E40M

Binary Numbers, Codes

Reading

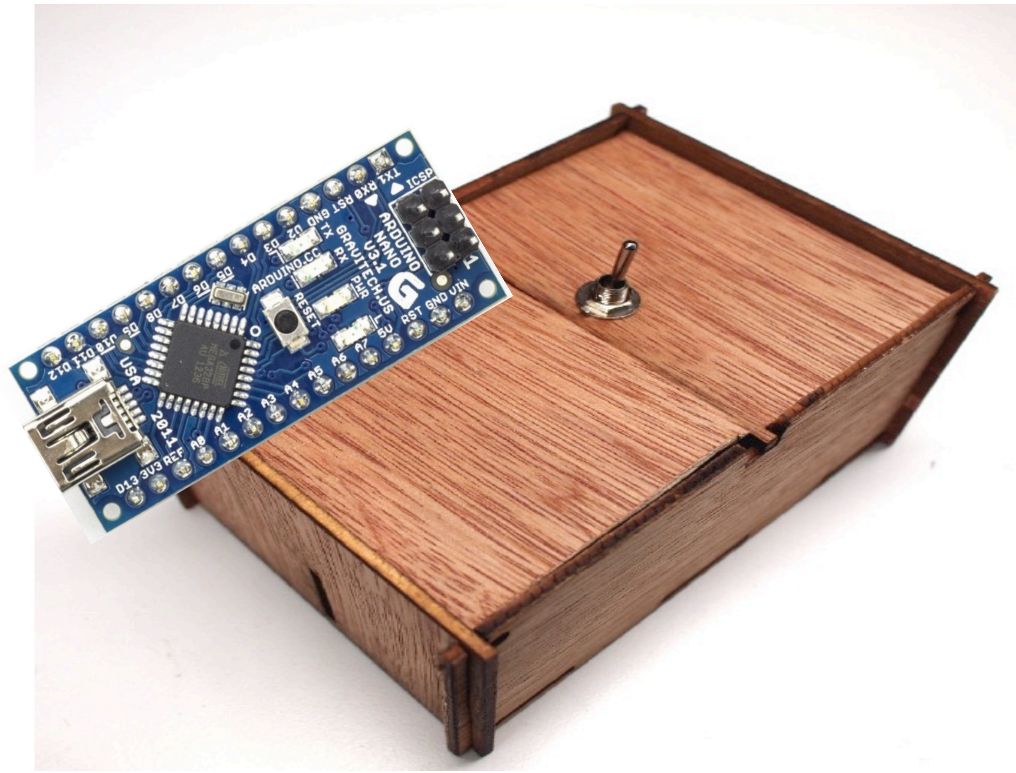
- Chapter 5 in the reader
- A&L 5.6

Useless Box Lab Project #2

Adding a computer to the Useless Box alows us to write a program to control the motor (and hence the box).

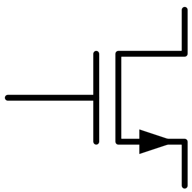
- To control the motor we need to study and use MOS transistors since our computer cannot drive the motor directly.

- The computer (Arduino) uses Boolean logic and CMOS logic gates to implement software we write for it. The MOS transistors amplify the Arduino output to control the motor.

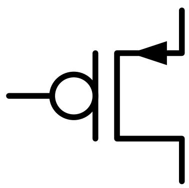


Last Lecture: MOSFETs

- nMOS
  - It is a switch which connects source to drain
  - If the gate-to-source voltage is greater than V_{th} (around 1 V)
    - Positive gate-to-source voltages turn the device on.



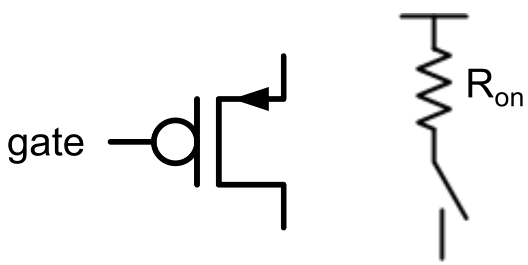
- pMOS
  - It is a switch which connects source to drain
  - If the gate-to-source voltage is less than V_{th} (around -1 V)
    - Negative gate-to-source voltages turn the device on



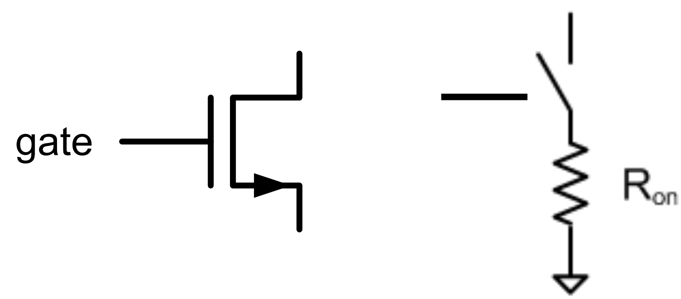
… and there's zero current into the gate!

Last Lecture: MOSFET models

- Are very interesting devices
  - Come in two “flavors” – pMOS and nMOS
  - Symbols and equivalent circuits shown below
- Gate terminal takes no current (at least no DC current)
  - The gate voltage* controls whether the “switch” is ON or OFF



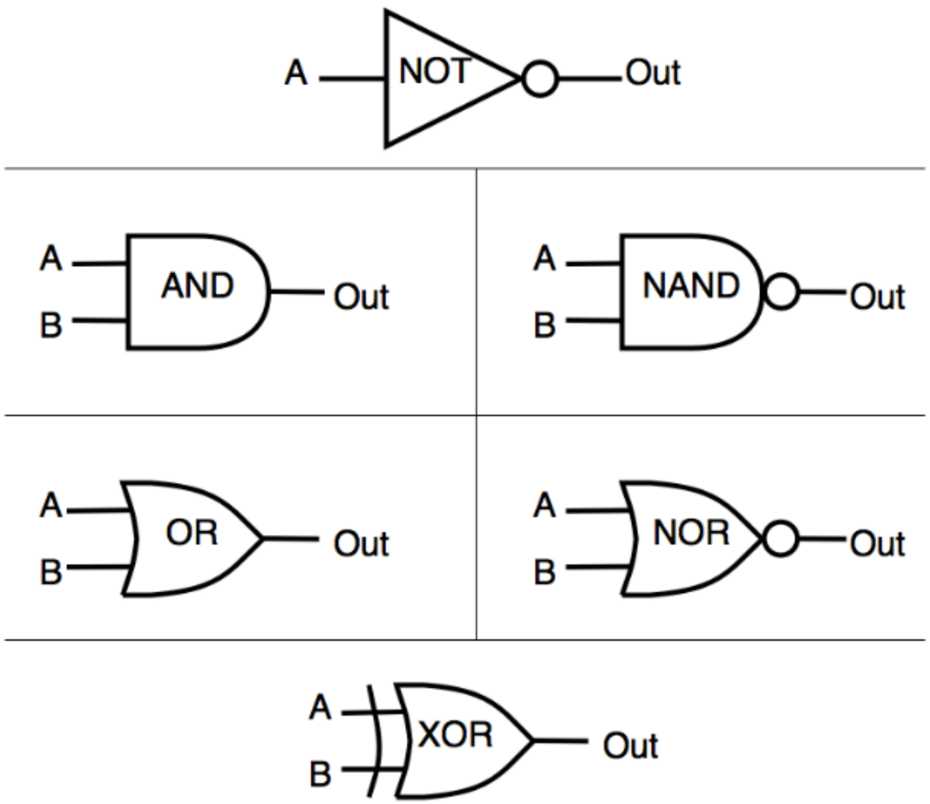
pMOS

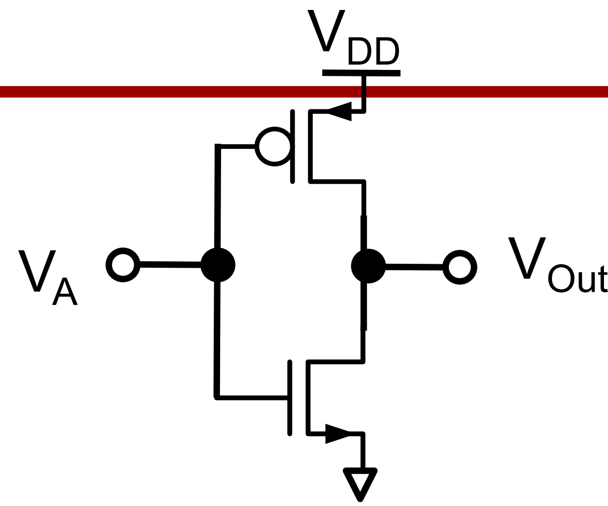


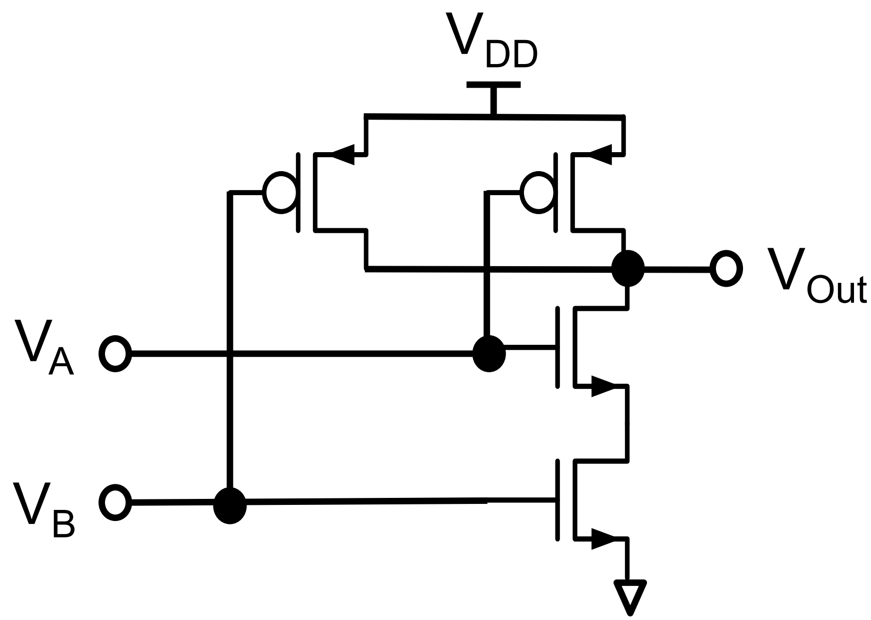
nMOS

* actually, the gate – to – source voltage, V_{GS}

Last Lecture: Logic Gates





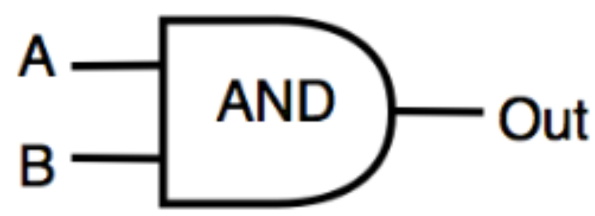


Logic Gates Deal With Binary Signals

- Wires can have only two values
  - Binary values,   0, 1; or True and False
- Generally transmitted as:
  - Vdd = 1 V; Gnd = 0
    - Vdd is the power supply, 1V to 5 V
    - Gnd is the “reference” level, about 0V
- So how do we represent:
  - Numbers, letters, colors, etc. ?

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

(A && B)   <->   AND

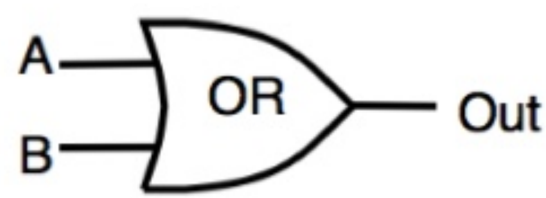


Logic Gates Deal With Binary Signals

- Wires can have only two values
  - Binary values, 0, 1; or True and False
- Generally transmitted as:
  - Vdd = 1; Gnd = 0
    - Vdd is the power supply, 1V to 5 V
    - Gnd is the “reference” level, about 0V
- So how do we represent:
  - Numbers, letters, colors, etc. ?

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

(A || B) <-> OR



Binary Numbers

- To represent anything other than true and false
  - You are going to need more then one bit
- Group bits together to form more complex objects
  - 8 bits are a byte, and can represent a character (ASCII)
- 24 bits are grouped together to represent color
  - 8 bits for R, G, B.
- 16 or 32 or 64 bits are grouped together and can represent an integer

Binary Numbers

- For decimal numbers
  - Each place is 10^n
    - 1, 10, 100, 1000 …
    - Since there are 10 possible values of digits

10^2	10^1	10^0
1	4	5

Decimal = 145

- For binary numbers
  - Each digit is only 0, 1 (only two possible digits)

- The place values are then 2^n
  - 1, 2, 4, 8, 16, …

- It is useful to remember that $2^{10} = 1024$

2^2	2^1	2^0
1	0	1

Decimal = $2^2 + 2^0$

= 5

Binary Numbers

Operation	Output	Remainder
$23/2^7$	0	23
$23/2^6$	0	23
$23/2^5$	0	23
$23/2^4$	1	7
$23/2^3$	0	7
$23/2^2$	1	3
$23/2^1$	1	1
$23/2^0$	1	0

23 in binary is 00010111

Carry		1	1	
11	1	0	1	1
3	0	0	1	1
Addition (14)	1	1	1	0

Add by carrying 1s

- Subtract by borrowing 2s.
- See class reader

Binary Numbers

- One can think of binary numbers as a kind of **code**:
  - In communications and information processing, **code** is a system of rules to convert information—such as a letter, word, sound, image, or gesture—into another form or representation, sometimes shortened or secret, for communication through a channel or storage in a medium.

https://en.wikipedia.org/wiki/Code

Many Different Types of Codes

- Figuring out “good” codes is a big part of EE
- Security
- Error correction
- Compression
  - .mp3, .mp4, .jpeg, .avi, etc. Are all a code of the source
  - .zip
- Whole theory of information (Information theory) guides codes
  - Sets what is possible and not possible (Claude Shannon)

Binary Code

- Advantages
  - Uses a small number of bits, log2(N), to represent a number
  - Easy to generate and decode
  - Easy to do math on this representation
- Disadvantages
  - The value is in the "clear" (the data is not encrypted)
  - If any bit is in error, the number is lost
    - If you use more bits, can recover from single bit error
  - Can only represent one number per value
    - Let's say we would like to drive a display with 64 lights …

How To Represent Negative Numbers?

- Could create sign / magnitude code
  - Add one bit to be the sign bit

7	00000111
-7	10000111

- But that is ***not*** what works out best
  - To add two numbers, you first need to look at the sign bits
    - If they are the same, add
    - If they are different, subtract

Two's Complement Numbers

- Positive numbers are normal binary
- Negative numbers are defined so when we add them to |Num|, a normal adder will give zero
  - So
    - -1 = 1111111111111111
    - -2 = 1111111111111110

Carry	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
+3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
-3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
Addition	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Generating Two's Complement Numbers

- Take your number
  - Invert all the bits of the number
    - Make all "0" "1" and all "1" "0"
  - And then add 1 to the result

- Lets look at an example:

42	0	0	1	0	1	0	1	0
Flip 1 and 0	1	1	0	1	0	1	0	1
Add 1	1	1	0	1	0	1	1	0

- So -42 in two's complement is 11010110

The Way This Works

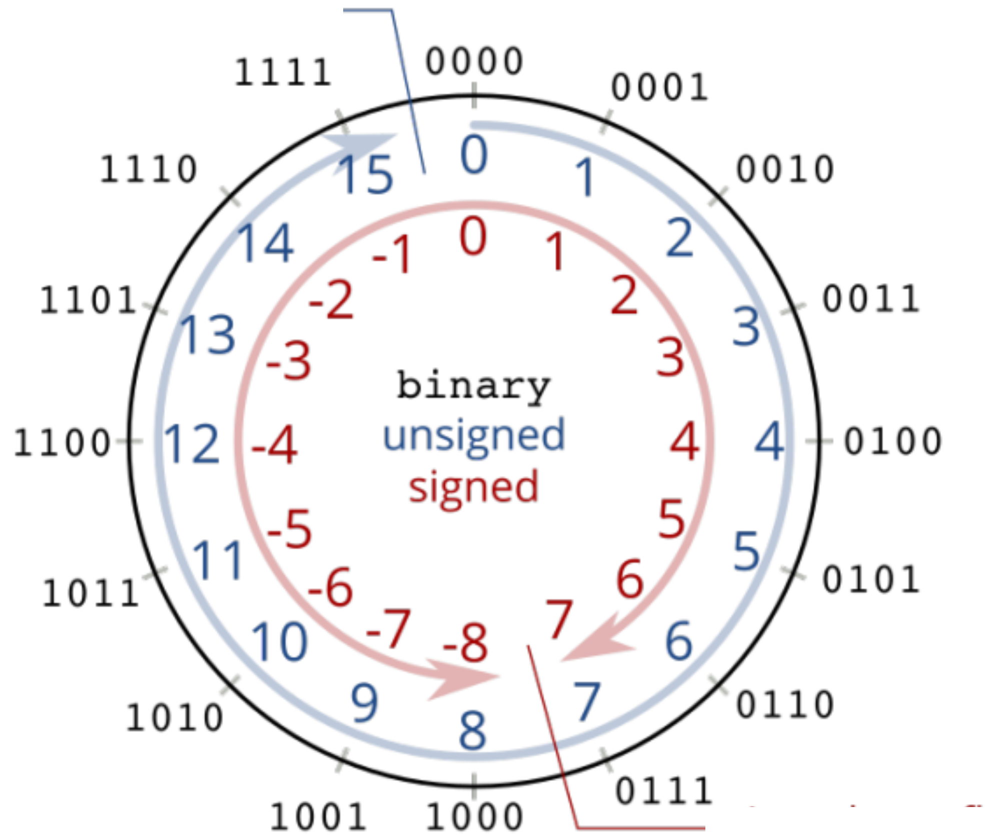


Figure 7.16: Two's complement circle

Funny Things Happen With Finite Numbers

- Your homework looks at some problems with numbers
  - That only have a finite number of bits
- Computers generally have two types of numbers
  - Signed numbers, and unsigned numbers
    - It interprets the MSB differently
  - Can get "interesting" results if you mistake
    - A signed number as unsigned and vice versa
- Also overflows have an interesting effect
  - Can add two positive numbers and get a negative result!

Overflow Situations

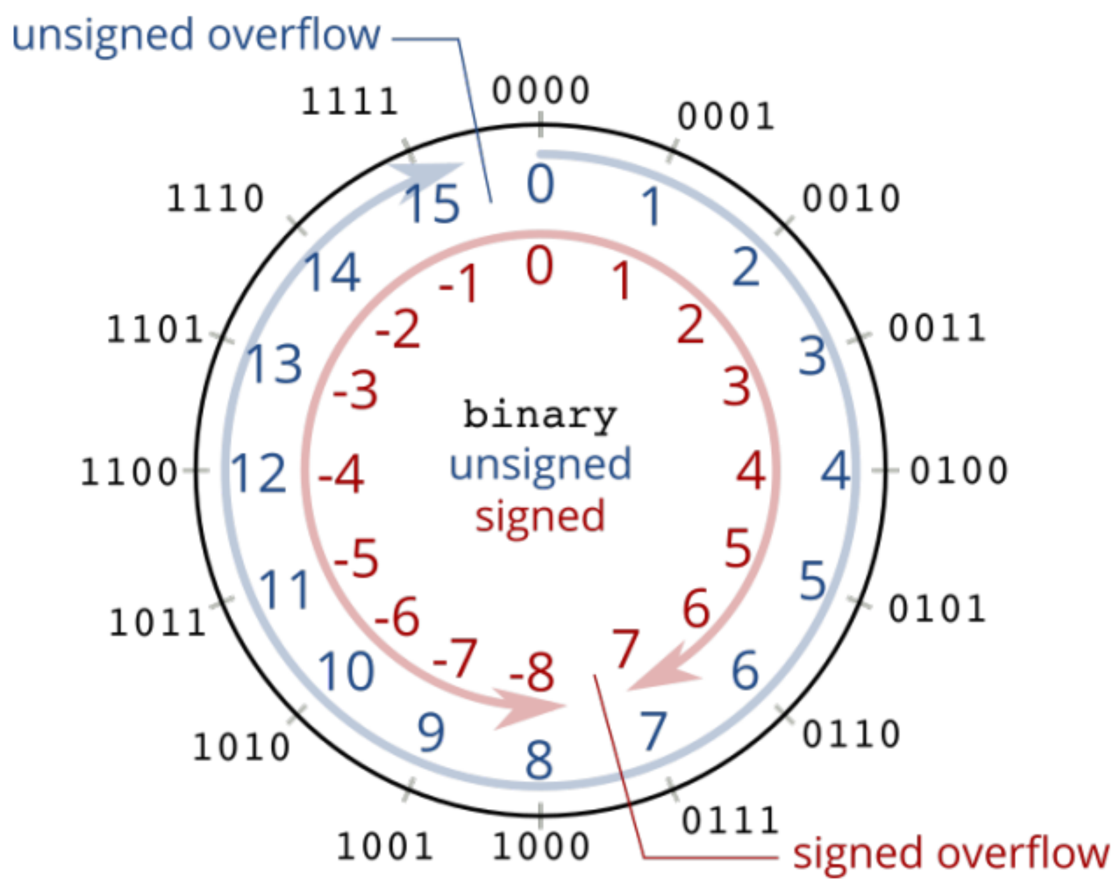


Figure 7.16: Two's complement circle

ERASURE CODES

An Example of Other Types of EE Coding Problems

- You want to send data to someone
  – But the communication link is not reliable
    - It will sometimes drop packets
    - But you will know which packets you got/lost
  – Say that the link might lose up to K of N packets
    - You have no control of which packets are lost
  – Called erasure channels, since it erases information

- How much data can you communicate in N packets?
  – How do you code the information to achieve that rate?

At First Life Seems Hard

- We need to communicate all the data reliably
  - If we can lose any 3 packets
  - Then if we transmit each value 3 times, we still might lose.
- Almost seems like we need to transmit each value 4 times
- Or for K erasures in N packets
  - Can transmit < N/K pieces of information
- Can we do better?

Upper Bound

- I don't know how to solve this but I do know one thing
- Impossible to transmit more than N-K packets of information
  - Since that is all the bits that you get at the receiver
- How close can we come to this upper bound?
  - We can actually achieve it!

Getting Around the Problem

- The problem is we don't know what data will be lost
  - To achieve the upper bound what data is lost can't matter
  - Any (N-K) code outputs can give us our (N-K) packet values
- For this to be true
  - Each packet value must be in at least K+1 code outputs
  - But since there are much less than (K+1)(N-K) outputs
    - Most code values must contain multiple packet values
    - Huh?
- Linear algebra to the rescue!

Simple Example

- Assume we want to transmit 3 values X1, X2, X3
  - And the channel may drop two in 5 packets

- What happens if we transmit:
  - X1
  - X2;
  - X3;
  - X1+X2+X3
  - X1+2*X2+3*X3

- Can we solve for X1, X2, X3 with any 3 packets?

Linear Algebra Formulation

- To solve for M = N-K variables
  - How many linear equations are needed?

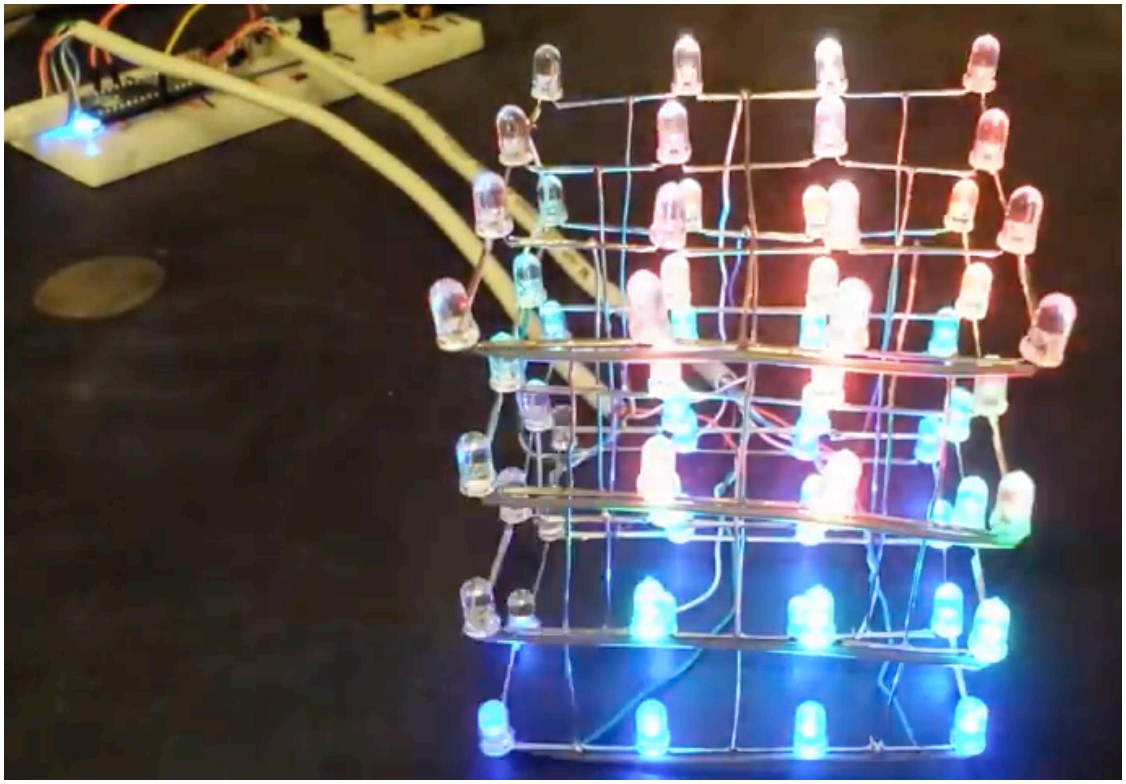
- If we create N independent linear combinations of M values
  - It doesn't matter which M equations we have
    - We have enough information to solve for the variables!

- Generally written in matrix notation
  - $Y = A X$, where A is a NxM matrix
    - $X = A^{*-1} Y^*$, where the * means the values you received

Time Multiplexing

- In the LED cube we will have 64 LEDs but they are wired so we can access rows and columns of the matrix.

- We'll explore a different type of coding later that will allow us to control each LED individually, even though we do not have a separate connection to each of them!



Learning Objectives

- Understand how binary numbers work
  - And be able to convert from decimal to binary numbers

- Understand what Electrical Engineers mean by a code
  - It is a set of rules to represent information
  - How to use two's complement codes
    - To represent positive and negative numbers
    - And what happens when an overflow occurs

- Understand how erasure codes allow you to recover your message even when some packets are dropped