Parity

The simplest form of error detection is parity.

A single bit is appended to a bit pattern.

In even parity, the number of '1' bits must add up to an even number.

In odd parity, the number of '1' bits must add up to an odd number.

Even parity is more common, but both are used.

e.g. ASCII letter H = 72_{10} = 100 1000_2

Using even parity we need to add a 0, so the transmission is 1001 0000

If a bit is flipped accidentally (e.g. 1011 0000) then the parity check detects an error.

The two downfalls of parity checks are:

Parity can only detect an odd number of errors
e.g. 1111 0000 would be falsely identified as correct

Parity can't be used to detect WHICH bit is in error.

The main advantage is the low overhead.

2D Parity

One possible solution is 2D parity. A series of messages each uses a single parity bit, with a whole parity byte at the end of the transmission.

e.g. H e l l o	Error:	
1001 000 0	1 0 0 1 0 0 0 0	1 0 0 1 0 0 0 0
1100 101 0	1 1 0 1 1 0 1 **0 – error**	1 1 0 **1** 1 0 1 **0**
1101 100 0	1 1 0 1 1 0 0 0	1 1 0 1 1 0 0 0
1101 100 0	1 1 0 1 1 0 0 0	1 1 0 1 1 0 0 0
1101 111 0	1 1 0 1 1 1 1 0	1 1 0 1 1 1 1 0
1000 010 0	1 0 0 **0** 0 1 0 0 **error**	1 0 0 **0** 0 1 0 0

2D parity increases the overhead in data transmission, but provides correction of ONE error and detection of MULTIPLE errors

Hamming Codes

To start with Hamming Codes, write down all the binary digits from 1 to 7:

1 0 0 1
2 0 1 0
3 0 1 1
4 1 0 0
5 1 0 1
6 1 1 0
7 1 1 1

All the powers of 2 (bit patterns that have JUST ONE 1 bit) are designated as parity bits. The rest are data bits.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	**P2**	d	**P4**	d	d	d	**P8**	d	d	d	d	d	d	d	**P16**	d	d	d	d

The parity bit at position 1 (0001) covers all of the bit positions that end in a 1.
The parity bit at position 2 (0010) covers all of the bit positions that end 1*.
The parity bit at position 4 (0100) covers all of the bit positions that end 1**.
The parity bit at position 8 (1000) covers all of the bit positions that end 1***.

This means that (looking at the binary digits above),

P1 covers bits 1, 3, 5, 7, 9, 11, etc…
P2 covers bits 2 & 3, 6 & 7, 10 & 11, etc…
P4 covers bits 4 – 7, 12 – 15, etc…

This means that an error in bit 5 would be highlighted by errors in P1 & P4.
An error in bit 7 would be highlighted by errors in P1, P2 & P4.
An error in bit 2 would only be highlighted by an error in P2

Note that 1 + 4 = 5 1 + 2 + 4 = 7 2 = 2

Adding the bit positions of the incorrect parity bits highlights the exact position of the error.

Also note that this only works provided there is only one error in the transmitted bit pattern.

Hamming Codes (how to)

If we wanted to transmit the pattern 1011 we would do the following:

Position	1	2	3	4	5	6	7
Bit	P1	P2	1	P4	0	1	1

P1 covers bits 1, 3, 5 & 7 (P, 1, 0, 1). Using even parity, P = 0

Position	1	2	3	4	5	6	7
Bit	0	P2	1	P4	0	1	1

P2 covers bits 2, 3, 6 & 7 (P, 1, 1, 1). Using even parity, P = 1

Position	1	2	3	4	5	6	7
Bit	0	1	1	P4	0	1	1

P4 covers bits 4, 5, 6 & 7 (P, 0, 1, 1). Using even parity, P = 0

Position	1	2	3	4	5	6	7
Bit	0	1	1	0	0	1	1

So the transmitted data is 011 0011

If there was an error and the message received was 011 0111 then check each parity bit:

P1 covers 1, 3, 5 & 7 (0 1 1 1) - fail
P2 covers 2, 3, 6 & 7 (1 1 1 1) – pass
P4 covers 4, 5, 6 & 7 (0 1 1 1) - fail

P1 and P4 failed, 1 + 4 = 5, therefore bit 5 is incorrect.

Therefore the correct data transmission should have been 011 0011 and the original data is 1011

Gray Codes

Imagine looking at a clock where the digits flip as the minutes change.

Now imagine the time is changing from 09:59 to 10:00. All 4 digits need to change, and because it is a mechanical operation it is impossible to guarantee absolute synchronicity.

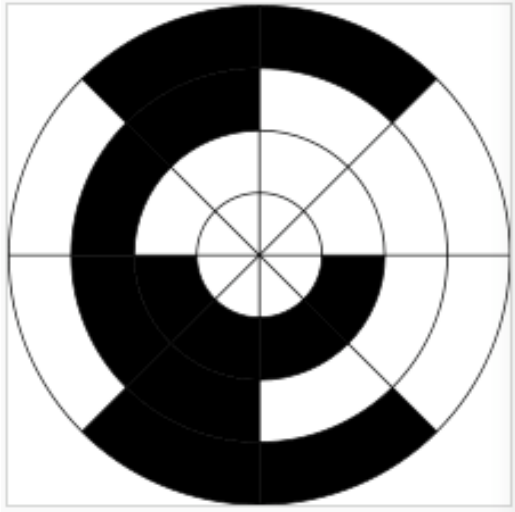
It is possible that we will look at the clock just as it says 19:00 – meaning that we would read the time wrong. Oh no!

The solution is to arrange the digits so that only 1 needs to change at any time.

If we have a 2-bit word we could use 00, 01, 11, 10 to represent the numbers 1 to 4.

This is NOT the same as the standard binary representation!!

Digital communications systems often use gray codes to track the modulation. Satellite dishes also use a form of gray coding to locate its position as it rotates.



Rotary encoder for angle-measuring devices marked in 3-bit binary-reflected Gray code (BRGC)