

Fall 24

# DIGITAL ELECTRONICS IN ROBOTICS

---

Roberto Martin-Martin

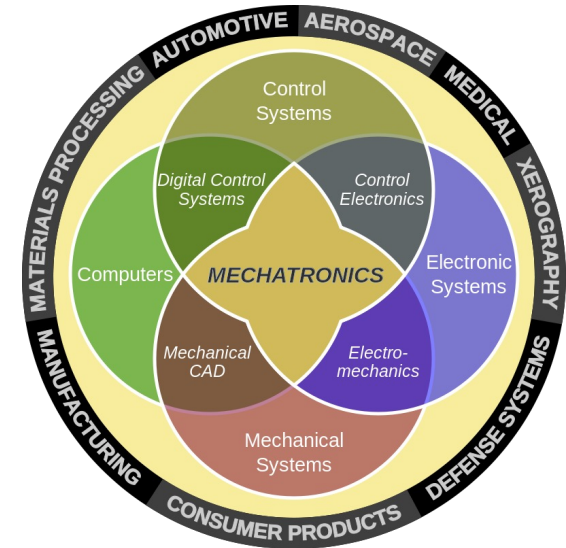
Assistant Professor of Computer Science

# Recap

- Ohm's Law
- Kirchoff's (current/voltage) Law
- Resistance, Capacitor, Inductor, Switches/Fuses
- Analysis of analog circuits
- Resistors in series and parallel
- RC circuit: charge and discharge
- RL circuit

# Common Topics in Mechatronics Course

- Electronic Circuits & Components
- Semiconductor Electronics
- System Response
- Analog Signal Processing using OpAmps
- **Digital Signals & Logic**
- Microcontroller Programming & Interfacing
- Data Acquisition
- Sensors
- Actuators (motors/gears)
- Component control methods (Bang-bang, PID, etc.)
- System control architectures (state machines)



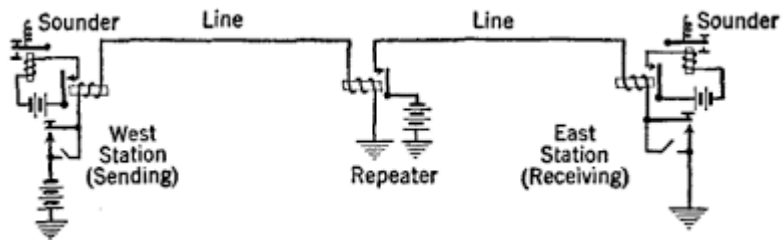
# What will you learn today?

- What are digital signals and logic and why do they matter?
  - High level overview...
  - ...with some practical considerations sprinkled in
- How do we
  1. Communicate digital signals?
  2. Represent alphanumeric data using digital signals?
  3. How do we perform calculations with digital signals?
    - Logic Circuits
    - Boolean Algebra

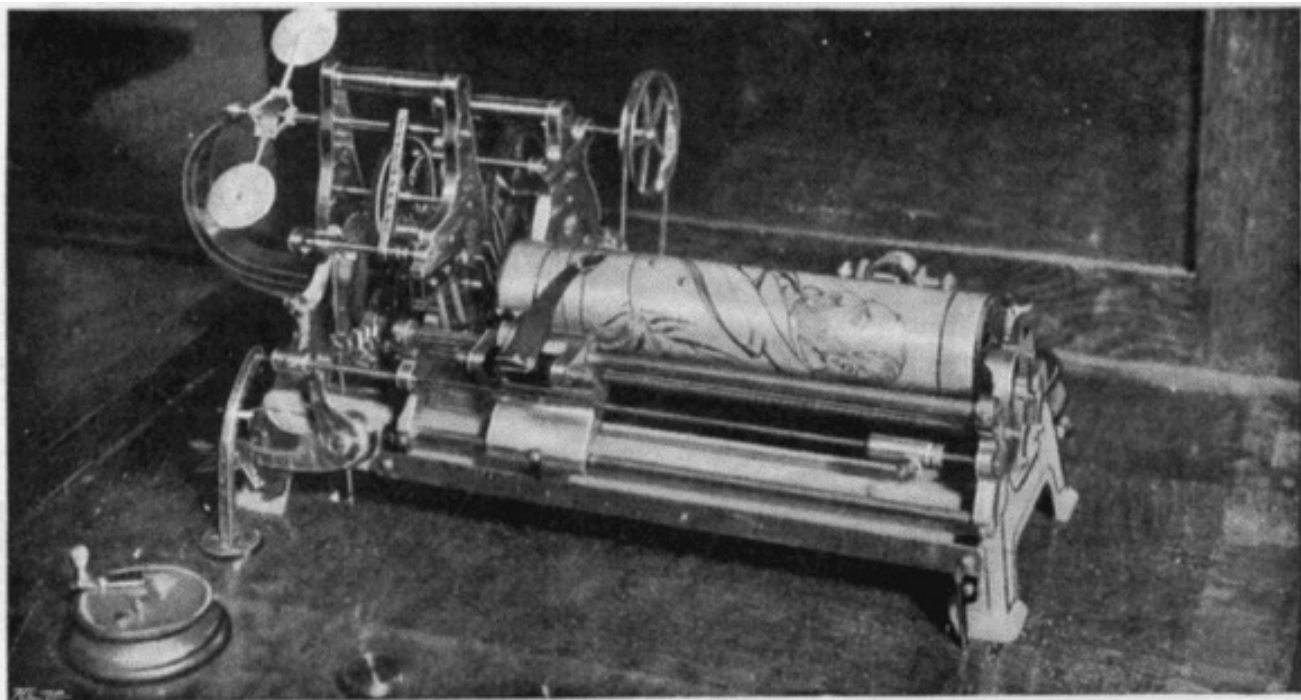


Michael William Sullivan

|    |         |    |         |    |         |
|----|---------|----|---------|----|---------|
| Aa | · —     | Jj | · — — — | Ss | ···     |
| Bb | — ···   | Kk | — · —   | Tt | —       |
| Cc | — · · · | Ll | · — · · | Uu | ·· —    |
| Dd | — · ·   | Mm | — —     | Vv | ·· · ·  |
| Ee | ·       | Nn | — ·     | Ww | · — —   |
| Ff | ·· — ·  | Oo | — — —   | Xx | — · · · |
| Gg | — — ·   | Pp | · — — · | Yy | ·· · ·  |
| Hh | ·· · ·  | Qq | — — · — | Zz | — —     |
| Ii | ··      | Rr | · — ·   |    |         |



# The Fax Machine

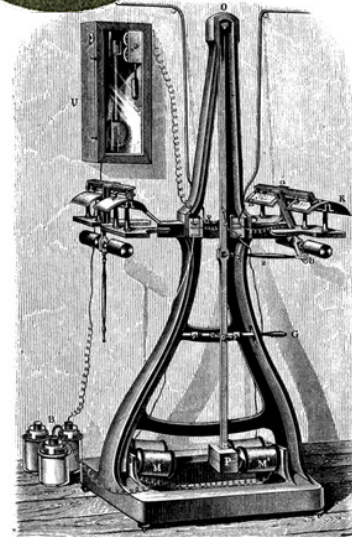


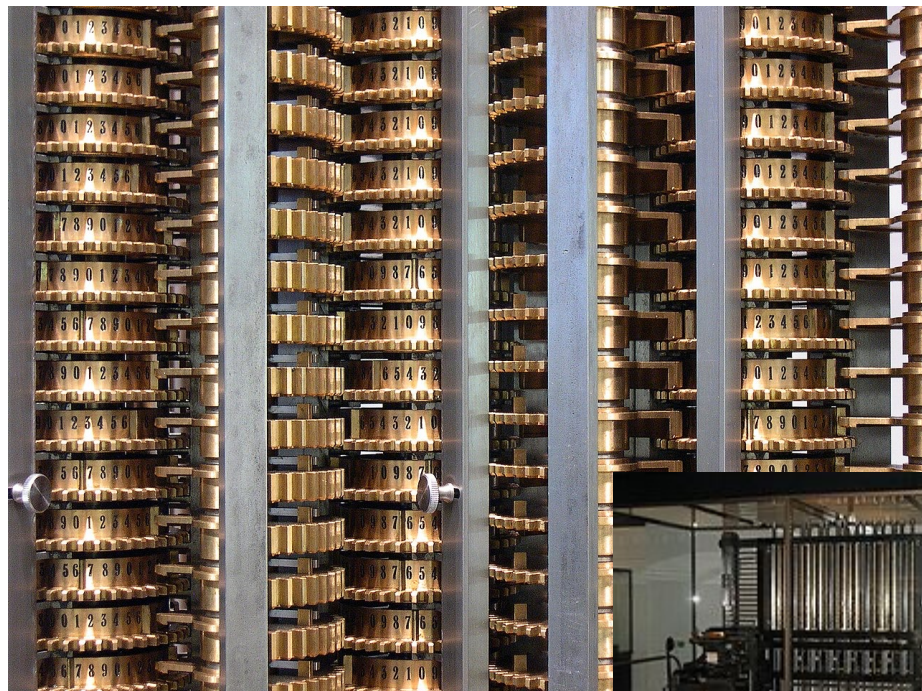
The transmitter in operation at the *New York Herald* office, showing Croker being telegraphed to the *Chicago Times Herald*, 1000 miles away.



Giovanni Caselli

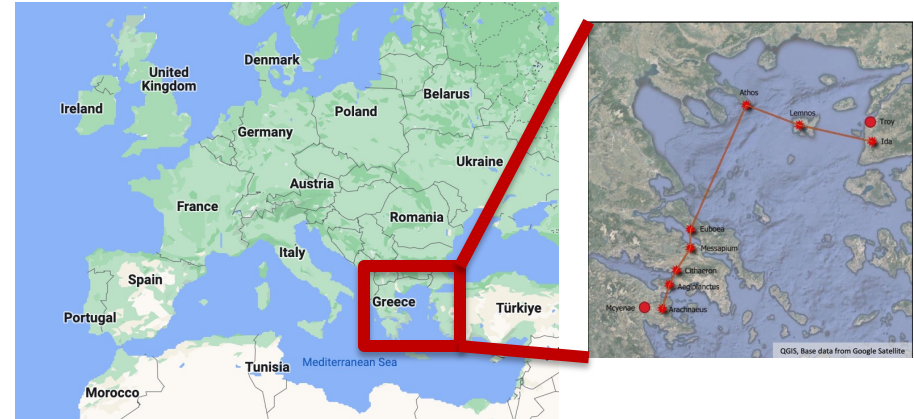
(Fig. 431.)



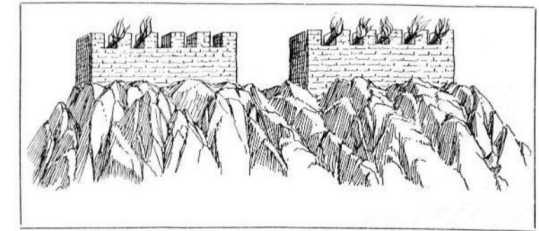


# Does digital communication need to be binary?

- Ancient Greeks used a different system
  - Base what?
- The system was called Phryctoria
- Two groups of torches on towers at the top of some mountains
- They communicated the fall of Troy with it



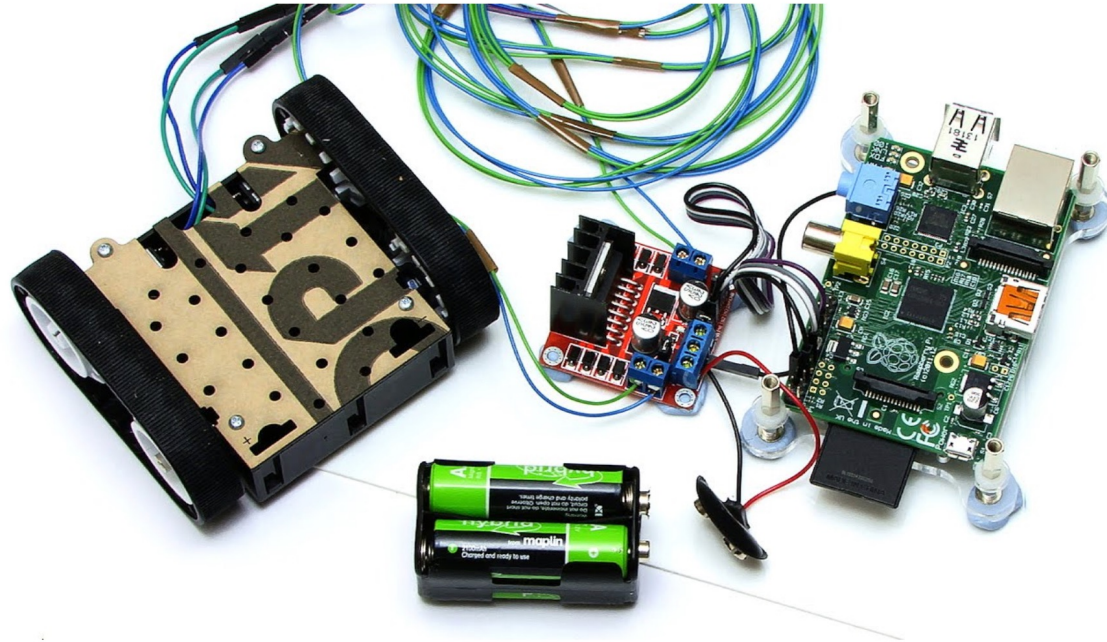
|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | A | B | Γ | Δ | E |
| 2 | Z | H | Θ | I | K |
| 3 | Λ | M | N | Ξ | O |
| 4 | Π | P | Σ | T | Y |
| 5 | Φ | X | Ψ | Ω |   |



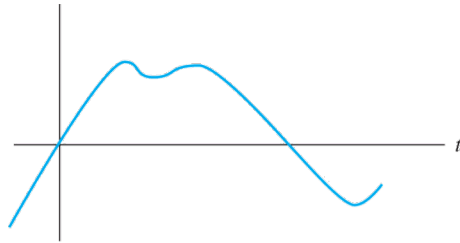
[458 BC]



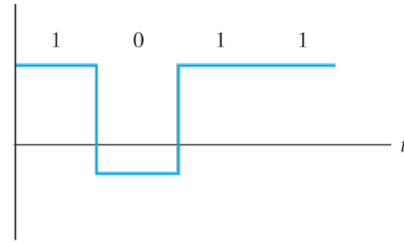
# In Robotics



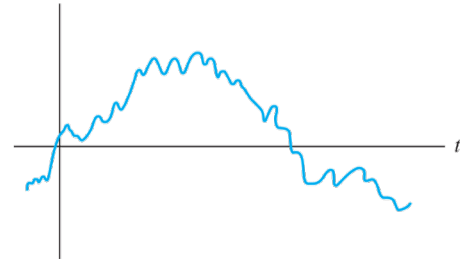
# Defining a bit (0 or 1) using an analog signal



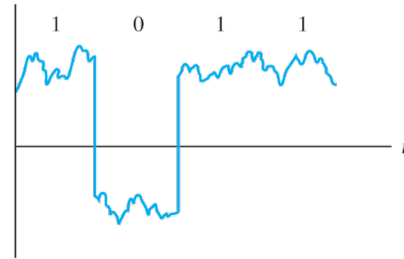
(a) Analog signal



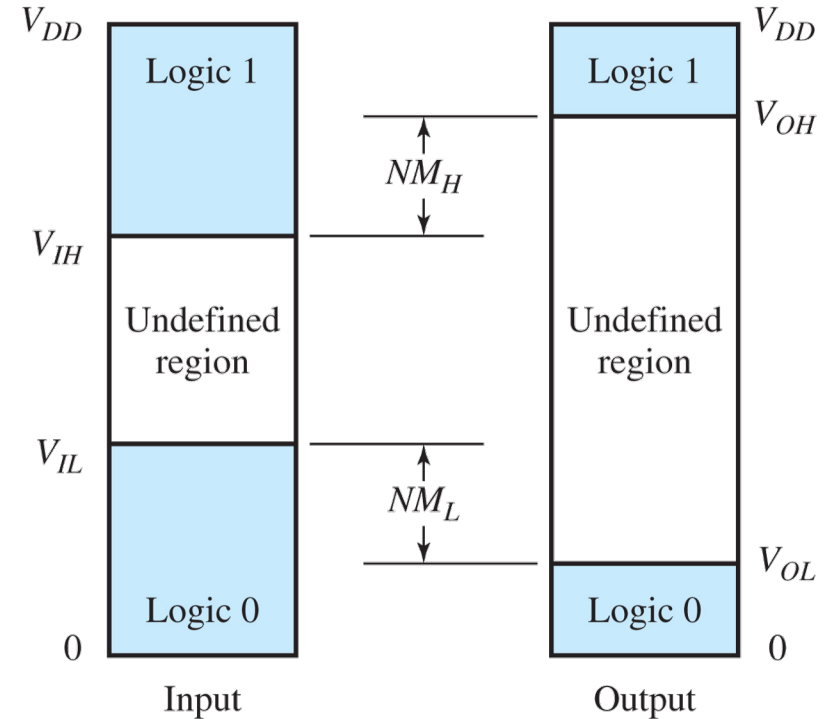
(b) Digital signal



(c) Analog signal plus noise



(d) Digital signal plus noise



# Advantages of Digital Over Analog

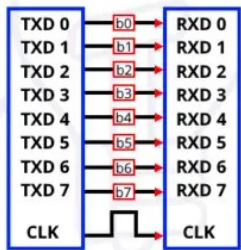
1. Minimizes need to accurately model the analog circuit
2. Less susceptible to noise (robust)
3. Digital circuits are easier and cheaper than analog circuits.
4. Hardware implementations are more flexible
5. Cross-talk issues are minimized.
6. Encryption and compression are possible.
7. Can employ error detection and correction in the algorithm.
8. Modern computers enable sufficient digital fidelity.
9. Etc. etc. etc.

# Some Definitions

- **Bit:** single binary digit (0 or 1)
  - **Positive logic:** 1=higher voltage & 0-lower voltage
- **Digital word:** collection of bits
  - **Nibble:** collection of 4 bits
  - **Byte:** collection of 8 bits
- **Bit order**
  - **LSB:** Least significant bit represents the lowest valued bit (typically on the right)
  - **MSB:** Most significant bit represents the greatest valued bit (typically on the left)

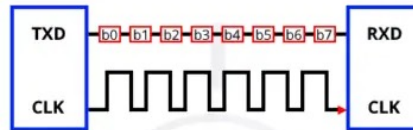
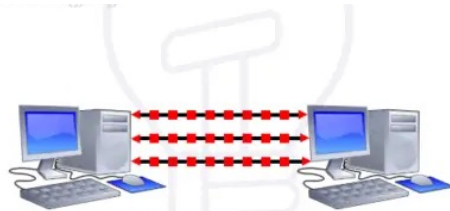
# How do we communicate digitally?

- **Parallel transmission:**  $n$ -bit word transferred on  $n$  wires (1 wire for each bit).
  - Includes a common or ground wire.
- **Serial transmission:** successive bits of an  $n$ -bit word transferred one after the other on a single pair of wires.
  - Requires a coordinated clock



[www.electricaltechnology.org](http://www.electricaltechnology.org)

**Parallel Communication**

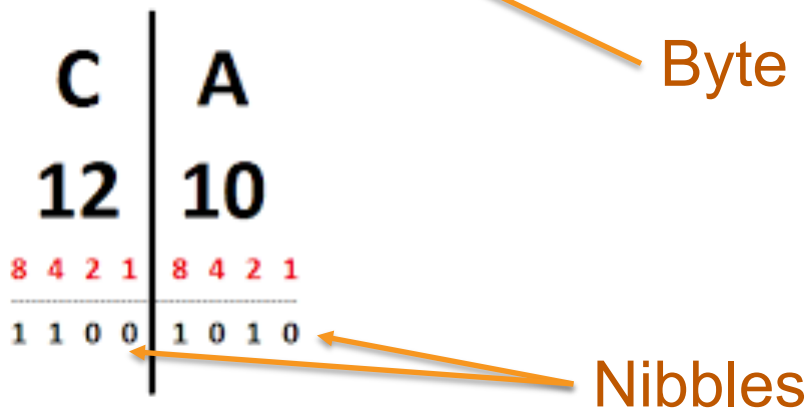
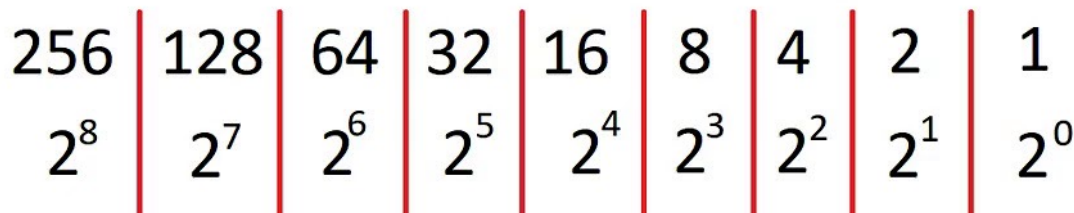


**Serial Communication**



# How do we represent numbers/symbols with 1's and 0's?

- Base-2, Base-10, Octal (Base-8) Hexadecimal (Base-16)



|   | Octal | Hexadecimal |
|---|-------|-------------|
| 0 | 000   | 0 0000      |
| 1 | 001   | 1 0001      |
| 2 | 010   | 2 0010      |
| 3 | 011   | 3 0011      |
| 4 | 100   | 4 0100      |
| 5 | 101   | 5 0101      |
| 6 | 110   | 6 0110      |
| 7 | 111   | 7 0111      |
| 8 |       | 8 1000      |
| 9 |       | 9 1001      |
| A |       | A 1010      |
| B |       | B 1011      |
| C |       | C 1100      |
| D |       | D 1101      |
| E |       | E 1110      |
| F |       | F 1111      |

<https://pollev.com/robertomartinmartin739>

# The (non) magic behind base-10



- Why do we use base-10?

# Using numbers to represent letters and symbols: ASCII Table

## ASCII TABLE

| Decimal | Hex | Char                   | Decimal | Hex | Char    | Decimal | Hex | Char | Decimal | Hex | Char  |
|---------|-----|------------------------|---------|-----|---------|---------|-----|------|---------|-----|-------|
| 0       | 0   | [NULL]                 | 32      | 20  | [SPACE] | 64      | 40  | @    | 96      | 60  | `     |
| 1       | 1   | [START OF HEADING]     | 33      | 21  | !       | 65      | 41  | A    | 97      | 61  | a     |
| 2       | 2   | [START OF TEXT]        | 34      | 22  | "       | 66      | 42  | B    | 98      | 62  | b     |
| 3       | 3   | [END OF TEXT]          | 35      | 23  | #       | 67      | 43  | C    | 99      | 63  | c     |
| 4       | 4   | [END OF TRANSMISSION]  | 36      | 24  | \$      | 68      | 44  | D    | 100     | 64  | d     |
| 5       | 5   | [ENQUIRY]              | 37      | 25  | %       | 69      | 45  | E    | 101     | 65  | e     |
| 6       | 6   | [ACKNOWLEDGE]          | 38      | 26  | &       | 70      | 46  | F    | 102     | 66  | f     |
| 7       | 7   | [BELL]                 | 39      | 27  | '       | 71      | 47  | G    | 103     | 67  | g     |
| 8       | 8   | [BACKSPACE]            | 40      | 28  | (       | 72      | 48  | H    | 104     | 68  | h     |
| 9       | 9   | [HORIZONTAL TAB]       | 41      | 29  | )       | 73      | 49  | I    | 105     | 69  | i     |
| 10      | A   | [LINE FEED]            | 42      | 2A  | *       | 74      | 4A  | J    | 106     | 6A  | j     |
| 11      | B   | [VERTICAL TAB]         | 43      | 2B  | +       | 75      | 4B  | K    | 107     | 6B  | k     |
| 12      | C   | [FORM FEED]            | 44      | 2C  | ,       | 76      | 4C  | L    | 108     | 6C  | l     |
| 13      | D   | [CARRIAGE RETURN]      | 45      | 2D  | -       | 77      | 4D  | M    | 109     | 6D  | m     |
| 14      | E   | [SHIFT OUT]            | 46      | 2E  | .       | 78      | 4E  | N    | 110     | 6E  | n     |
| 15      | F   | [SHIFT IN]             | 47      | 2F  | /       | 79      | 4F  | O    | 111     | 6F  | o     |
| 16      | 10  | [DATA LINK ESCAPE]     | 48      | 30  | 0       | 80      | 50  | P    | 112     | 70  | p     |
| 17      | 11  | [DEVICE CONTROL 1]     | 49      | 31  | 1       | 81      | 51  | Q    | 113     | 71  | q     |
| 18      | 12  | [DEVICE CONTROL 2]     | 50      | 32  | 2       | 82      | 52  | R    | 114     | 72  | r     |
| 19      | 13  | [DEVICE CONTROL 3]     | 51      | 33  | 3       | 83      | 53  | S    | 115     | 73  | s     |
| 20      | 14  | [DEVICE CONTROL 4]     | 52      | 34  | 4       | 84      | 54  | T    | 116     | 74  | t     |
| 21      | 15  | [NEGATIVE ACKNOWLEDGE] | 53      | 35  | 5       | 85      | 55  | U    | 117     | 75  | u     |
| 22      | 16  | [SYNCHRONOUS IDLE]     | 54      | 36  | 6       | 86      | 56  | V    | 118     | 76  | v     |
| 23      | 17  | [ENG OF TRANS. BLOCK]  | 55      | 37  | 7       | 87      | 57  | W    | 119     | 77  | w     |
| 24      | 18  | [CANCEL]               | 56      | 38  | 8       | 88      | 58  | X    | 120     | 78  | x     |
| 25      | 19  | [END OF MEDIUM]        | 57      | 39  | 9       | 89      | 59  | Y    | 121     | 79  | y     |
| 26      | 1A  | [SUBSTITUTE]           | 58      | 3A  | :       | 90      | 5A  | Z    | 122     | 7A  | z     |
| 27      | 1B  | [ESCAPE]               | 59      | 3B  | ;       | 91      | 5B  | [    | 123     | 7B  | {     |
| 28      | 1C  | [FILE SEPARATOR]       | 60      | 3C  | <       | 92      | 5C  | \    | 124     | 7C  |       |
| 29      | 1D  | [GROUP SEPARATOR]      | 61      | 3D  | =       | 93      | 5D  | ]    | 125     | 7D  | }     |
| 30      | 1E  | [RECORD SEPARATOR]     | 62      | 3E  | >       | 94      | 5E  | ^    | 126     | 7E  | ~     |
| 31      | 1F  | [UNIT SEPARATOR]       | 63      | 3F  | ?       | 95      | 5F  | _    | 127     | 7F  | [DEL] |

- 8b per character
- Other codes
  - EBDIC
  - UNICODE
  - etc.



<https://pollev.com/robertomartinmartin739>

## Exercise

- Write “hello” in ASCII in binary
  1. 01001000 01100101 01101100  
01101100 01100000
  2. 11001000 01100101 01101100  
01101100 01101111
  3. 01001000 01100101 01101100  
01101100 01101111
  4. 01001000 01100101 01101101  
01101101 01101111

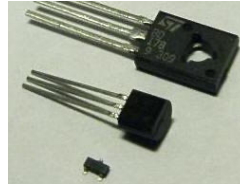


# How do we get from binary signals to calculation?

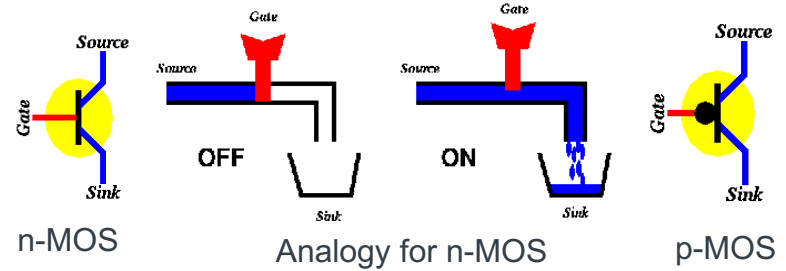
- **Logic Circuits**

- A fundamental building block of to enable digital circuits to perform calculations.
- Perform logical operations using binary inputs to produce binary outputs.
- Transistors enable digital logic gates
- 7 fundamental logic gates.
  - (BUF), AND, OR, XOR, NOT, NAND, NOR, and XNOR

# Transistors

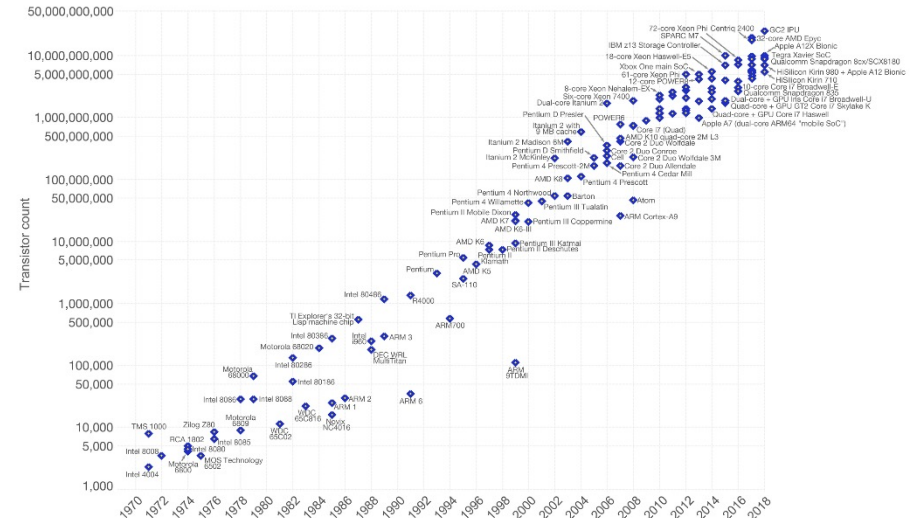


- Have three connections
  - Source, gate, sink.
- Today's chips have >1 million transistors per square millimeter
- Simple faucet analogy
  - When the gate is ON (value 1), then the sink connected to the source and transistor is ON.
  - When the gate is OFF (value 0), then the no flow from source to sink and transistor is OFF.
- Today's transistors are CMOS
  - Complementary Metal Oxide Semiconductors
- Two types:
  - n-MOS if the gate has a positive voltage, transistor is ON
  - p-MOST if the gate has a positive voltage, the transistor is OFF



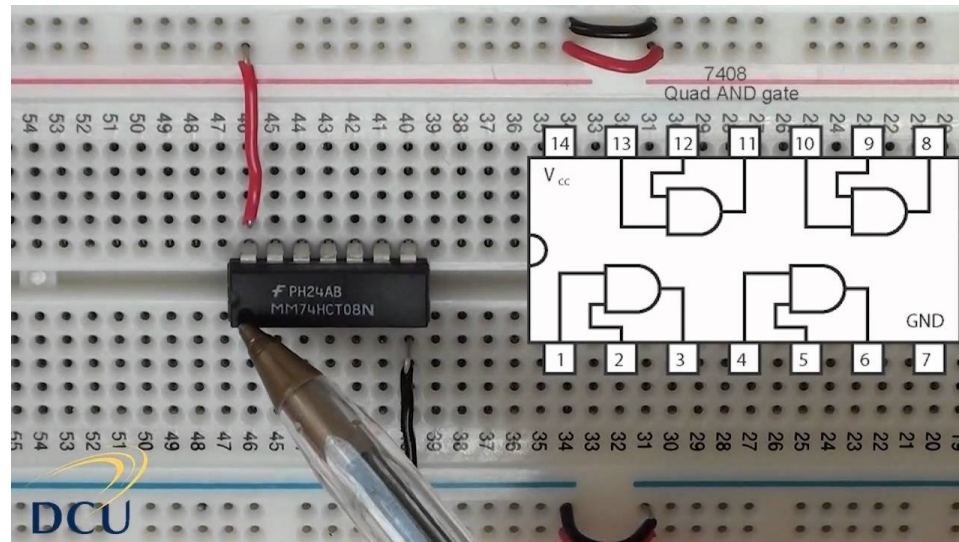
## Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



# How do Logic Elements look like?

- Integrated circuits
  - These enable us to use simple logic chips for simple tasks.
  - Not everything needs a microcontroller.

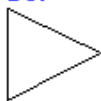
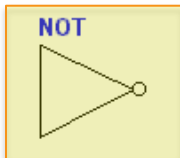


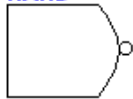
# How do we get from binary communication to calculation?

- **Logic Circuits**

- A fundamental building block of to enable digital circuits to perform calculations.
- Perform logical operations using binary inputs to produce binary outputs.
- Transistors enable digital logic gates
  - 0 False realized by a +5V
  - 1 True realized by a +0V
- 7 fundamental logic gates.
  - BUF, NOT, AND, NAND, OR, NOR, XOR, and XNOR

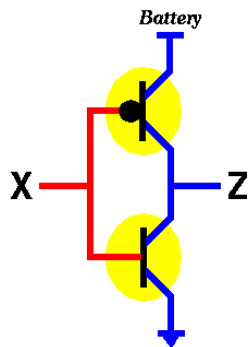
# The NOT (Inverter) Logic Gate

**BUF**

**NOT**

**AND**

**NAND**

**OR**

**NOR**

**Exclusive OR**

**Exclusive NOR**


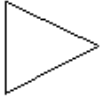
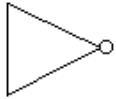
- If  $X=0$ :
  - bottom n-transistor is OFF
  - top p-transistor is ON
  - Z is 1
- If  $X = 1$ 
  - top p-transistor is OFF
  - no flow into the n-transistor even though it is ON
  - Z is 0

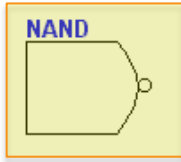
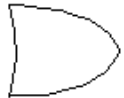
**NOT**


| INPUT |  | OUTPUT |
|-------|--|--------|
| A     |  |        |
| 0     |  | 1      |
| 1     |  | 0      |

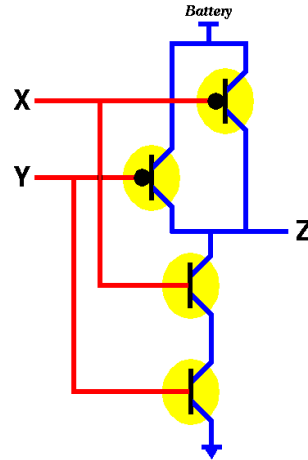
**Truth Table**

# The NAND (not AND) Logic Gate

**BUF**

**NOT**

**AND**

**NAND**

**OR**

**NOR**

**Exclusive OR**

**Exclusive NOR**


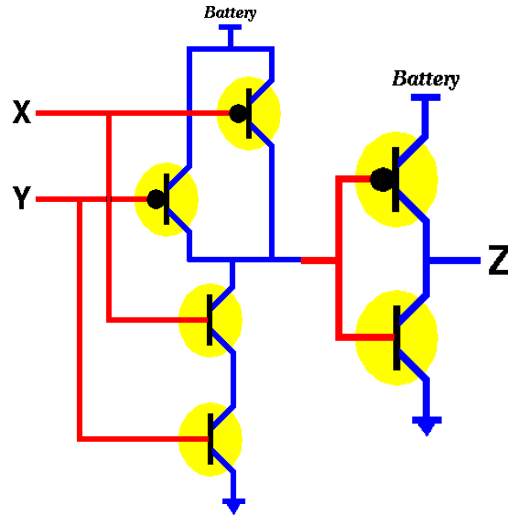
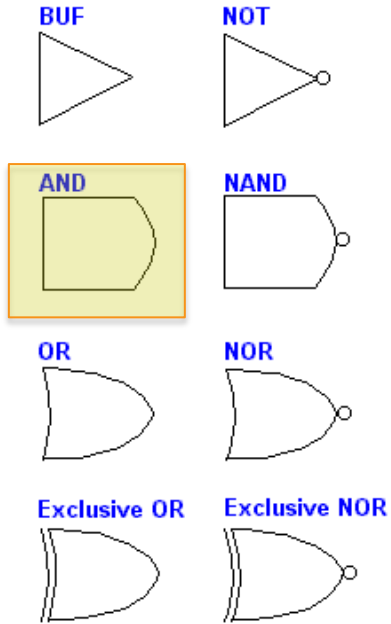

- If  $X=0$  and  $Y = 0$ 
  - Both the bottom transistors are OFF
  - Both the top transistors are ON
  - Electricity will flow to Z which is ON
- If  $X = 1$  and  $Y = 0$ 
  - Then a circuit similar to OR exists with X
  - Same with Y
  - Z is 1
- Etc.



| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 1      |
| 1     | 0 | 1      |
| 0     | 1 | 1      |
| 1     | 1 | 0      |

Truth Table

# The AND Logic Gate

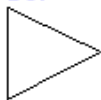
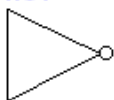
| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 0      |
| 1     | 0 | 0      |
| 0     | 1 | 0      |
| 1     | 1 | 1      |

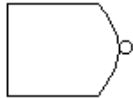
Truth Table

- Combine the NAND and the NOT gates.



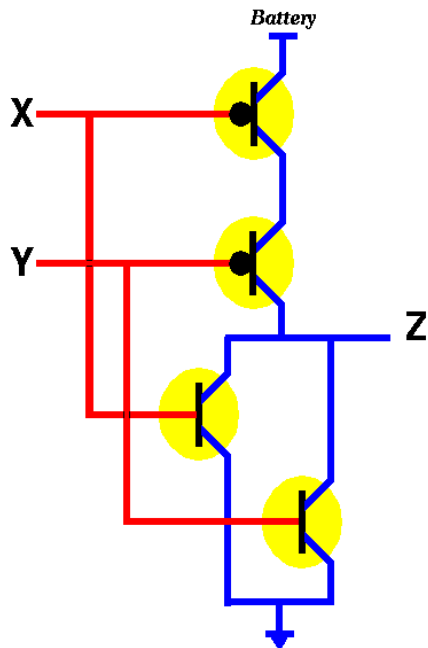
# The NOR Logic Gate

**BUF**

**NOT**

**AND**

**NAND**

**OR**

**NOR**

**Exclusive OR**

**Exclusive NOR**


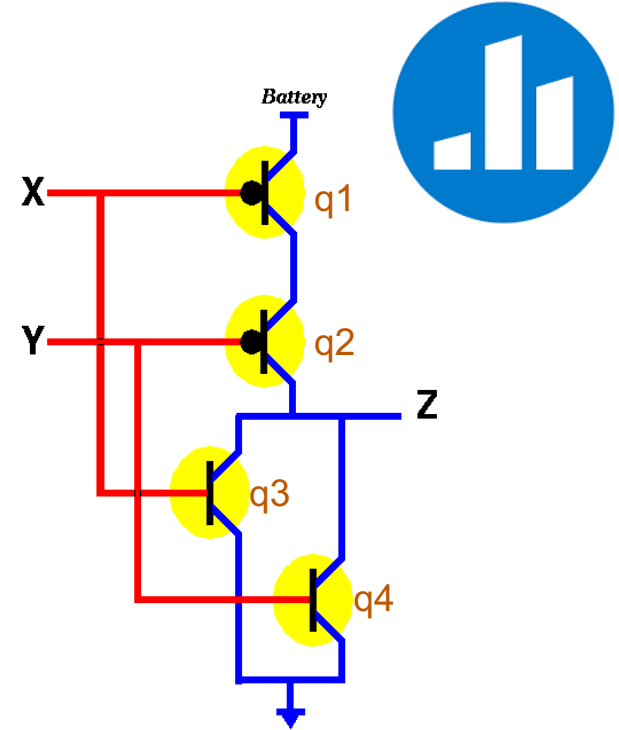
| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 1      |
| 1     | 0 | 0      |
| 0     | 1 | 0      |
| 1     | 1 | 0      |

Truth Table

<https://pollev.com/robertomartinmartin739>

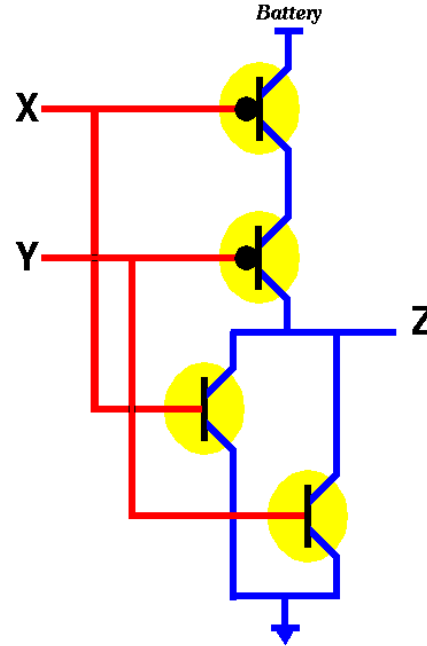
## Exercise

- What transistors are ON if  $X=1$  and  $Y=0$  in the following NOR logic gate?
- What is then the value at  $z$ ?

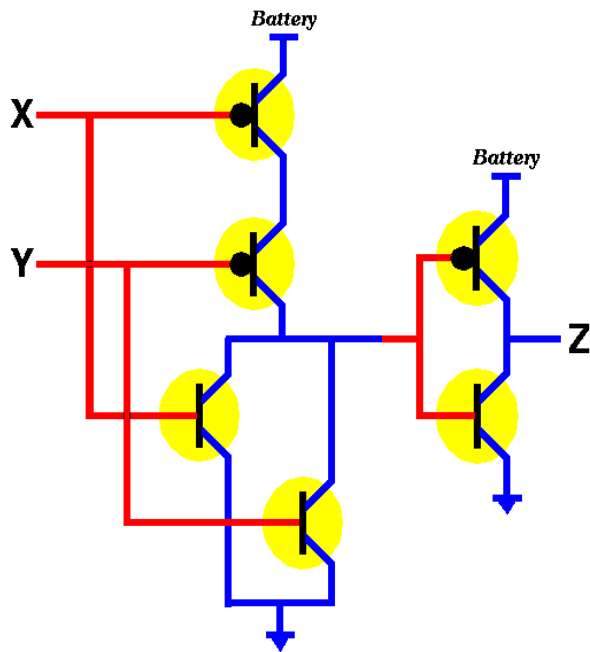
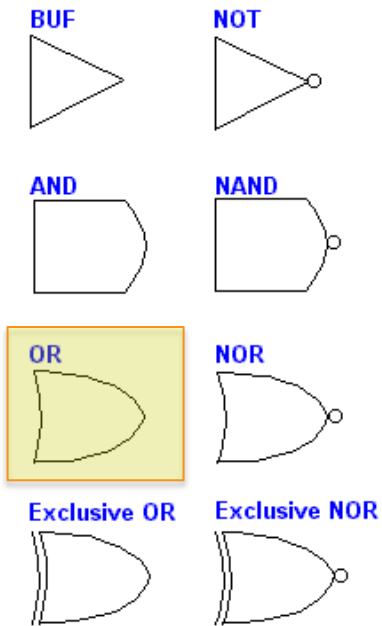


# Exercise

- What is the truth table of the NOR gate?



# The OR Logic Gate



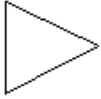
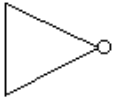
- Combine the NOR and the NOT gates

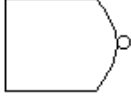
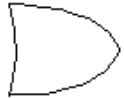


| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 0      |
| 1     | 0 | 1      |
| 0     | 1 | 1      |
| 1     | 1 | 1      |

Truth Table

# The XOR (eXclusive OR) and XNOR Logic Gates

**BUF**

**NOT**

**AND**

**NAND**

**OR**

**NOR**

**Exclusive OR**

**Exclusive NOR**


| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 0      |
| 1     | 0 | 1      |
| 0     | 1 | 1      |
| 1     | 1 | 0      |



| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 1      |
| 1     | 0 | 0      |
| 0     | 1 | 0      |
| 1     | 1 | 1      |

Truth Tables

# Summary of Truth Tables

- Using transistors, we can create the foundational logic blocks to then construct higher functions.
- These functions are critical for
  - Core understanding of microcontrollers
  - IC integration
  - PLC programming

YES



| INPUT |  | OUTPUT |
|-------|--|--------|
| A     |  |        |
| 0     |  | 0      |
| 1     |  | 1      |

NOT



| INPUT |  | OUTPUT |
|-------|--|--------|
| A     |  |        |
| 0     |  | 1      |
| 1     |  | 0      |

AND



| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 0      |
| 1     | 0 | 0      |
| 0     | 1 | 0      |
| 1     | 1 | 1      |

OR



| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 0      |
| 1     | 0 | 1      |
| 0     | 1 | 1      |
| 1     | 1 | 1      |

XOR



| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 0      |
| 1     | 0 | 1      |
| 0     | 1 | 1      |
| 1     | 1 | 0      |

NAND



| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 1      |
| 1     | 0 | 1      |
| 0     | 1 | 1      |
| 1     | 1 | 0      |

NOR



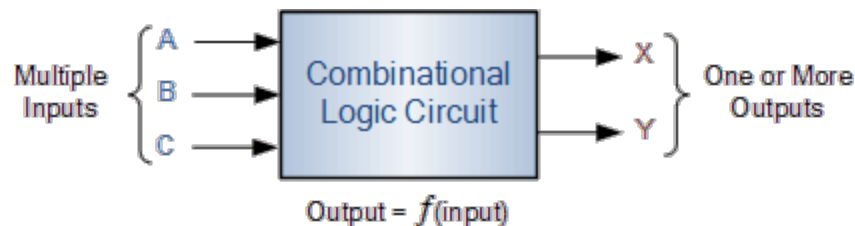
| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 1      |
| 1     | 0 | 0      |
| 0     | 1 | 0      |
| 1     | 1 | 0      |

XNOR



| INPUT |   | OUTPUT |
|-------|---|--------|
| A     | B |        |
| 0     | 0 | 1      |
| 1     | 0 | 0      |
| 0     | 1 | 0      |
| 1     | 1 | 1      |

# Combinatorial Logic Circuits



- **Combinatorial Logic Circuits:** Memoryless digital logic circuits whose output at anytime only depends on the combination of its inputs.
  - Akin to Functional Programming – There are no stored internal states
  - There is no feedback – previous output(s) are not known.
  - Built from the foundational logic gates presented above.
- Three ways to specify
  - **Boolean Algebra:** Uses only '0' and '1' and logic operators. CLCs formed resulting from algebraic expressions.
  - **Truth Tables:** A concise table showing all possible outputs given all possible combinations of inputs.
  - **Logic Diagram:** Graphical representation of the CLC showing wiring and connections between each individual logic gate.
- Also known as **Decision Making Circuits**

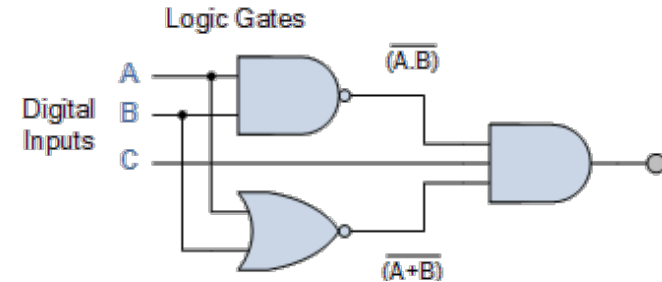
# Example

Boolean Expression

$$Q = \overline{(A \cdot B)} \cdot \overline{(A + B)} \cdot C$$

Typical  
Truth Table

| C | B | A | Q |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

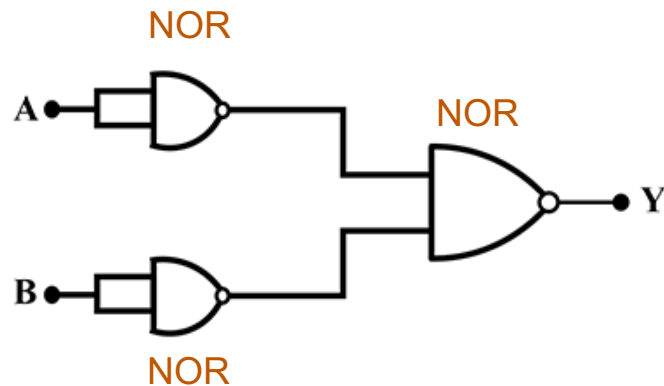




<https://pollev.com/robertomartinmartin739>

## Exercise

- What is the truth table of the given circuit?
- What is it equivalent to?



# Introduce Boolean Algebra

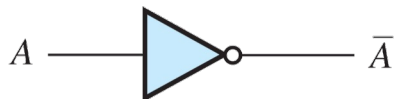
- Succinctly represent a logic gate or truth table as an *algebraic expression*
- The algebraic properties of the expression must be defined and strictly followed
  - What are the symbolic representations
  - What are operations
  - What are valid transformations on expressions



# Boolean Algebra: NOT Gate

| $A$ | $\bar{A}$ |
|-----|-----------|
| 0   | 1         |
| 1   | 0         |

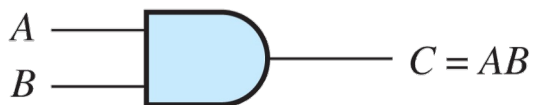
- $\bar{\bar{A}} = A$



# Boolean Algebra: AND Gates

| $A$ | $B$ | $C = AB$ |
|-----|-----|----------|
| 0   | 0   | 0        |
| 0   | 1   | 0        |
| 1   | 0   | 0        |
| 1   | 1   | 1        |

(a) Truth table

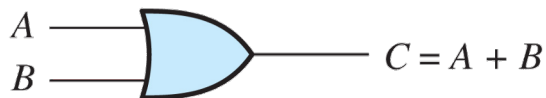


- $AA = A$
- $A1 = A$
- $A0 = 0$
- $A\bar{A} = 0$
- $AB = BA$
- $A(BC) = (AB)C = ABC$

# Boolean Algebra: OR Gates

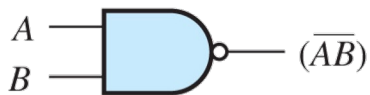
| $A$ | $B$ | $C = A + B$ |
|-----|-----|-------------|
| 0   | 0   | 0           |
| 0   | 1   | 1           |
| 1   | 0   | 1           |
| 1   | 1   | 1           |

(a) Truth table

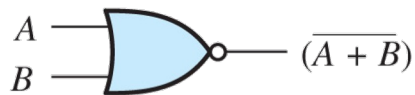


- $A + 0 = A$
- $A + 1 = 1$
- $A + \bar{A} = 1$
- $A + A = A$
- $A(B + C) = AB + AC$
- $(A + B) + C = A + (B + C) = A + B + C$

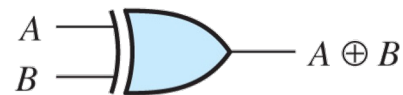
# Other logic gates



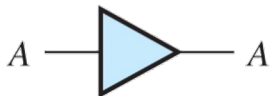
(a) NAND gate



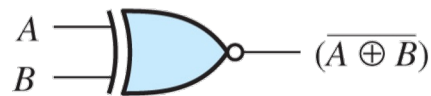
(b) NOR gate



(c) XOR gate



(d) Buffer



(e) Equivalence gate

# Summary of Properties in Boolean Algebra

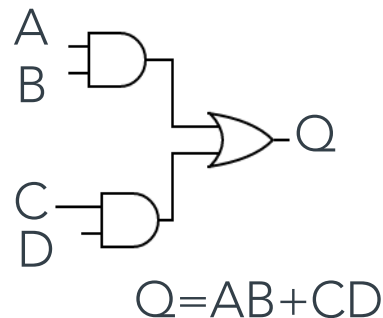
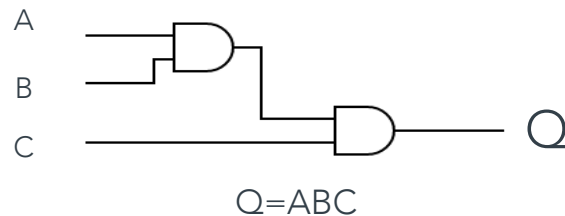
1.  $0 + X = X$
2.  $1 + X = 1$
3.  $X + X = X$
4.  $X + \bar{X} = 1$
5.  $0 \cdot X = 0$
6.  $1 \cdot X = X$
7.  $X \cdot X = X$
8.  $X \cdot \bar{X} = 0$
9.  $\overline{\bar{X}} = X$
10.  $X + Y = Y + X$
11.  $X \cdot Y = Y \cdot X$
12.  $X + (Y + Z) = (X + Y) + Z$
13.  $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
14.  $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$
15.  $X + X \cdot Z = X$
16.  $X \cdot (X + Y) = X$
17.  $(X + Y) \cdot (X + Z) = X + Y \cdot Z$
18.  $X + \bar{X} \cdot Y = X + Y$
19.  $X \cdot Y + Y \cdot Z + \bar{X} \cdot Z = X \cdot Y + \bar{X} \cdot Z$

} Commutative law

} Associative law

Distributive law

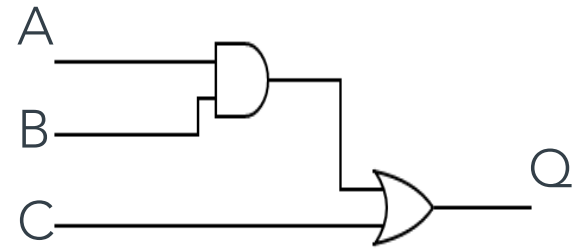
Absorption law



<https://pollev.com/robertomartinmartin739>

## Exercise

- What is the algebraic expression of the given circuit?

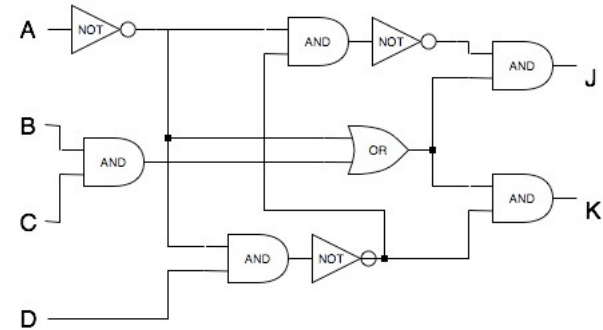




<https://pollev.com/robertomartinmartin739>

# Exercise

- What is the algebraic expression of the given circuit?



# Conclusions

- From analog voltage to abstract Boolean algebra in one lecture!
- Digital communication has been understood as necessary since ancient times → clean signal!
- The transistor enabled the use of digital concepts to make decisions using combinations of logical gates.
- We can design decision making circuits using
  - Boolean Algebra
  - Truth Table Analysis
  - Looking at the Combinatorial Logic Circuits
- Common Decision-Making Circuits are commonly found in robotics in ICs to perform common functions.
- Next up: State machines (~CLCs + memory)