# 第五组通原实验报告

## 一 概述

我们组基于 PlutoSDR[1]，设计并完成了以下两个实验目标.
一: FM实现音频文件的传输.
我们利用FM模块和 MATLAB 中的 `Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio` (下面简称 `Communication Toolbox`) 插件所提供的已经封装好的函数, 成功传输了一段音频文件. 在传输过程中,我们能完整做到自发自收和单收单发这两种模式.
二: 基于BPSK的文字传输系统.
在 Analogodevice 公司所提供的 libiio 硬件库源代码以及从网上找到的 BPSK.demo（Matlab实现）的基础上，我们设计并实现了可靠的无线通信传输系统。最终的效果是: 我们既能在一台 Pluto 设备上实现自发自收，也在此基础上，实现了两台 Pluto 设备的相互传输。在实际的无线信道中，可以传输任意长度的文字。同时, 我们为了传输的可靠性，使用了简化的停止等待协议。

## 操作系统： Manjaro(arch linux)

## 二 实验原理

### 2.1 PlutoSDR 的简单介绍

> PlutoSDR 通常在主机（x86）Windows，Linux或Mac或嵌入式Linux平台（Raspberry Pi，Beaglebone，96boards.org）上使用MATLAB，Simulink，GNU Radio或自定义C，C ++，C＃或Python环境与RF信号进行交互。通过USB插入您喜欢的嵌入式Linux平台）。

### 2.2 FM音频传输的设计思路

1. Pluto生产商 Analog 在 MATLAB 中提供了 `Communications Toolbox`这样一个插件. 在这个插件中集成了很多的函数, 可以很方便的对于 Pluto 的属性, 收发等进行开发.
2. 其次, 在 MATLAB 的 SIMULINK 库中, 带有 FM 模型, 也就意味着我们也可以使用这个封装好的模块, 利用 FM 进行收发测试.
3. 设计流程: 发送音频 --> FM调制 --> PLUTO发送 --> PLUTO接收 --> FM解调 --> 播放音频
4. 结果验证: 发送的音频和接受的音频播放一致, 无刺耳杂音或丢失即可

### 2.3 BPSK调制的原理

> BPSK (Binary Phase Shift Keying)-------二进制相移键控。是把模拟信号转换成数据值的转换方式之一，利用偏离相位的复数波浪组合来表现信息键控移相方式。
> BPSK使用了基准的正弦波和相位反转的波浪，使一方为0，另一方为1，从而可以同时传送接受2值(1比特)的信息。
> 由于最单纯的键控移相方式虽抗噪音较强但传送效率差，所以常常使用利用4个相位的QPSK和利用8个相位的8PSK。

### 2.4 BPSK文字传输系统

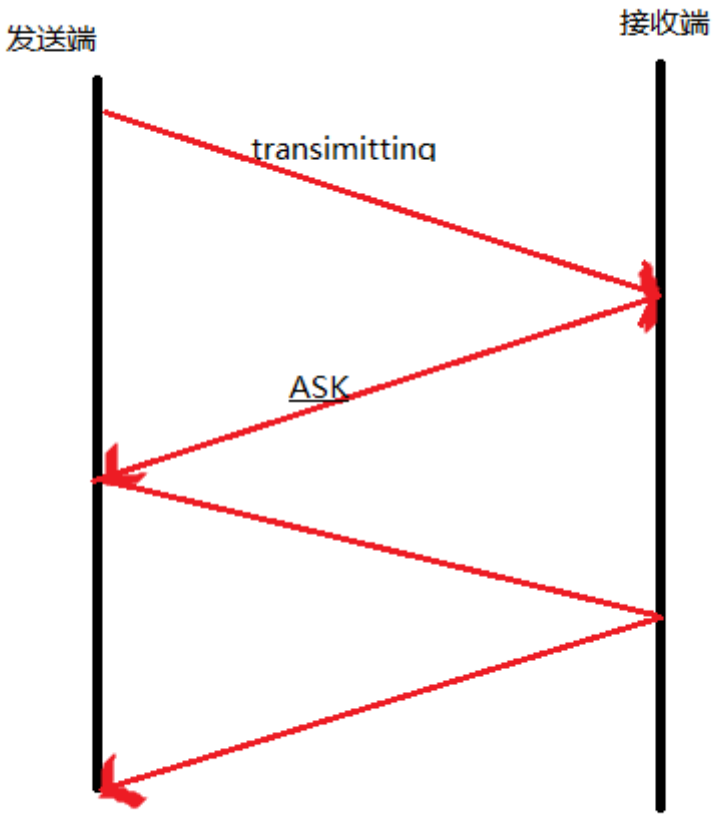
1. 我们使用 MATLAB+Libiio 的方式, 实现了基于 BPSK 的文字传输, 为了行文方便, 我们以自发自收为例阐述我们的设计.
2. 设计思路: 发射段发送一段文字 --> BPSK调制 --> Pluto发送 --> Pluto接受 --> BPSK解调 --> 显示结果.

3. 结果验证: 使用不同长度, 含有不同字符的文字串, 比较发送和接受的结果. 如果有丢失或者错误, 则分析误差原因.
4. 对于很长的字符串,为了保证传输的精确度和准确度, 我们设计了**停止等待协议**; 同时, 为了确保显示效果, 我们使用了**窗口滑动协议**来达到更好的显示效果.

### 2.4.1 停止等待协议.

停止等待协议的主要作用是处理很长的发送字符集, 我们可以将其分割成规定长度的数据帧, 来确保发送和接受准确度. 同时, 这是我们实现 `./code/BPSK/stop_wait.m` 的主要思路.我们设计的停止等待协议是这样的:

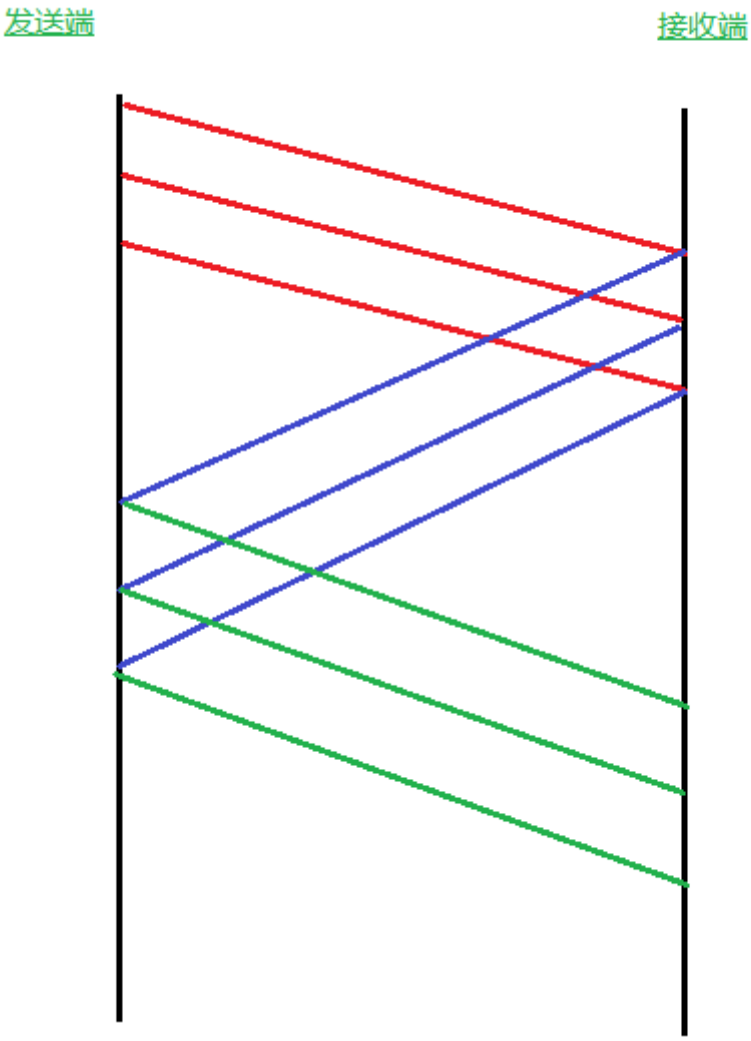
1. 帧序号有 0 和 1 两种，在两种之间跳变
2. 每帧的前 3 个字符用作帧序号和其他控制信息。即有效信息从 60 个字符减少到 57 个字符
3. 发送方每次发送一个帧，并开始计时
4. 接收方如果收到一个帧，且该帧的序号是自己期望的，则把收到的帧序号返回，并保存相应数据
5. 发送方如果没有到规定的超时时间（这里是 10 秒）：
    - 持续监听返回值
        - 如果收到自己刚刚发送到的帧序号
        - 发送下一帧
    - 否则
        - 继续监听,直到超时
6. 发送方如果超时,重新传输刚才的帧
7. 停止等待协议原理图:



### 2.4.2 滑动窗口协议.

这是我们实现`./code/BPSK/slide_window_recv.m`的主要思路.我们设计的滑动窗口协议规则如下:

1. 发送方和接收方窗口大小均设置为 5
2. 为了避免序号回滚时引起歧义，序号的个数设置为窗口大小的 2 倍，即取 1- 10
3. 发送方维护发送窗口，在等待发送的数据数组上滚动
4. 接收方维护接收窗口，判断收到的数据帧号是否落在自己的窗口内，并酌情保存数据
5. 接收方发现有一段序号连续的数据后（顺序正常），滑动自己的窗口，并返回这些连续的数据中的最大序号
6. 发送方一次发完从上次发送的位置到窗口末尾中的所有数据，并开始计时
7. 发送方如果没有到规定的超时时间（这里是 10 秒）
    - 持续监听返回值
    - 如果收到的帧序号落在自己的发送窗口中，且与之前收到的帧序号不一样
        - 滑动窗口并开始发送下一串数据
    - 否则
        - 继续监听，直到超时
8. 发送方如果超时，重新传输窗口中的所有数据
9. 滑动窗口协议原理图:




### 2.4.3 需要使用的主要函数

1. 收发函数部分. `./library/matlab/iiio_sys_obj_matlab.m` 中,函数 `function ret = stepImpl(obj, varargin)` 实现了发射和接受数据.具体格式如下:

```
% Implement the data transmit flow
writeData(obj.libiio_data_in_dev, varargin{1});

% Implement the data capture flow
[~, data] = readData(obj.libiio_data_out_dev);
```

   其中, `obj` 是代表了一个对象,而这个对象就是我们使用的 Pluto 开发板,这个开发板有很多的属性(Properties), 我们就可以使用`obj.libiio_data_in_dev` 这样的形式来设置我们的开发板. 使用这样面向对象的代码能极大方便我们的开发流程.

2. 收发数据设置. 在 `./code/matlab/BPSK/transmitter/bpsk_tx_func.m` 中,原作者设置了要发送的128*4 = 512 bit 的数据。其中前 480 比特是有用信息（60 个字符），后 32 位用作循环冗余校验（CRC）。由于在其他地方设置了收发的数据长度而不好更改，这里我们就用了这 512 比特作为一个数据帧。为了支持任意长度的数据，需要把给定的不足 60 个字符的消息结尾补没有意义的空白字符填充到 60.

```
function txdata = bpsk_tx_func(msgStr)
%......
    for k = length(msgStr)+1 :60
        msgStr = [msgStr, char(0)];
    end
%......
end
```

   当发送长消息时，以 60 个字符一切割，在接受到之后只需要抹去最后的空白字符就可以恢复原始消息。

# 三 实验过程

## 3.1 实验流程概述

我们的实验流程主要是分为三部分, 环境安装和配置; 基于FM的音频传输; 基于BPSK的文字传输.从接受任务,我们首先在windows下进行了 MALTAB 环境配置, 主要是安装插件和学习参考的例程. 然后,我们实现了基于 FM 音频传输, 从而对于开发的流程有了更加深刻的认识. 在此基础上, 我们又从公司的 wiki 文档上获知了可以使用 libiio + MATLAB 的方式实现 BPSK 的传输, 我们便作为主要的方向. 但是很遗憾, 在windows下我们没有成功地搭建环境, 所以我们转向使用 linux 作为主力系统, 调试成功. 下面是对于每一部分的详细阐述.

## 3.2 环境搭建和配置

1. 在Matlab上安装 `Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio`
2. 安装的 libiio 库文件： windows 和 linux 下配置的方法稍有不同
   - windows. 到 analog官网[2] 上下载相应的 `.exe` 文件安装即可
   - linux. 需要从 analog 中下载源码,自己在本地编译(编译过程不在此展开,请自己查阅官方 wiki 文档实现).
3. 配置 MingW -- C语言编译器
   - windows. 下载 MingW --> 设置环境变量 --> 在 Matlab 相应配置
   - linux. linux 下自带 gcc+gdb  编译环境,所以不需要配置

## 3.3 基于FM的音频传播

我们在上面实验思路的指导下, 实现了单收单发和自发自收. 为了行文方便, 下文以自发自收为例, 解释主要模块的使用和作用.

1. 首先, 我们需要使用 `Communication Toolbox`, 设置 Pluto 的参数. 这样做的目的主要是设置发送和接受参数, 确保能顺利完成收发. MATLAB 中的实现方法如下:

```
%% 发射初始化
txPluto = sdrtx('Pluto','RadioID','usb:0',...
    'CenterFrequency',92.4e6,...
    'Gain',-0,...
    'ChannelMapping',1,...
    'BasebandSampleRate',228000);
txPluto.ShowAdvancedProperties = true;

%% 接收初始化
sigSrc=comm.SDRRxPluto(...
'CenterFrequency',92.4e6,... %The channel you want to listen to (Hz)
'GainSource','Manual',...
'Gain',50,... %can control volume
'ChannelMapping',1,...
'BasebandSampleRate',228000,... %228000
'OutputDataType','single',...
'SamplesPerFrame',45600*5/2); %5.2:发射有少量间断;5似乎行;4.2:接收有间断;4.8:接收过一段时间间断一次
```

   - **注意:** 对于 Pluto 的设置, 主要是设置 `CenterFrequency` 这个频率相同, 否则不能接受到发送的音频信号.

2. 使用 MATLAB 的 FM 模块. FM 有两部分组成--发送和接受模块.

```
%% 发射部分
% 加载音频并对音频进行采样等数字处理
afr=dsp.AudioFileReader('Scarborough Fair.flac','SamplesPerFrame',44100/2);
adw = audioDeviceWriter('SampleRate', afr.SampleRate);
% FM调制
mod=comm.FMBroadcastModulator('AudioSampleRate',afr.SampleRate, ...
'SampleRate',txPluto.BasebandSampleRate,'Stereo',false);

%% 接收部分
% Create FM broadcast receiver object and configure based on user input
fmBroadcastDemod = comm.FMBroadcastDemodulator(...
    'SampleRate', 228000, ...
    'FrequencyDeviation', 75e3, ...
    'FilterTimeConstant', 7.5e-5, ...
    'AudioSampleRate', 45600, ...
    'Stereo', true);

% Create audio player to play the received music.
player = audioDeviceWriter('SampleRate',45600);
```

3. 评价

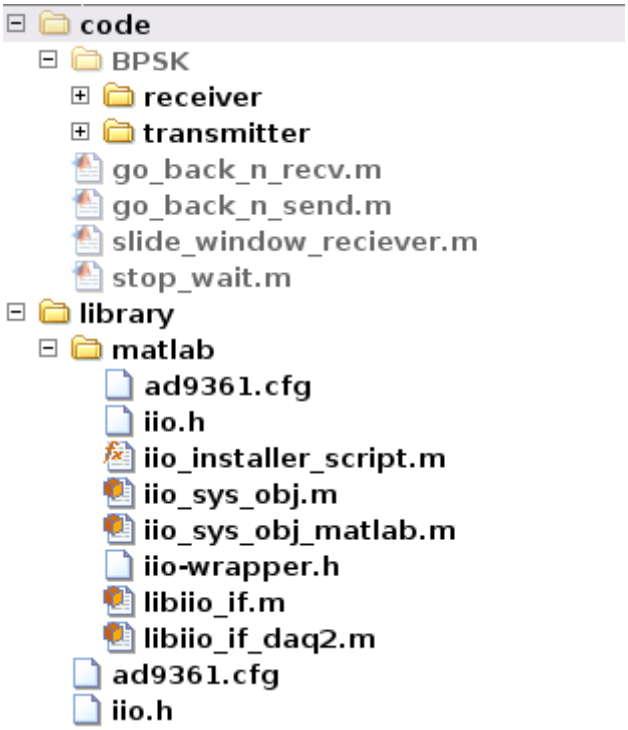
   - 使用已经封装好的函数和面向对象式编程的方式, 确实极大简化了开发的流程, 而且很容易见到现象. 我们在基本上阅读开发文档的基础上, 便可以实现.

    - 由于已经封装好了, 对于内部的一些原理和现象便变得不再明显.
4. 不足: 由于我们只能通过去听这种方式来验证是否发送成功, 没有严谨的方式(如通过画出功率谱等数学方式)来验证是否成功, 这也是我们这个实验的一个很大的不足.

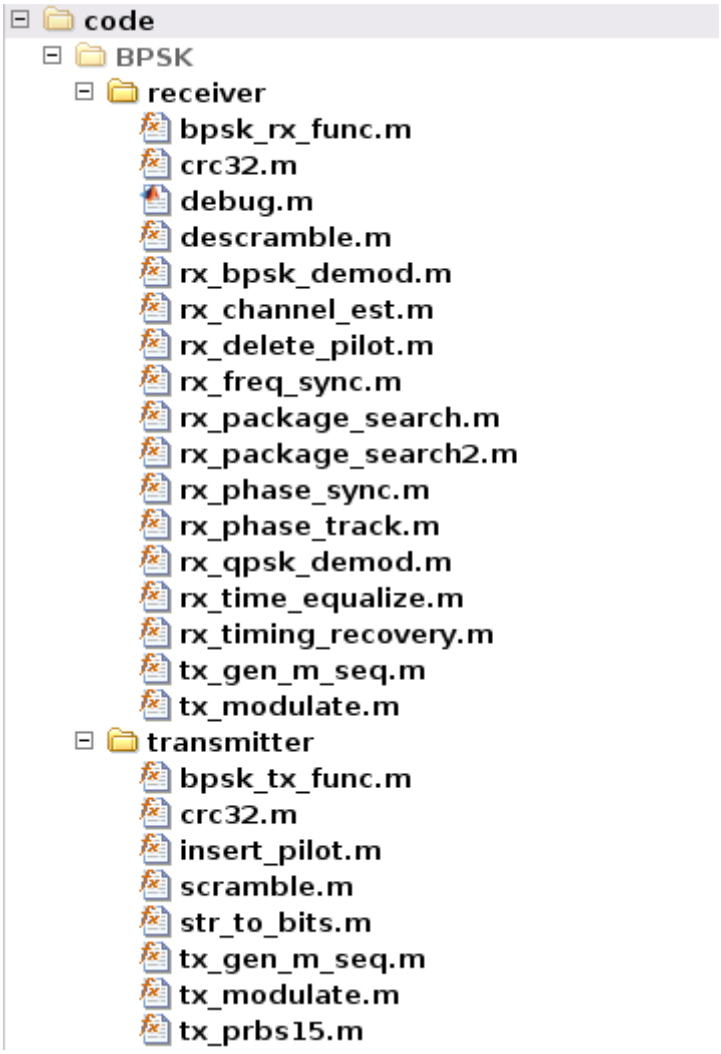
## 3.4 基于BPSK的文字传输系统

我们以自发自收为例, 对于核心部分进行解读.

### 3.4.1 代码结构&文件作用分析




- `/library/matlab`
    - 这下面保存着主要的 libio库的matlab函数实现. 这个库的作用就是作为 Matlab 和 PlutoSDR 进行交互的文件.
    - 我们主要使用的是其中的 `iiio_sys_matlab.m` 文件中的函数, 比如说发送和接受数据, 设置 pluto 的频段等参数等等作用.
    - `ad9361.cfg` 则是对于 PlutoSDR 的芯片参数进行配置的文件.
    - `iio.h` 是用C语言编写的硬件函数库,而这也是我们需要在基本环境配置中添加C语言编译器的原因,在运行过程中,需要编辑头文件来达到操作硬件的目的.
- `/code/`
    - `go_back_n_recv.m` & `go_back_n_send.m` 是我们作为单独收和单独发的主要函数文件.
    - `slide_windows_receiver.m` 则是实现了窗口滚动接受的效果,相当于在原先接受的基础上进行了提升.
    - `stop_wait.m` 则是利用了停止等待协议,实现了对一段文本的自发自收.




- `/code/BPSK/`
  - `../receiver` & `../transmitter` 这是我们找到的参考例程. 主要的作用是实现了 BPSK 的调制和解调. 我们只需要将这些文件的位置添加到路径当中即可调用.

### 3.4.2 自发自收系统

要实现自发自收, 我们以"停止等待协议"为思路, 设计了 `stop_wait.m` 文件. 在此之中, 结合了调用库文件来完成 Pluto 的数据收发, BPSK调制和解调等. 对应的主要文件有:

- `stop_wait.m`: 以 停止等待协议 为根本的**主函数**文件.
  - `slide_windows_receiver.m` : 则是以 窗口滑动协议 为基础的另一个主函数文件.
  - 两个文件的结构相似,我们主要是 `stop_wait.m` 为主要部分进行解释
- `iio_sys_obj_matlab.m`: Pluto 的配置文件
- `./code/BPSK/transmitter & receiver`: BPSK调制与解调的实现文件

接下来, 我们首先用主函数文件 `stop_wait.m` 来解释我们的实现思路, 然后对于 Pluto 的配置文件和 BPSK 进行解释.

### 3.4.3 添加库文件和基本路径

如果想要使用相应的 libiio 和 BPSK 调制解调方法, 必须要先添加路径:

```
% 添加 libiio 库文件
addpath ..\library
addpath ..\library\matlab

% 设置 pluto 需要使用的ip地址
% 添加 BPSK 调制解调代码路径
ip = '192.168.2.1';
addpath BPSK\transmitter
addpath BPSK\receiver
```

### 3.4.4 使用库文件配置 Pluto

我们没有使用 `Communication Toolbox` 来进行配置, 而是使用了 `iio_sys_obj_matlab.m` 中的函数. 结合面向对象式编程方法, 我们是将 Pluto 作为一个对象, 而像是 ip, dev_name 等等则是属性, 同时, 它也具有方法, 比如收发数据等.

```
%% System Object Configuration
s = iio_sys_obj_matlab; % MATLAB libiio Constructor
s.ip_address = ip;
s.dev_name = 'ad9361';
s.in_ch_no = 2;
s.out_ch_no = 2;
s.in_ch_size = 42568;
s.out_ch_size = 42568*8;

s = s.setupImpl();

input = cell(1, s.in_ch_no + length(s.iio_dev_cfg.cfg_ch));
output = cell(1, s.out_ch_no + length(s.iio_dev_cfg.mon_ch));

% Set the attributes of AD9361
input{s.getInChannel('RX_LO_FREQ')} = 2e9;
input{s.getInChannel('RX_SAMPLING_FREQ')} = 40e6;
input{s.getInChannel('RX_RF_BANDWIDTH')} = 20e6;
input{s.getInChannel('RX1_GAIN_MODE')} = 'manual';%% slow_attack manual
input{s.getInChannel('RX1_GAIN')} = 10;
% input{s.getInChannel('RX2_GAIN_MODE')} = 'slow_attack';
% input{s.getInChannel('RX2_GAIN')} = 0;
input{s.getInChannel('TX_LO_FREQ')} = 2e9;
input{s.getInChannel('TX_SAMPLING_FREQ')} = 40e6;
input{s.getInChannel('TX_RF_BANDWIDTH')} = 20e6;
```

- 请注意 `s.dev_name`, 这个地方的设备名不能写为 `Pluto` 或者是 `ad9364`. 这是因为 analog 公司只提供了 ad9361.cfg 这样一个文件, 并没有提供 9364 版本的文件, 而且在库文件的路径中也只有 `ad9361.cfg` 这样一个文件. 这个地方如果出错了, 则会出现 `Cannot find device` 这样一个错误.
- 下图就是建立连接的过程:

```
>> stop_wait
libiio_if: Connected to IP 192.168.2.1
libiio_if: Remote libiio version: 0.17, (git-v0.17  )
libiio_if: Local libiio version: 0.18, (git-c0012d0)
libiio_if: Found 5 devices in the system
libiio_if: cf-ad9361-dds-core-lpc was found in the system
libiio_if: Found 2 output channels for the device cf-ad9361-dds-core-lpc
libiio_if: cf-ad9361-dds-core-lpc output data channels successfully initialized
libiio_if: Connected to IP 192.168.2.1
libiio_if: Remote libiio version: 0.17, (git-v0.17  )
libiio_if: Local libiio version: 0.18, (git-c0012d0)
libiio_if: Found 5 devices in the system
libiio_if: cf-ad9361-lpc was found in the system
libiio_if: Found 2 input channels for the device cf-ad9361-lpc
libiio_if: cf-ad9361-lpc input data channels successfully initialized
libiio_if: Connected to IP 192.168.2.1
libiio_if: Remote libiio version: 0.17, (git-v0.17  )
libiio_if: Local libiio version: 0.18, (git-c0012d0)
libiio_if: Found 5 devices in the system
libiio_if: ad9361-phy was found in the system
```

### 3.4.5 设置发送数据

在这里, 我们为了测试系统稳定性, 直接发送长文本的方式来测试.在这个长文本中既有字母也有符号.

```
%% Transmit and Receive using MATLAB libiio

% 这是我们需要发送的数据
stringToSend = 'Once a heroic Jedi Knight, Darth Vader was seduced by the
dark side of the Force, became a Sith Lord, and led the Empire"s eradication
of the Jedi Order. He remained in service of the Emperor -- the evil Darth
Sidious -- for decades, enforcing his Master''s will and seeking to crush the
fledgling Rebel Alliance. But there was still good in him ...';
```

### 3.4.6 停止等待协议

要实现停止等待协议, 根据上文第二部分的思路, 主要是分割数据帧 --> 发送 <-(校验)-> 接受.通俗来讲,就是让发送端知道什么时候该发.

**分割要发送的数据集,变成数据帧**

```
```matlab
arrLength = ceil(length(stringToSend)/57);
sendArray = cell(1,arrLength);
seqNum = 0;
for index = 1:arrLength
    seqNumStr = ['00', int2str(seqNum)];
    if seqNum == 0
        seqNum = 1;
    else
        seqNum = 0;
    end
    if index*57 > length(stringToSend)
        sendArray(index) = {[seqNumStr,stringToSend(index*57-
56:length(stringToSend))]};
    else
        sendArray(index) = {[seqNumStr,stringToSend(index*57-56:index*57)]};
    end
end
```
```

**数据发送**

这之中, 先进行 BPSK 调制, 然后根据设计的停止等待协议, 进行传输.

```
index = 1;
% 接受成功与否的标志位
isRecieved = 0;
RcurrentSeq = '0';
% 保存接收的数据
recievedStr = '';
while(index <= arrLength)
    TcurrentSeq = sendArray{index}(3);
    fprintf('Transmitting Data Block %s ...\n',TcurrentSeq);

    % BPSK 调制
    txdata = bpsk_tx_func(sendArray{index});
    txdata = round(txdata .* 2^14);
    txdata=repmat(txdata, 8,1);
    input{1} = real(txdata);
    input{2} = imag(txdata);
    % 发送数据
    sendData(s, input);
```

**数据接收**

我们为了保证数据的成功接收, 每次收到数据便会返回`ASK`的标志位, 代表可以继续发送下一个数据帧

```
while(index <= arrLength)
    % 数据发送部分
    ...

    % 数据接受
    sendTime = clock;
    while (etime(clock, sendTime) < 10)
        output = recieveData(s);
        I = output{1};
        Q = output{2};
        Rx = I+1i*Q;
        % BPSK解调
        [rStr, isRecieved] = bpsk_rx_func(Rx(end/2:end));
        if (~isRecieved)
           continue;
        else
            if (rStr(1, 1:3) == 'ACK')
                if (rStr(1, 4) == TcurrentSeq)
                    fprintf('Data Block %s ACKed...\n',TcurrentSeq);
                    index = index + 1;
                    break;
                end
            else
                if (rStr(1, 3) == RcurrentSeq)
                    fprintf('Data Block %s Received...\n',RcurrentSeq);
                    if (length(rStr) == 16)
                        temp = [rStr(1, 4:16), rStr(2,:), rStr(3,:),
rStr(4,:)];
                        recievedData = temp(1:find(ismember(temp, char(0)),
1 ) - 1);
```

```
                    else
                        recievedData = rStr(1, 4:length(rStr));
                    end
                    fprintf('recievedData: %s\n', recievedData);
                    recievedStr = [recievedStr, recievedData];
                    if (RcurrentSeq == '0')
                        RcurrentSeq = '1';
                    else
                        RcurrentSeq = '0';
                    end
                end
                txdata = bpsk_tx_func(['ACK', rStr(1, 3)]);
                txdata = round(txdata .* 2^14);
                txdata=repmat(txdata, 8,1);
                input{1} = real(txdata);
                input{2} = imag(txdata);
                fprintf('Transmitting ACK...\n');
                sendData(s, input);
                output = {};
            end
        end
    end
end
fprintf('Transmission and reception finished\n');
fprintf('recievedData: %s\n', recievedStr);

% Read the RSSI attributes of both channels
rssi1 = output{s.getOutChannel('RX1_RSSI')};
% rssi2 = output{s.getOutChannel('RX2_RSSI')};

s.releaseImpl();
```

### 3.4.7 测试过程

```
Transmitting Data Block 0 ...
Data Block 0 Received...
recievedData: Once a heroic Jedi Knight, Darth Vader was seduced by the
Transmitting ACK...
Data Block 0 ACKed...
Transmitting Data Block 1 ...
Data Block 1 Received...
recievedData:  dark side of the Force, became a Sith Lord, and led the
Transmitting ACK...
Data Block 1 ACKed...
Transmitting Data Block 0 ...
Data Block 0 Received...
recievedData: Empire"s eradication of the Jedi Order. He remained in se
Transmitting ACK...
Data Block 0 ACKed...
Transmitting Data Block 1 ...
Data Block 1 Received...
recievedData: rvice of the Emperor -- the evil Darth Sidious -- for dec
Transmitting ACK...
Data Block 1 ACKed...
Transmitting Data Block 0 ...
Data Block 0 Received...
recievedData: ades, enforcing his Masters will and seeking to crush t
Transmitting ACK...
Data Block 0 ACKed...
Transmitting Data Block 1 ...
Data Block 1 Received...
recievedData: he fledgling Rebel Alliance. But there was still good in
Transmitting ACK...
Data Block 1 ACKed...
Transmitting Data Block 0 ...
Data Block 0 Received...
recievedData: him ...
Transmitting ACK...
Data Block 0 ACKed...
```

**过程分析:**

- `Transmitting Data Block *` 是准备开始发送的标志
- `Data Block * received` 代表数据已收到
- `ReceivedData` 会显示当前收到的数据帧
- `Transimitting ACK` 是从接收端发送到发送端, 代表可以发送下一个数据帧

### 3.4.8 测试结果

```
Transmission and reception finished
recievedData: Once a heroic Jedi Knight, Darth Vader was seduced by the dark side of the Force, be
```

### 3.4.9 结果分析

经过比对,我们可以发现发送的字符串和接受到的字符串相符. 并且经过不同数据的多次测试, 都可以保证信息准确传输而没有传输错误.

# 四 问题总结

这次的开发, 周期很短,而且公开的资料只有官网的 wiki 文档和有限的 MATLAB 文档, 上手不容易, 不过最后基本上全部实现了当时定下的目标, 所以还是可以的. 当然, 我们总结还是有很多问题的:

## 4.1 环境搭建和配置

1. 在Matlab上安装 `Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio` 失败
    - 解决方案一： 需要在 Matlab 中设置 `web_proxy` ，使用外网下载相应的配置文件；
    - 解决方案二： 在 Analogodevice 的 github[3] 仓库中下载已经打包好的 toolbox 文件

2. 在 Linux 上编译 libiio 库, 根据官方文档需要使用超级管理员权限,同时,在编译的最后需要和 Matlab 做结合.因此,本地的 Matlab 需要已经以超级管理员身份运行和激活,否则最后一步就会报错, 从而导致环境配置失败.

## 4.2 BPSK只实现了传输文字

理论上来说 BPSK 不仅可以传输文字, 也可以传输图像, 文件等. 我们本准备使用 Base64 编码做传输, 但是没有成功. 所以, 这也算是不足.

## 4.3 BPSK 文字传输速度并不是很理想

这个是和我们设置的数据帧长度有关. 受限于 BPSK 调制方式, 我们为了稳妥起见没有使用更长的帧长度. 当然, 我们也可以使用多进制的调制方式, 比如 4PSK, 16QAM等.

# 五 心得体会

在开发的过程中, 刚开始我们也不了解什么是 BPSK 等通信原理知识. 而且由于参考资料很少, 且绝大部分都是外文参考资料, 我们克服了很大的困难, 尽力排除每一个 Bug, 思考每个问题的原因. 在这个过程中, 我们熟悉了 PlutoSDR 这个设备, 而且使用 MATLAB 让通信原理的理论知识和实践相结合, 加深了对于理论知识的理解.

# 六 注释和参考文档

[1] PlutoSDR 器件介绍: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html#eb-overview
[2] analog官网: https://wiki.analog.com/
[3] github仓库网站： https://wiki.analog.com/resources/tools-software/
[4] 本项目所有的代码都托管在了 Github： https://github.com/rongyupan/PlutoSDR-BPSK