Spring 2011: 18551

# PROJECT SDR

~ Remote Terminal Access Control System

Group # 7


Anish Menon

anishm@andrew.cmu.edu

Akash Gopisetty

agopiset@andrew.cmu.edu

# Abstract:

*Transferring sensitive data wirelessly using the internet is not the safest option. Also depending upon the internet implies relying on the service provided by your ISP for the functioning of your system.*

*With Project SDR we aim to establish a private, safe wireless channel between two terminals using SDRs (Software Defined Radios). This channel can be used to transfer access request data from users (here a picture of the user's face) which can then be verified at the receiving terminal and access to the user can be either granted or denied.*

# CONTENTS:

# 1. Introduction:

The internet is the most popular and convenient mode of communication. But it is plagued with several problems such as:

- Lack of Security :

Since the internet is easily accessible and open to all, any terminal connected to the internet can be hacked. This could be the cause of potential security threat as the hacker can get access to confidential information and spoof identity.

- Virus Threat:

Any terminal connected to the internet is open to the potential threat of a virus attack which can disrupt the normal functioning of the system. Private and confidential information could be extracted and misused.

- Reliability:

Using the internet implies that the system is dependent on an ISPs for service. Thus any failure on the ISP's part would affect the system.

- Limited by wired distance:

Setting up a private network would require a combination of wired and wireless medium, bringing in complexity.

Through Project SDR we provide an alternate to the use of the internet for short range communication for transferring sensitive data. The goal of our project is to establish a private wireless link which is secure and robust.

- Ensuring Security:
  Since the specifications of the system are designed by us, security is ensured. We control the frequency of transmission, modulation, encoding schemes and can implement encryption techniques to keep the network exclusive.

- Robustness:
  Since the system setup, running and maintenance is all performed by us, we ensure no third party dependence.

- Medium Homogeneity:

Since the system deals only with wireless mode of communication, setting up and maintaining the system becomes comparatively simple.

Since the aim of our project is to design our own channel, we implemented different modulation and encoding schemes and compare their performance while transferring an image.

# 2. System Implementation:

## 2.1 HARDWARE:

We implemented our wireless channel using Software Defined Radios. Working with SDRs is a lot easier than working with radios with fixed specifications as they offer a lot of flexibility. SDRs utilize an external general purpose processor instead a specialized one on its hardware. This feature allows the SDR to be compatible with a variety of platforms. SDRs typically consist of an ADC, DAC and RF front end.

## 2.2 ADVANTAGES OF SDRs:

SDRs offer the following advantages over traditional fixed radios:

a) They have the ability to receive and transmit using various modulation and encoding methods using a common set of hardware.
b) Allow user to alter functionality by downloading and running new software at will.
c) The possibility of adaptively choosing an operating frequency and a mode best suited for prevailing conditions.
d) The opportunity to recognize and avoid interference with other communications channels.
e) Elimination of analog hardware and its cost, resulting in simplification of radio architectures and improved performance.
f) They provide the chance for new experimentation.

## 2.3 DISADVANTAGES OF SDRs:

While SDRs offer benefits as outlined above, a few obstacles remain to their universal acceptance. Those include:

a) The difficulty of writing and maintaining software for various target systems.
b) The need for interfaces to digital signals and algorithms.
c) Poor dynamic range as compared to fixed radios

Due to the above drawbacks, the use of SDRs is usually restricted to academic and research purposes for testing and experimentation.

## 2.4 WHICH SDR?

We choose to work with the SDR, USRP1 (Universal Software Radio Peripheral) manufactured by the Ettus Research Group. We are grateful to Prof. Negi providing us with a pair of USPR1 hardware kits for the duration of our project.


Fig: USPR1 Hardware

## 2.5 KEY FEATURES OF USRP1:

- Four 64 MS/s 12-bit analog to digital converters
- Four 128 MS/s 14-bit digital to analog converters
- Four digital down converters with programmable decimation rates
- Two digital up converters with programmable interpolation rates
- High-speed USB 2.0 interface (480 Mb/s)
- Capable of processing signals up to 16 MHz wide
- Modular architecture supports wide variety of RF daughter boards
- Auxiliary analog and digital I/O support complex radio controls such as RSSI and AGC
- Fully coherent multi-channel systems (MIMO capable)

For details about the Ettus Research Group:
http://www.ettus.com/

Further details and specifications of USPR1 can be found on:
http://www.ettus.com/downloads/ettus_ds_usrp_v7.pdf

## 2.6 SOFTWARE INTERFACE:

The USRP1 can be interfaced to a computer with a few options, GNU Radio or MATLAB or SIMULINK. For our implementation we use MATLAB to link to the SDRs.

### 1) GNU Radio:

GNU Radio is a free software development toolkit that provides the signal processing runtime and processing blocks to implement software radios using readily-available, low-cost external RF hardware and commodity processors. It uses C++ to carry out the signal processing in the system.

It is widely used in hobbyist, academic and commercial environments to support wireless communications research as well as to implement real-world radio systems.

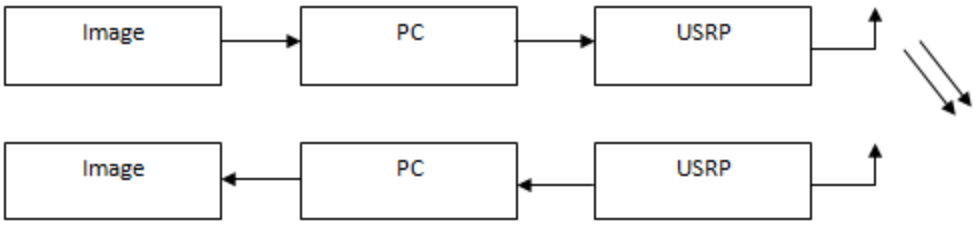
2) MATLAB and SIMULINK:
Interfacing the USRP hardware with MATLAB and SIMULINK is explained in detail in: http://www.tools4sdr.com/wiki/Tools4SDR.
The MATLAB codes we used to interface the SDRs with our computers is detailed in the codes section.

We did encounter problems with the MATLAB interface codes provided by tool4sdr , while transferring continuous stream of data. The code works perfectly for transferring short bursts of data.

From our experience with the MATLAB interface we recommend using the approach mentioned by GNU Radio and the use of C++ to interface with the radios.

# 3. Block Diagram of the System:

## 3.1 The Big Picture of the System:




All signal preprocessing, implementation of modulation, encoding schemes have to be implemented on a PC before passing the digital data to the SDR. The USRP uses it D/A converters, and modulates the signal at a specified signal frequency and then transmits the signal over the wireless channel. At the receiver, the USRP receives the signal demodulates the carrier and converts it into a digital signal and passes it to the computer for processing.

Since we were unable to transfer large streams of data we simulated the image transfer in MATLAB.

3.2 Transmitter (Detailed Block Diagram):

We convert the image to be transferred into a stream of pixels and then into a stream of bits. These bits are then encoded, followed by digital and then analog modulation.

We implemented the rate ¼ and ¾ coding, decoding algorithms and the M-QAM and M-PSK modulation/ demodulation algorithms.

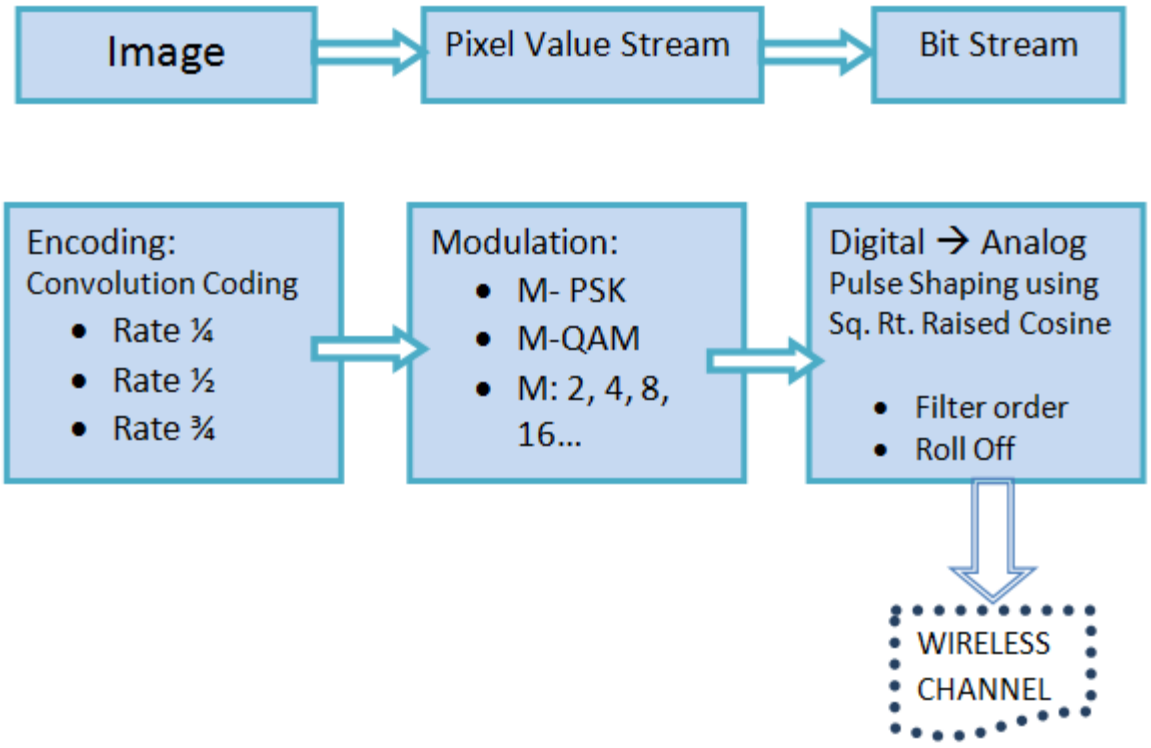



Fig: Block Diagram of the Transmitter

3.3 Receiver (Detailed Block Diagram):

The transmitted signal plus noise is received at the receiver, this anolog is first converted into a digital form. This digital signal goes through the reverse process as that in the transmitter. It first gets demodulated then decoded. This gives us a stream of bits which we 'guess' was sent from the transmitter.

The number of bits different in the transmitted stream and the guess of the bit stream gives the number of bit errors.

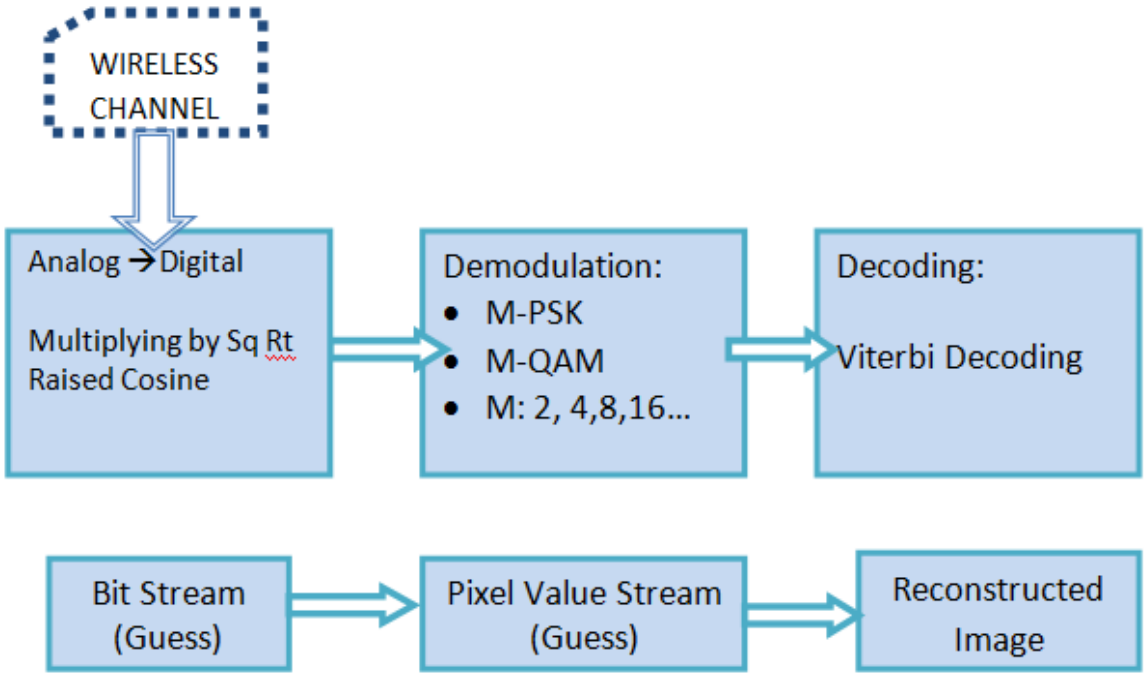



Fig: Block diagram of the receiver

# 4. Algorithms:

## 4.1 Repetition Codes

They are a type of forward error correcting codes. Described by k/n where k inputs give n outputs (n>=k).

In the transmitter, the FEC code's encoder encodes the information bits into coded bits, which are then modulated by the bits-to-symbol mapper to yield the symbol sequence (xk). The symbol sequence is transmitted into the equivalent baseband discrete-time channel, as usual. In the receiver, the received samples (zk) are decoded by the code's sequence decoder to yield estimates of the information bits. In particular, symbol-by-symbol detection is not appropriate, due to the dependencies in (xk) created by the encoding operation.

A convolution code is so called, because its encoder is simply a convolution operator, i.e., an LTI system. The only difference from LTI systems that we otherwise encounter is that the encoder processes bits (bits in and bits out), rather than real or complex numbers.

Repetition code (Why we opt for convolution codes)

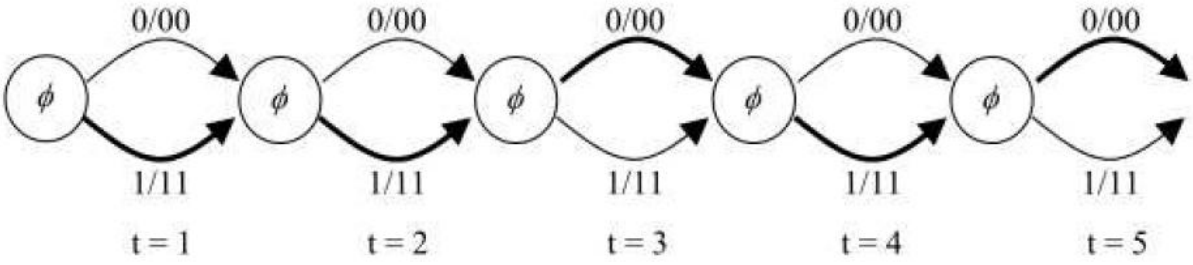



Figure 12.1: Trellis diagram of a repetition code of rate $\frac{1}{2}$.

The description of the trellis above is as follows. Each discrete time t is called a stage of the trellis. Discrete-time t controls the encoder operation. It is related, but different, from the discrete-time k, which controls the symbol and sampling operations. The circle is called the state of the trellis. (Note the difference between stage and state.) The repetition code has only one state, which we have labeled as the null state.

What is a state of a system? The fundamental property of the state of any system is that the output of the system in the future depends only on the current state and on future inputs, but not on the past inputs. Thus, a state "summarizes all the past inputs" (which could be a long sequence of numbers) into one concise description. This is the convenience provided by a state machine.

In the repetition code, there is only one state required to summarize all the past inputs, i.e., the null state. As time t progresses, the state changes, in general. However, in a repetition code, since there is only one state, the new state is always a null state. The arrows are called branches, and they are labeled as information bits/coded bits.

## 4.2 Convolutional codes: Encoding

The flaw in the repetition code can be observed from its trellis in Figure above. As there is only one state, any two paths that diverge from the state must immediately come back to the same state. Therefore, it is not possible to have two paths, which are at minimum distance d, differ by more than one branch. This puts an upper bound on how large the minimum distance d can be. We want d to be large, so that the BER is small.

It is likely that the larger the number of stages in which two paths at minimum distance differ, the larger is the minimum distance, and so, the smaller is the BER. We can increase the minimum distance doubling the number of states from one to two, as shown by the trellis in Figure below. This trellis represents a 2-state convolutional code.

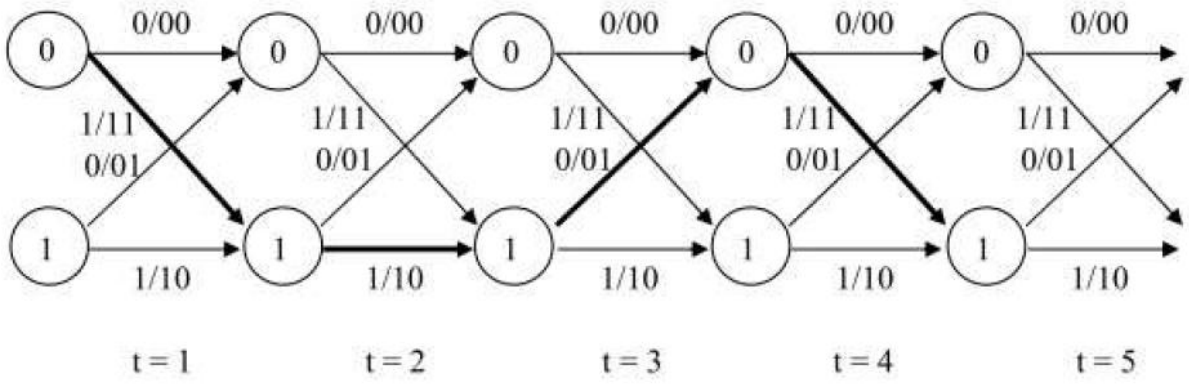



Figure 12.4: Trellis diagram of a 2-state, rate $\frac{1}{2}$ convolutional code.

As in the case of the repetition code, the circles are called states of the code. We now have two states, labeled as '0' and '1'. As time t progresses, the state can change, in general. The branches are again labeled as information bits/coded bits, just as in Repetition code. Since each information bit produces 2 coded bits, hence the rate of this code is ½. As time progresses, information bits enter the encoder, and get encoded into coded bits. The bold branches indicate the path through the trellis, which is taken when the information bit sequence is 1, 1, 0, 1, . . .. The path shown produces the coded bit sequence 11, 10, 01, 11, . . ., which is different from that produced by the earlier example of rate ½ repetition coding.

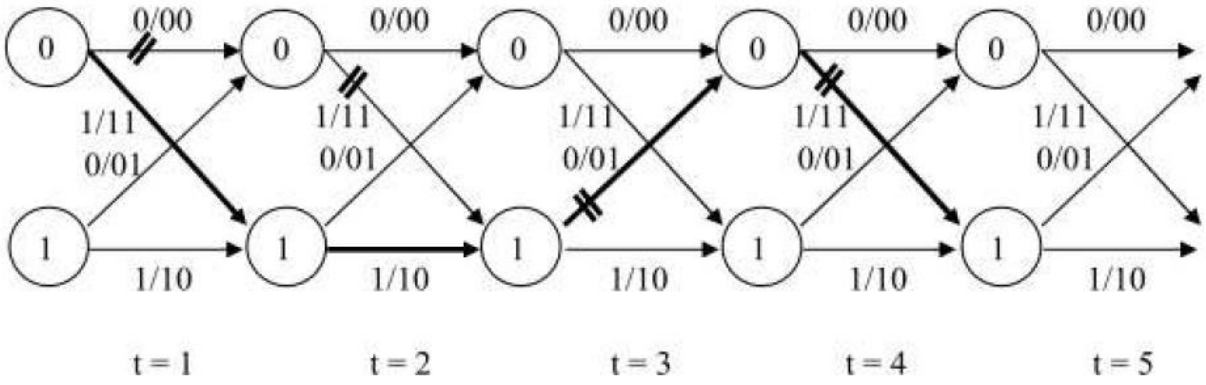



Figure 12.5: Two paths $(x_k), (y_k)$, which are at minimum distance to each other.

The path (xk) is shown by the bold branches, while the path (yk) is shown by the branches crossed by two segments. Enabled by the larger number of states of the trellis, these two paths differ from each other in two stages, unlike the case of the

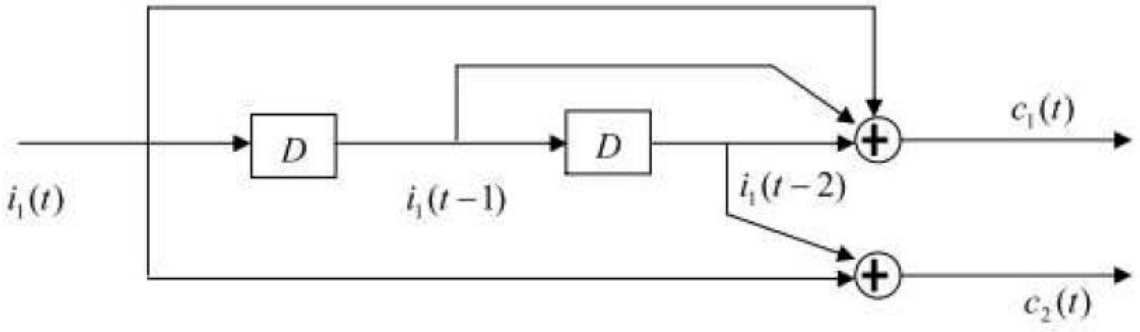
repetition code (No pairs of paths differ in only one stage.) Thus, they differ from each other in 3 coded bits (i.e., 11, 10, 01, 11,... versus 00, 11, 01, 11, . . .).

The coding gain of 3 of this rate 1/2 convolutional code is more than the coding gain of 2 of the rate 1/2 repetition code, resulting in lower BER. However, both these codes have the same information bit rate, since they are both rate 1/2 codes.

To summarize, the key idea is that by introducing more states in the trellis, we prevent divergent paths (which represent two competing allowed symbol sequences – the error events) from re-merging quickly. Thus, any two allowed symbol sequences differ in several stages, which we hope results in a larger minimum distance. (There is no guarantee that longer error events will result in a larger minimum distance – this will require an intelligent choice of trellis.) So, if we want a larger coding gain code at the same rate of 1/2, we need to double the number of states in the trellis again.

| $2^\nu$ | $g_{11}(D)$ | $g_{12}(D)$ | $d_{free}$ | $\gamma_f$ | (dB) | $N_e$ | $N_1$ | $N_2$ |
|---|---|---|---|---|---|---|---|---|
| 4 | 7 | 5 | 5 | 2.5 | 3.98 | 1 | 2 | 4 |
| 8 | 17 | 13 | 6 | 3 | 4.77 | 1 | 3 | 5 |
| 16 | 31 | 27 | 7 | 3.5 | 5.44 | 2 | 3 | 4 |
| 32 | 53 | 75 | 8 | 4 | 6.02 | 1 | 8 | 7 |
| 64 | 155 | 117 | 10 | 5 | 6.99 | 11 | 0 | 38 |

Table 7.1: Rate 1/2 Maximum Free Distance Codes




Encoder of a 4 state rate ½ code

4.3 Convolutional Codes: Decoding

The Viterbi algorithm is a special type of dynamic program, which casts the sequence detection problem for a convolutional code as a shortest path computation problem.

For a rate k/n code, pick a branch appearing in the th stage of the trellis and calculate its

$$\text{branch metric} \doteq \sum_{k=n(t-1)}^{nt-1} |z_k - x_k|^2$$

The path metric is the cumulative branch metric over a particular path and the path with the shortest metric is the selected path.

Notice also that at a given code rate k/n, the code can be made stronger by increasing the number of states 2^v , but the price paid is the increased decoding complexity of the Viterbi algorithm.

(Terminal bits): A practical point to note is that the encoder of a convolutional code is a causal (and LTI) system. This means that each information bit only affects future coded bits, and thus, only affects future symbols (xk). Therefore, in a finite length packet, the last few users bits are not protected sufficiently against errors. For example, the last user bit would only affect encoding in the last stage. Such a bit would not receive a coding gain of code, since that gain is achieved by making the error events span several stages. To reduce errors in these last few user bits, as a practical matter, several ‘0’ bits, which are called terminal bits are

appended to the end of the user bit sequence. For example, if the user bits are 11011, then the information bit sequence can be produced by appending, say, terminal ‘0’s, to get 11011000. This information bit sequence is now encoded by the transmitter, as usual. (Since the memory of the encoder is v bits, the terminal ‘0’ bits return the encoder to the all-zeros state. Thus, now, the source as well as the destination state is the all-zeros state.) At the receiver, the detector knows that the last bits are ‘0’ terminal bits, and thus, selects the ‘0’ labeled branches at the end of the packet, returning it to the all-zeros state. Essentially, the terminal bits serve to increase the length of the coded bit sequence, and thus allow full error protection to the last few user bits.

## 4.4 Modulation:

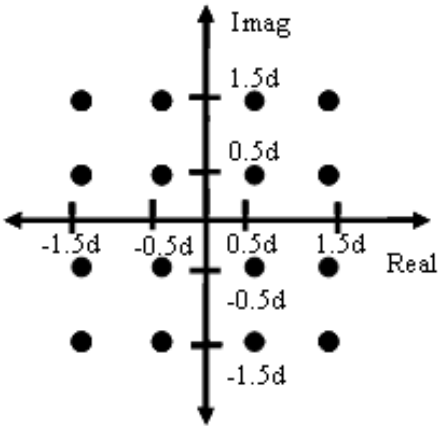
Modulation is the process of converting information bits into signals which can be transmitted via an analog medium. The message presented to the communication sequence is digital for our system, this bit sequence has to be grouped into bit symbols which are translated into analog voltage levels which are transmitted into the channel.
Modulation schemes are required to increase the bit rate and yet keep the bit error rate low.

### 4.4.1 M- QAM

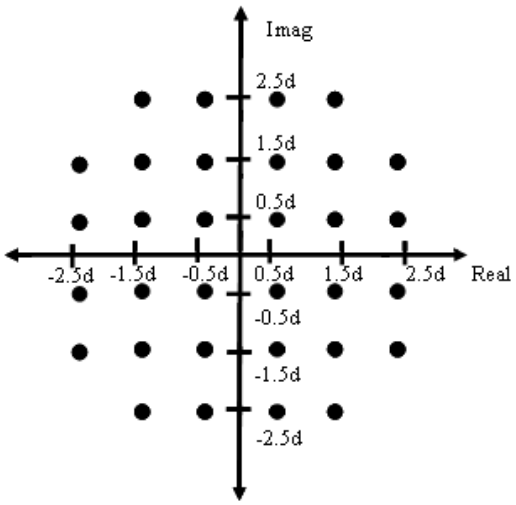
In M-QAM, we assume that the symbols (xk) are complex-valued. Thus, M-QAM is applicable only to pass band modulated systems. This constellation maps b (b>2)bits into one complex-valued symbol xk. Note that the number of points in the constellation is M = 2b. There are three different possibilities here:

• Square-QAM:

Here, b > 2 is an even integer. A square QAM constellation is produced by defining root M = 2b/2 equi-spaced levels on each axis, the real and the imaginary axis, of the signal space.
Therefore, the total number of points is 2b/2 × 2b/2 = 2b = M. This is shown in figure below, for a 16-QAM constellation, which has b = 4. The minimum distance of the constellation is defined to be d. The requirement that b should be an even integer arises from the requirement that pM = 2b/2 must be an integer.



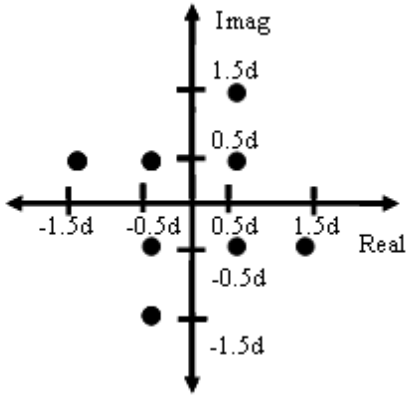

16 QAM




32 QAM

• Cross-QAM: Here, b>5 is an odd integer. Therefore, a Square-QAM cannot be defined because root M is not an integer. However, a 'Cross-QAM' constellation can be produced as follows. First, draw a M/2 = 2b−1 point Square-QAM constellation. This is possible, because (b − 1) is even. Then, add the remaining M/2 = 2b−1 points evenly on all four sides. Thus, these remaining 2b−1 points are placed in four groups of 2b−3 points each, along the four sides of the Square-QAM constellation. This is shown in the figure above for a 32-QAM constellation, which has b = 5. From the shape of the constellation, it is clear that the 'corners' of the constellation do not contain points. The intuition behind this idea is that, the further away the points are from the origin, the more is the power required for transmitting them. So, keeping them closer to the origin saves power.

- 8-QAM: The special case of 8-QAM, which has b = 3 (odd integer), but does not precisely follow the construction of Cross-QAM



4.4.2 M- PSK:

In M-PSK, we assume that the symbols (xk) are complex-valued. Thus, M-PSK is applicable only to passband modulated systems. This constellation maps b bits (b>1) into one symbol. The number of points in the constellation is M = 2b. The M points are arranged uniformly on a circle of radius r.

The M-PSK constellation is not as dense as the M-QAM constellation, because it does not utilize the interior of the disk. Therefore, it is not very power efficient. However, |xk| is constant (equal to radius r) for the M-PSK constellation, unlike the M-QAM constellation. Hence, wireless systems that use the PSK constellation can tolerate non-linearities in the various electronic amplifiers (primarily in the high power transmitter amplifiers). Such Further, in PSK, only the phase carries information, and so, even if the channel attenuation is unknown (i.e., unknown |h0|), decoding of PSK is unaffected. Thus, PSK is a robust constellation.

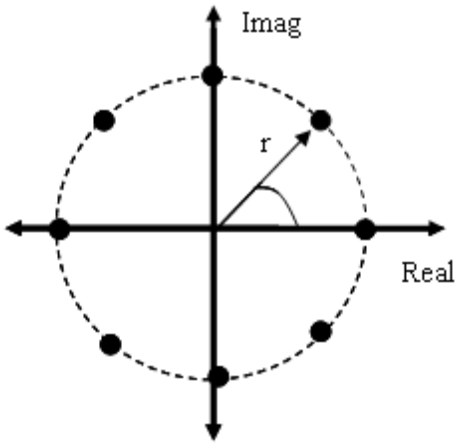
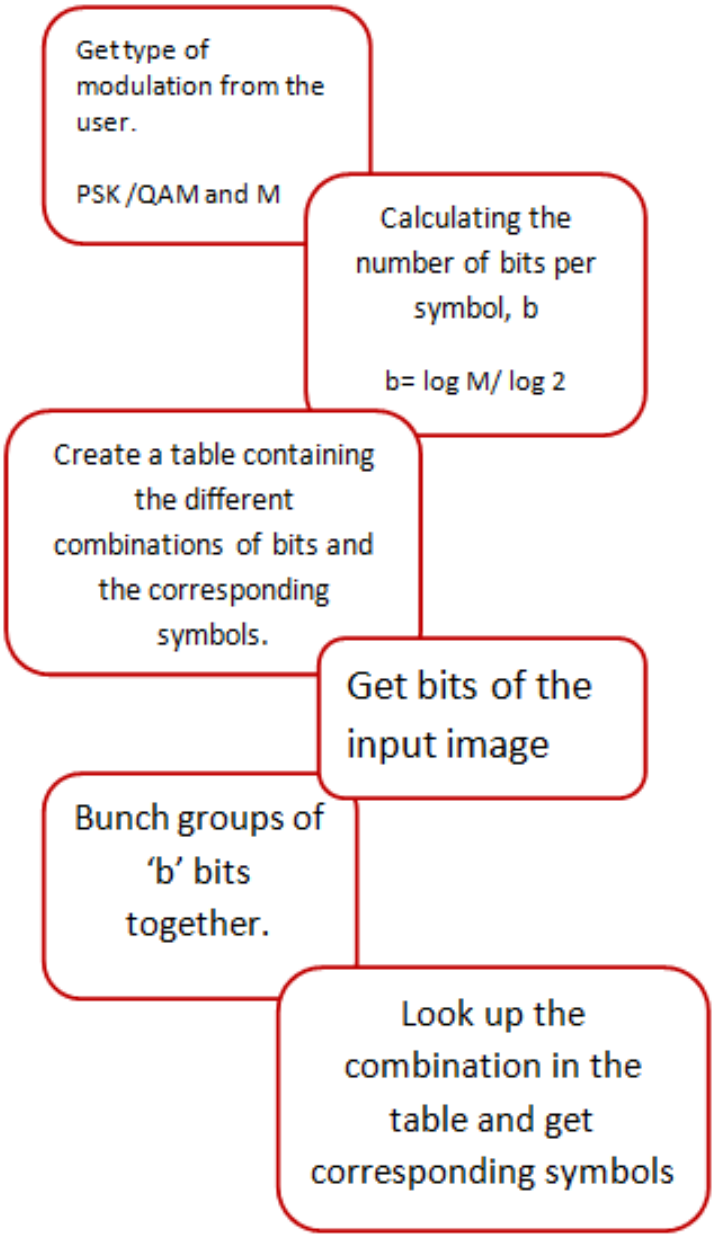



Fig: 8 PSK Constellations

4.4.3 Algorithm for Modulation:



4.4.4 Example of Modulation:

1) Input stream of bits: 10001011011100
2) For 4 QAM modulation.
3) b= log(4)/log(2)= 2
4) Group bunches of 2 bits, so we get 10 00 10 11 01 11 00.
5) Create conversion table of bit combinations to symbols.

| Bit Combination | Symbol |
|---|---|
| 00 | -1 -j |
| 01 | -1 + j |
| 10 | 1 - j |
| 11 | 1 + j |

6) Looking up corresponding symbols from the table:
Symbols Transmitted: 1-j -1-j 1-j 1+j -1+j 1+j -1-j

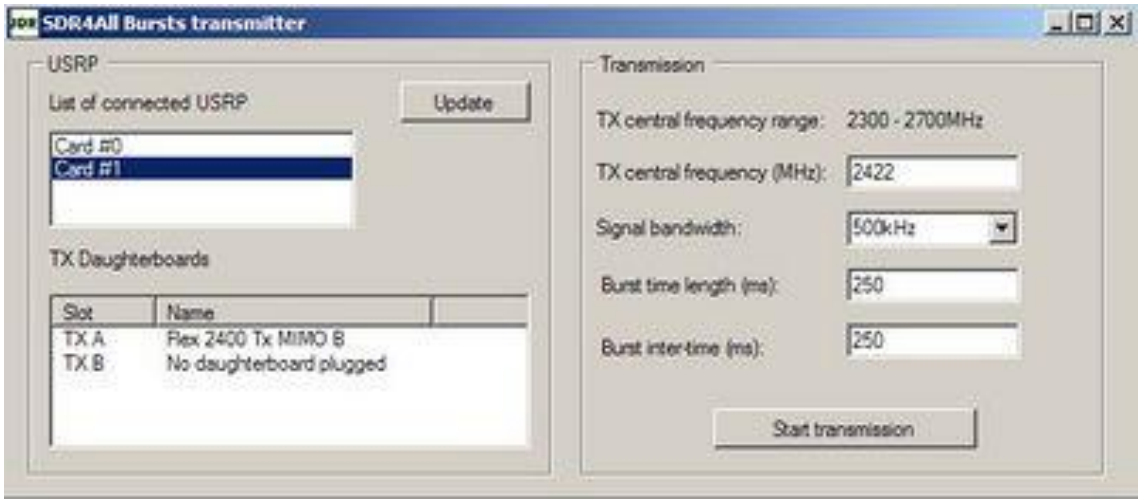
4.5 Demodulation:
At the receiver we use minimum distance demodulation; the received symbol with noise is compared to all the possible symbols. The symbol from which it has least distance, would be the symbol that was most likely have been transmitted at the transmitter, this symbol then becomes the guess of the symbol transmitted.

# 5. Implementation:

In the first part of this section we describe how we transferred a short burst of signal from the transmitting SDR to the receiving SDR.
In section 5.2 we describe what we actually implemented for the demo. Since we were unable to transfer an image using the SDRs we simulated the wireless transfer of the image and then carried out a face recognition algorithm at the receiver.

## 5.1 Transferring Short Bursts of Signal:

We setup the SDRs transmitter with the following specifications:




```
sock=SDR4All_Connect(0,'SlotA','TX'); % 0 stands for USRP #0
```

Set the communication parameters:

```
SDR4All_SetGain(sock,20); % Maximal TX gain
SDR4All_SetFreq(sock,2422e6);
SDR4All_SetInterpRate(sock,256); % set signal sampling period (and bandwidth)
```

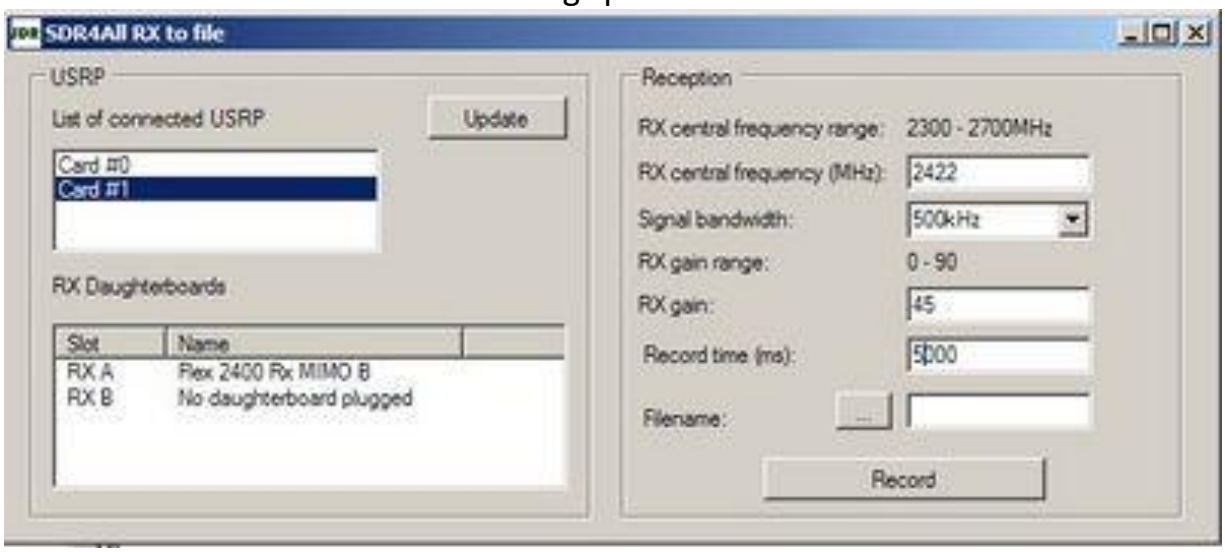
Generate the signal to be transmitted:

```
Basic  = kron(ones(5e5/10/2/4,1),[ones(10,1);-ones(10,1)]);
Vide = zeros(125000,1);
Base = [Basic;Vide];
Sig = kron(ones(20,1),Base);
Te = (1:length(Sig))/(500e3);
plot(Te,Sig);
```

And perform the transmission:

```
SDR4All_SendData(sock,Sig);
```

We set the receiver with the following specifications:




```
sock=SDR4All_Connect(0,'SlotA','RX'); % 0 stands for USRP #0
```

The communication with the USRP should be confirmed with a message like:

```
Number of connected USRP: 2
Communication set with USRP 1 and daugtherboard on slot A
```

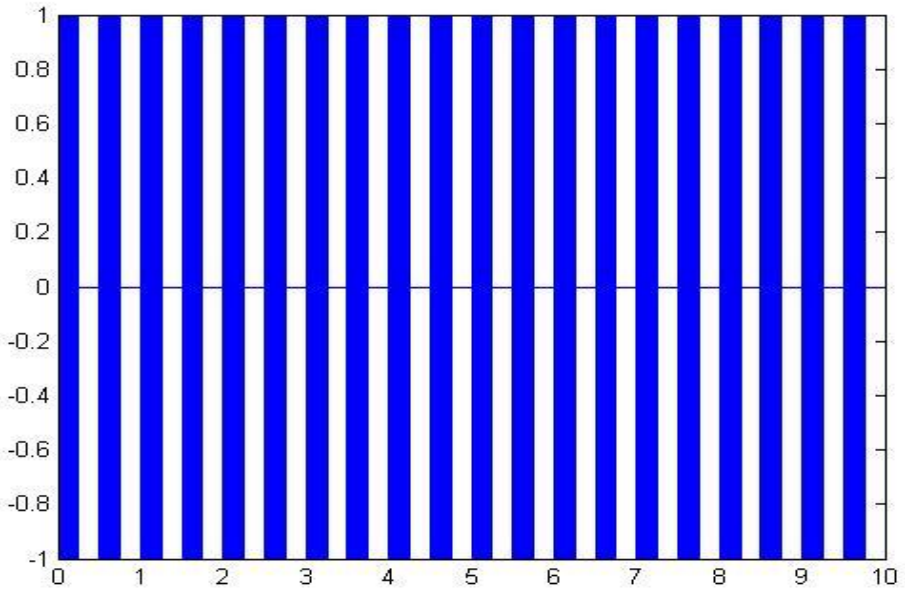
The next commands are:

```
[gain_min,gain_max,gain_step] = SDR4All_GetGain(sock);
[freq_min,freq_max] = SDR4All_GetFreq(sock);
SDR4All_SetGain(sock,(gain_max+gain_min)/2);
SDR4All_SetDecimRate(sock,128); % set signal sampling period (and bandwidth)
SDR4All_SetFreq(sock,2422e6);
```

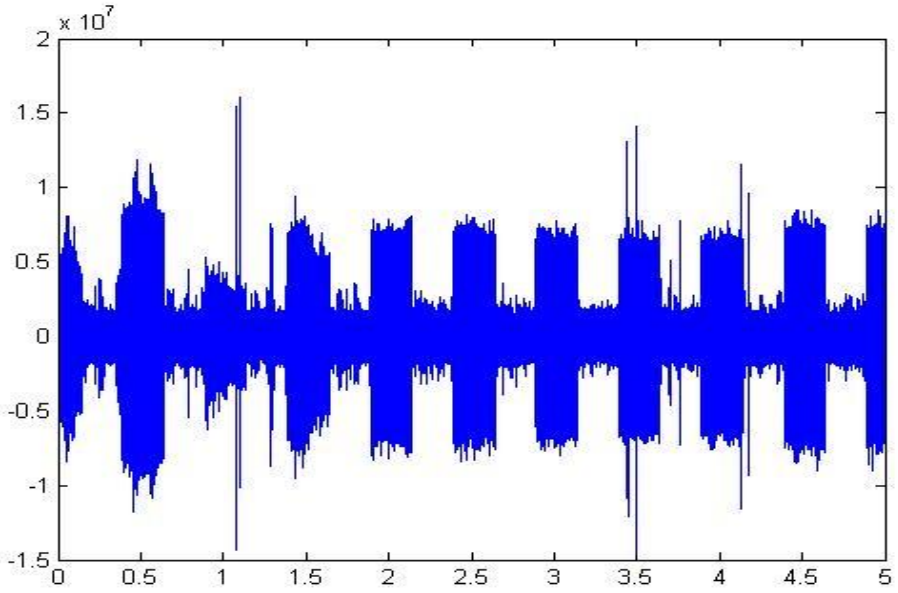
You can now start the transmission with the TX bursts soft. Use the following command with matlab to record 5 seconds of signal (at 500kHz):

```
[Data] = SDR4All_GetData(sock,5*1000*500);
Te = (1:length(Data))/(500e3);
```

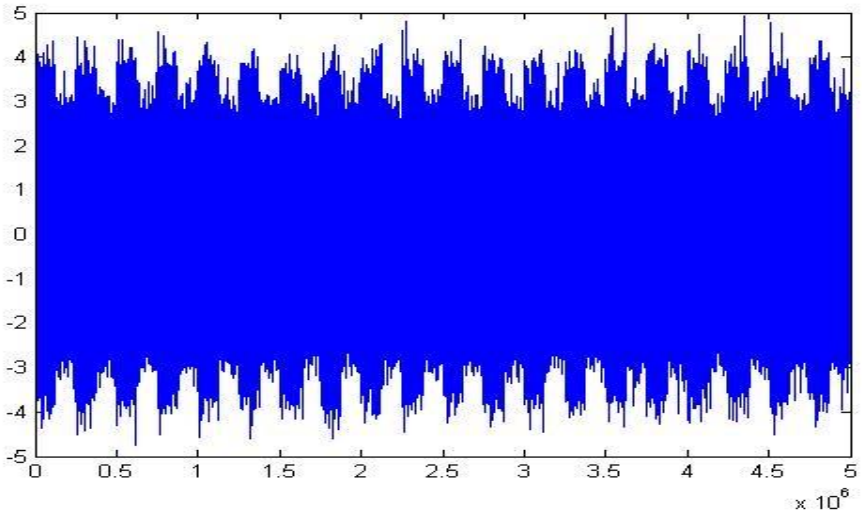
With the above setup, the signal which we transmitted was:



The signal we received at the receiving SDR was:




The signal received at the receiver in our simulation




## 5.2 Simulating Transfer of Image in MATLAB.

To reduce the time taken for processing in MATLAB we use wavelet compression on the image to be transmitted. This data is converted into a stream of bits, modulated using any one of the modulating schemes, encoding schemes and then sent over the simulated channel.

Transmitted Image

Received Signal with noise

This received signal was then feed into the face recognition algorithm to search in the OpenCV database for a match. Even with significant amount of noise in the image, the algorithm was successful in finding a match in the database.

# 6. Results:

We tried running different combinations of modulation and encoding schemes for transferring the different images for different SNR values. Following are the results for transferring a particular image at an SNR of 10dB.

| | | M-QAM | | M-PSK | |
|---|---|---|---|---|---|
| | | Time(s) | BER | Time(s) | BER |
| M=4 | 1/2 | 1120.96 | 14.5% | 941 | 15.8% |
| | 2/3 | 697.507 | 3.4% | 578 | 3.9% |
| M=16 | 1/2 | 545.138 | 22 % | 419 | 18.66% |
| | 2/3 | 368.906 | 4% | 304 | 4% |

Time here refers to the time taken for MATLAB to finish processing. And the bit error rate is calculated as the number of bits in error to the total number of bits transmitted.

# 7. Inferences:

- For the same SNR, M= 4 produces lesser errors than M=16. This is expected as distance between symbols is more for M=4, hence the chances of an error occurring is lowered.

- M=4 modulation schemes takes longer to run and process this can be justified as M =4 produces a longer chain of symbols to be transmitted. Also the number of bits per symbol is lesser than M=16, hence it takes longer to transfer the same amount of data using M=4.

- Rate ½ code should have produced less errors than the 2/3 code, as rate ½ introduces more redundancy hence reduces chances of making an error but our results are show otherwise.

- M-QAM shows better results than M-PSK for this channel. This channel is a pure AWGN channel. If multipath and fading effects were considered then M-PSK would have done better, as wireless channel non-linearities would badly distort the QAM constellation.

# 8. Schedule:

| Plan | Tentative Deadline |
|---|---|
| Acquiring hardware | 2nd February |
| Identifying and installing software bundles | 5th February |
| Achieving successful interface between USRP1 and Laptop | 13th February |
| Face detection | 10th March |
| Implementing the various schemes | 31st March |
| Addressing frequency offset and channel estimation issues | 9th April |
| Transfer of image between client and server MTALAB | 3rd April |
| Comparing these by evaluating performance parameters | 9th April |
| Optimal scheme for project | 11th April |

Work Distribution:

Anish Menon: Codes for Encoding Schemes, Decoding, Face Recognition

Akash Gopisetty: Digital & Analog Modulation/ Demodulation schemes

# 9. References:


- The Ettus Research Group Website : www.ettus.com
  Understanding of the hardware and the available software bundles

- MATLAB interfacing for USRP1 by Tools4SDR: www.tools4sdr.com
  The software interface we used with our hardware and MATLAB, setting the transmitter and receiver parameters.

- Notes from Prof.Negi's Class 18-450 : Digital Wireless Communication concepts

- Eigen Vector Face Recognition code by Santiago Serrano modified by Karoly Pados. The code was used to verify the robustness of wireless communication.

- http://groups.google.com/group/simulink-usrp:
  A software interface between simulink and usrp, this group helped us with our problems of installing the same though we were not able to get it working.