Data and measurement

How to make and store measurements on a computer.

Or a brief introduction on how computers think.

Are they really intelligent, even if its artificial?

Bits

Bit = binary digit.

- This is the smallest unit of information on a computer.
- A bit is either ‘on’ or ‘off’, ‘yes or no’, ‘high or low’.
- There is no intermediate state.
- All information is stored in bits.

Bits

Bit = binary digit.

- Each bit has only two possible states, but computers can do more complex tasks than ‘yes’ or ‘no’.
- This is possible by organizing bits into groups.

How do we count with Bits?

Remember positional notation from (elementary school)?

- In Base 10, we have: 0,1,2,3,4,5,6,7,8,9 (10 symbols).
- 00009 = 9.
- If we want a number > 9, we have to *increment to a new position*.
- 100009 = (1×10^5) (0×10^4) (0×10^3) (0×10^2) (0×10^1) (0×10^0)

How do we count with Bits?

Positional notation also applies for computers, but with fewer symbols.

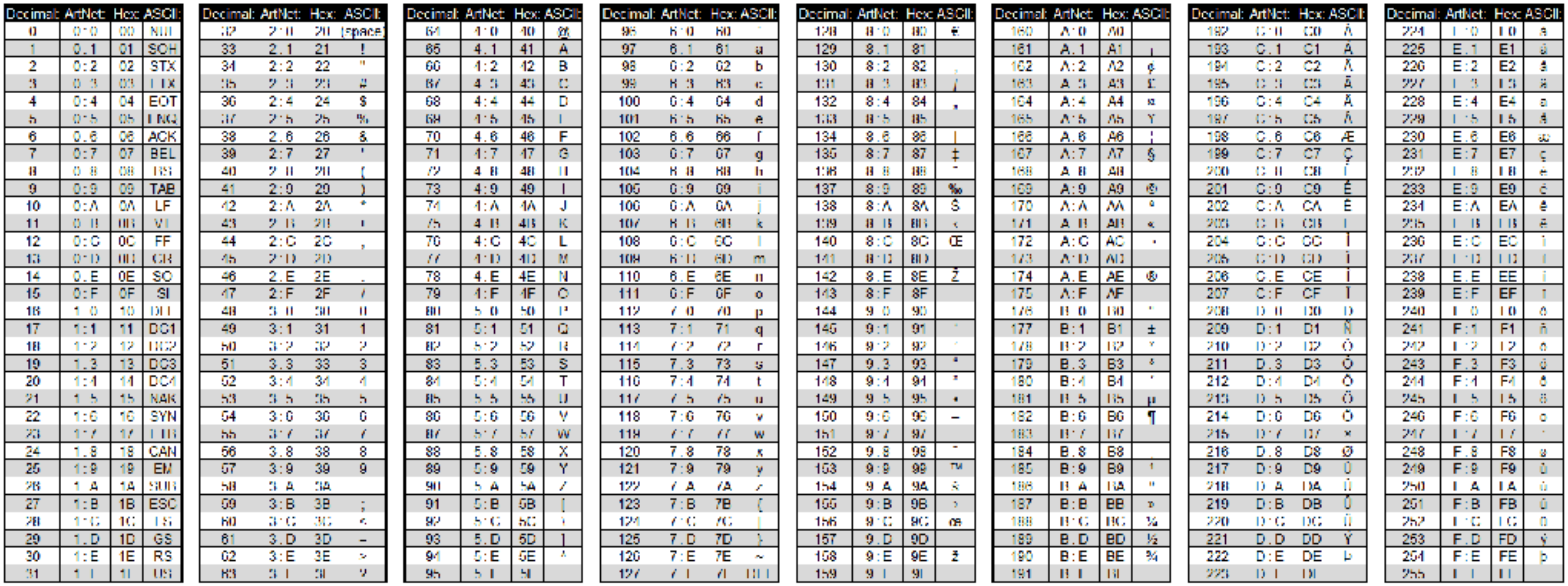
- In Base 2, we have: 0,1 (2 symbols).
- If we want a positional number > 1, we have to *increment to a new position*.
- 100001 = $(1x2^5)$ $(0x2^4)$ $(0x2^3)$, $(0x2^2)$ $(0x2^1)$ $(1x2^0)$

Bytes

- 0000000 = 1 byte.   Also known as a binary number.
- Each ‘position’ in the byte has 2 possible states - 1 or 0.
- The number of possible numbers represented by a byte is captured by the following formula
- 00000001 = $0^7+0^6+0^5+0^4+0^3+0^2+0^1+2^0 = 1$

Bytes

- A grouping of bits - usually 8 bits.
- Why 8? Because this was the minimum number of bits required to represent all ascii characters.



Decimal:	ArtNet:	Hex:	ASCII:
0	0:0	00	NUL
1	0:1	01	SOH
2	0:2	02	STX
3	0:3	03	ETX
4	0:4	04	EOT
5	0:5	05	ENQ
6	0:6	06	ACK
7	0:7	07	BEL
8	0:8	08	BS
9	0:9	09	TAB
10	0:A	0A	LF
11	0:B	0B	VT
12	0:C	0C	FF
13	0:D	0D	CR
14	0:E	0E	SO
15	0:F	0F	SI
16	1:0	10	DLE
17	1:1	11	DC1
18	1:2	12	DC2
19	1:3	13	DC3
20	1:4	14	DC4
21	1:5	15	NAK
22	1:6	16	SYN
23	1:7	17	ETB
24	1:8	18	CAN
25	1:9	19	EM
26	1:A	1A	SUB
27	1:B	1B	ESC
28	1:C	1C	FS
29	1:D	1D	GS
30	1:E	1E	RS
31	1:F	1F	US
32	2:0	20	[space]
33	2:1	21	!
34	2:2	22	"
35	2:3	23	#
36	2:4	24	$
37	2:5	25	%
38	2:6	26	&
39	2:7	27	'
40	2:8	28	(
41	2:9	29	)
42	2:A	2A	*
43	2:B	2B	+
44	2:C	2C	,
45	2:D	2D	-
46	2:E	2E	.
47	2:F	2F	/
48	3:0	30	0
49	3:1	31	1
50	3:2	32	2
51	3:3	33	3
52	3:4	34	4
53	3:5	35	5
54	3:6	36	6
55	3:7	37	7
56	3:8	38	8
57	3:9	39	9
58	3:A	3A	:
59	3:B	3B	;
60	3:C	3C	<
61	3:D	3D	=
62	3:E	3E	>
63	3:F	3F	?
64	4:0	40	@
65	4:1	41	A
66	4:2	42	B
67	4:3	43	C
68	4:4	44	D
69	4:5	45	E
70	4:6	46	F
71	4:7	47	G
72	4:8	48	H
73	4:9	49	I
74	4:A	4A	J
75	4:B	4B	K
76	4:C	4C	L
77	4:D	4D	M
78	4:E	4E	N
79	4:F	4F	O
80	5:0	50	P
81	5:1	51	Q
82	5:2	52	R
83	5:3	53	S
84	5:4	54	T
85	5:5	55	U
86	5:6	56	V
87	5:7	57	W
88	5:8	58	X
89	5:9	59	Y
90	5:A	5A	Z
91	5:B	5B	[
92	5:C	5C	\
93	5:D	5D	]
94	5:E	5E	^
95	5:F	5F	_
96	6:0	60	`
97	6:1	61	a
98	6:2	62	b
99	6:3	63	c
100	6:4	64	d
101	6:5	65	e
102	6:6	66	f
103	6:7	67	g
104	6:8	68	h
105	6:9	69	i
106	6:A	6A	j
107	6:B	6B	k
108	6:C	6C	l
109	6:D	6D	m
110	6:E	6E	n
111	6:F	6F	o
112	7:0	70	p
113	7:1	71	q
114	7:2	72	r
115	7:3	73	s
116	7:4	74	t
117	7:5	75	u
118	7:6	76	v
119	7:7	77	w
120	7:8	78	x
121	7:9	79	y
122	7:A	7A	z
123	7:B	7B	{
124	7:C	7C	|
125	7:D	7D	}
126	7:E	7E	~
127	7:F	7F	DEL
128	8:0	80	€
129	8:1	81	
130	8:2	82	‚
131	8:3	83	ƒ
132	8:4	84	„
133	8:5	85	
134	8:6	86	†
135	8:7	87	‡
136	8:8	88	ˆ
137	8:9	89	‰
138	8:A	8A	Š
139	8:B	8B	‹
140	8:C	8C	Œ
141	8:D	8D	
142	8:E	8E	Ž
143	8:F	8F	
144	9:0	90	
145	9:1	91	‘
146	9:2	92	’
147	9:3	93	“
148	9:4	94	”
149	9:5	95	•
150	9:6	96	–
151	9:7	97	
152	9:8	98	˜
153	9:9	99	™
154	9:A	9A	š
155	9:B	9B	›
156	9:C	9C	œ
157	9:D	9D	
158	9:E	9E	ž
159	9:F	9F	
160	A:0	A0	
161	A:1	A1	¡
162	A:2	A2	¢
163	A:3	A3	£
164	A:4	A4	¤
165	A:5	A5	¥
166	A:6	A6	¦
167	A:7	A7	§
168	A:8	A8	
169	A:9	A9	©
170	A:A	AA	ª
171	A:B	AB	«
172	A:C	AC	¬
173	A:D	AD	
174	A:E	AE	®
175	A:F	AF	
176	B:0	B0	°
177	B:1	B1	±
178	B:2	B2	²
179	B:3	B3	³
180	B:4	B4	´
181	B:5	B5	µ
182	B:6	B6	¶
183	B:7	B7	
184	B:8	B8	¸
185	B:9	B9	¹
186	B:A	BA	º
187	B:B	BB	»
188	B:C	BC	¼
189	B:D	BD	½
190	B:E	BE	¾
191	B:F	BF	
192	C:0	C0	À
193	C:1	C1	Á
194	C:2	C2	Â
195	C:3	C3	Ã
196	C:4	C4	Ä
197	C:5	C5	Å
198	C:6	C6	Æ
199	C:7	C7	Ç
200	C:8	C8	È
201	C:9	C9	É
202	C:A	CA	Ê
203	C:B	CB	Ë
204	C:C	CC	Ì
205	C:D	CD	Í
206	C:E	CE	Î
207	C:F	CF	Ï
208	D:0	D0	Ð
209	D:1	D1	Ñ
210	D:2	D2	Ò
211	D:3	D3	Ó
212	D:4	D4	Ô
213	D:5	D5	Õ
214	D:6	D6	Ö
215	D:7	D7	×
216	D:8	D8	Ø
217	D:9	D9	Ù
218	D:A	DA	Ú
219	D:B	DB	Û
220	D:C	DC	Ü
221	D:D	DD	Ý
222	D:E	DE	Þ
223	D:F	DF	
224	E:0	E0	à
225	E:1	E1	á
226	E:2	E2	â
227	E:3	E3	ã
228	E:4	E4	ä
229	E:5	E5	å
230	E:6	E6	æ
231	E:7	E7	ç
232	E:8	E8	è
233	E:9	E9	é
234	E:A	EA	ê
235	E:B	EB	ë
236	E:C	EC	ì
237	E:D	ED	í
238	E:E	EE	î
239	E:F	EF	ï
240	F:0	F0	ð
241	F:1	F1	ñ
242	F:2	F2	ò
243	F:3	F3	ó
244	F:4	F4	ô
245	F:5	F5	õ
246	F:6	F6	ö
247	F:7	F7	÷
248	F:8	F8	ø
249	F:9	F9	ù
250	F:A	FA	ú
251	F:B	FB	û
252	F:C	FC	ü
253	F:D	FD	ý
254	F:E	FE	þ
255	F:F	FF	

Bytes

- 11111111 = $2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0$ = 255 = 2^8
- An 8-bit microprocessor (computer) can resolve a number as big as 255.
- By analogy, 2^{64} ~ 1.8447e+19
- The actual biggest integer a 64-bit microprocessor can resolve is 9223372036854775807.

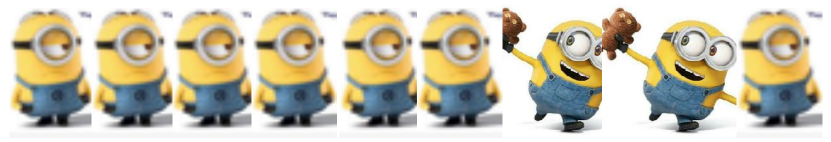
A cartoon version of bits

- 1 =

- 0 =

A cartoon version of bits

- 00010000 = 2^4 = 16



- 00000110 = $2^2 + 2^1$ = 6

A cartoon version of bits

http://www.wallpapermania.eu/wallpaper/millions-of-minions-despicable-me-2013/5120x3200

Bytes

- The previous slides explain how integers are stored. What about rational numbers?
- Rational numbers: This is done with scientific notation: $123 \times 10^{-1} = 12.3$.
- Rational number on 32-bit machine = 23 bits for significant figures + 1 bit for sign + 8 bits for exponent.
- 32-bit signed integer = (**0**0000000) (00000000) (00000000) (00000000)

  One integer keeps track of the sign
- Text: ASCII Table.
  - 01000001 = A (capital A).
  - 01011010 = Z (capital Z).

Binary arithmetic

If a computer only knows 1 or 0, how can it do complex math?

- All math operations can be broken down into a series of sums.
- Example: 7 + 2 = 00000111 + 0000010

$$\begin{array}{r} 7 = 00000111 \\ 2 = 00000010 \\ \hline 9 = 00001001 \end{array}$$

Binary arithmetic

If a computer only knows 1 or 0, how can it do complex math?

- All math operations can be broken down into a series of sums.
- What about?: 3 - 2 = 11000000 + (-) 01000000. Signed integer. A separate bit keeps track of the sign of the integer.
- What about?: 3 × 2 = 3 + 3.
- What about?: 3÷2 = 3 + (-2) + (-2) until the value goes negative.

Summary

- int - an integer number that computers can represent easily in binary.

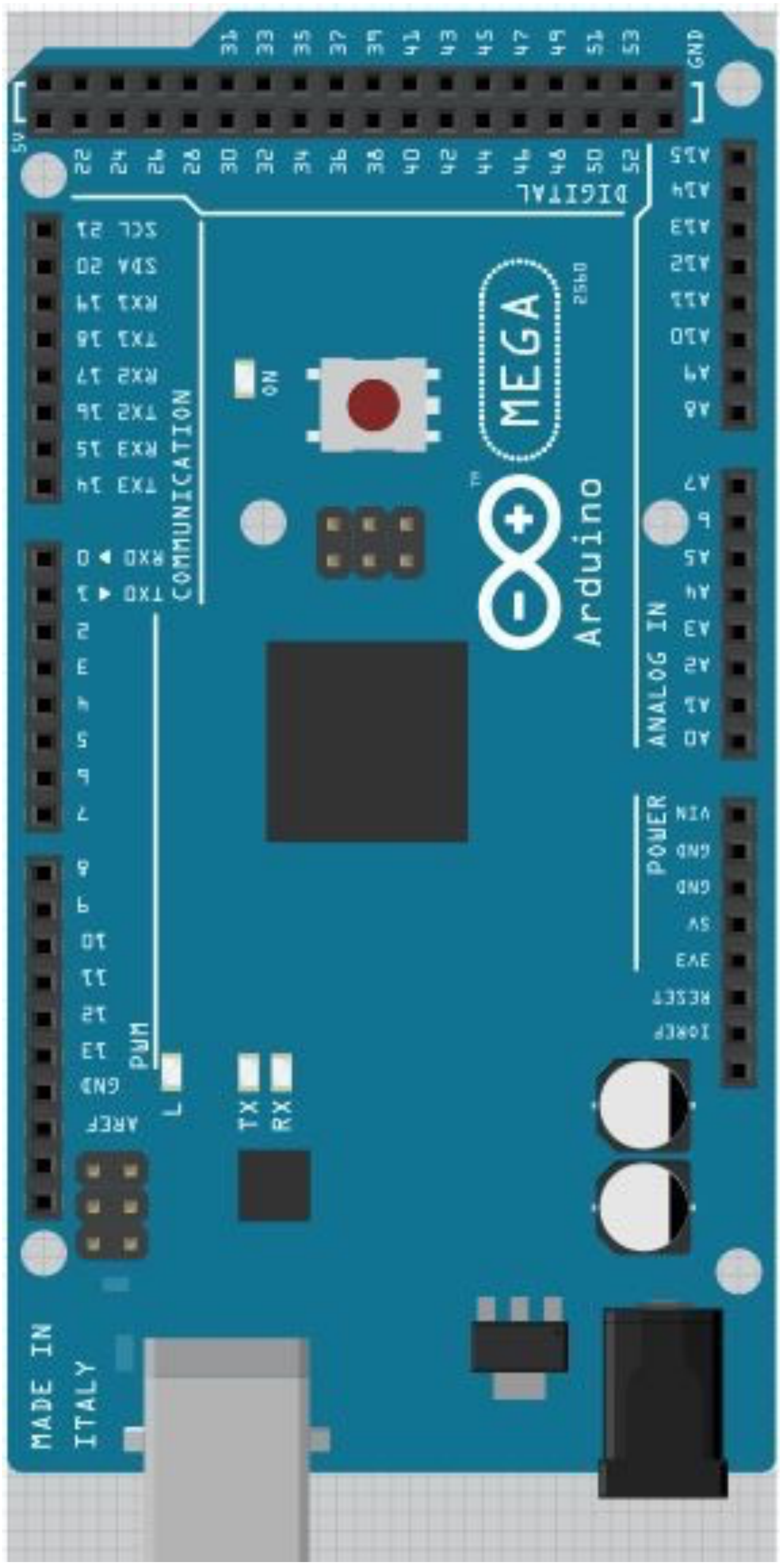
- floats - a rational number that computers can represent in binary using scientific notation and one bit for the sign.

- str - a table lookup for characters that can be represented by binary.

- arithmetic - bitwise addition. Everything else requires an algorithm

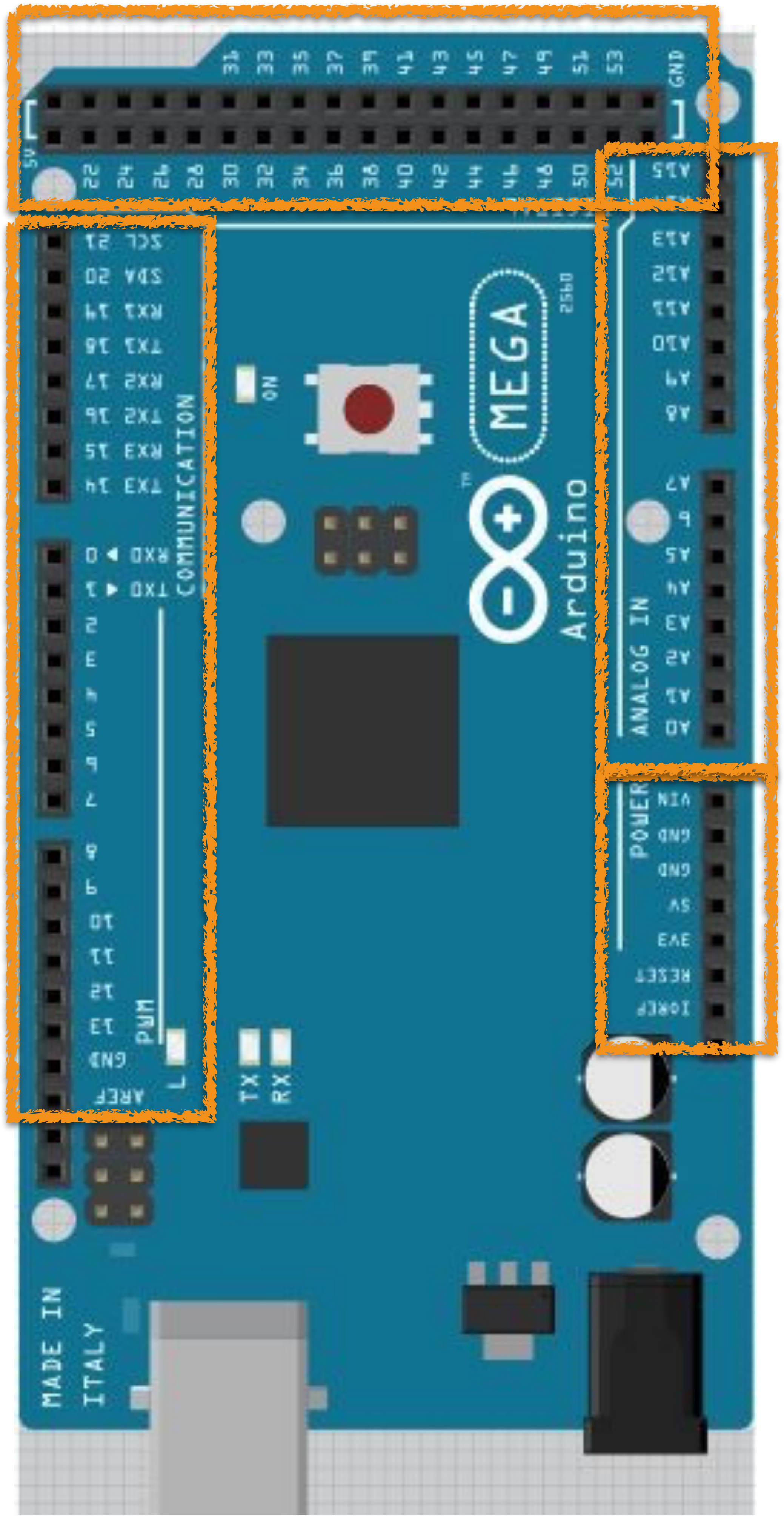
Microcontrollers

- **Arduino microprocessor:**
- **Microcontroller:** ATmega2560
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limit): 6-20V
- Digital I/O Pins: 54 (of which 15 provide PWM output)
- Analog Input Pins: 16
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 256 KB of which 8 KB used by bootloader
- SRAM: 8 KB
- EEPROM: 4 KB
- Clock Speed 16 MHz
- LED_BUILTIN: 13
- Length: 101.52 mm
- Width: 53.3 mm
- Weight: 37 g



Digital Pins In/Out:

Serial pins:
Reads UART, I2C, SPY



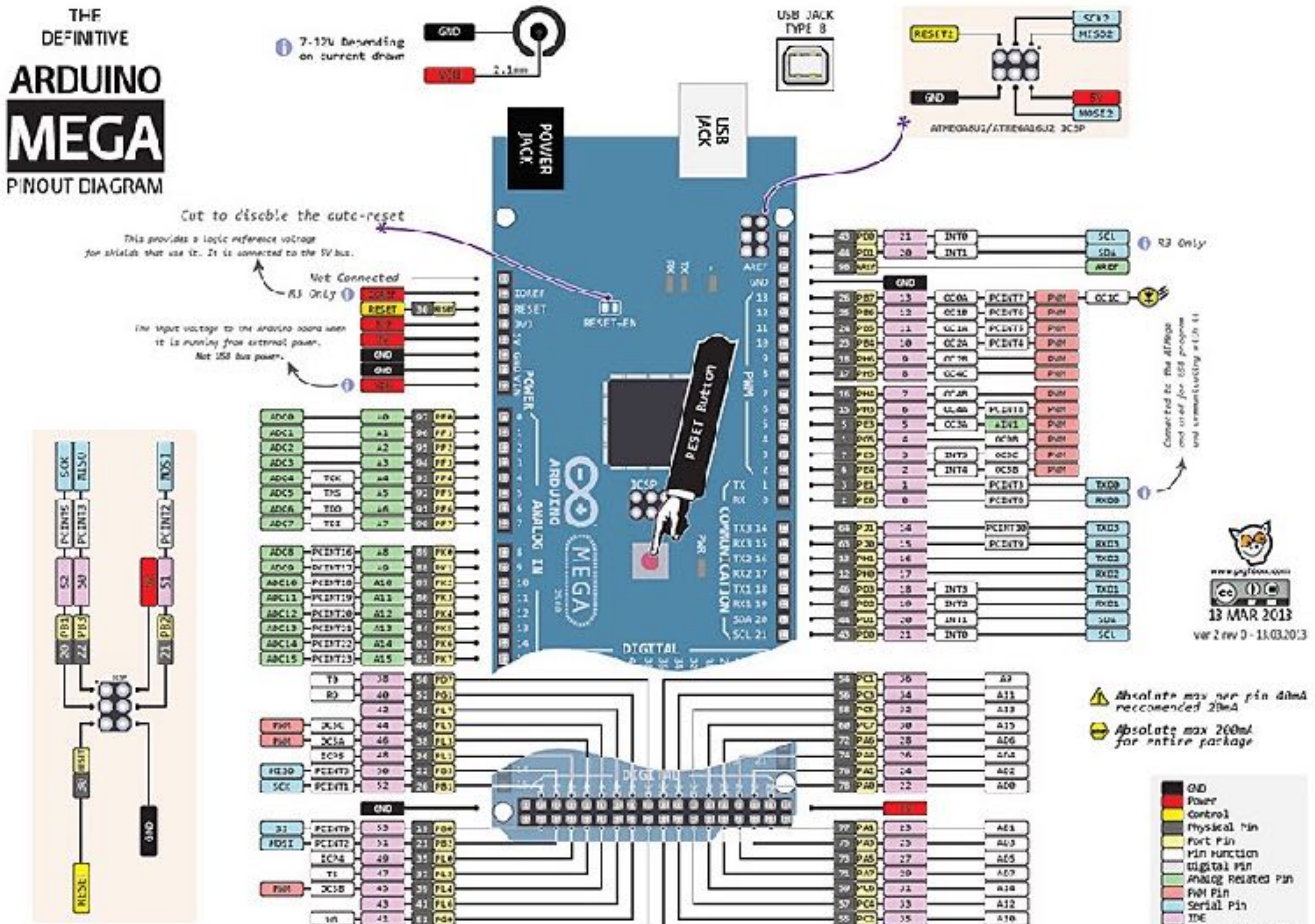
Analog inputs:

- Reads variable 0 to 5 V.
- Converts voltage to digital number

Power/Ground:

- Use to complete your circuit

Microcontrollers



Input/Output

Most Common forms of I/O:

- Analog Input: Read 0 to +5V and convert from voltage to engineering units.
- Analog Input: 4 to 20 mA and convert from current to engineering units.
- Digital Output: Hi/Lo to send a ‘yes’ or ‘no’ signal.
- Serial I/O: Data sent 1 bit at a time.
- There are others, but these are the most common.

Serial I/O

Benefits of Serial I/O:

- Cabling is less expensive.
- Easy to read.
- What uses serial? USB, Ethernet, Firewire, DV, coaxial.
- We will use serial called RS-232.

Analog Input

Analog to Digital Conversion:

- Microprocessor reads voltage.
- Microprocessor converts to an integer because this is what a computer stores - binary numbers.
- To analyze a circuit, we need to convert back to voltage:
- V_{sens} = V_{in}/Digital_scale.   Digital_scale depends on the bit-size of the microprocessor.
- **Arduino is 10-bit A to D microprocessor: 2^{10} = 1024 digital units (this is important for your code).**

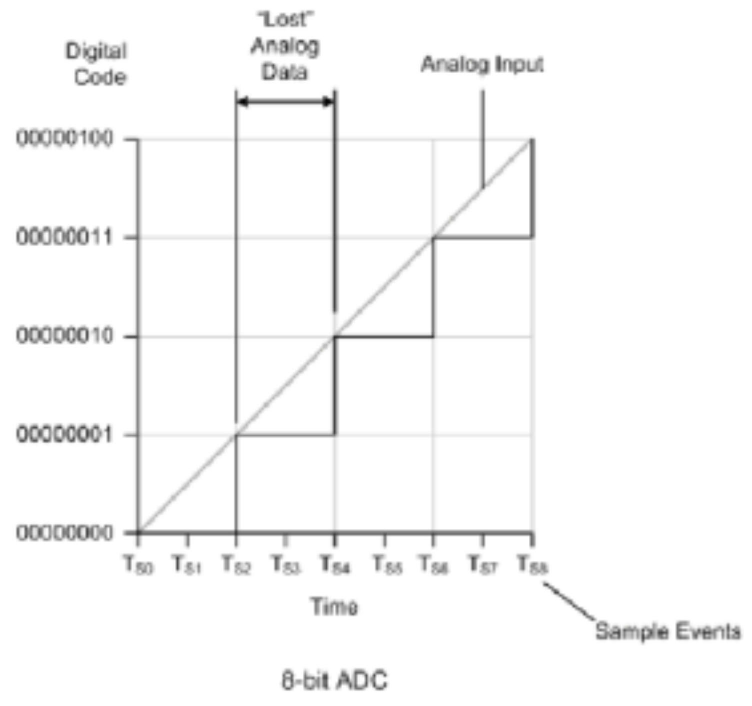
Analog Input

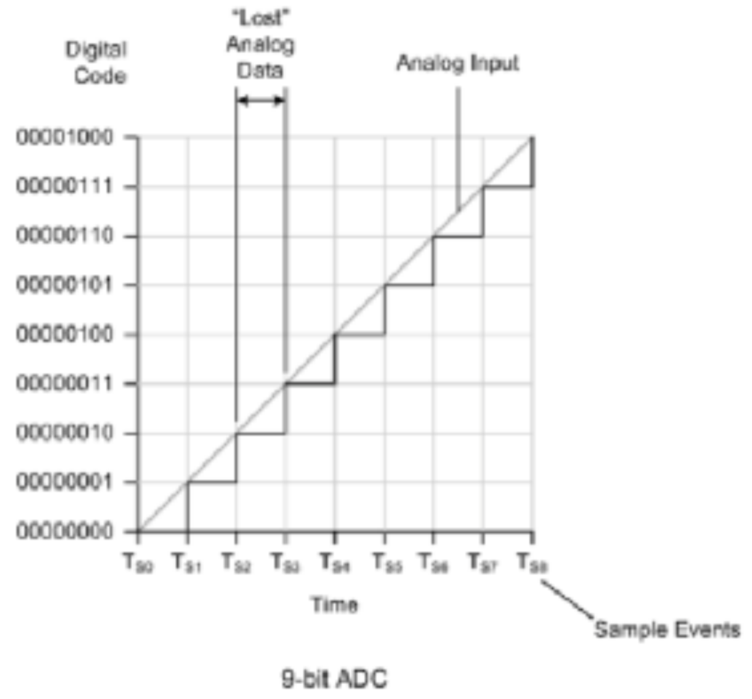
Analog to Digital Conversion:

- Resolution = $V_{in}/(2^n-1)$.

Example:

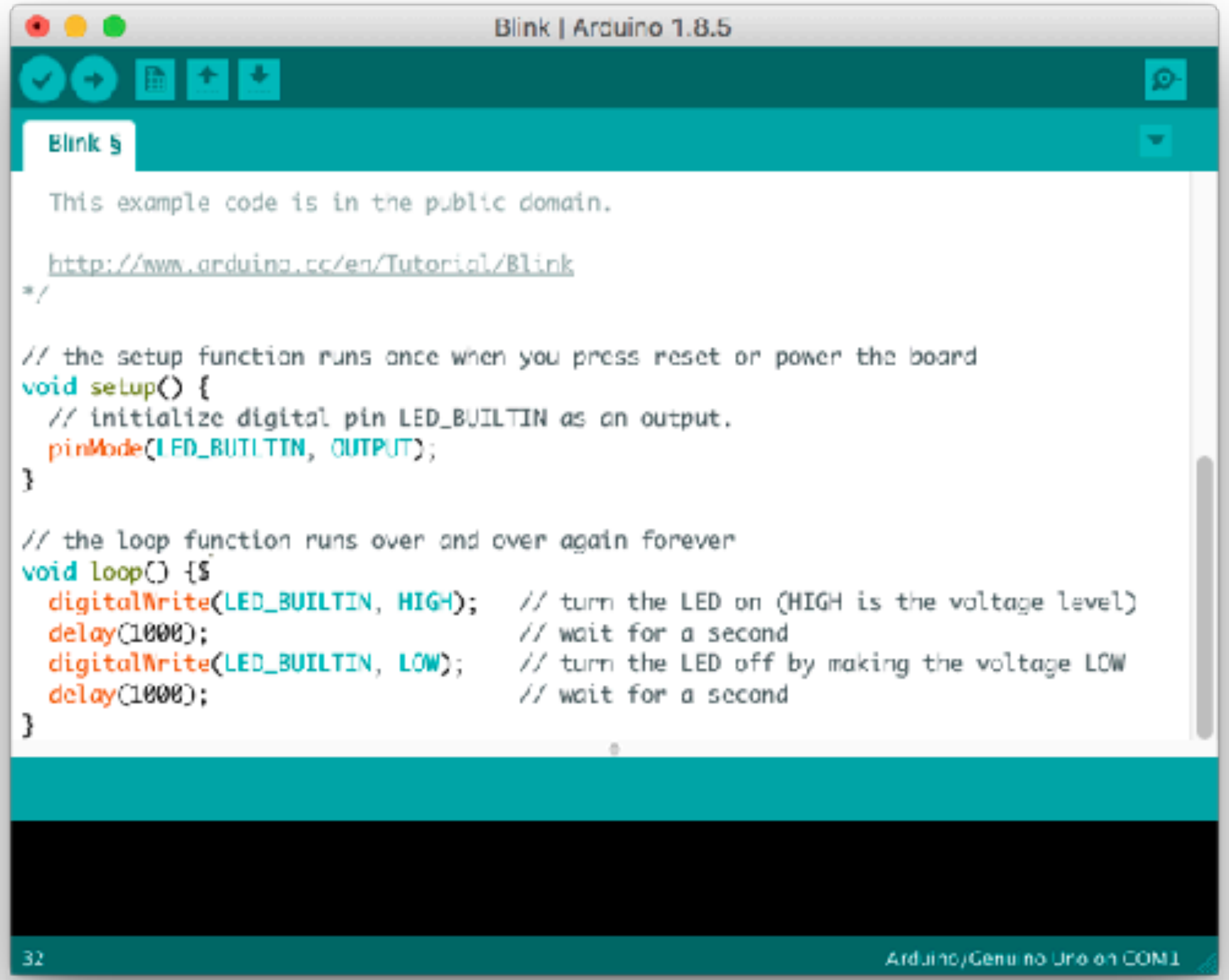
- We connect to Analog Input 4 (A4).
- A voltage of Vin = +5V is applied to the circuit.
- We read A4 and get 880. What does that tell us?
- $V_{forward}$ (at A4) = Vin*880/(2^n-1).





Arduino IDE

- https://www.arduino.cc/en/Main/software
- https://www.tinkercad.com/



```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {$
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```