2019 年 SEU-Xilinx 国际暑期学校团队项目设计文档

(Project Paper Submission Template)

作品名称	高帧率 Respeaker 方位识别与语言识别及在 AWS 上的物联网应用
组员姓名	萧嘉乐　中山大学 电子与信息工程学院 袁霄亮　南京航空航天大学 自动化学院 孔昶　　中山大学 电子与信息工程学院
房间号及桌号	720 房间 26 桌

第一部分

小组成员分工

（各成员任务分配）

	姓名	任务分配
组长	萧嘉乐	搭建 Respeaker DOA 算法 PL-PS 系统
组员 1	袁霄亮	搭建 AWS GreenGrass Core 系统
组员 2	孔昶	调试基于 PS 的 DOA 算法与 STT 算法

第二部分

设计概述 /Design Introduction

（请简要描述一下你的设计：1. 功能说明；2. 所用到的设备清单）

本项目的设计是基于 PYNQ 开发板与 AWS 平台，利用 4-mic Respeaker 外设实现声源方向检测算法 DOA（Direction Of Arrival）与语言识别算法 STT（Speech To Text），并将实现的结果在 PYNQ Jupyter 上显示出来。DOA 算法的实现是部署在利用 RTL 搭建起的 PL 上，利用硬件加速使 DOA 算法的处理速度比 I2S 协议采集的速度更快，以实现 DOA 声源定位的实时性。

在应用上，声源检测算法 DOA 是利用 Respeaker 分布在 4 个方向的麦克风采集到的声音数据，来分析判断出声源的方向，由此可在不同的应用场景下做不同的相应。比如在安防领域，摄像头可以根据声源方向控制摄像头的朝向方向；在机器人领域可以作为声控机器人的控制信号；此外还有视频会议、目标定位、助听装置、发生物体识别等领域都可以有应用。而语言识别算法 STT 是利用 Respeaker 对声音进行录制，然后上传到百度智能云服务进行语音文字识别，得到结果后在 Jupyter 上显示出来。STT 算法的应用更加广泛，比如可以作为语音转文字输入，解放用户的双手；通过语音进行人机交互（如 Siri），声控机器人应用等等 。

同时我们基于亚马逊云服务 AWS 平台，搭建一个 IoT 系统。我们在 PYNQ 开发板上部署 GreenGrass Core 以作为另外两个设备（Controller 和 Sensor）通信的处理核。部署完成后，Controller 可以向 Core 发出外设的控制指令，然后 Core 再发送信号到 Sensor 核进行采集数据或者控制其他外设，以实现一个 Things to Things 的 IoT 概念的控制。我们将 DOA 算法处理得到的方位结果作为控制指令，经过 GreenGrass Core 的处理对另外一个设备的外设如 LED 灯进行控制，从而实现一个基于 AWS IoT 的应用。此 IoT 系统实现的 Demo 可以作为各个场景下 IoT 的应用，比如通过智能家居、车载电子中通过声音、手势控制各种电子设备。

设备清单

材料名称	数量
PYNQ-Z2 开发板	4
Respeaker	1
网线	3
Base Shield	1
无线网卡	1
LED 点阵	1

第三部分

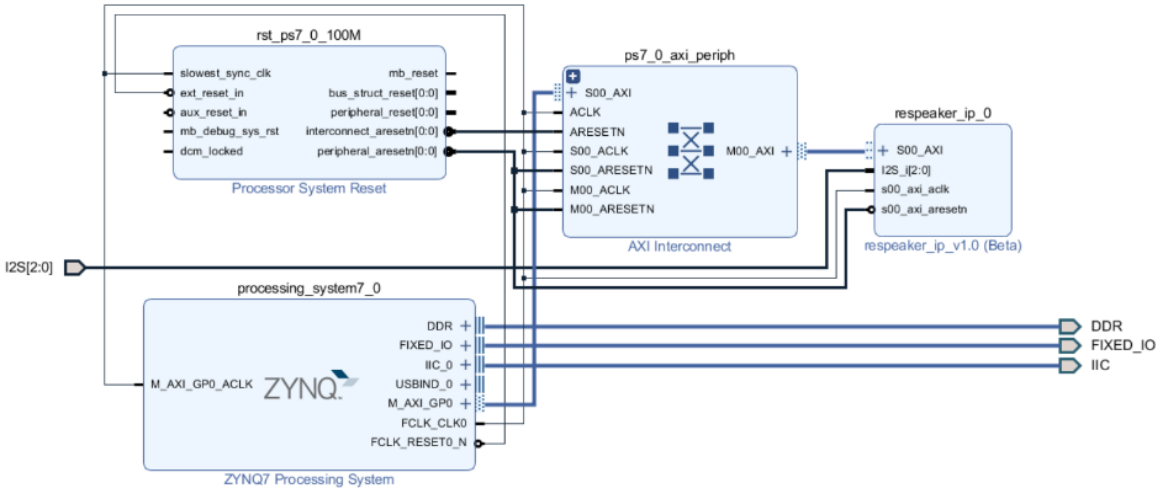
详细设计 /Detailed Design

（请详细说明你作品要实现的所有功能以及如何组建系统以实现该功能，还包括为实现该功能需要用到的所有参数和所有操作的详细说明，必要的地方多用图表形式表述）
如因文档篇幅限制无法详述，可提供附件。

本设计分为三大部分，分别是 Respeaker DOA 算法与 STT 算法在 PS 上的实现；Respeaker DOA 算法基于 PS-PL 系统的搭建； AWS GreenGrass Core 的部署与 Demo 设计。

（1） Respeaker DOA 算法与 STT 算法在 PS 上的实现

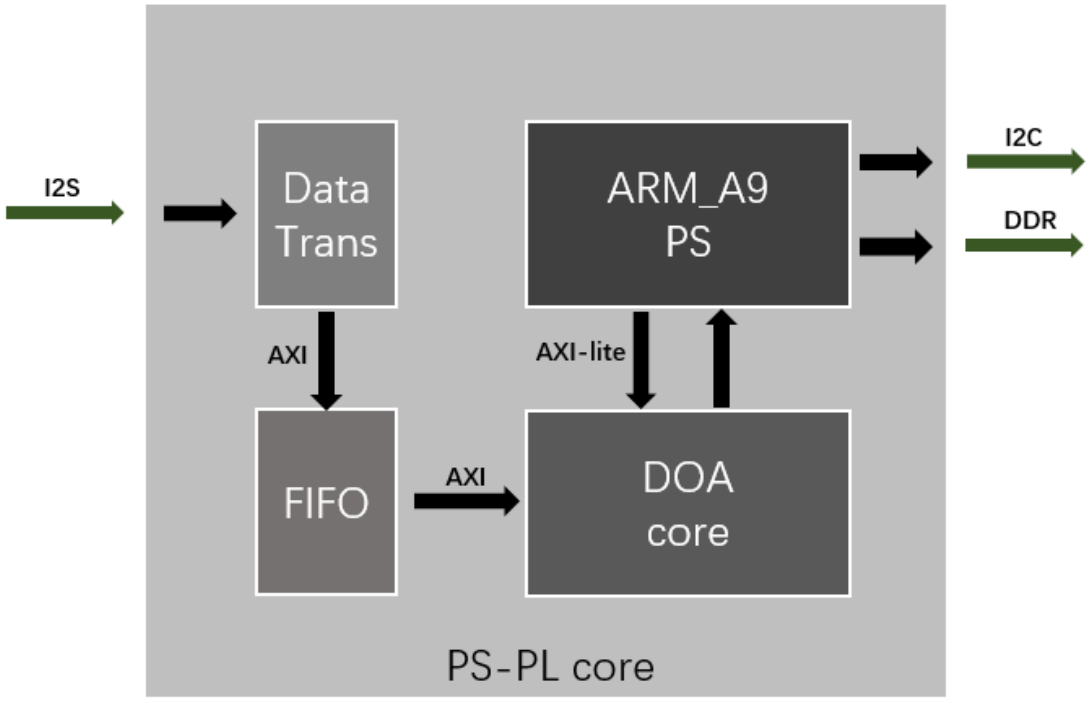
我们利用已提供的.bit 文件和.tcl 文件，将已经设计好的 respeaker_ip 工程烧录到 PYNQ 开发板中。然后基于现有的 DOA 算法和 STT 算法和，利用 PYNQ 提供的 Jupyter 开发环境，将 DOA 算法与 STT 算法用 Python 实现出来，编译成程序放在 PS 上运行，并实现方向定位功能和语音转换为文字的功能。此工作可以得到一个完整的可实现方向定位与语音转换文字的完整设备，可以投入应用中。下图为已提供的工程的 block diagram



（2） Respeaker DOA 算法基于 PS-PL 系统的搭建

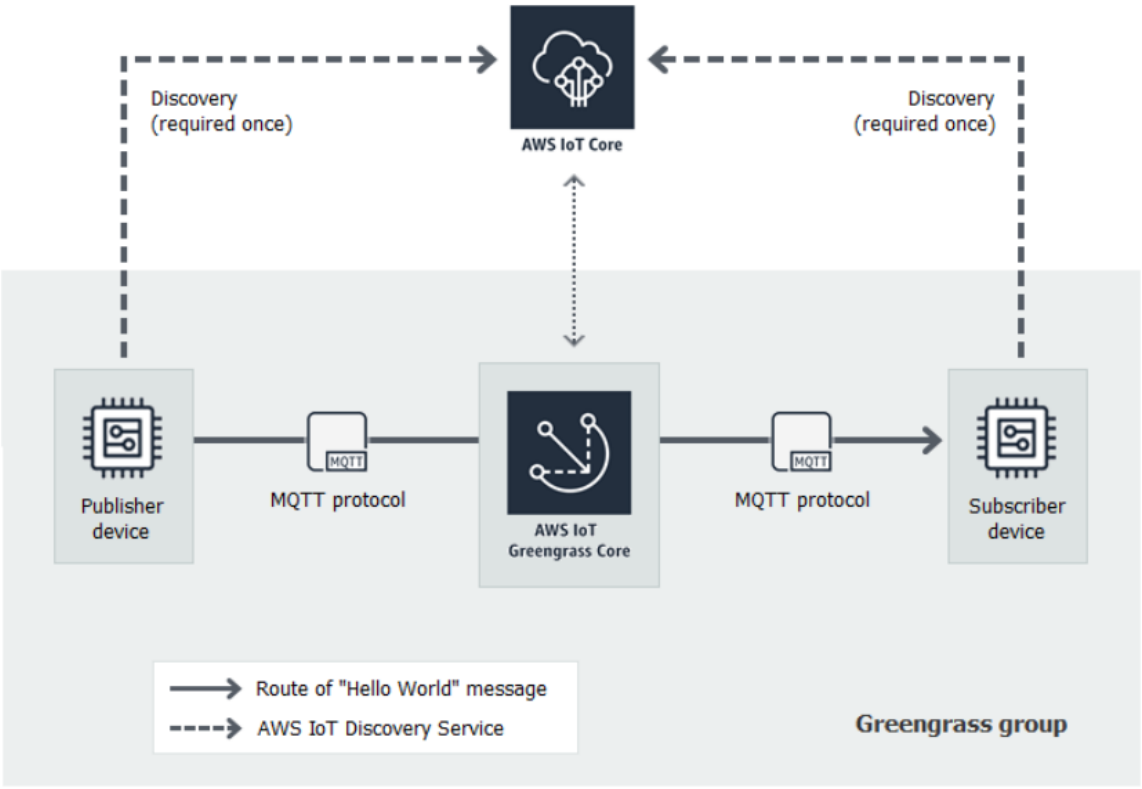
该系统将实现 Respeaker 通过 I2S 协议实时提取音频数据，将数据缓存后传输到 PL 中的 DOA_core 中进行算法处理，得到方向信息。随后再利用通过 Jupyter 控制的 PS 核采集 PL 的 DOA 计算结果，在 Jupyter 显示出来。

我们首先学习已有的在 PS 上实现的 DOA 算法，将整个计算流程分解为可用 Vivado IP 分别实现的小步骤，利用 Vivado Block Diagram 搭建起可实现 DOA 算法的 RTL 数据通路，得到一个专用处理 DOA 算法的 IP。随后将已有的 I2S_decode IP 进行 AXI-Stream 协议的封装，以实现更高的数据传输速度。然后将两者组合成专用 DOA 算法处理 PL，与 Zynq7 组成一个 PS-PL 处理系统。系统完成后，我们可以在 Jupyter 中用 Python 代码直接通过 PS 向 PL 获取 DOA 算法的处理结果，然后在 Jupyter 中显示出将结果。由此，我们将 DOA 算法的实现平台从 PS 转移到了 PL，利用 PL 更快的数据缓存和算法处理速度更快的得到方向数据，并避免了 PS 通过 MMIO 读取数据的带宽瓶颈，以实现高帧率的音频数据采集和 DOA 算法定位声源方向。



（3） AWS IoT GreenGrass 的部署与 Demo 设计

该设计我们将根据 AWS 提供的 GreenGrass 入门教程指南，下载安装 AWS IoT GreenGrass 核心软件，配置 GreenGrass 上的核心函数，以在 PYNQ 上部署一个可以作为 IoT 处理核的 GreenGrass Core。然后通过 AWS 控制台的 GreenGrass 组与已经部署好 Core 的设备进行通信。验证 GreenGrass Core 可以通信之后，我们再利用另外两个 PYNQ 开发板分别作为 Controller 和 Sensor，让三块 PYNQ 连上局域网后，由 Controller 发送信号到 Core，Core 处理完后再控制 Sensor 产生动作如点亮 LED 灯，实现一个 IoT 概念的控制通路。然后我们将前面实现的 DOA 算法得到的声源定位结果作为 Controller 的控制条件，控制 Sensor 的动作。这个设计得到了一个完整的 IoT 概念的控制链，并实现在另一个设备用 LED 灯显示声源方向的应用场景。



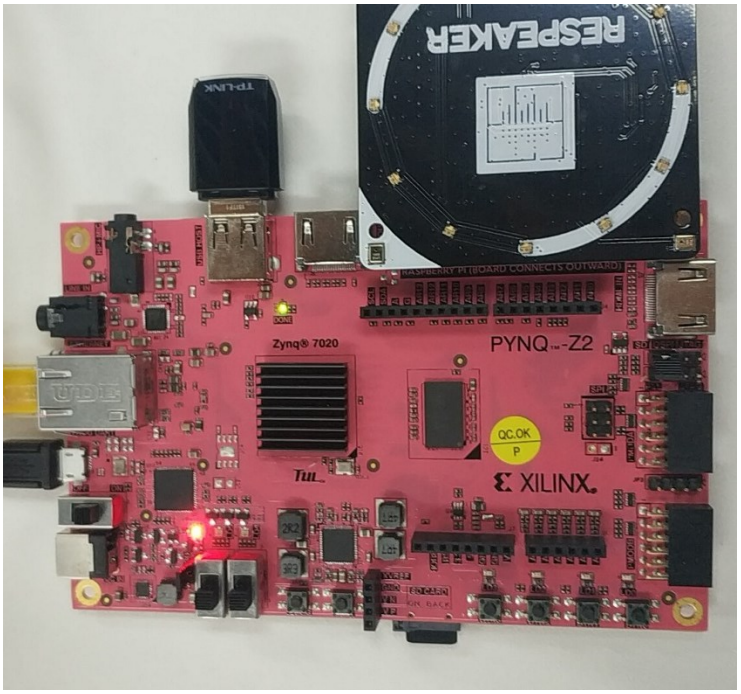
第四部分

完成情况及性能参数 /Final Design & Performance Parameters

（作品完成情况，附照片/视频链接）

（1） Respeaker DOA 算法与 STT 算法在 PS 上的实现

搭建起了可以运行 DOA 算法和 STT 算法的硬件系统，如下图。



我们成功完整地实现了 DOA 算法识别声源方向和 STT 算法识别语音内容。

循环执行 DOA 算法可以持续进行声源方向的检测，检测速度为 2 次/s。减少每次检测的采集的音频数据量可以提高检测速度。在误差合理的范围下，检测速度最高可达 5 次/s 左右。

STT 算法可以实现将时长为 5s 的音频进行语音识别并转换为文字，要求音频的语言为英语。识别的速度限制在百度智能云的处理速度与返回速度上。可以通过增加录制的音频时长来达到语句层面上的识别。下图为识别的结果“hello hello hello”

8. Recognize

recognize the speech and print the result

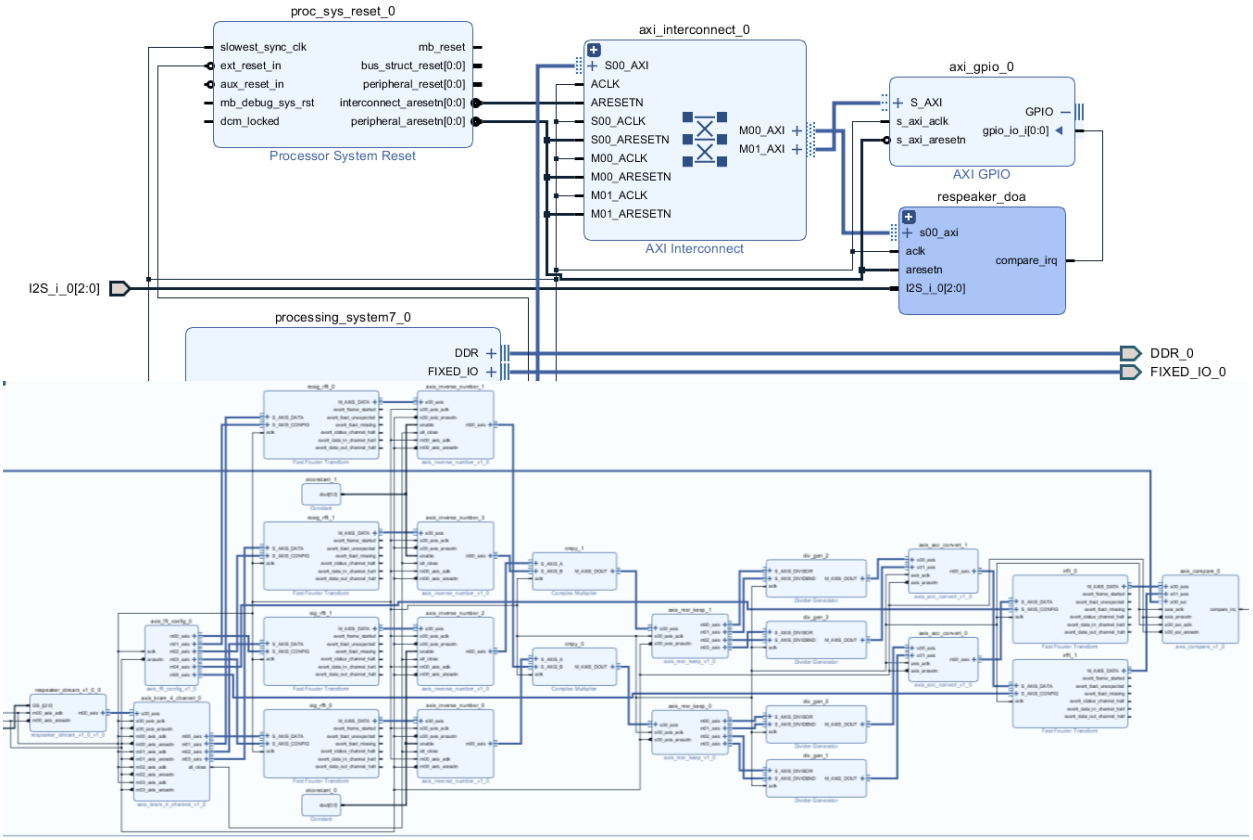
```
In [20]: result = client.asr(data[0], 'pcm', 16000, {
             'dev_pid': 1737,
         })

         print(result)
```

```
{'corpus_no': '6713431216646348018', 'err_msg': 'success.', 'err_no': 0, 'resul
t': ['hello hello hello'], 'sn': '942363796115630925l1'}
```

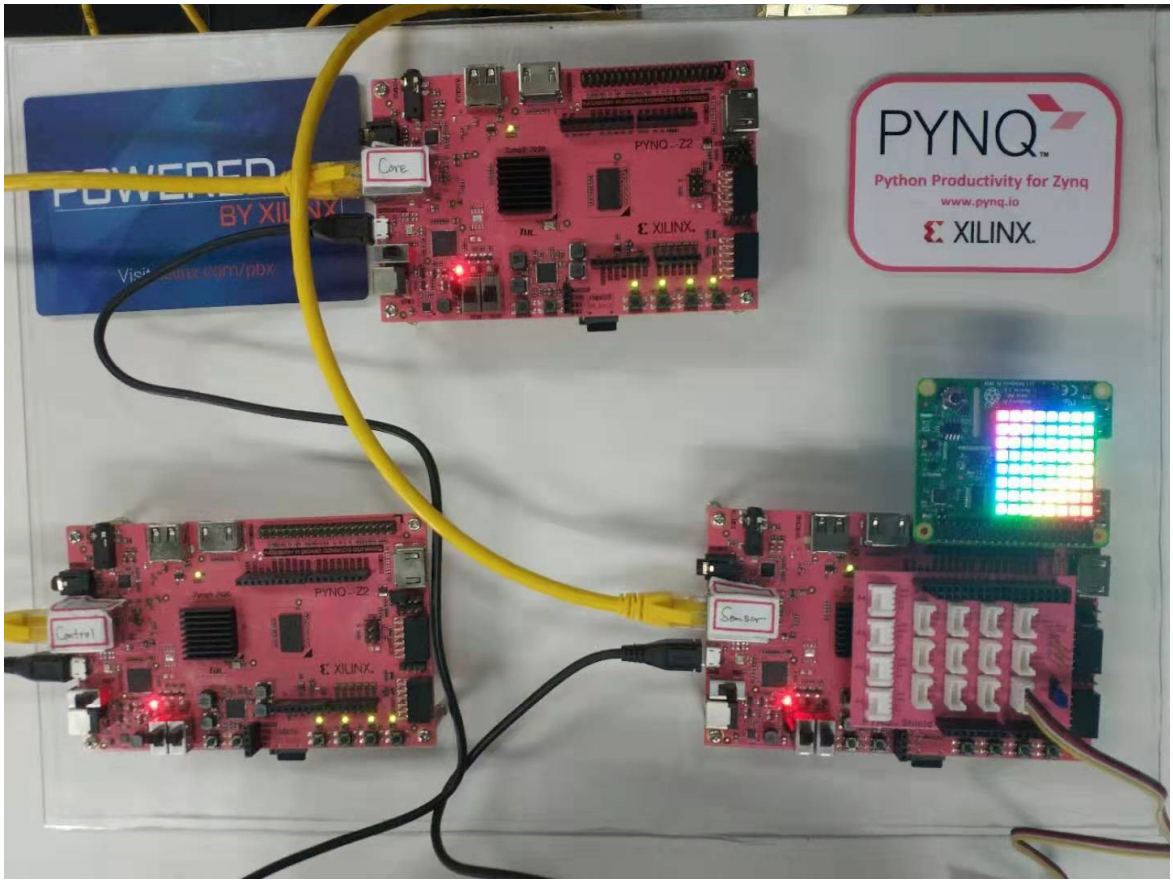
（2）Respeaker DOA 算法基于 PS-PL 系统的搭建

成功利用 Vivado Block Diagram 搭建起了专用处理 DOA 算法的 ip，仿真测试可以实现 I2S 数据缓存、DOA 算法中的复数乘法、复数求模、快速傅里叶变换等功能并输出声源方向检测结果。PL 处理一组数据到得到一个方向需要 30ms 的时间，远低于原来使用 PS 实现的 500ms，检测速度大幅提高。此外，PL 的 DOA 算法 IP 的处理速度远高于采集速度，消除了计算处理的瓶颈，实现实时 DOA 算法检测声源方向。下图为修改后的 block diagram。

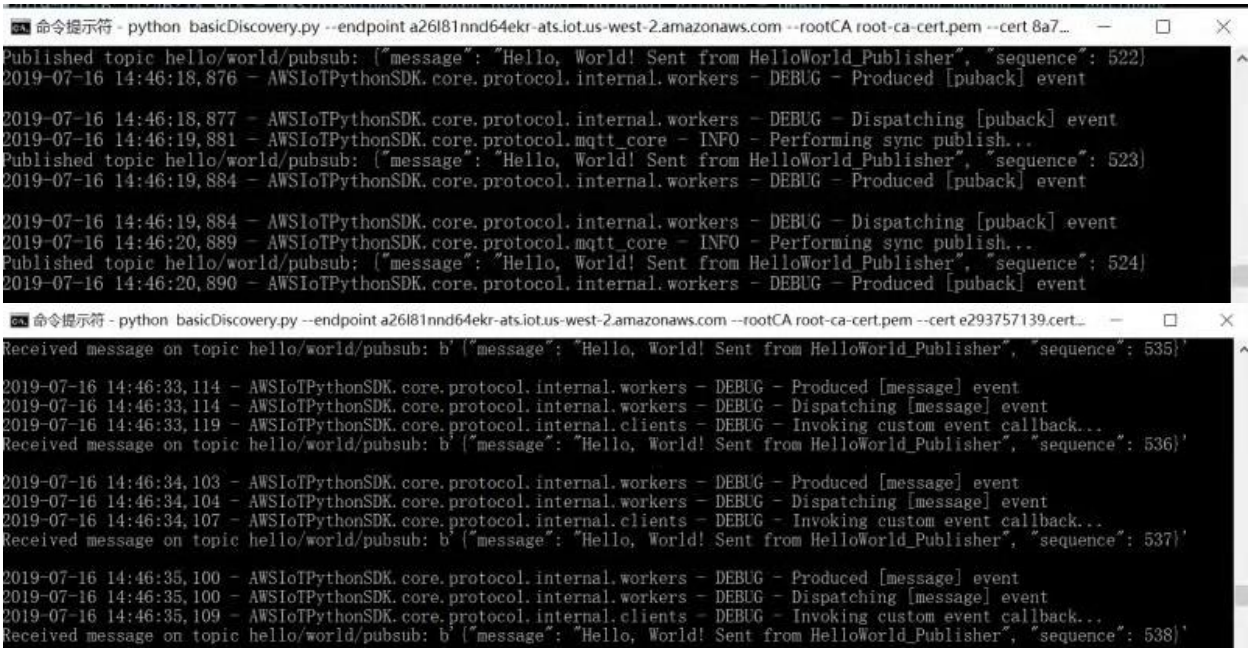


（3）AWS IoT GreenGrass 的部署与 Demo 设计

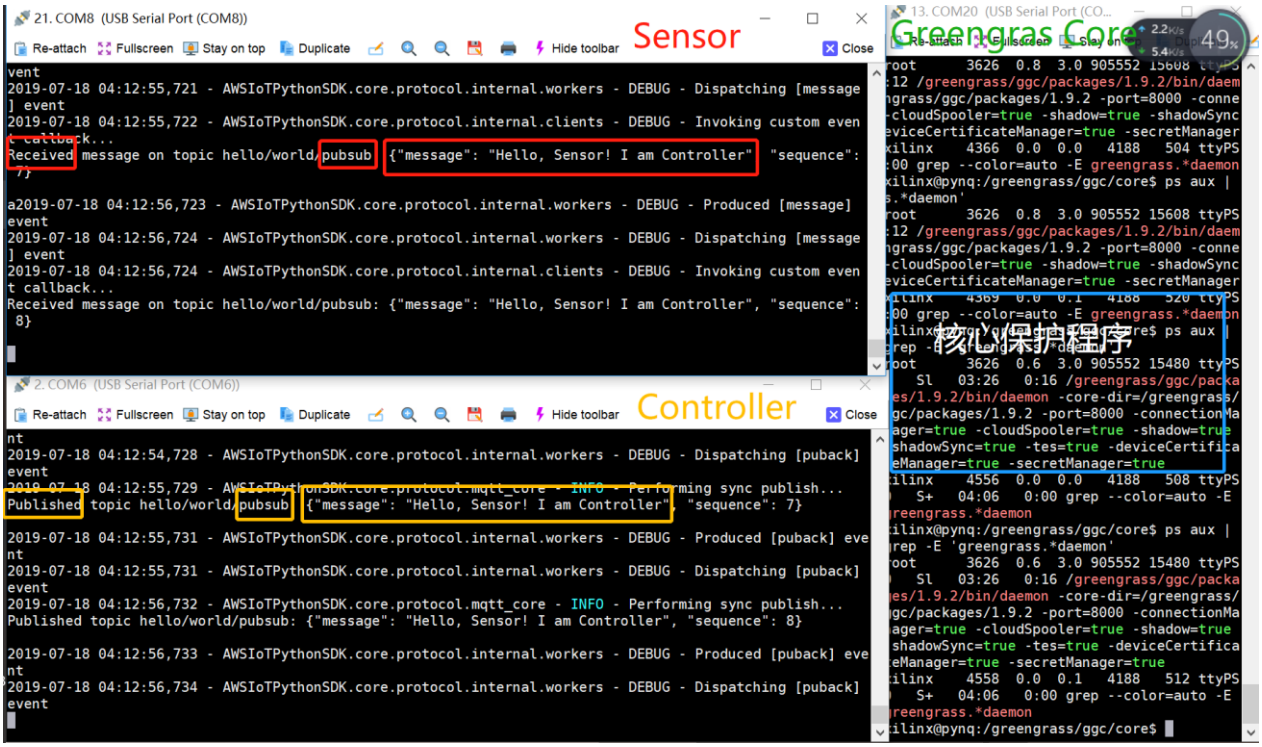
搭建起了系统，如下图：



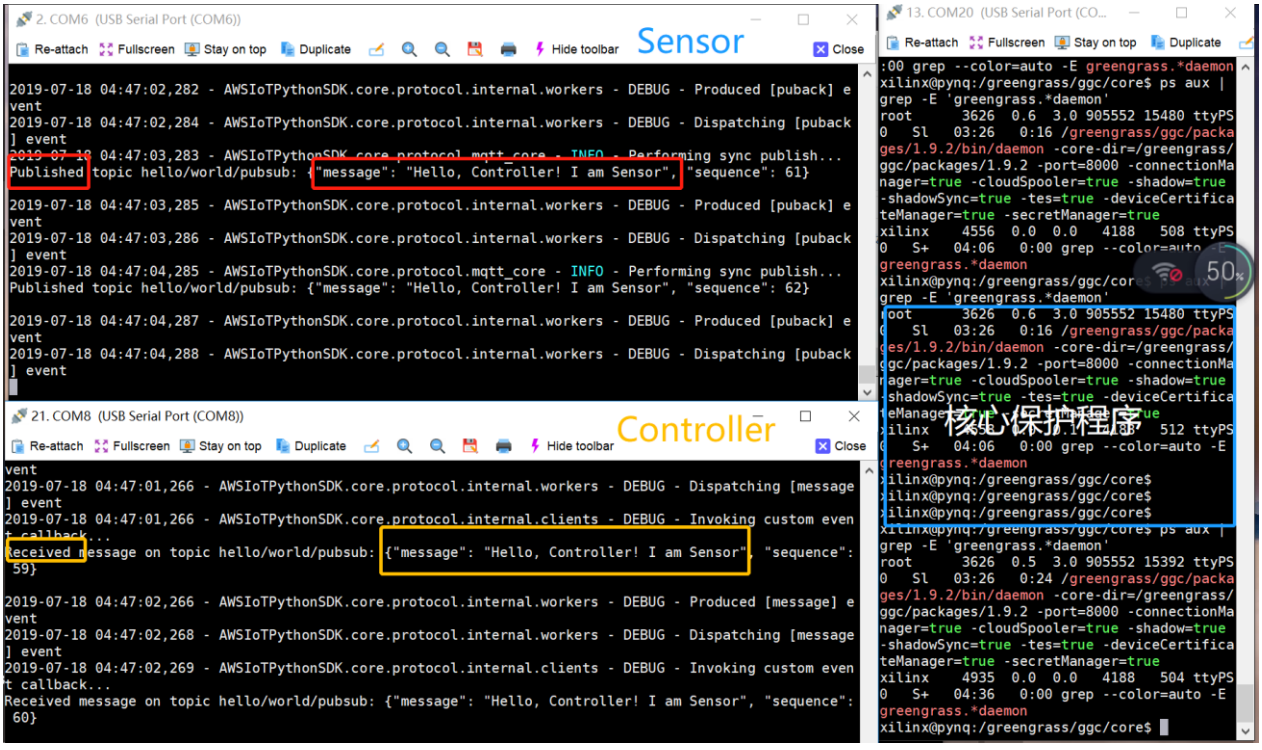
成功部署在 PYNQ 上 GreenGrass Core，并可以让 PC 作为发射端和接收端，实现数据从发射端传输到 GreenGrass Core，并通过 GreenGrass Core 处理并传输到接收端的功能，下图为在 PC 配置发射端和接收端，通过 PYNQ 上的 GreenGrass Core 进行中转的结果。



在另外两个 PYNQ 板上设置好 Controller 和 Sensor 的环境后，可以通过在 PC 上用命令行，在 Controller 发出“Hello，Sensor! I am Controller”字符串，通过 GreenGrass Core 中转传输，发送到 Sensor 并接收到“Hello，Sensor! I am Controller”。



Sensor 接收到信息后，可以返回“Hello，Controller，I am Sensor”。



附录：视频的百度云链接
链接：https://pan.baidu.com/s/19Dnv4GRFuOzNrLMx2mK1Fg
提取码：1mjo

第五部分

项目总结 /Conclusions

(项目中所用到的知识点，项目收获及心得)

纵观整个项目，我们三人分别负责不同的工作，综合下来用到的工具和知识点繁多。在 PS 实现 DOA 和 STT 算法的过程中，我们用到了 PYNQ 相关的各种知识点，比如 Python 编程，Jupyter 环境运行，ubuntu 环境下的网络配置，在 ubuntu 上安装无线网卡等知识点。STT 算法需要用网络访问百度智慧云的语音识别云服务，因此也学到了云服务的一些配置方法。

在 Vivaodo 中搭建 block diagram 是个繁琐的工程。工程的开始首先要对流水线的延迟和瓶颈进行评估，分析出搭建出来的 PL 速度可以提升多少。然后我们需要将用 Python 实现的 DOA 算法剖析，分解成若干个用 Vivado IP 实现的步骤之外，这个过程我们要学习原本不熟悉的 Python。此外在封装 IP 中我们还学习了 IP 封装的各种知识，比如接口转换等等。

AWS 的搭建则是用到 AWS 的云服务的各种配置及 ubuntu 的知识。

第六部分

源代码/Source code

（作品源码 Github 链接，github 操作步骤见 *Github 项目上传简单教程*）

Github 链接

https://github.com/CarlosShiu/Pynq-Respeaker-DOA-Vivado.git

第七部分（技术总结/心得/笔记等分享）

（可分享的技术文档附件或已发表的文章链接）

关于算法硬件化的过程，DOA 算法的过程在 Python 上运行均是以浮点的形式运行，而若要实现算法的硬件化，那则是需要将浮点运算转化为定点运算以提高计算效率。正确的做法是应该在软件上模拟硬件化中定点处理时所带来的精度误差，并判断是否会影响最终结果，同时软件端提供一个可供硬件 Debug 的逐点数据。但是本次项目由于时间限制，并没有进行如此详细的过程，对于精度的转化也只是使用直接截去尾端数据的方式。且由于其中用到 FFT IP 核仿真时间过长，也没做到逐点 debug，最终实现的计算结果有错误。