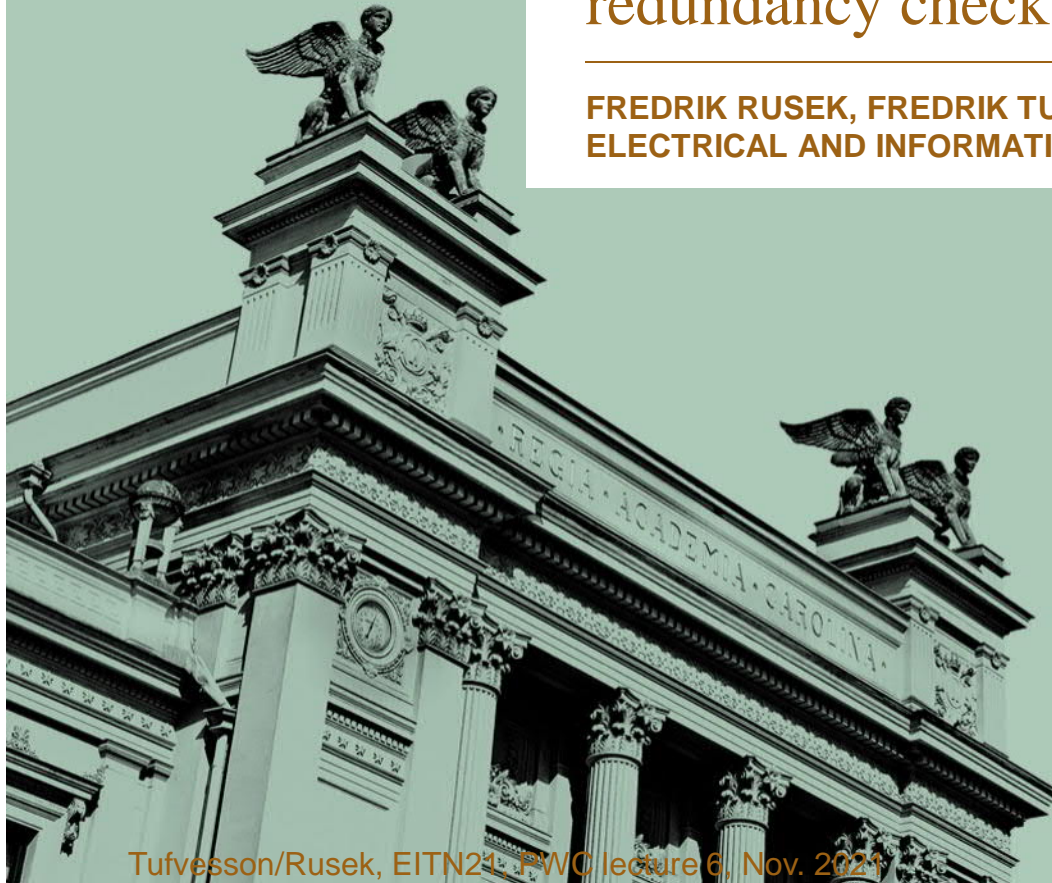




LUND
UNIVERSITY

Project in Wireless Communication Lecture 6: Project overview and cyclic redundancy check (CRC) codes

FREDRIK RUSEK, FREDRIK TUFVESSON
ELECTRICAL AND INFORMATION TECHNOLOGY



Project overview, part two: the audio channel

Part two is divided into two tasks

Task 1 – the basic link: Implement an OFDM system and send a data sequence from one computer to the other via the audio channel and decode.

Task 2 – the advanced link: Implement the packet based full duplex system on the audio channel with ARQ.

- **Deadline is Friday Dec 3, 2021 for part two (Tasks 1 and 2)**



Project overview, the audio channel

System should include:

- OFDM, minimum 64 carriers
- Four sub-channels for continuous pilot symbols, and a preamble
- Packet based system
- ARQ, i.e., receiver should send ACK/NACK for each packet.
- Re-transmissions of the incorrectly received packets
- Cyclic redundancy check (CRC) code
- Minimum bit-rate during transmissions: 0.5 kbit/s
- Convolutional code is optional (you need to find out yourself if you need it or not)
- Minimum size of file: 20 kbits
- Max packet length: 1 kbits



Project overview, part three: the radio channel

Implement a basic OFDM based file transfer system over the radio channel using the ADALM Pluto SDR

Part three is also divided into two tasks

Task 1 – the basic link: Implement an OFDM transceiver and send a file between the Tx and Rx part of the same Pluto SDR.

Task 2 – the advanced link: Transfer the file from one Pluto to another Pluto over the radio channel.

- **Deadline is Friday Jan 7, 2022 for part three (Tasks 1 and 2)**



A problem to be encountered in Matlab

One particular problem is that in Matlab, one have to record sound for a pre-defined amount of time.

Since this is a "Matlab-problem" and not a "communication-theory-problem", you are **allowed to make use of the built in clock-function** in matlab. The internal clocks of the receiver and the transmitter are allowed to be synchronized with each other.

If you choose to use C/C++/Python, this problem is completely alleviated since one can then record sound "until something happens" – for example "until there is no sound to record". You can also start recording when "there is something to record

However, the overhead of using C/C++/Python is rather large if you are not experienced.



Some tips and tricks

- Plot the signals you are transmitting, do they look as you intended?
- Plot the received signal, can you see it visually?
- Make a scatter plot of the received constellation, does it make sense?
- Think about a suitable carrier frequency and bandwidth before starting to code
- Verify your code after each step
- Use external, cable connected, microphone and loud speaker, verify your hardware settings in the computer



Cyclic redundancy check, CRC

A CRC is used for **error detection**, not for error correction.

Example: Single parity check bit

Suppose one wants to transmit the 5 bits

[0 0 1 0 1]

If one receives the bits

[0 0 0 0 1]

This error will pass by un-detected.



CRC Example: Single parity check bit

Suppose one wants to transmit the 5 bits

[0 0 1 0 1]

If one receives the bits

[0 0 **0** 0 1]

This error will pass by un-detected.

We can fix this by adding a single parity bit so that the total number of 1s is always even.

We then have

[0 0 1 0 1 **0**] ← *Parity bit*

Total number of 1s = even

CRC Example: Single parity check bit

Suppose one wants to transmit the 5 bits

[0 0 1 0 1]

If one receives the bits

[0 0 0 0 1]

This error will pass by un-detected.

We can fix this by adding a single parity bit so that the total number of 1s is always even.

We then have

[0 0 1 0 1 **0**]

Parity bit

*Total number of 1s = odd
-> not correct*

If we receive

[0 0 0 0 1 **0**]

we know that there has been 1 bit error on the channel, and we will ask for a re-transmission



CRC

The previous example was just meant as illustration, and in reality, much more advanced systems are used. **But, they are based upon the same principle!**

Suppose that we should send K bits, $[u_0 \dots u_{K-1}]$. We denote these by the D -transform

$$u(D) \stackrel{\text{def}}{=} u_{K-1}D^{K-1} + u_{K-2}D^{K-2} + \dots + u_0$$

Hence, $1 + D + D^6 = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1]$ etc etc

The powers of the indeterminate D can be thought of as keeping track of which bit is which. The CRC is represented by another polynomial,

$$c(D) \stackrel{\text{def}}{=} c_{L-1}D^{L-1} + c_{L-2}D^{L-2} + \dots + c_0$$

The entire frame of data and CRC is then $x(D) = u(D)D^L + c(D)$, that is

$$x(D) \stackrel{\text{def}}{=} u_{K-1}D^{L+K-1} + u_0D^L + c_{L-1}D^{L-1} \dots + c_0$$



How to find $c(D)$?

The check bits $c(D)$ depend of course on the information bits $u(D)$.

Question: How to find the check bits $c(D)$ given a particular set of information bits $u(D)$ in a structured fashion?

How to find $c(D)$?

Define a generator polynomial $g(D)$ of degree L

$$g(D) \stackrel{\text{def}}{=} D^L + g_{L-1}D^{L-1} + \dots + g_1D + 1$$

For a given generator polynomial, $g(D)$, the mapping from the information bits, $u(D)$, to the CRC, $c(D)$, is given by

$$c(D) = \text{Remainder} \left[\frac{u(D)D^L}{g(D)} \right]$$



Long Division

$$c(D) = \text{Remainder} \left[\frac{u(D)D^L}{g(D)} \right]$$

- This is just an ordinary long division of one polynomial with another.
- All operations are modulo 2. Thus $(1+1) \bmod 2 = 0$, and $(0-1) \bmod 2 = 1$.
- Subtraction using modulo 2 arithmetic is the same as addition

Example:

$$\underline{D^3 + D^2 + 1} \overline{) D^5 + \quad D^3}$$

Find remainder of $D^5 + D^3$ divided with $D^3 + D^2 + D$

Long Division

$$c(D) = \text{Remainder} \left[\frac{u(D)D^L}{g(D)} \right]$$

- This is just an ordinary long division of one polynomial with another.
- All operations are modulo 2. Thus $(1+1) \bmod 2 = 0$, and $(0-1) \bmod 2 = 1$.
- Subtraction using modulo 2 arithmetic is the same as addition

Example:

$$\begin{array}{r} D^2 \\ \hline D^3 + D^2 + 1 \mid D^5 + \quad D^3 \end{array}$$

Long Division

$$c(D) = \text{Remainder} \left[\frac{u(D)D^L}{g(D)} \right]$$

- This is just an ordinary long division of one polynomial with another.
- All operations are modulo 2. Thus $(1+1) \bmod 2 = 0$, and $(0-1) \bmod 2 = 1$.
- Subtraction using modulo 2 arithmetic is the same as addition

Example:

$$\begin{array}{r} D^2 \\ D^3 + D^2 + 1 \overline{) D^5 + + D^3} \\ \underline{+ D^5 + D^4 + + D^2} \\ D^4 + D^3 + D^2 \end{array}$$

Long Division

$$c(D) = \text{Remainder} \left[\frac{u(D)D^L}{g(D)} \right]$$

- This is just an ordinary long division of one polynomial with another.
- All operations are modulo 2. Thus $(1+1) \bmod 2 = 0$, and $(0-1) \bmod 2 = 1$.
- Subtraction using modulo 2 arithmetic is the same as addition

Example:

$$\begin{array}{r} D^2 + D \\ D^3 + D^2 + 1 \overline{) D^5 + D^3} \\ \underline{+ D^5 + D^4 + D^2} \\ D^4 + D^3 + D^2 \end{array}$$

Long Division

$$c(D) = \text{Remainder} \left[\frac{u(D)D^L}{g(D)} \right]$$

- This is just an ordinary long division of one polynomial with another.
- All operations are modulo 2. Thus $(1+1) \bmod 2 = 0$, and $(0-1) \bmod 2 = 1$.
- Subtraction using modulo 2 arithmetic is the same as addition

Example:

$$\begin{array}{r}
 D^2 + D \\
 \hline
 D^3 + D^2 + 1 \overline{) D^5 + + D^3} \\
 + D^5 + D^4 + + D^2 \\
 \hline
 D^4 + D^3 + D^2 \\
 + D^4 + D^3 + + D \\
 \hline
 D^2 + D = \text{Remainder}
 \end{array}$$



Long Division

$$c(D) = \text{Remainder} \left[\frac{u(D)D^L}{g(D)} \right]$$

- This is just an ordinary long division of one polynomial with another.
- All operations are modulo 2. Thus $(1+1) \bmod 2 = 0$, and $(0-1) \bmod 2 = 1$.
- Subtraction using modulo 2 arithmetic is the same as addition

Example:

$$\begin{array}{r}
 \text{quotient} \\
 \text{\textcircled{D}^2 + D} \\
 \hline
 D^3 + D^2 + 1 \overline{) D^5 + + + + + 1} \\
 \underline{+ D^5 + D^4 + + + + } \\
 D^4 + D^3 + D^2 \\
 \underline{+ D^4 + D^3 + + + } \\
 \text{\textcircled{D}^2 + D} = \text{Remainder} \\
 \hline
 D^5 + D^3 = (D^2 + D)(D^3 + D^2 + 1) + (D^2 + D)
 \end{array}$$



CRC

Let $z(D)$ denote the quotient. We then have:

$$u(D)D^L = g(D)z(D) + c(D)$$

Subtract $c(D)$ from both sides and use "+" = "-" in modulo 2 arithmetic

$$x(D) = u(D)D^L + c(D) = g(D)z(D)$$

Thus, all valid code words $x(D)$ are divisible by the generator polynomial $g(D)$

Receiver operation

Assume $x(D)$ is transmitted and that $y(D)$ is received. Let the errors on the channel be $e(D)$.

Hence, $y(D)=x(D)+e(D)$.

The receiver knows that a valid $y(D)$ should leave no remainder if divided by $g(D)$.

So, the receiver declares:

ACK if Remainder $\frac{y(D)}{g(D)} = 0$

NACK if Remainder $\frac{y(D)}{g(D)} \neq 0$



When does it fail?

Since we have shown that $x(D)$ is divisible by $g(D)$,

$$\text{Remainder } \frac{y(D)}{g(D)} = \text{Remainder } \left[\frac{x(D)+e(D)}{g(D)} \right] = \text{Remainder } \frac{e(D)}{g(D)}$$

If no errors occur, i.e., $e(D)=0$, then this remainder is zero, and the receiver declares a successful transmission.

If $e(D)$ is not zero, the receiver fails to detect the error only if $\text{Rem}[e(D)/g(D)]=0$.

This is the same as saying that $e(D)$ is a valid code word, i.e.,

$$e(D) = g(D)z(D)$$

For some non-zero polynomial $z(D)$

When does it fail?

Suppose that a single error occurs, i.e., $e(D) = D^i$, for some integer i .

We have an un-detected error if and only if

$$e(D) = g(D)z(D) \quad \text{for some } z(D)$$

But since $g(D)$ have at least two non-zero terms (1 and D^L), so must $g(D)z(D)$ have.



When does it fail?

Suppose that a single error occurs, i.e., $e(D) = D^i$, for some integer i .

We have an un-detected error if and only if

$$e(D) = g(D)z(D) \quad \text{for some } z(D)$$

But since $g(D)$ have at least two non-zero terms (1 and D^L), so must $g(D)z(D)$ have.

Hence, $g(D)z(D)$ cannot possibly equal D^i and we can conclude

All single event errors are detectable

Burst errors

We next consider bursts of errors $e = [\dots 000 \underbrace{110 10110}_{P} \dots 01101 000 \dots]$

We know that it will pass un-detected if and only if $e(D) = g(D) z(D)$ for some $z(D)$

But,

$$\underbrace{(D^L + \dots + 1)}_{g(D)} \underbrace{(D^j + \dots + D^i)}_{z(D)} = D^{L+j} + \dots + D^i$$

Hence, $g(D)z(D)$ will consist of a burst of at least length L .

Burst errors

We next consider bursts of errors $e = [\dots 000 \underbrace{110 10110}_{P} \dots 01101 000 \dots]$

We know that it will pass un-detected if and only if $e(D) = g(D) z(D)$ for some $z(D)$

But,

$$\underbrace{(D^L + \dots + 1)}_{g(D)} \underbrace{(D^j + \dots + D^i)}_{z(D)} = D^{L+j} + \dots + D^i$$

Hence, $g(D)z(D)$ will consist of a burst of at least length L .

If $P < L$, $e(D) = g(D)z(D)$ is not possible!

All error bursts of length L and less are detectable

What about double errors?

What about $e(D)$ of the type $D^j + D^i$?

We already know that if $j-i < L+1$, then it is detectable.

For $j-i > L$, more advanced theory must be used (theory of finite fields – Galois theory)

When the smoke clears, the result is

if $g(D)$ is primitive, all double errors are detectable (if $K < 2^L - 1$)

Some known results

With a primitive $g(D)$ times $(1+D)$, that is $g(D) = (1+D)g_p(D)$

- All single and double errors are detectable
- Burst-detecting capability of at least K
- Probability of detecting a completely random $e(D)$: 2^{-L}

Standard $g(D)$ s with $L=16$.

- $D^{16} + D^{15} + D^2 + 1$ CRC-16
- $D^{16} + D^{12} + D^5 + 1$ CRC-CCITT

Luckily, there is Matlab. Play around with the CRC-class (just type **help crc** in Matlab)





LUND
UNIVERSITY