An Implementation of an Asynchronous Communication System using the USRP

A Project Report

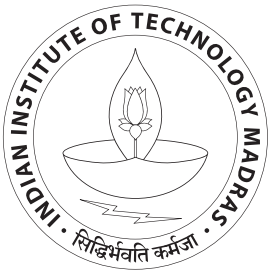
submitted by

MARDAVA RAJUGOPAL GUBBI

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

under the guidance of
Dr. Venkatesh Ramaiyan



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2015

THESIS CERTIFICATE

This is to certify that the thesis entitled **An Implementation of an Asynchronous Communication System using the USRP**, submitted by **Mardava Rajugopal Gubbi**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Venkatesh Ramaiyan
Research Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to thank my guide, Dr. Venkatesh Ramaiyan, for his invaluable support and guidance for the entire duration of my project. He was always available to listen to and offer advice on the problems I was facing. With his guidance, I gained a wonderful insight into the field of communication networks, and into the research environment in general.

In addition, I would like to thank Sundaram for providing a valuable input every step of the way. I would also like to thank my lab mates for all of the help and support they provided me during my project. The countless conversations we had, bouncing ideas off of each other, made my project experience a truly fulfilling one.

I would also like to thank my parents for always being there for me over the years. Their love and support have played a large part in making me who I am today.

Finally, I would like to thank my friends. College life is nothing without a good set of friends, and I can honestly say that I had the best.

ABSTRACT

This project focuses on the asynchronous communication problem involving the intermittent transmission of packets from a single transmitter to a receiver. We deal with two aspects of the asynchronous communication problem, the first involving minimizing the packet length, and the second involving an analysis of the transmission power with respect to the asynchronism level of the system. The first part of our project dealing with the minimization of the packet length involved building a two-way short packet communication system from scratch. The second part of our project involves an analysis of the asynchronism level of the communication system. Recent results indicate that the Signal to Noise Ratio (SNR) required to maintain a particular Bit Error Rate is related to the entropy of the arrival process of the packets. This part of the project involves transmitting a single packet over an extended period of time, and attempting to accurately detect and decode the packet. We determine the packet Acquisition Error Rates and the Bit Error Rates wherever appropriate under various operating conditions. We chose to build both of these systems from scratch on a set of USRP N210 kits, using the UHD API.

TABLE OF CONTENTS

CHAPTER 1

Introduction

The recent emergence of frameworks like the Internet of Things and new applications in the field of wireless sensor networks has led to a renewed interest in the problem of asynchronous communication. Asynchronous communication between a transmitter and receiver refers to the situation where the transmitter sends data intermittently in the form of packets, with random intervals between successive packets. The receiver must then detect and decode these packets without prior information about the instants of their transmission. The various aspects of this problem have been studied at length for a number of years in early papers such as [9, 13, 16, 17], as well as more recent works like [8, 15, 18]. Decoding the received data requires the synchronization of each received frame, the recovery of the carrier frequency and phase, and the bit timing synchronization. This synchronization problem is traditionally mitigated at the packet design level by using a good synchronization sequence, a large packet size, or both. Receiver designs making use of these packet designs have been extensively studied, primarily as data-aided, non-data-aided [12, 14], or code-aided synchronization [7, 19] techniques. However, some of these techniques are costly in terms of power and time, and result in higher transmission overheads than are strictly necessary. With a rising demand for sensors with increased lifespans, optimizing the sensors' communication modules becomes a problem worth exploring. We focus on two aspects of the asynchronous communication setup. The first aspect seeks to minimize the length of the packet exchanged between the transmitter and the receiver. Small packet sizes imply small communication delay and large multiplexing gain. The second aspect corresponds to the characterization of the overheads (power) needed for reliable communication in the presence of asynchronism.

Both of these operations are required to be performed under certain conditions, while still maintaining a specified Bit Error Rate (BER). To address these problems, we designed and built two different communication systems using the USRP N210 RF transceiver. The USRP serves as an affordable and fairly popular system used by Software Defined Radio (SDR) enthusiasts across the world. As a result, there is an abundance of support for setting up SDR projects. The API provided by the manufacturers is easy to work with, and supports building a system from scratch. As a result, the USRP series of devices is ideal for projects such as ours, which require us to build two different communication systems and test them across a wide variety of operating conditions.

The first problem necessitated the construction of a short packet communication system from the ground

up, with the help of the UHD API. The system forms a star topology, with one transmitter and two receivers. The data packets transmitted contain 1 byte of data, with various MAC and PHY layer overheads. The receiver then transmits much shorter acknowledgement packets. In this system, we assume no fading, and a manageable timing and frequency offset. We also assume that we can boost the signal power as and when required to counter Additive White Gaussian Noise (AWGN). With these assumptions in place, we can demonstrate that at a sampling rate of 250 kHz, we can achieve a turnaround time of < 1 ms. Apart from the original low power application, this offers uses for systems requiring short response times, and may be applied to control systems like the vertical pendulum.

The motivation behind the second part of the asynchronous communication problem arises from the emergence of standalone sensor nodes transmitting data back to a central data collection centre at regular intervals. It has been shown theoretically that the signal to noise ratio required to support an asynchronism is proportional to the entropy of the arrival distribution. We seek to demonstrate the theoretical results in a practical setup using the USRP kits. For this problem, we use a hardware setup similar to the short packet communication system to determine the performance of a sequential detector at various SNR values.

Thus, this project seeks to explore new methods to reduce the transmission overheads both for a single packet and across packets. Each of these aspects is explored by implementing and testing a wireless communication system on a group of USRP RF transceivers. The rest of this paper is organised as follows. Chapter 2 discusses the USRP, the UHD API, GNURadio, and the rationale behind the decision to choose UHD over GNURadio for our work. Chapter 3 discusses the short packet transceiver system, and goes into detail both about the various algorithms implemented and some of the algorithms discarded. Chapter 4 discusses the sequence detection problem, and the system designed for this problem. Chapter 5 concludes with a summary and discusses possible future work.

CHAPTER 2

The Experimental Setup Using the USRP Hardware

2.1 An Introduction to USRP

The Universal Software Radio Peripheral (USRP) is a series of RF transceivers, designed to be used in consort with a PC, for the purpose of designing and testing Software Defined Radio (SDR) systems. For the purpose of our project, we chose to use the USRP N210 device. It consists of two modules, a motherboard, which is fixed for a given USRP device, and a daughtercard, a drop-in module used for passband transmission and reception. The motherboard contains the clock generation and synchronization circuitry, FPGAs, ADCs, DACs, the host processor interface, and the power regulation system. This allows the motherboard to handle baseband signals. The daughtercard contains the analog circuitry for up/down conversion, filtering, etc. The various daughtercards offered are each optimized for a particular range of frequencies, allowing the user to select whichever suits their application the best. The USRP N210 is also provided with 2 ports for antennae or a loopback cable, one a receiver and the other a transceiver. The USRP also requires a 6V-3A DC power supply, as shown in Figure 2.1.

Thus, in order to set up a simple SDR system on a USRP kit, one requires the following equipment:

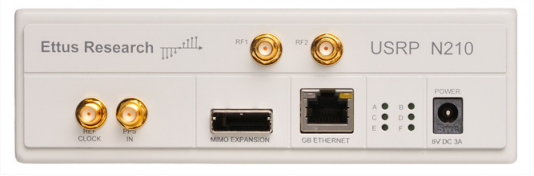


Figure 2.1: Front view of the USRP N210 RF transceiver

- A PC with a Gigabit Ethernet capable port, and the corresponding ethernet cable
- A 6V-3A DC power supply (the adapter is provided with the USRP kit)
- A daughtercard compatible with the USRP motherboard
- A VERTx (900/2450) antenna or loopback cable, depending on one's needs

Both the short packet transceiver system and the sequence detector work at frequencies in the region 2.2-2.4 GHz. Hence, for our experiments, we chose the CBX daughtercards, with the range 1.2-6 GHz,

and the VERT2450 antennae, optimized for the frequency 2.45 GHz.

2.2 A Brief Look at UHD

The next part of setting up a USRP system involves getting the host PC to interface with the USRP device. The USRP Hardware Driver (UHD) serves as an open source API to the USRP series of devices. We used the version UHD_003.007.002 for our application. The driver is written in C++, and provides access to all of the USRP's transmit and receive parameters, as well the actual samples to be sent and received by the USRP device. The UHD API functions as a base on top of which the user can implement an entire system. The UHD API allows the user to initialize the USRP device and then treat it as a black box that reads from or writes to vectors of complex samples. In essence, the UHD API allows the user to completely configure and run the USRP with a few simple lines of code. However, the UHD API does not help with any signal processing beyond a simple up/down conversion of samples to/from the passband. All of the code required for baseband signal processing must be written by the user.

2.3 UHD vs. GNURadio

GNURadio is a graphical toolkit used to develop modular SDR systems, which can then be run using USRP devices, using the UHD API. GNURadio uses the block as its functional unit, allowing users to develop and share algorithms as blocks, which can then be connected to each other to form a coherent whole. A typical GNURadio flowgraph consists of a source, a sink, and a few blocks linking the two. The source or sink block representing a USRP device will contain all the information needed to configure the USRP device for this flowgraph. Figure 2.2 shows the GNURadio flowgraph for a simple spectrum analyzer. The first block is a USRP Source block, reading samples from the USRP at a given carrier frequency and sampling rate. The second block in the flowgraph is a GUI Sink block. This block plots the spectrum of the received signal, with updates at regular intervals. Thus, this flowgraph is designed to plot the spectrum of the received signal at a given frequency and sampling rate.

Since GNURadio has a very active community, a wide variety of applications are already available in the form of blocks for the user to simply download and install on their PC. This allows for a much easier initial learning curve, and potentially a faster turnaround time, assuming that the user is using pre-existing blocks. However, the block system leads to a decrease in flexibility due to the steep learning

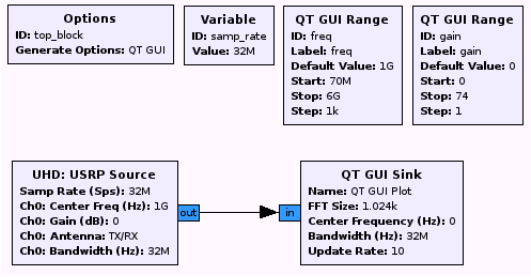


Figure 2.2: A simple GNURadio flowgraph involving a USRP source block

curve involved with developing a new block, or even significantly modifying an existing block. The extra layers between the UHD codebase and the GNURadio blocks lead to a greater difficulty in testing and debugging complex blocks. Thus, we decided to go with the UHD codebase over GNURadio. Nonetheless, it would be worthwhile to install GNURadio for the extra utilities **uhd_fft**, which allows the user to observe the spectrum of the signal received by the connected USRP device, and **uhd_siggen** and **uhd_siggen_gui**, which provide an easy interface to generate and transmit a signal.

CHAPTER 3

Short Packet Transceiver System

The motivation behind designing the short packet transceiver system is to determine the various overheads involved with the transmission and reception of short packets, and to determine optimized algorithms to speed up the packet reception and decoding. The transceiver system uses a star topology depicted in Figure 3.1, with one transmitter and two receivers. The transmitter sends out a single data packet to one of the receivers and waits for an acknowledgement before moving on to the next block of data, effectively making this a half-duplex system. The receivers only respond to packets addressed to them. The system is designed for a single transmitter and a small number (≤ 16) of receivers. The data packets are transmitted on the frequency 2.42 GHz, and the acknowledgement packets are transmitted on the 2.22 GHz. The choice of different frequencies for the data and acknowledgement packets was necessitated by problems with a full duplex system. Overcoming this problem and putting some effort into maintaining a packet queue will help speed up the transceiver system. Both the transmitter and receivers use a common sampling rate of 250 kHz.

The discussion of the short packet transceiver system is organized as follows. Section 3.1 discusses

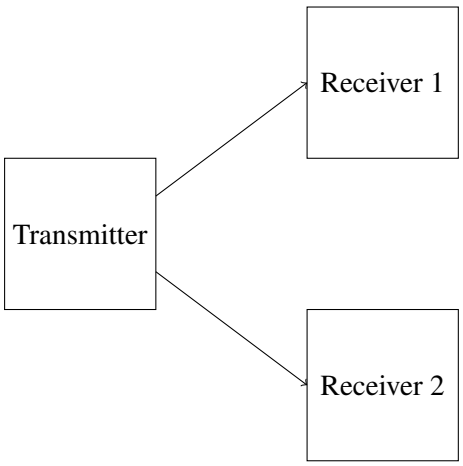


Figure 3.1: The star topology used by the short packet transceiver system

the transmitter design, starting from the sequence of bits, to the generation of the baseband signal corresponding to the packet. Section 3.2 elaborates on the receiver design, starting from the reception of a sequence of samples containing a packet, to the demodulation of the constituent bits. Since a majority

Figure 3.2: Front view of the Short Packet Transceiver System

of optimizations are required at the receiver end of the system, this section also takes a look at various algorithms that were discarded along the way, and the reasons for discarding them. Section 3.3 details the design of the data and acknowledgement packets as bit sequences.

3.1 Packet Transmitter Design

The transmitter design involves the construction of the baseband signal corresponding to the packet. All of the operations are written in C++, from the generation of the bit sequence corresponding to the data that needs to be sent, to the pulse shaping of the upsampled symbol sequence. The final sequence of samples is then passed as a vector of complex samples to the USRP's transmit streamer for the purpose of transmission. The signal processing operations on the transmitter side are listed and discussed in detail in the following subsections.

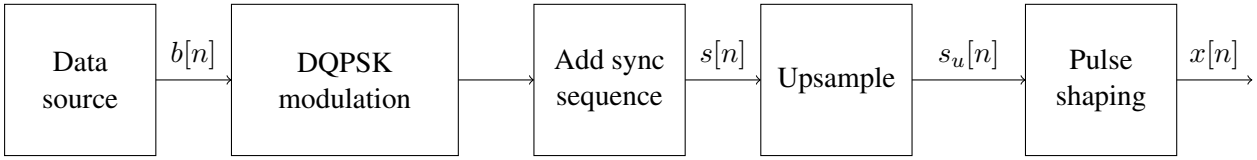


Figure 3.3: Transmitter System

3.1.1 DQPSK Modulation

Every transmission system begins with the generation of a sequence of bits, followed almost immediately by the modulation of that bit sequence into a sequence of symbols. The generation of the bit sequence will be discussed in section 3.3. A Differential Quaternary Phase Shift Keying (DQPSK) modulation scheme is used to encode the bit sequence $b[n]$ into the symbol sequence $s[n]$. In the standard QPSK modulation scheme, the n^{th} symbol $s[n]$ depends on the bits $b[2n]$ and $b[2n+1]$, as follows:

$$s[n] = \exp\left(j\pi\left(\frac{2b[2n]+b[2n+1]}{2}\right)\right) \quad n = 0, 1, ...$$

The differential modulation scheme adds a dependence on the phase $\angle s[n-1]$ of the previous symbol, as follows:

$$s[n] = \begin{cases} \exp\left(j\left(\frac{\pi(2b[2n]+b[2n+1])}{2} + \angle s[n-1]\right)\right) & n = 1, 2, ... \\ \exp\left(j\left(\frac{\pi(2b[2n]+b[2n+1])}{2} + \frac{\pi}{4}\right)\right) & n = 0 \end{cases}$$

$$b[n] \in \{0, 1\} \quad n = 0, 1, ...$$

The differential modulation scheme helps simplify the receiver design by allowing it to neglect the phase difference between the transmitter and receiver. This modulation scheme also helps us neglect the symbol amplitude. Since all symbols in the constellation have the same amplitude, we assume that they are of unit magnitude, for the sake of convenience. These symbols can be transmitted at a higher gain to boost the SNR if required.

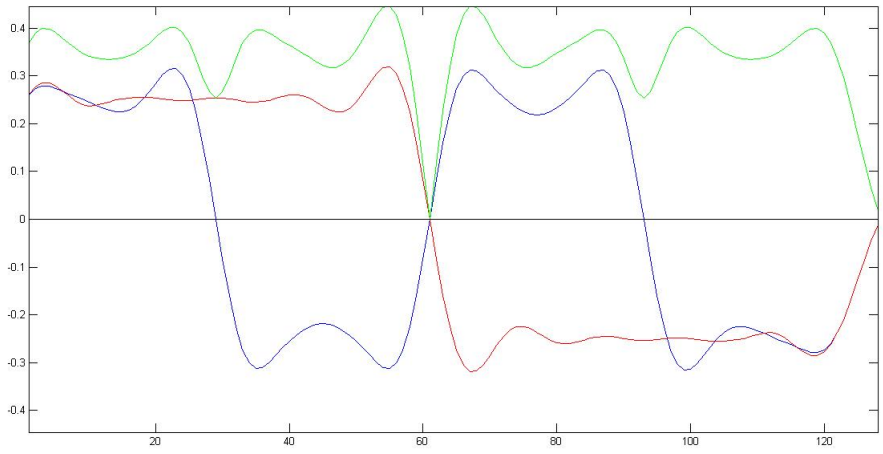


Figure 3.4: DQPSK modulated symbols of the bit stream [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]

3.1.2 Synchronization Sequence

At the receiver, we require certain synchronization information in order to decode the packet. This information is provided by attaching a synchronization sequence to the beginning of the packet. [4], [13] and [10] detail the binary and quaternary Barker sequences of various short lengths as having ideal correlation properties. Barker sequences have a correlation function $C\tau \leq 1 \forall \tau \neq 0$. We chose to use the quaternary Barker sequence of length $L_B = 13$, two copies of which are attached to the beginning of every packet. This allows the receiver to determine the exact start of the packet via a block-wise correlation operation. The quaternary Barker sequence of length 13 is given below.

$$\frac{e^{j(\frac{\pi}{4})}}{\sqrt{2}}(1, -j, -1, j, 1, j, 1, j, 1, j, -1, -j, 1)$$

3.1.3 Pulse Shaping

We chose to use the Root Raised Cosine (RRC) pulse for our pulse shaping module. The RRC pulse has the following formula:

$$r(t) = \begin{cases} \frac{1}{\sqrt{T}}\left(1 - \beta + 4\frac{\beta}{4\pi}\right) & t = 0 \\ \frac{\beta}{\sqrt{2T}}\left[\left(1 + \frac{2}{\pi}\right) sin\left(\frac{\pi}{4\beta}\right) + \left(1 - \frac{2}{\pi}\right) cos\left(\frac{\pi}{4\beta}\right)\right] & t = \pm\frac{T}{4\beta} \\ \frac{1}{\sqrt{T}} \frac{sin\left[\pi\frac{t}{T}(1-\beta)\right] + 4\beta\frac{t}{T}cos\left[\pi\frac{t}{T}(1+\beta)\right]}{\pi\frac{t}{T}\left[1-\left(4\beta\frac{t}{T}\right)^2\right]} & \text{otherwise} \end{cases}$$

where β, the roll off factor, is set to 0.35, and T is the symbol period. The time t varies from $-4T$ to $4T$.

We use an oversampling factor of $M = 2$ in our transmitter and receiver. Thus, pulse shaping the symbol stream $s[n]$ involves upsampling it by a factor of M to obtain the signal $s_u[n]$, and convolving $s_u[n]$ with the RRC pulse $r[n]$, generated once at the beginning of the transmission. The transmitted signal $x[n]$ is given by:

$$x[n] = s_u[n] * r[n]$$

where $s_u[Mn] = s[n]$

The generated sample stream $x[n]$ is then transmitted via the USRP.

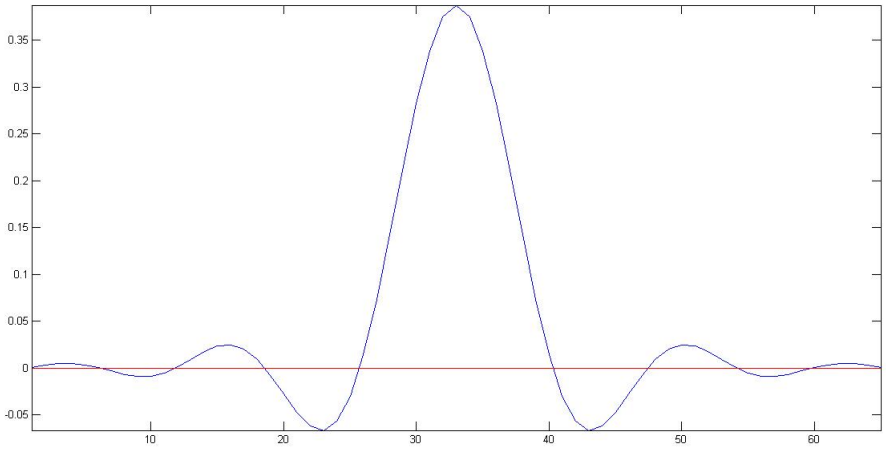


Figure 3.5: The RRC pulse

3.2 Packet Receiver Design

[3], [5], and [15] cover the basic architecture of all digital receivers under various operating conditions. We chose to keep a similar modular structure, with each step in the receive chain represented by a single module. Each of the receiver algorithms was chosen based on a compromise between simplicity and speed. The various modules were then linked together in a specific order, based on the requirements of individual algorithms. First on the list is the energy based packet detector, which obtains a rough location of the start of the packet (accurate to within 2 symbols), and then extracts the samples corresponding to the packet. The next module after the detection module is the timing acquisition module, which obtains the optimum sampling instant, and downsamples the signal to an oversampling rate of 1 sample per symbol. Timing acquisition is second on the list because all of the remaining algorithms work best with a unit oversampling rate. The third process to be performed is the carrier frequency offset estimation and correction. For this, we use a simple peak detection on the spectrum of the fourth power of the signal. Subsequently, the signal is correlated with itself to accurately determine the start of the packet. The DQPSK demodulation module is the last in the receiver chain. Our choice of a differential modulation scheme allows us to avoid a phase offset estimation and correction. Our assumption of a line-of-site transmission allows us to neglect channel estimation. These two processes are significant to most receiver systems, but also cause significant overheads in the decoding time. Some other options explored to reduce the decoding time include highly optimizing the averaging and FFT functions, and modifying the buffer size, to reduce the number of samples taken at each point of time. [3] also deals with introducing a pipeline system to facilitate reception at high data rates. However, as our receiver is

expected to respond immediately to a single packet, we felt that pipelining the data stream would not improve the overall efficiency of our system.

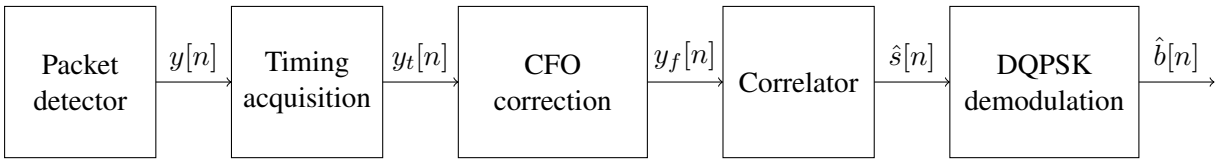


Figure 3.6: Receiver system

3.2.1 Energy Based Packet Detector

We chose to use an energy based packet detector for its simplicity, ease of implementation, and speed. The energy based packet detector scans the received signal, and does the following:

- Compute the average energy E_{av} of all of the samples obtained thus far.
- For each sample, compute the average energy E of the next $2M$ samples.
- If the value $\left(\frac{E}{E_{av}}\right)$ is above a certain threshold $E_{Th} = 2.0$, then assume that the current sample is close the start of a packet. Extract the surrounding samples that constitute the packet.

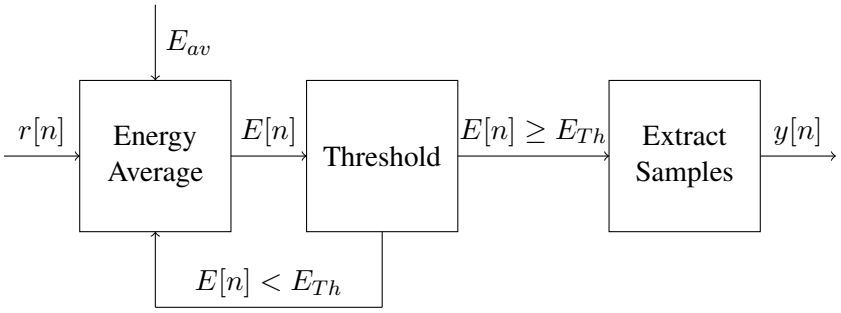


Figure 3.7: Energy based packet detector

If $x[n]$ is the received signal, then the running average of the energy is computed by:

$$e[n] = \frac{\sum_{i=0}^{2M-1} \|x[n+i]\|^2}{2M}$$

The number of samples to extract depends on whether the packet in question is a data packet or an acknowledgement packet. If the number of bits in the packet is N, then the number of samples to extract is $(2L_BM) + \left(\frac{NM}{2}\right) + L_R - 1$, L_R being the length of the RRC pulse used by the transmitter to generate the packet.

Further improvements can made to the speed of the algorithm, by reducing the number of comparisons to 1 in M samples instead of every sample, and by reducing the number samples to take the running average on.

Since AWGN is not considered to be a problem for the short packet transceiver system, further improvements can be made simply by considering the sample amplitude instead of the average energy. However, the energy based packet detector would be more reliable in the event of a high occurrence of zero crossings, which can happen if the packet data consists of alternating 1's and 0's. In this event, an average would give a better estimate of the actual packet size. Also, the computation of the average energy did not form a bottleneck in the receiver design, and thus improving its speed would not significantly speed up the receiver.

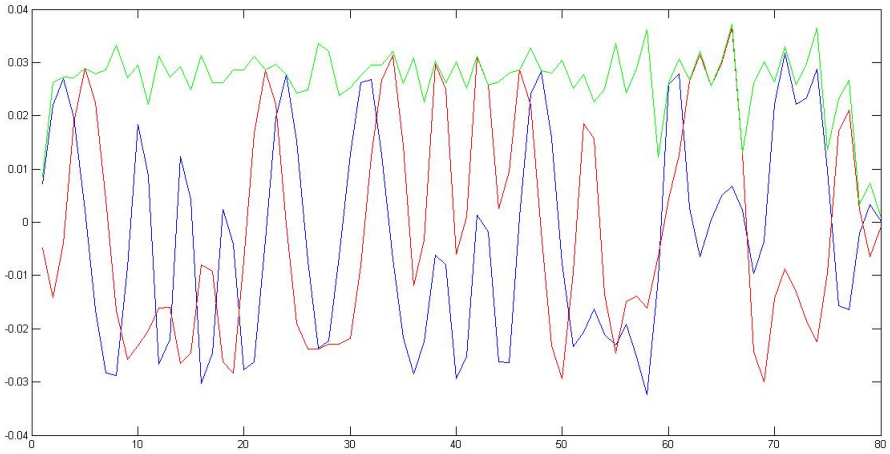


Figure 3.8: The samples extracted by the energy based packet detector

3.2.2 Timing Acquisition

The timing acquisition module forms the most expensive module in terms of time. [16], and [9] detail various algorithms for timing acquisition and tracking. However, our packet size is so small that timing tracking is never a concern. Also, our algorithm requires as small a training set as possible. Hence, the algorithm we chose is derived from the Harris Rice Synchronization algorithm in [12]. This is a timing acquisition algorithm involving a convolution with a matched filter and an energy averaging to determine the optimum sampling time. The RRC pulse was chosen as the pulse shaping filter in the transmitter, so we use an RRC pulse as a matched filter to obtain the symbols from the received signal. The resulting filtered signal becomes equivalent to a symbol stream convolved with a Raised Cosine

(RC) pulse, which satisfies the Nyquist criteria for a minimum Inter-Symbol Interference (ISI). We use a polyphase filter bank, whose filters are derived from an RRC pulse with an oversampling factor of $4M$. The signal is convolved with each filter in the bank, and the filter and sample index with the maximum average energy are chosen. We assume that the packet is small enough for there to be no clock slip across the packet. The signal is then downsampled by a factor of M to obtain the symbol stream with the maximum energy. It is essential that timing acquisition be the first block in the receiver system, because all the other modules in the receiver design require a minimum ISI for optimum functioning.

The polyphase filter bank is an array of 4 filters derived from a base filter. The base filter is an RRC pulse with one main lobe, 3 side lobes, at an oversampling rate of $4M$. The i^{th} filter is designed as follows:

$$f_i(x) = f_{base}(4x + i) \quad i = 0, 1, 2, 3$$

An alternate timing acquisition algorithm can be found in [17]. This involves a spectral analysis of the input signal, which is advaantageous in terms of speed, but loses accuracy with a drop in the number samples taken. Increasing the size of the FFT does not improve the accuracy of the algorithm, and thus, it was dropped in favour of the Harris Rice filter.

Further improvements to the Harris Rice synchronization algorithm can be made by running each of filtering and energy averaging operations in parallel.

3.2.3 Carrier Frequency Offset (CFO) Estimation and Correction

The next stage in the receiver design is the CFO estimation and correction. [11] offers a comparison of 3 different CFO estimation methods: a Maximum Likelihood (ML) method, a method based on the Kurtosis of the signal, and the FFT method, which as we have chosen to do, involves analyzing the spectrum of the fourth power of the signal. While the paper details their performance at low SNR, it goes on to say that above an SNR of 10 dB, all three estimators are identical in performance. The FFT method is the simplest of the three both in terms of implementation and computational complexity. Thus, it was chosen over the other available algorithms.

Once the symbols $s[n]$ corresponding to a packet are extracted, a crude CFO estimation and correction algorithm is performed to obtain an estimate of the transmitted symbols with a phase offset. The signal $y[n]$ is computed such that,

$$y[i] = s[i]^4$$

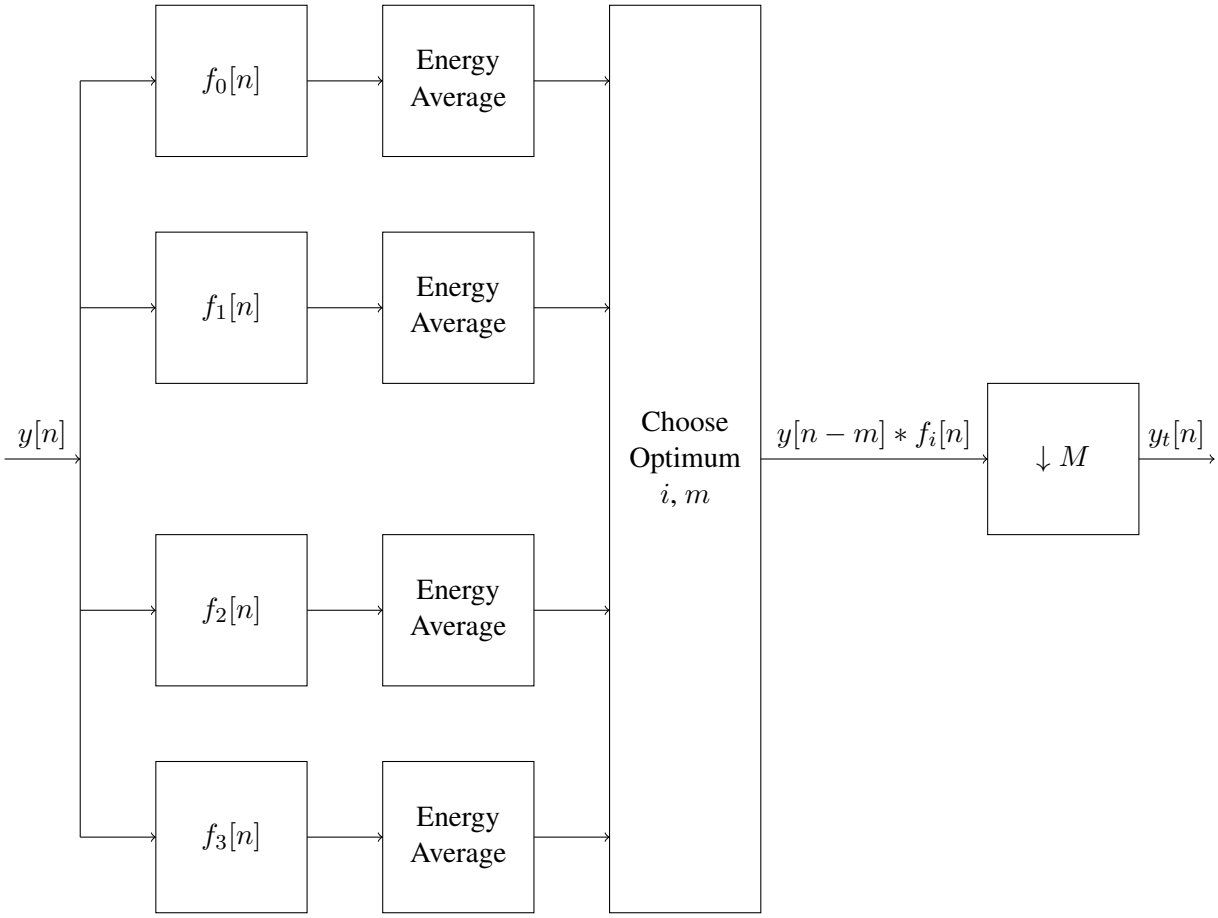


Figure 3.9: Timing acquisition module

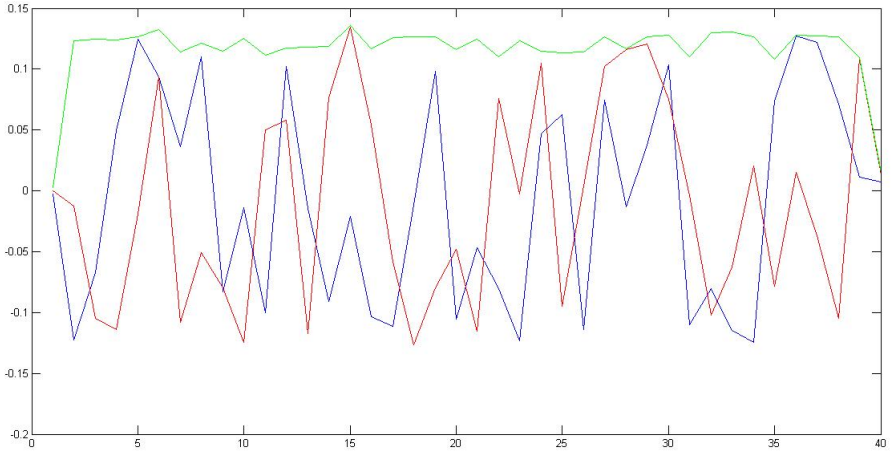


Figure 3.10: Packet symbols output by the Harris Rice synchronization algorithm

The peak of the spectrum of $y[n]$ gives us a rough estimate of the carrier frequency offset for the signal $s[n]$. $s[n]$ is then shifted by $-\frac{f_{err}}{4}$ to obtain $\tilde{s}[n]$, where f_{err} is the distance of the peak from the centre of the spectrum of $y[n]$.

$$\tilde{s}[n] = s[n] \exp\left(-j\frac{2\pi f_{err} n}{4N_{FFT}}\right)$$

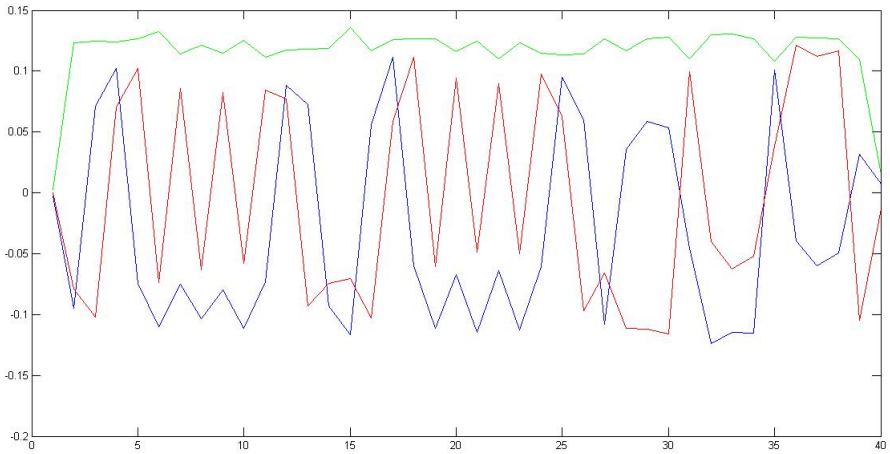


Figure 3.11: Packet symbols after CFO correction

3.2.4 Correlator

After the CFO estimation and correction is performed, a block-wise correlation is performed to find the location of the Barker sequence at the start of the received packet. If $\tilde{s}[n]$ is the signal obtained after the frequency offset correction, then the correlation signal $c[n]$ is computed as:

$$c[n] = \left\| \sum_{i=0}^{L_B-1} \tilde{s}^*[n+i]\tilde{s}[n+i+L_B] \right\|$$

The peak of the correlation corresponds to the start of the Barker sequence, which is transmitted twice before the packet. Thus, the start of the packet n_0 can be determined as:

$$n_0 = (\arg\max c[n]) + 2L_B$$

[13] discusses the use of a correlation operation and a correction term to compensate for the occurrence of a plateau in the correlation values. In our system, it was determined that the signal power would always be maintained significantly higher than the noise power. With that assumption, in the event of

a plateau, the start of a plateau would correspond to the start of the packet. Thus, no correction term would be needed.

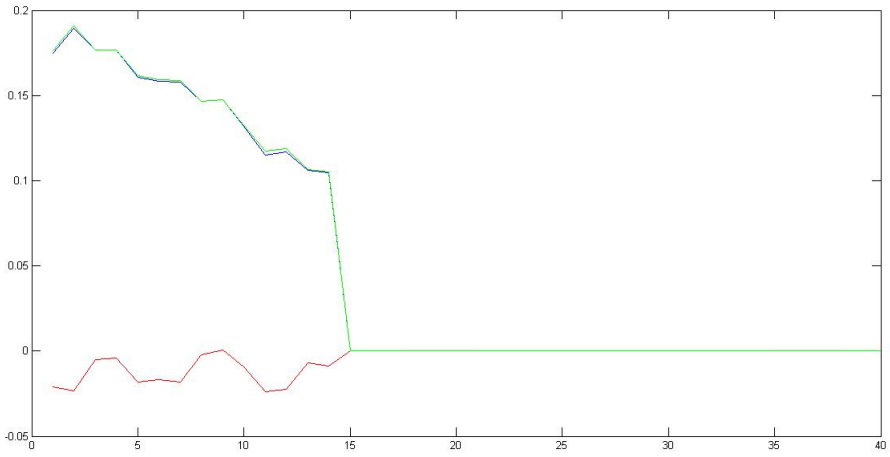


Figure 3.12: Correlation output indicating the start of the Barker synchronization sequence

3.2.5 DQPSK Demodulation

Once the symbols corresponding to a packet are extracted, a DQPSK demodulation is performed on the given symbols, with the previous phase as a reference for the demodulation.

$$\phi[n] = \angle \tilde{s}[n] \quad n = 0, 1, ...$$

$$\phi[-1] = \text{reference\_phase}$$

$$(\hat{b}[2n], \hat{b}[2n+1]) = \begin{cases} (0,0) & \phi[n] - \phi[n-1] \leq \frac{\pi}{2} \\ (0,1) & \frac{\pi}{2} < \phi[n] - \phi[n-1] \leq \pi \\ (1,0) & \pi < \phi[n] - \phi[n-1] \leq \frac{3\pi}{2} \\ (1,1) & \frac{3\pi}{2} < \phi[n] - \phi[n-1] \leq 2\pi \end{cases}$$

3.3 Packet Design

Existing standards, including [1] and [2] were looked at in an effort to determine the best possible solution for the data and acknowledgement bit structures. Straight away, it was determined that Orthogonal Frequency Division Multiplexing (OFDM) would not be an ideal solution due to the large size of in-

dividual OFDM symbols. Thus, we took our cues from 802.15.4, and minimized both the data being transmitted as well as the remaining packet information like the packet index, source and destination addresses, etc. A 4-bit Cyclic Redundancy Check (CRC) is used with both data and acknowledgement packets to add some measure of error detection within the system itself.

3.3.1 Data Packet Bit Structure

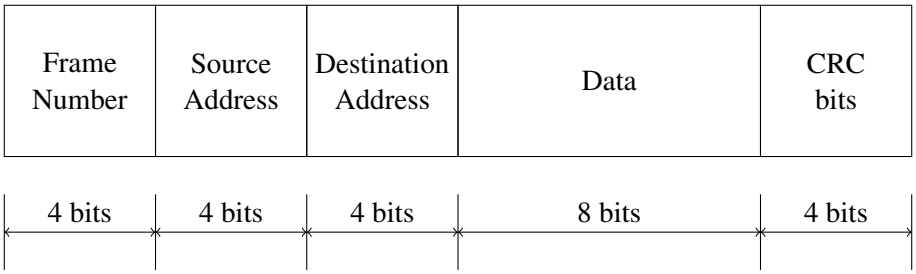


Figure 3.13: Data Packet Design

The data packet consists of:

1. 4 bits to denote the frame number
2. 4 bits for the source address
3. 4 bits for the destination address
4. 8 bits for the data
5. 4 bits for the Cyclic Redundancy Check (CRC)

Due to the lack of a packet queue, the frame numbers are recycled as and when required. This is possible because at any given point of time, the transmitter is waiting for an acknowledgement corresponding to a single packet. The system contains only one source for now, so that address is set to 0000. There are two possible destinations right now. The receiver with the address 0001 receives any packets containing the characters '0' through '9' as their data. The receiver with the address 0000 receives all other packets. The data consists of a single character, obtained from the keyboard input to the PC connected to the USRP that acts as the transmitter. The CRC sequence is used by the corresponding receiver to validate the packet before displaying the data and transmitting the acknowledgement. The generator sequence used for the CRC sequence is 11011.

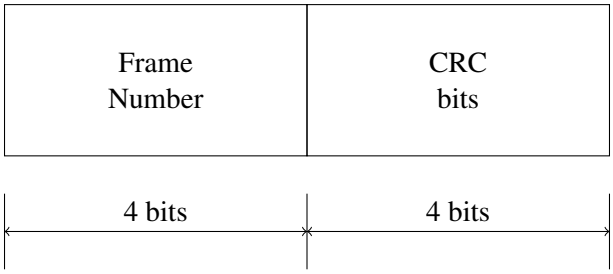


Figure 3.14: Acknowledgement Packet Design

3.3.2 Acknowledgement Packet Bit Structure

The acknowledgement packet consists of:

1. 4 bits to denote the frame number
2. 4 bits for the CRC

The acknowledgement packet contains only the frame number because at any given point of time, only one receiver is transmitting an acknowledgement to the only transmitter in the system. The CRC generator sequence used is 11011, the same as that of the data packet.

3.4 System Design

Both the transmitter and receiver are transceiver systems equipped with packet transmission and reception capabilities as described in sections 3.1 and 3.2. The system is setup to transmit a keypress event in a terminal on one computer to another, and to display the key pressed on the second computer. The transmitter waits for a key press event involving an ASCII character c before sending one data packet containing the ASCII value of c as the data. The destination address of the packet is determined based on the actual key pressed. If c is a number between 0 and 9, then the packet is addressed to Receiver 1. Otherwise, the packet is addressed to Receiver 0. Each receiver listens for a packet, detects and decodes the packet and determines the destination address as well as the data $\hat{c}$. If the destination address corresponds to the given receiver, it prints the character $\hat{c}$, and transmits an acknowledgement containing the frame number of the received data packet. The transmitter waits for the acknowledgement before moving on. If the acknowledgement is not received within a given time frame, then the transmitter enters a loop of repeating the transmission and waiting for the acknowledgement.

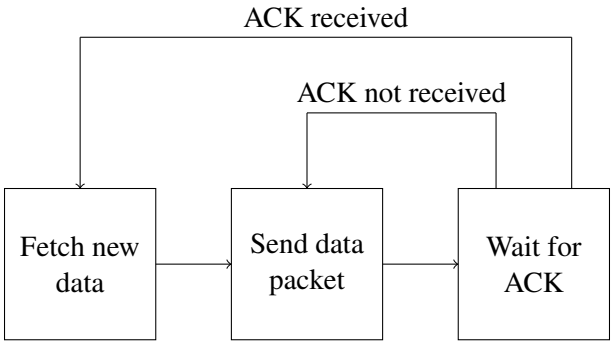


Figure 3.15: Transmitter Design

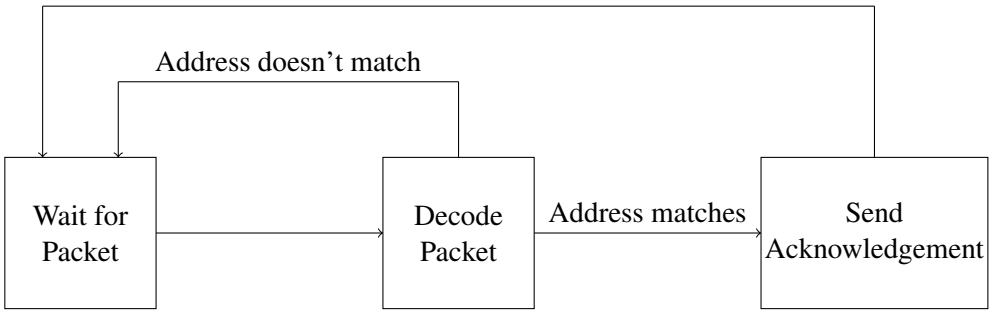


Figure 3.16: Receiver Design

3.5 Observations

The data packets are transmitted at a frequency of 2.42 GHz at a rate of 250 kHz, and the acknowledgement packets are transmitted at a frequency of 2.22 GHz at a rate of 250 kHz, making this system a half duplex system. The total number of samples in a data packet is 76, and the total number of samples in an acknowledgement packet is 60. This system has a turnaround time of < 1 ms, at a sampling rate of 250 kHz. With a faster computer and a higher number of parallel processes, the sampling rate can potentially be brought all the way up to 40 MHz, bringing the turnaround time to ≈ 1 μs. Further improvements can be made to the speed by one or more of the following processes:

- Reducing the length of the synchronization sequence.
- Turning the transceiver system into a full duplex system.
- Optimizing and combining some of the synchronization algorithms.

However, at some point, transferring the samples from the USRP buffer to the PC might begin to cause a significant overhead. At this point, the only way to further increase the transmission speed would be to offload as much as possible of the signal processing functions onto the in-built FPGA on the

USRP's motherboard. The steep learning curve and slow compilation time involved with FPGA coding make this prohibitive, and thus, it might be better to have this as a last option.

CHAPTER 4

Sequence Detection

4.1 Problem Statement

The second aspect of the asynchronous communication problem we look at involves the optimization of the transmission process. [6] discusses a one-shot frame synchronization problem. In this paper, a pattern is transmitted in a time-slotted channel with a known asynchronism level of A, over an AWGN channel with average noise power N_0. In other words, a pattern of a fixed length N is transmitted at a random slot ν within a window of A slots. This pattern is then detected and decoded with a sequential detector. The paper states that there exists a synchronization threshold A_0 beyond which even a maximum likelihood detector will not be able to reliably detect the pattern in the window. The paper considers a time-slotted system, indicating at least some level of synchronization between the transmitter and receiver. The level A_0 is found to grow exponentially with N, the length of the packet. This is found to be a direct consequence of the asynchronism level's dependence on the entropy of the arrival process, and our choice of a uniform arrival process. Thus, we get the following result for A:

$$A = 2^{\alpha\left(\frac{P}{N_0}\right)N}$$

where α depends on the required BER to be maintained.

The complication on the receiver side is caused by the noise levels in the channel. With a higher noise power, or a higher asynchronism level for a given noise power, it becomes increasingly likely that a sequence of noise samples will be erroneously marked by the receiver as the required pattern. Also, the sequential decoder might simply miss the pattern and assume that the window in question does not contain the required pattern. The design of the sequential decoder, and the choice of its threshold strikes a compromise between these two events. In our experiments we minimize the probability that a given window is declared void of a packet, while increasing the probability of an erroneous detection.

We seek to experimentally verify the results stated in [6] by setting up a transmitter and receiver as detailed in the following sections. We also seek to extend the experiment to allow for a variable power of transmission, and to remove the slotted assumption made in the paper. The experiment we designed has been made repeatable, to allow us to obtain meaningful BER and pattern acquisition error rate plots.

We use small pattern lengths of $N = 1$ and $N = 4$, a fixed transmission power $P = 0.1$, SNR values in the range of 5-25 dB, and asynchronism levels A in the range 256-65536. We run the experiment for K iterations at a time. Since our goal is to determine the value of A at which the error rate begins to climb, we need not press for precise values of the error rate, and hence, $K = 1000$ is taken to be sufficient for our purposes. The following sections detail the transmitter and receiver designs, and the results.

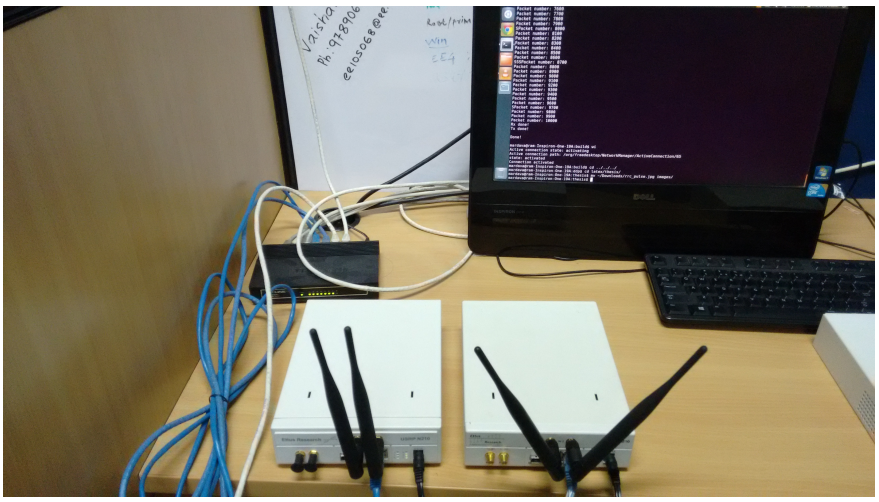


Figure 4.1: Front view of the Sequence Detection System

4.2 Transmitter Design

The objective of this system is accurately determine the location of and to decode a pattern of a known length and power, transmitted at a random instant of time ν within a window of A samples. The instant of transmission $\nu \sim U(0, A)$. Prior to transmitting the window of A samples, the transmitter sends a known synchronization sequence at a much higher transmission power than that of the pattern. The window of A samples is then generated with the pattern located at a random instant ν from the start of the window. The pattern is an On Off Keying modulated stream of a bit stream of a pre-determined size N. The experiment also calls for maintaining an SNR value of $\left(\frac{P}{N_0}\right)$, where P is the average power of the pattern, and N_0 is the average noise power. Since the ambient noise in the system of two USRP kits is much lower than our required noise levels, a sequence of complex normal samples is generated and added to the transmitted signal. This signal is then transmitted at a sampling rate of 250 kHz, and a central frequency of 2.42 GHz.

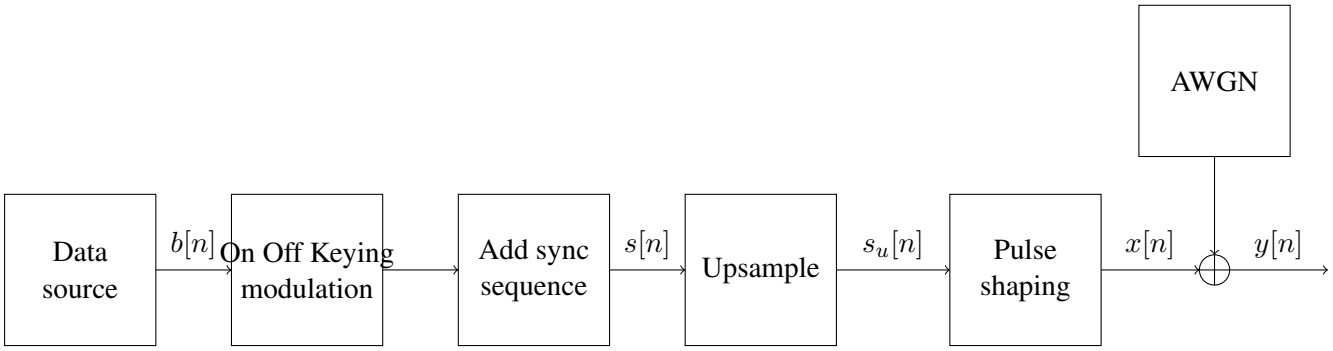


Figure 4.2: Transmitter System

4.3 Receiver Design

The receiver is designed to search for the synchronization sequence at the start of every window of length A samples. Once the synchronization sequence has been found and decoded, the start of the window of A samples is located. The window of A samples is then received in a block of $L = MN$ samples at a time. A running kurtosis is then computed on each of these blocks. A trough in the kurtosis values will indicate the location of the sequence. Since On Off Keying is the modulation scheme used, a simple timing acquisition is sufficient to determine the original sequence transmitted. The location of the sequence and the sequence itself are then written to file. The transmitter output is also written to file, and compared with the receiver output to determine pattern acquisition error rate, and the bit error rate.

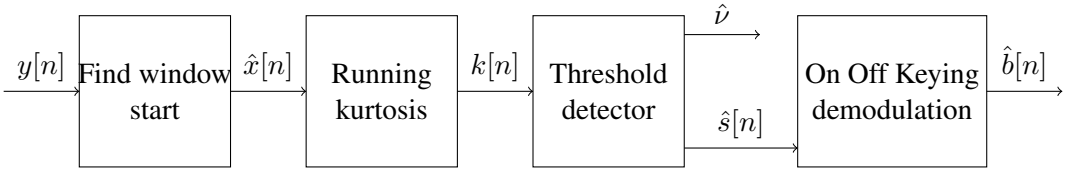


Figure 4.3: Receiver Design

4.3.1 Finding the Start of a Window

We use the same synchronization sequence that we used for the transceiver system, and thus, finding the start of the window becomes a problem very similar to detecting and decoding a packet in the transceiver system. The same steps are followed to locate the start of the window of A samples. The synchronization sequence is first detected via an energy averaging method, which makes it imperative that the synchronization sequence be transmitted at a significantly higher power than the pattern or

noisy samples in the window. The sequence is then subjected to a timing acquisition, frequency offset estimation and auto-correlation to determine the exact start of the synchronization sequence. Following this, the start of the window is determined by a simple increment, and the next A samples are then taken in a block-wise manner for processing.

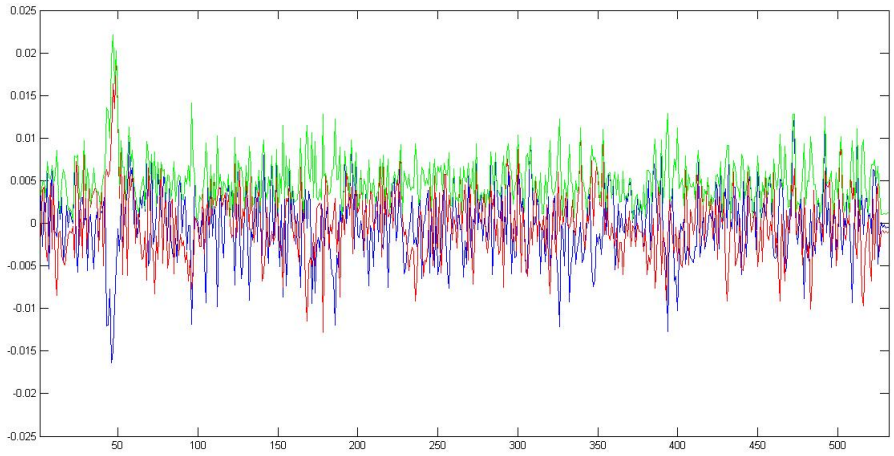


Figure 4.4: The received window for a window size of $A = 512$, pattern transmission power of $P = 0.1$ and SNR $= 10$

4.3.2 Running the Kurtosis

A block-wise kurtosis is computed on the group of samples, with the block-length comparable to the size of the pattern. The samples are taken 200 at a time to prevent the USRP buffer from overflowing due to our code's slow execution rate. The block-wise kurtosis was chosen because the 4^{th} moment of a Gaussian random variable is zero. As a result, the kurtosis of the signal offers us a better chance at locating the packet in a low-SNR situation than an energy based detector. For a pattern length $N = 4$ and transmission power $P = 0.1$, a good threshold value was computed to be $-5e - 8$, for all SNR values considered, indicating that the threshold value depends only on the block-length of the kurtosis and the transmission power.

4.3.3 On Off Keying demodulation

The On Off Keying demodulation module also contains a timing acquisition function to reduce the over-sampling rate to 1 sample per symbol. Then, the energy of the symbols is computed and a hard threshold

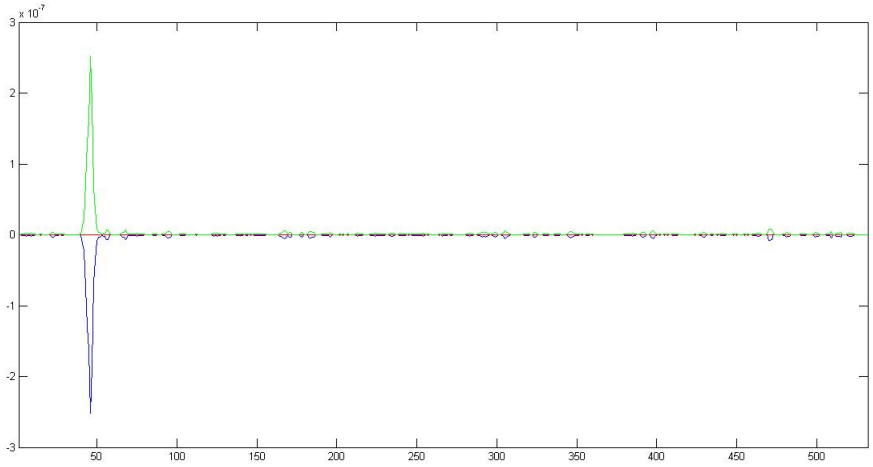


Figure 4.5: The kurtosis values for the above received window

is used to decode the bits from the given samples. These bits are then written to file, along with the computed location of the pattern and the window index. After the pattern is found, the receiver waits for the approximate start of the next window before searching for the synchronization sequence, regaining synchronization with the transmitter, and taking the next window of A samples into consideration.

4.4 Computing the BER

The transmitter and receiver outputs are written to file. Each transmitter output entry consists of the window number n, the index ν_n at which the pattern is located, and the bit sequence $\mathbf{b}_n$ from which the pattern was constructed. Each receiver output entry contains the window number n, the index $\hat{\nu}_n$ at which the pattern was found, and the bit sequence $\hat{\mathbf{b}_n}$ recovered from the pattern. The acquisition error rate is defined as the fraction of windows where the pattern was not located correctly. Due to small inaccuracies in the synchronization process, we use an error threshold of 4 samples. Hence, the acquisition error rate (AER) is computed as:

$$\text{AER} = \frac{\sum_{n=0}^{K-1} \mathbf{I}\left(\|\nu_n - \hat{\nu}_n\| \geq 4\right)}{K}$$

The transmitter runs longer than the receiver, and thus outputs a higher number of windows than K. Thus, we need to find the transmitted windows which correspond to K. Normally it would be sufficient

to find the offset m such that,

$$m = \arg\min\left(\text{AER}_m\right)$$

$$\text{AER}_m = \frac{\sum_{n=0}^{K-1} \mathrm{I}\left(\|\nu_{n+m} - \hat{\nu}_n\| \geq 4\right)}{K}$$

However, the USRP receiver code runs slow for large values of A, in the range 20000-70000. At these values of asynchronism, the receiver loses roughly 1 window in 2000. Hence, a simple text comparison of the outputs is not sufficient to determine the accurate error rate. Instead, we adopt a greedy algorithm, which seeks to minimize the AER for each group of 10 windows received. Once that is done, the BER is computed by comparing the patterns transmitted with the patterns received.

4.5 Observation

The resulting packet acquisition error rate is plotted for various SNR values with the asynchronism level on the x-axis. As we can see the general trend is an increase in the error rate with the asynchronism level. As the SNR increases, the asynchronism level capable of supporting a given BER also increases.

The plot was obtained for a pattern of length $N = 4$, with SNR values ranging from $\frac{P}{N_0} = 5$ to

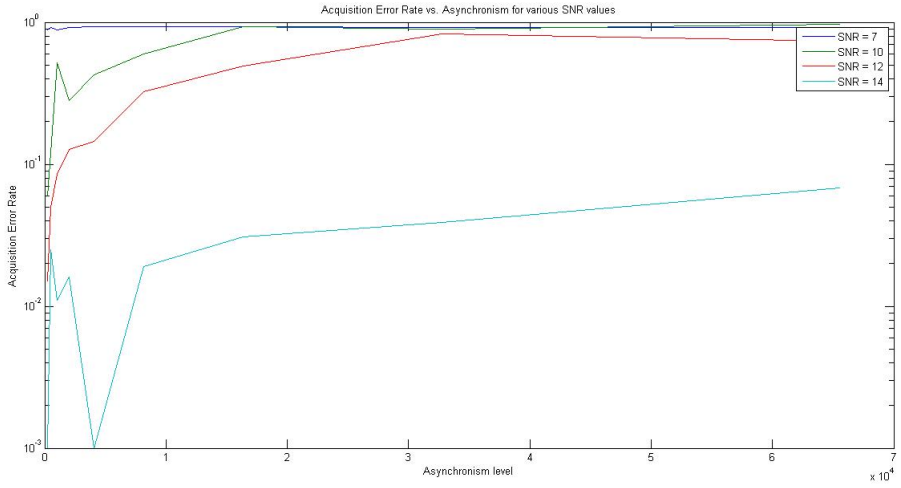


Figure 4.6: Log plot of the packet acquisition error rate against the asynchronism level for various values of SNR

$\frac{P}{N_0} = 25$, and asynchronism values from $A = 256$ to $A = 65536$. The number of iterations for each plot is 1000. This experiment should be repeated with different values of N. It would also be beneficial

to vary P for a given N, and see how the results change. The new kurtosis thresholds wwould have to be computed in these cases.

CHAPTER 5

Conclusion

This project seeks to explore the asynchronous communication problem using the USRP RF transceivers in its experimental setup. Two aspects of the asynchronous communication problem are studied in this manner, each requiring the construction of a communication system implemented on the USRP N210 kits. The first aspect dealt with constructing a short packet transceiver system, with the intent of minimizing the transmission time. The short packet transceiver system was designed to have a best-case turn-around time of < 1 ms, although this can be further improved by offloading some of the signal processing algorithms onto the FPGA, and optimizing and combining existing synchronization algorithms to reduce the number of instructions executed by the PC per packet.

The pattern transmission and reception system was designed to analyse the asynchronism level present in the system, and its relation to other system parameters like the pattern length and the SNR value. Theoretical results indicated that for an AWGN channel, there was an exponential relation between the asynchronism level and the length of the packet, for a given BER. Experimental verification of this result necessitated the construction of a one-way pattern transmission system. With this system, we saw that SNR values of 10-15 dB have their asynchronism thresholds located in the 256-65536 range, and similarly, the asynchronism thresholds can be computed for other SNR values and pattern lengths. Future work on the pattern transmission system can involve scaling up the pattern length and the asynchronism level to mimic a real-world application. Also, some work can be done to optimize the synchronization process, as for such a small pattern length, the Harris Rice synchronization algorithm becomes an overkill. Further research into a better metric than a block-wise kurtosis, and the performance of the kurtosis metric on non-AWGN channels would also be beneficial.

With a rising demand for low-power devices and stand-alone data acquisition units, the idea of drastically reducing the transmission power by exploiting the arrival process offers potential for further study. The already existing demand for systems with a low response latency ensures that the short packet transceiver system will find applications in either its current, or an optimized form. Thus, more than 40 years after Massey's seminal paper on frame synchronization, the problem of asynchronous communication remains a relevant part of the field of communication systems.

REFERENCES

[1] (1999). Ieee 802.11a: Wireless lan medium access control (mac) and physical layer (phy) specifications: High-speed physical layer in the 5 ghz band.

[2] (2003). Ieee 802.15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans).

[3] **Ascheid, G.**, **M. Oerder**, **J. Stahl**, and **H. Meyr** (1989). An all digital receiver architecture for bandwidth efficient transmission at high data rates. *IEEE Transactions on Communications*, **37**, 804–813.

[4] **Barker, R. H.** (1953). Group synchronizing of binary digital systems. *Communication Theory. London: Butterworth*, 273–287.

[5] **Benvenuto, N.**, **A. Salloum**, and **L. Tomba** (1996). An all digital receiver for dect radios. *Communications, 1996. ICC '96, Conference Record, Converging Technologies for Tomorrow's Applications. 1996 IEEE International Conference on*, **2**, 862–866.

[6] **Chander, V.**, **A. Tchamkerten**, and **G. Wornell** (2008). Optimal sequential frame synchronization. *IEEE Transactions on Information Theory*, **54**, 3725–3728.

[7] **Freedman, A.**, **Y. Rahamim**, and **A. Reichman** (2006). Maximum-mean-square soft-output (m2s2o): a method for carrier synchronisation of short burst turbo coded signals. *Communications, IEE Proceedings*, **153**, 245–255.

[8] **Fujita, T.**, **D. Uchida**, **Y. Fujino**, **O. Kagami**, and **K. Watanabe** (2007). A short-burst synchronization method for narrowband wireless communication systems.

[9] **Gardner, F. M.** (1986). A bpsk/qpsk timing-error detector for sampled receivers. *IEEE Transactions on Communications*, **34**, 423–429.

[10] **Golomg, S. W.** and **R. A. Scholtz** (1965). Generalized barker sequences. *IEEE Transactions on Information Theory*, **11**, 533–537.

[11] **Gunther, J.** and **T. Moon** (2009). Burst mode synchronization of qpsk on awgn channels using kurtosis. *IEEE Transactions on Communications*, **57**, 2453–2462.

[12] **Harris, F. J.** and **M. Rice** (2001). Multirate digital filters for symbol timing synchronization in software defined radios. *IEEE Journal on Selected Areas in Communications*, **19**, 2346–2357.

[13] **Massey, J. L.** (1972). Optimum frame synchronization. *IEEE Transactions on Communications*, **20**, 115–119.

[14] **McKilliam, R.**, **A. Pollok**, and **W. Cowley** (2014). Simultaneous symbol timing and frame synchronization for phase shift keying. *IEEE Transactions on Communications*, **62**, 1114–1123.

[15] **Meyr, H.**, **M. Oerder**, and **A. Polydoros** (1994). On sampling rate, analog prefiltering and sufficient statistics for digital receivers. *IEEE Transactions on Communications*, **42**, 3208–3214.

[16] **Mueller, K. H.** and **M. Muller** (1976). Timing recovery in digital synchronous data receivers. *IEEE Transactions on Communications*, **24**, 516–531.

[17] **Oerder, M.** and **H. Meyr** (1988). Digital filter and square timing recovery. *Communications, IEEE Transactions on*, **36**, 605–612.

[18] **Vaidyanathan, P. P.** (1990). Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proceedings of the IEEE*, **78**, 804–813.

[19] **Zhang, L.** and **A. Burr** (2002). Appa symbol timing recovery scheme for turbo codes. *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, **1**, 44–48.