**Lab 3: Packet Transmission using USRP**

|  |  |
| --- | --- |
| **Author** | Name: 孙逸涵 (Yihan Sun) Student ID: 12012128  Name: 张旭东 (Xudong Zhang) Student ID: 12011923 |
| **Introduction**  In this lab, what we need to do is to master the method of using USRP based on MATLAB. We get to know the Software-Defined Radio(SDR), the process of connecting MATLAB and USRP and the function of driving USRP in MATLAB. We have realized the recovered result of pre-recorded data, the running result of the example of QPSK Transmitter/Receiver with USRP Hardware in MATLAB and the running result of 16-QAM tranceiver.  **Theoretical analysis**   1. **Introduction to Software Defined Radio**   The important value of Software-Defined Radio lies in that the traditional hardware radio communication device only serves as the basic platform of wireless communication, while lots of functions of communication are realized by softwares, which breaks the pattern that the realization of function of communication of devices only depends on the development of hardware in history. The emergence of Software-Defined Radio is the third revolution in the field of communication after fixed communication to mobile communication and analog communication to digital communication.  The main components of software radio are the RF front end for RF signal conversion, ADC and DAC for digital to analog signal conversion, and software processing components such as DSP. Among these three parts, ADC and DAC play a major role because different sampling methods will determine the composition of the RF front end and the processing of digital signals by subsequent devices. According to different sampling methods, software radio can be divided into three types: low-pass sampling software radio structure, RF direct bandpass sampling software radio structure and broadband mid-frequency bandpass sampling software radio structure.  The first architecture is the low-pass sampling software radio architecture. According to Nyquist sampling theorem, the low-pass sampling frequency should be more than twice of the maximum working frequency. Such a high sampling frequency cannot be realized by current DAC, which is not only a major challenge for ADC, but also a difficult problem to be solved for the improvement of the performance of ADC subsequent signal processing (FPGA/DSP) devices.  The second structure is the band-pass sampling software radio structure, which can correspondingly reduce the requirements of the first software radio structure on digital signal processing and DAC. This structure not only reduces the sampling rate of DAC transform, but also reduces the requirement of digital signal processing. Although it has many advantages, it is difficult to realize the functions of high broadband and pre-narrow band filter in practice.  The third structure is the wideband medium-frequency bandpass sampling software radio structure, from which it can be known that the superheterodyne mechanism (multiple mixing) is used. The main advantage of this radio structure is that it will make the bandwidth of the intermediate frequency wider, and all the functions of modulation and demodulation can be realized through software programming. Its disadvantage is that the RF front-end (the analog preprocessing circuit before A/D) is more complex. The main function of the RF front-end is to convert the RF signal into the broadband IF signal suitable for ADC conversion. This reduces the rate requirement for ADC sampling digitization.  The structure of Software-Defined Radio is as below:    For Software-Defined Radio, three kinds of structure are mainly learned, including Low-IF Receiver, Heterodyne receiver and Direct-Conversion Receiver.   1. **Heterodyne receiver**   The structure of Heterodyne receiver is as below:    The heterodyne receiver has been a staple of receiver design since its first appearance in 1917. The heterodyne structure consists of the following modules: input bandpass filter, low noise amplifier, image suppression filter, mixer, IF filter and AD. Input bandpass filters are usually used to suppress out-of-band interference signals to prevent strong interference signals from blocking low-noise amplifiers. Generally, the input bandpass filter has a wide bandwidth and consists of multiple channels. The mirror filter is used to suppress the mirror frequency. The IF bandpass filter after mixing determines the channel selectivity of the receiver and is used to suppress adjacent channel signal power. At the same time, the IF bandpass filter is usually used as the anti-aliasing filter in the front end of AD.   1. **Direct-Conversion Receiver**   The structure of direct-conversion receiver is as below:    Direct-conversion receivers overcome the problem of image suppression by converting the signal directly to baseband (0Hz). The local frequency (LO) and RF signal frequency (RF) of direct-conversion receiver are equal, and the mirror frequency is the signal frequency itself. The image suppression filter and if filter in the original superheterodyne receiver structure can be omitted. In this way, the external components are eliminated, which is conducive to the single chip implementation of the system.   1. **Low-IF Receiver**   The structure of Low-IF Receiver is as below:    Low IF receiver tries to solve the problem of DC bias and flicker noise of zero IF receiver, but at the same time maintain the high integration of zero IF receiver. Many wireless standards require that the suppression of adjacent channel interference is relatively weak compared with that of other channels. Low IF receiver makes full use of this regulation and selects the appropriate IF frequency to take the adjacent channel signal as its mirror signal.   1. **USRP and MATLAB connection process**     Besides, another thing needed to do is to program transmitter and receiver on MATLAB to realize this simulation using USRP. The process of transmitter is as below:    Through the flow graph of transmitter, what we need to do firstly is to configure the parameters of SDRu. Then the object, named ThSDRu, is created. After that, the process of transmission begins, which is read and write in a loop. Finally, when the process of transmission is completed, exit the loop and release resources. Otherwise, the tx would be busy.  The process of receiver is as below:    Through the flow graph of receiver, the process is similar to that of the transmitter. The step needed to do is the same as that of receiver.   1. **Introduction of USRP driver function in MATLAB** 2. **comm.SDRuTransmitter**   The SDRuTransmitter System object sends data to a Universal Software Radio Peripheral (USRP) hardware device, allowing simulation and development for various software-defined radio applications. The SDRuTransmitter System object enables communication with a USRP board on the same Ethernet subnetwork. We can write a MATLAB application that uses the System object or we can generate code for the System object without being connected to a USRP radio.  This object accepts a column vector or matrix input signal from MATLAB and transmits signal and control data to a USRP board using the Universal Hardware Driver (UHD) from Ettus Research. The SDRuTransmitter System object is a sink that sends the data it receives to a USRP board. The first call to this object could contain transient values, which would result in packets containing undefined data.  This object contains the following properties:   |  |  |  |  | | --- | --- | --- | --- | | Platform | IPAddress | SerialNum | ChannelMapping | | CenterFrequencySource | CenterFrequency | ActualCenterFrequency | LocalOscillatorOffsetSource | | LocalOscillatorOffset | ActualLocalOscillatorOffset | GainSource | Gain | | ActualGain | ClockSource | InterpolationFactorSource | InterpolationFactor | | ActualInterpolationFactor | TransportDataType | UnderrunOutputPort | EnableBurstMode | | NumFramesInBurst | MasterClockRate | ActualMasterClockRate |  |   And there are four methods for comm.SDRuTransmitter:   |  |  |  |  | | --- | --- | --- | --- | | info | isLocked | release | step |   In addition, we need to notice that we can set our desired values in the SDRuTransmitter System object for center frequency, gain, and bandwidth. However, due to quantization or range issues, it is possible that the actual values are not the same as our desired values. The actual values are stored in the ActualPropertyName properties.   1. **comm.SDRuReceiver**   The SDRuReceiver System object receives data from a Universal Software Radio Peripheral (USRP) hardware device, allowing simulation and development for various software-defined radio applications. The SDRuReceiver System object enables communication with a USRP board on the same Ethernet subnetwork. We can write a MATLAB application that uses the System object, or we can generate code for the System object without connecting to a USRP radio.  This object receives signal and control data from a USRP board using the Universal Hardware Driver (UHD) from Ettus Research. The SDRuReceiver System object receives data from a USRP board and outputs a column vector or matrix signal of fixed length. The first call to this object could contain transient values, which would result in packets containing undefined data.  This object contains the following properties:   |  |  |  |  | | --- | --- | --- | --- | | Platform | IPAddress | SerialNum | ChannelMapping | | CenterFrequencySource | CenterFrequency | ActualCenterFrequency | LocalOscillatorOffsetSource | | LocalOscillatorOffset | ActualLocalOscillatorOffset | GainSource | Gain | | ActualGain | ClockSource | DecimationFactorSource | DecimationFactor | | ActualDecimationFactor | TransportDataType | OverrunOutputPort | SampleRate | | OutputDataType | FrameLength | EnableBurstMode | NumFramesInBurst | | MasterClockRate | ActualMasterClockRate |  |  |   And there are four methods for comm.SDRuReceiver:   |  |  |  |  | | --- | --- | --- | --- | | info | isLocked | release | step |   In addition, we need to notice that we can set our desired values in the SDRuReceiver System object for center frequency, gain, and bandwidth. However, due to quantization or range issues, it is possible that the actual values are not the same as our desired values. The actual values are stored in the ActualPropertyName properties.  Furthermore, the SDRuReceiver System object has an optional lost samples output port. When this port is active, it outputs a logical signal that indicates whether the System object is processing data in real time. If the System object is not keeping up with the hardware, the signal indicates the approximate number of lost samples. This port is a useful diagnostic tool for determining real-time operation of the System objects.  **Lab results & Analysis**   1. **Pre-recorded data recovery results** 2. **QPSK**     The figure shown above is the received signal’s spectrum diagram, text recovery result, and BER of the QPSK pre-recorded data recovery result.  From the figure above, we can find that we have recovered the pre-recorded data successfully.   1. **16-QAM**     The figure shown above is the received signal’s spectrum diagram, text recovery result, and BER of the 16-QAM pre-recorded data recovery result.  From the figure above, we can find that we have recovered the pre-recorded data successfully.   1. **Result of MATLAB example “QPSK Transmitter/Receiver with USRP Hardware”**     From the recovered text, it is obvious that the last digit of the text has a higher probability of error and the error rate is 0.088158, which is a little high. The constellation of result is a little far from the expected result. The reason behind this is that we have changed the data length to 112 and there was a lot of interference in the environment, which means multiple groups were sending data at the same time.   1. **16-QAM Transceiver** 2. **Programming procedure**   The following figures show what the program changed:   * SimParams      * QPSKTransmitterR      * QPSKBitsGeneratotR      * QPSKReceiverR      * QPSKDataDecoderR        1. **Program flow chart**   The overall program flow chart is as follows:    In the process of configuration parameters, some important parameters should be considered.    The details of creating objects are as below:    The details of Loop Write/Read and releasing resources is as below:    In addition to the above details, the parameters of SDRu transmitter and the object of transmitter, named ThSDRu should be configured and created.      What’s more, the parameters of SDRu receiver and the object of receiver, named RhSDRu should be configured and created, too.       1. **Experimental results**  * **USRPCenterFrequency = 3GHz**     The figure shown above is the experimental result of the 16-QAM when the  USRP center frequency is 3GHz.   * **USRPCenterFrequency = 4GHz**       The figures shown above are the experimental results of the 16-QAM when the  USRP center frequency is 4GHz. And the only difference between those two results is the USRP gain of the transceiver.  The figures above show the spectrum diagram of the received signal, the constellation of the received signal before and after matched filtering, the text recovery result, and BER. From the text recovery result, BER, or the constellation of the received signal after matched filtering, we can find that we have carried out the 16-QAM packet transmission using USRP successfully.  In addition, we can easily find that the performances of the 16-QAM packet transmission using USRP is a little different under different USRP center frequencies. | |
| **Experience**  **Experience**  孙逸涵:   1. After this lab, I was more familiar with the framework of the QPSK or 16-QAM transceiver, and I can also find the corresponding parameters of the modulation mechanism faster. 2. In this lab, we first use MATLAB programming to recover the pre-recorded QPSK and 16-QAM data from USRP. The experimental results show that the bit error rate of QPSK is smaller than that of 16-QAM, which is completely consistent with our theoretical analysis. 3. When we do packet transmission using USRP, we need to notice that each character in the transmission data or the transmission message is encoded to 7-bits decimal values, i.e., 7 ASCII characters, so that the number of message bits per frame, i.e., MessageLength, must match that of the transmission message. 4. When we implemented the 16-QAM transceiver and did packet transmission using USRP, we found that the performances of the 16-QAM packet transmission using USRP is a little different under different USRP center frequencies.   张旭东:   1. I briefly understand the structure, function, application and important value in industry about Software-Defined Radio. 2. I have mastered the connection method between USRP and MATLAB. 3. I have run the example of OPSK Transmitter/Receiver with USRP Hardware in MATLAB and analyzed the result. 4. I help to get 16-QAM Tranceiver.   **In-class lab screenshot**  孙逸涵:    张旭东:    **Duration of experiment by using USRP**  5 hours for QPSK and 16-QAM packet transmission  4 hours for QPSK Transmitter/Receiver with USRP Hardware | |
| **Score** | 100 |