的实现不再赘述，对于一次性画多条数据，我们使用了set()函数。set可以对plot函数返回的句柄名进行操作，设置其数据，线型，颜色等。所以我们事先声明变量时绘制了一个数据为0的30x1的空数据（综合算来我们依次需要显示3x30条线），得到它的句柄，再用set函数对每条线进行赋值，SNR值的表达前文已经说过。

在实际进行操作的时候也出现了一些问题，就是用普通的plot绘制出的图像，实时性非常差，最后甚至出现了缓存区溢出的状况。指导学长给出了指导，建议我使用matlab的擦除模式。为什么擦除模式提高了我们的绘图的效率呢？前面说过，我们绘制图像实际上是每次只更新一次三条图像，把旧的图像更新成新的。但是实际上系统在绘图时还是按照一次画30条线的办法进行的，可是每次我们其实只需要更新三条就可以了。擦除模式正是解决了这个问题，他会按照前后图像的差别，从图像上“擦除”新图像有旧图像没有的，又绘制新图像新更新的部分。这种选项实际上提高了我们的系统的实时性能。但可惜的是该项功能在matlab 2014b以后版本被移除了，所以我们做了个版本控制，并建议在matlab 2014b以下版本运行我们的代码。以上就基本是matlab端实时绘图功能的实现。

**代码4.13 matlab端擦除模式及版本控制 read\_bf\_file.c**

...

%% Starting in R2014b, the EraseMode property has been removed from all graphics objects.

%% https://mathworks.com/help/matlab/graphics\_transition/how-do-i-replace-the-erasemode-property.html

[VER DATESTR] = version();

if datenum(DATESTR) > datenum('February 11, 2014')

p = plot(t1,m1,'MarkerSize',5);

else

p = plot(t1,m1,'EraseMode','Xor','MarkerSize',5);

end

...

4.3.2 Python端

Python和matlab绘图方式有着显著差别，相比之下Python的绘图方式更为多种多样。为了实现是实时显示的目标，我们对于几种常用的Python库进行了调研和尝试。

首先尝试了所谓的matplotlib这个库来实现Python端的绘图。这个库的官网介绍[16]中写道，matplotlib是一个基于Python的平面图像库，并以高质量图像，跨平台性与可交互性而著称。本身绘图风格继承了matlab又比matlab绘图风格更多样化，又因为在机器学习教程中频繁出现，是我最先接触并决定尝试这个画图库。

由于Python端没有发现有类似擦除模式这种功能的绘图方法，所以我们决定保守一点，放弃了一次显示最近十次结果的方法，我们只一次显示一次3条子载波的图像就好。matplotlib尽管并不以实时交互式界面而出名，但是它确实提供了实时化的处理代码，下面某教程[17]提供的实时显示的Python代码：

**代码4.14 matplotlib实时更新数据教程示例**

import matplotlib.pyplot as plt

import numpy as np

import time

from math import \*

plt.ion()

plt.figure(1)

for i in range(2000):

’’’

在这里实现更新数据的代码，更新t(横坐标),m(数据)

’’’

plt.plot(t,m,'-r')

plt.draw()

time.sleep(0.01)

关键的问题实时性，我们测试了大概10000组数据，一开始非常快后面就开始卡顿，并且在最后快结束时停止运行并无响应。况且若要设置计时器来设置刷新间隔的方式，和我们希望系统越快越好的期望不符合。

在进一步调研中，我们发现用户们的评价[18]也对matplotlib的实时性不是很看好，“不要推荐matplotlib, 我要的实时性，…在官网写着它的目标是‘produces publication quality figures’”。用户们也推荐了几款很适合实时绘图的软件：PyQwt, PythonQwt, GuiQwt, PyQtGraph。这里解释一下，这四款应用库其实是一脉相承的[17]。PyQwt是最早的常用实时显示库，用户评价非常好，但是前几年突然宣布放弃维护，也因此没有支持新出来的PyQt5 (解释一下，PyQt系列库一般是Python系统默认的GUI显示窗口，也就是我们绘图需要实时显示的在的那个窗口，用于其开发的库)，也不支持Python 3.6以后的版本。为了解决这个问题，有人开发了PythonQwt，这是一种现在常用的实时绘图库。另外有个作者在PythonQwt上又做了功能扩展，这就是GuiQwt, 这个作者还有一款实时化库的作品，号称是强于GuiQwt，用户的评价也是全方位完胜GuiQwt，这就是PyQtGraph。所以我们最后使用的正是PyQtGraph。值得一提的是，在PyQtGraph的官网介绍[19]中也评价道，’ If you are doing anything requiring rapid plot updates, video, or realtime interactivity, matplotlib is not the best choice.’ (当你做任何需要快速图像更新，视频或者实时交互的东西的时候，matlibpolt都不是你最好的选择)，这也正验证了我们之前的表现性能差别。

用PyQtGraph需要先安装这个包，Python3下进入Python命令行，运行pip3 install pyqtgraph. 用PyQtGraph绘图有多种方式，分别是pyqtgraph.plot()，PlotWidget.plot()以及初始化一个窗口GraphicsWindow.addPlot()再对添加对象用set函数修改数据，最后一种正是我们采用的方法。由于PyQtGraph它是针对PyQt4和PyQt5进行开发的，所以为了实现实时化我们需要了解一些PyQt的GUI编程相关知识，在这里实现绘图窗口的消息循环的正是pg.QtGui.QApplication.processEvents()，功能是将未处理的图像即时进行处理（显示），以此来达到实时化的目标而最后pg.QtGui.QApplication.closeAllWindows()则是销毁窗口以为下一次连接绘图做准备。具体代码如下：

**代码4.15 Python端绘图处理代码节选 read\_bf\_file.py**

while True:

win = pg.GraphicsWindow()

p = win.addPlot()

p.setLabel('left', "SNR", units='dB')

p.setLabel('bottom', "subcarrier index")

p.setRange(xRange = [0, 30], yRange=[-150, 0])

p1 = p.plot(pen = 'r')

p2 = p.plot(pen = 'y')

p3 = p.plot(pen = 'b')while True:

...

while True:

...

while True:

...

if code == 187:

csi\_entry = read\_bfee.read\_bfee(Bytes);

...

csi = get\_scaled\_csi.get\_scaled\_csi(csi\_entry)

x = np.arange(30)

p1.setData(x, get\_scaled\_csi.db(abs(csi[0,0,:].flatten())));

p2.setData(x, get\_scaled\_csi.db(abs(csi[0,1,:].flatten())));

p3.setData(x, get\_scaled\_csi.db(abs(csi[0,2,:].flatten())));

pg.QtGui.QApplication.processEvents()

csi\_entry = []

...

pg.QtGui.QApplication.closeAllWindows()

4.4 系统效果

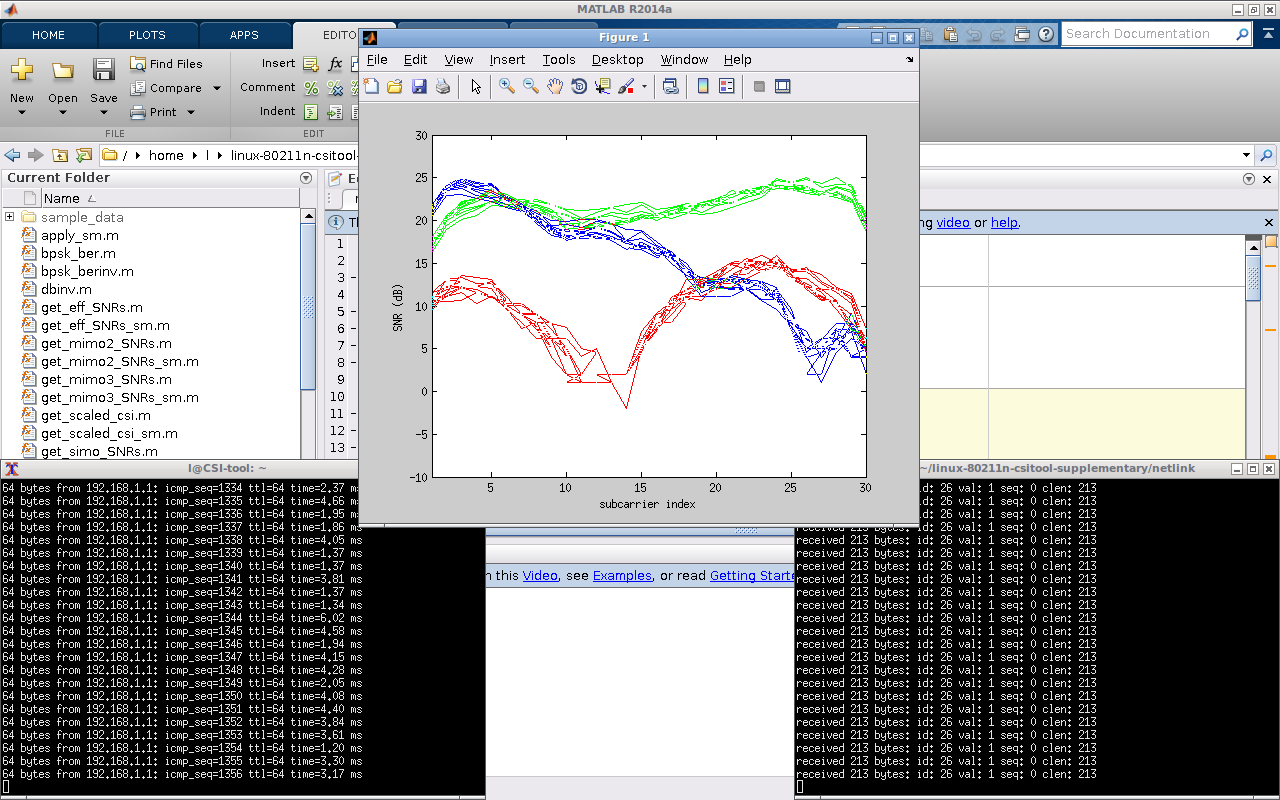
根据以上对于系统的设计及要求，我们给出我们测试环境下的配置，在该配置环境下系统运行无误。

发送端（CSI接收端）：装有Intel 5300网卡的linux系统的电脑

接收端（图像数据接收端）：装有MATLAB 2014b或以下版本的任意电脑或者装有Python3.6以上版本的任意电脑。不符合以上所述版本要求可能对性能有影响。

运行效果截图如下：

如图4.5所示，图上的三个颜色的线分别表示三根天线接收信号的CSI近十次数据。

图4.5 matlab端界面展示

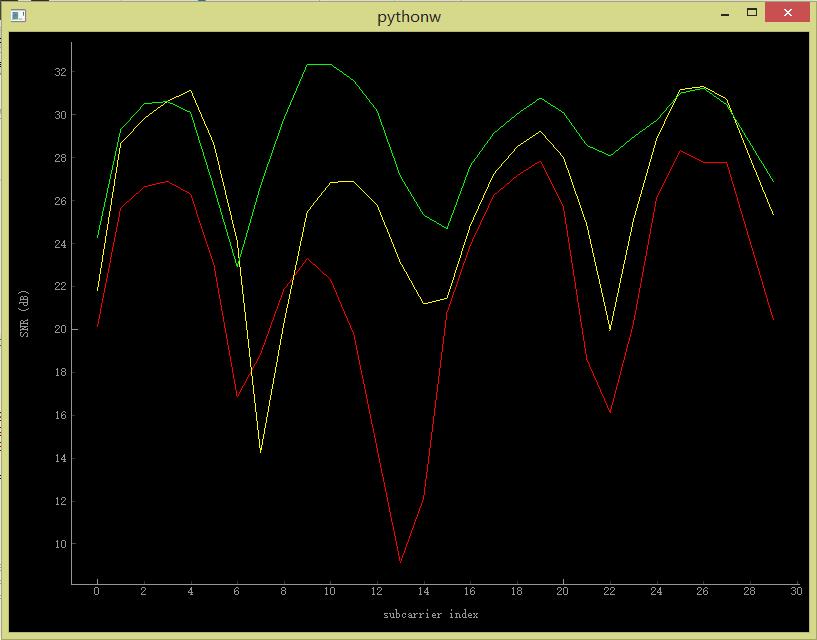


图4.6 Python端界面展示

如图4.6所示，Python端的数据显示在QT5的GUI界面上，三条颜色的线分别代表三根天线

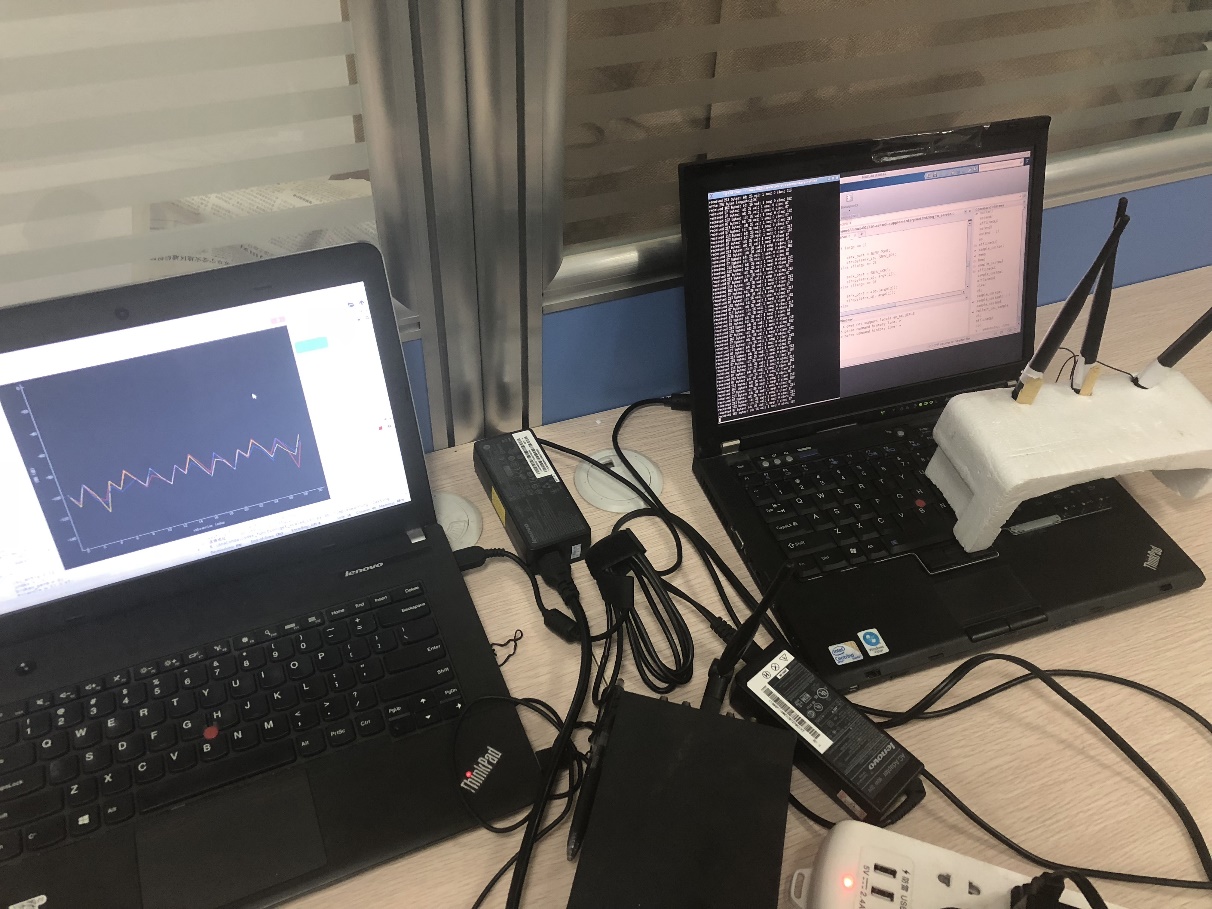


图4.7 Python端发送端与接收端绘图展示

如图4.7所示，我们的工作设备基本上是装有5300网卡的发送端和绘图的接收端，实时进行发送与绘图。

性能方面，我们测试了100包每秒和200包每秒的采样率。发送端100包每秒的速度发包时，matlab端和Python端基本运行正常，可以达到实时化的目标；200包每秒的时候，matlab端除了在大概采集数据5到10分钟会出现延迟和可能的报错现象之外，基本运行正常，Python端除了刚开始会有些许的卡顿之外，也是达到了实时化的目标。

5 总结及不足之处

该系统基于CSI-Tool，基本实现了matlab端和Python端实时显示CSI信息的功能。

这次的设计开发过程，让我对应用的迭代开发和二次开发有了很深的体会，也通过潜移默化的接触和老师学长的指导对基于CSI的室内定位技术有了长足的理解。这次的开发和学习中，我接触并深入了解了到了Octave，matlab，linux环境下的开发，以及对socket编程也有了一个深入的认知。matlab和Python对一开始的我而言更是不怎么熟悉的东西，通过此次开发我掌握了这两个平台开发的技能，也对脚本语言和编程语言之间实现的差异有了个直观的认识。整个开发过程，我从我的老师和学长学姐身上学到很多东西。

该系统也存在很多待改进的不足之处。我们没能开发出更为友好的界面和UI，基本还是基于命令行的界面。在实时性方面，对于200包每秒及更高的速度系统性能还亟待提高，对于matlab2014b以上版本的支持还是一个未解决的问题，而Python端方面的实时显示效果也应该可以做到更好。并且我们现在已经可以在非Intel 5300网卡设备上采集CSI数据了，比如ESP32的可穿戴设备，而对其也就是接收端的程序，我们还没有开放对5300网之外设备的支持。这些都是待改进的点，我们会在以后的版本中根据需求逐渐改进这套系统。

致 谢

感谢王雷老师，让我进入他的实验室，接触到无线室内定位和CSI相关的领域研究，作为我论文的导师，给我很多有意义的指导和建议，在各种方面都给了我很大帮助和提携。衷心感谢老师的支持。

感谢在完成该项目中给我很大支持的卢炳先学长，该项目需求一开始也是学长提出来的，在技术，学习以及工程方面给了莫大的帮助，指导与支持，没有他就没有这个项目的完成。

感谢实验室的学长学姐和我的朋友们，在论文完成过程中给了我很多鼓励和支持，陪伴我度过实验室的日常。

感谢我的父母，对我论文的工作表示了理解与支持，在家牺牲了旅行的时间陪我完成了论文初稿，对此我表示愧疚和感动。

也感谢阅读论文的你，花了一番时间理解我的工作，感谢你花费时间阅读这篇论文。

参考文献

[1] Adib F, Katabi D. See through walls with WiFi![J]. Computer Communication Review, 2013, 43(4):75-86.

[2] IBM. Pervasive Computing[EB/OL]. [2018.8.6] https://www.ibm.com/support/knowledgecenter/ssw\_ibm\_i\_73/rzahg/rzahgebpervasive.htm

[3] mazi. 究竟什么是RSSI[EB/OL].(2017.3.13)

https://blog.csdn.net/sinat\_22991367/article/details/61921223

[4] 姜芳, 杜志强, 谷晓鹏. 对多径效应引起测向误差的分析[J].通信对抗, 2004(4):30-32.

[5] Yang Z, Zhou Z, Liu Y. From RSSI to CSI: Indoor localization via channel response[J]. Acm Computing Surveys, 2013, 46(2):1-32.

[6] 黄亚萍. 基于TDOA和TOA的无线定位技术研究[D]. 南京：南京邮电大学, 2012.

[7] 王永良，陈辉，彭应宁，万群．空间谱估计理论与算法：清华大学出版社，2004

[8] Schmidt R O. Multiple emitter location and signal parameter estimation[J]. IEEE Transactions on Antennas & Propagation, 1986, 34(3):276-280.

[9] 杨铮, 刘云浩. WiFi雷达：从RSSI到CSI[J]. 中国计算机学会通讯.2014, 10(11):55-60

[10] Dan Halperin (dhalperi). linux-80211n-csitool-supplementary[DB/OL]. (2015.11.15) https://github.com/dhalperi/linux-80211n-csitool-supplementary/.

[11] 清华大学可信网络与系统研究所. Wi-Fi Radar[DB/OL] [2018.6.6] http://tns.thss.tsinghua.edu.cn/wifiradar/index\_chi.html

[12] Cai Chao. livestreamcsi[DB/OL].(2017.3.21) https://github.com/caichao/livestreamcsi.

[13] Daniel Haimanot. python\_wifi\_csi\_parsers[DB/OL] (2018.7.30) https://github.com/DanielHaimanot/python\_wifi\_csi\_parsers

[14] Daniel Haimanot. csi\_framework [DB/OL] (2018.7.30) https://github.com/DanielHaimanot/csi\_framework

[15] Dan Halperin (dhalperi). Linux 802.11n CSI Tool: FAQ, Things to Know, and Troubleshooting[EB/OL] [2018.6.6] http://dhalperi.github.io/linux-80211n-csitool/faq.html

[16] matplotlib. matplotlib version2.2.2[DB/OL].[2018.6.6] https://matplotlib.org/

[17] \_风起了\_. python中plot实现即时数据动态显示方法[EB/OL]. (2017.2.8) https://blog.csdn.net/u013468614/article/details/58689735

[18] V2EX. Python 的实时绘图库选哪个好[EB/OL] (2018.1.4) https://www.v2ex.com/t/420027

[19] yy123xiang. Python 实时绘图库小结[EB/OL]. (2018.2.10) https://blog.csdn.net/yy123xiang/article/details/79304023

[20] pyqtgraph. PyQtGraph - Scientific Graphics and GUI Library for Python.[DB/OL].[2018.6.6] http://www.pyqtgraph.org/

[21] Magnus Lie HetLand. Beginning Python: From Novice to Professional: Second Edition[M]. Berkerley. Apress L.P. 2014.