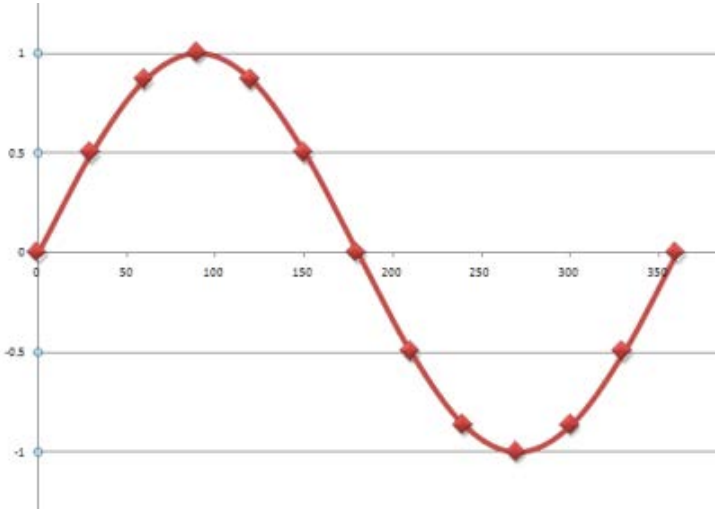
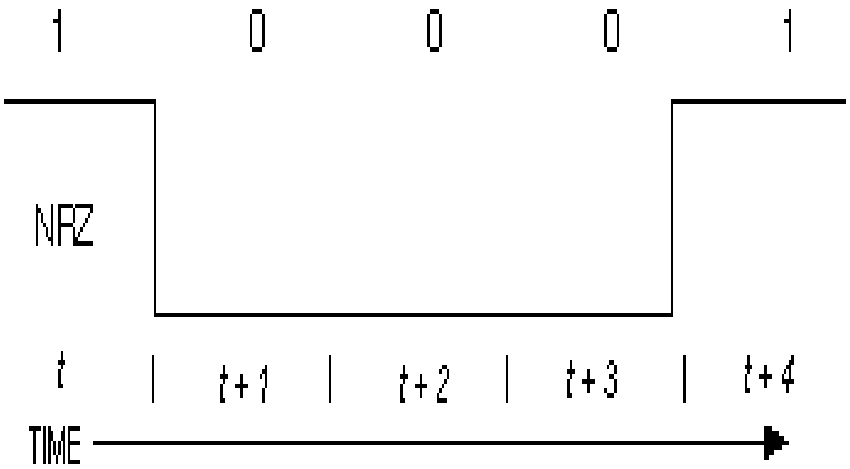
Inventing… 4

with Software and Electronics

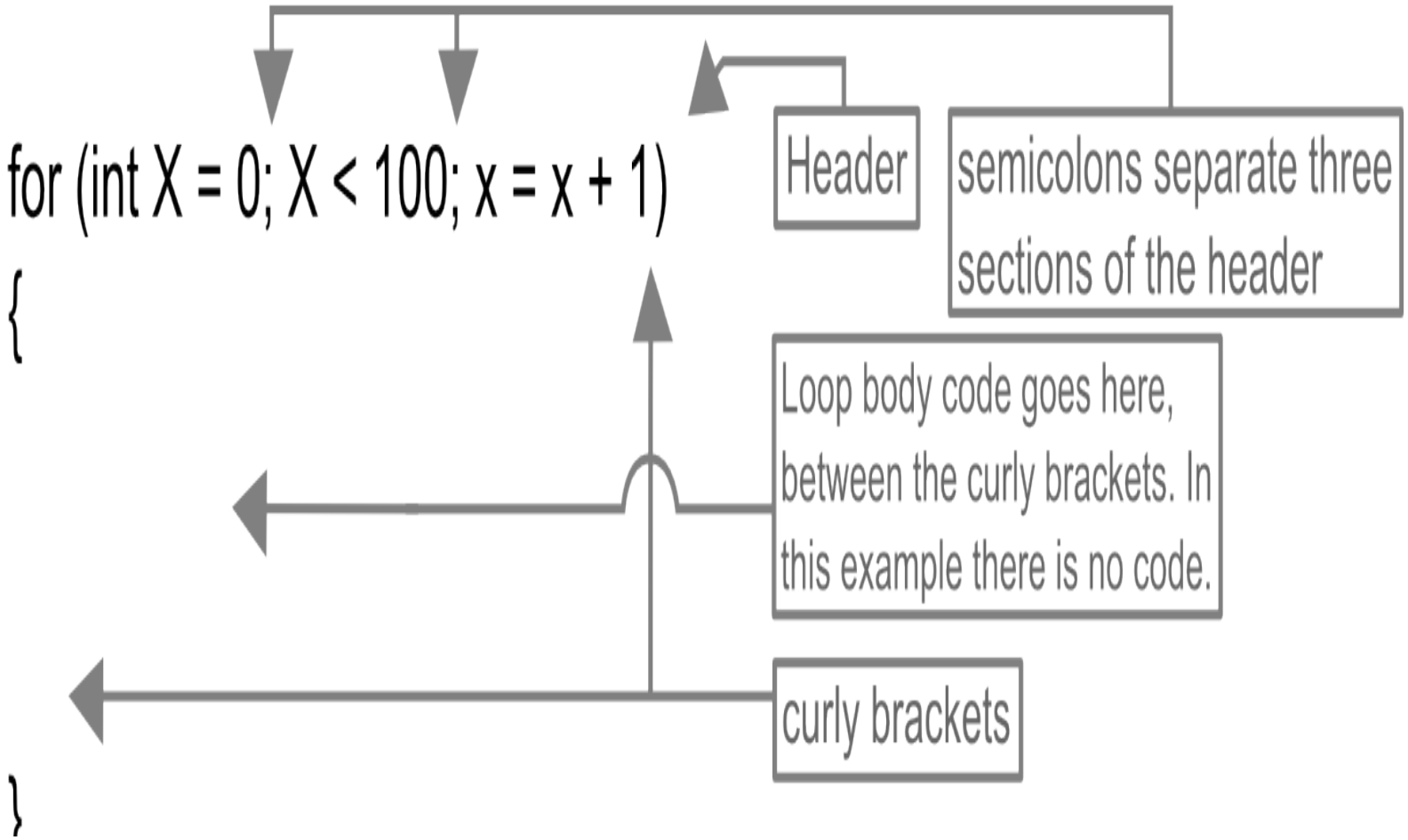
PWM


1
0.5
0
-0.5
-1
0
50
100
150
200
250
300
350


1
0
0
0
1
NRZ
t
t+1
t+2
t+3
t+4
TIME

Loops
&
Motors
&
Amps

(Oh, Boy!)


for (int X = 0; X < 100; x = x + 1)
{
}
Header
semicolons separate three sections of the header
Loop body code goes here, between the curly brackets. In this example there is no code.
curly brackets

For Loop

```
variableN = 0;

while (variableN < 10)      {

variableN = variableN + 1;

}
```

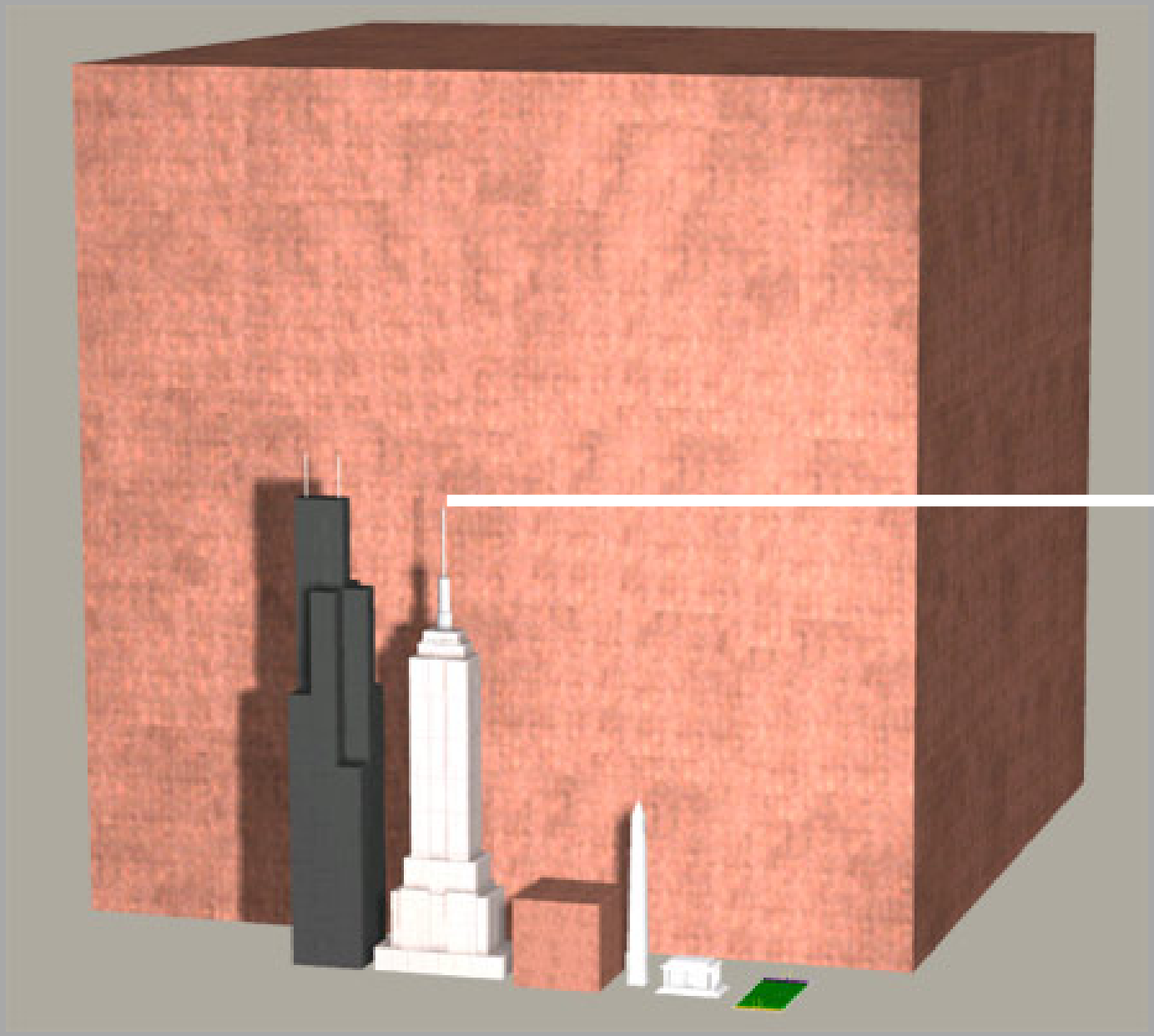
This is not a part of the while loop, it just sets variableN equal to zero before the while loop happens.

Header

Loop body code goes here, between the curly brackets. In this example it changes variableN so you're not stuck in the while loop forever.

curly brackets

The while loop

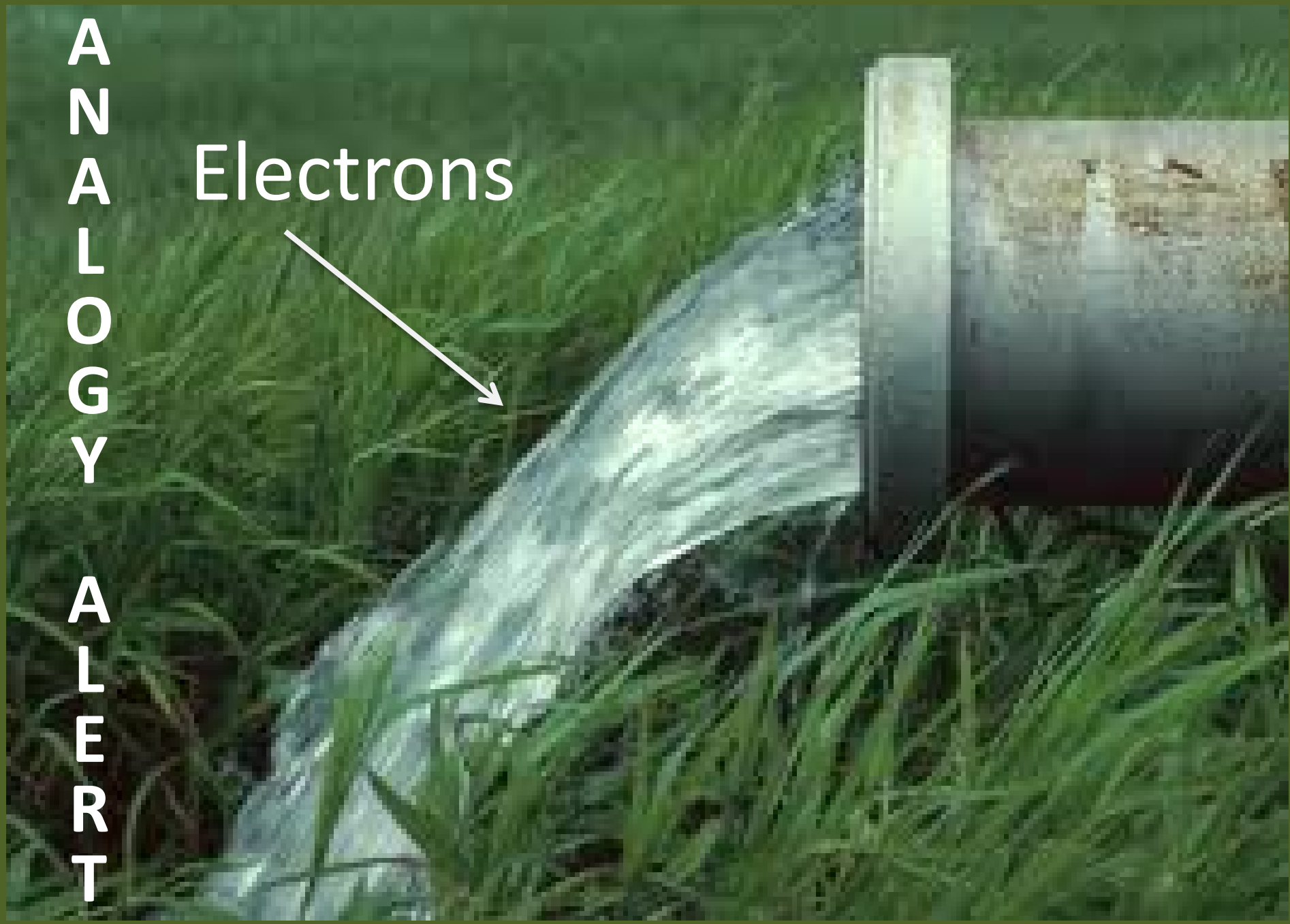


Empire State Building
102 Stories

Or – a pile of pennies
986,426,768 Miles
High….

ANALOGY ALERT

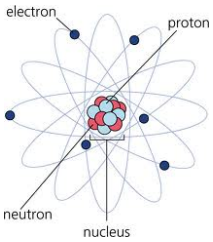
Conductor (Wire)


ANALOGY ALERT
Electrons

ANALOGY ALERT

Flow Rate (Gallons/Hour)

ANALOGY ALERT



Current (Amps)

Arduino Uno Datasheet

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)

Circuit Happiness

Dead

Stressed

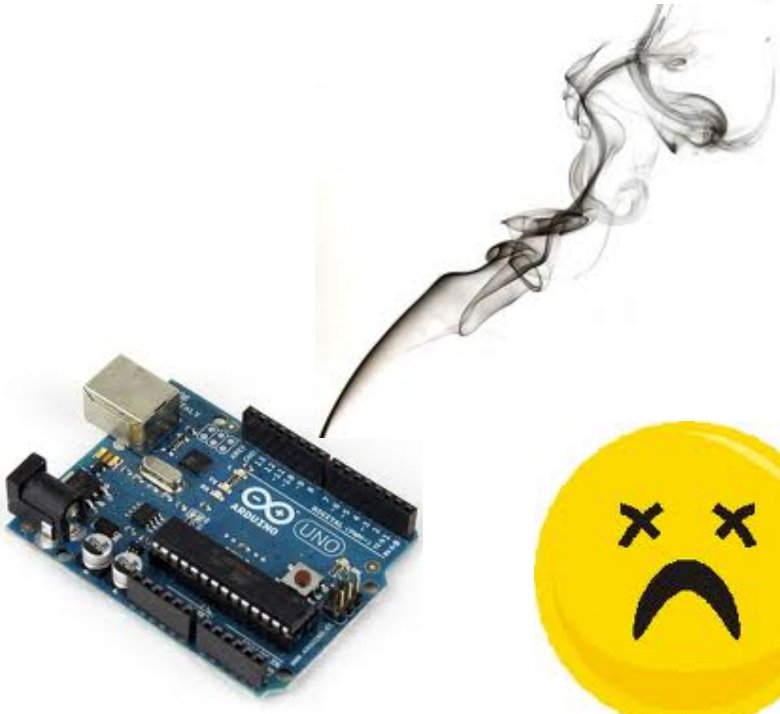
Healthy

Underpowered

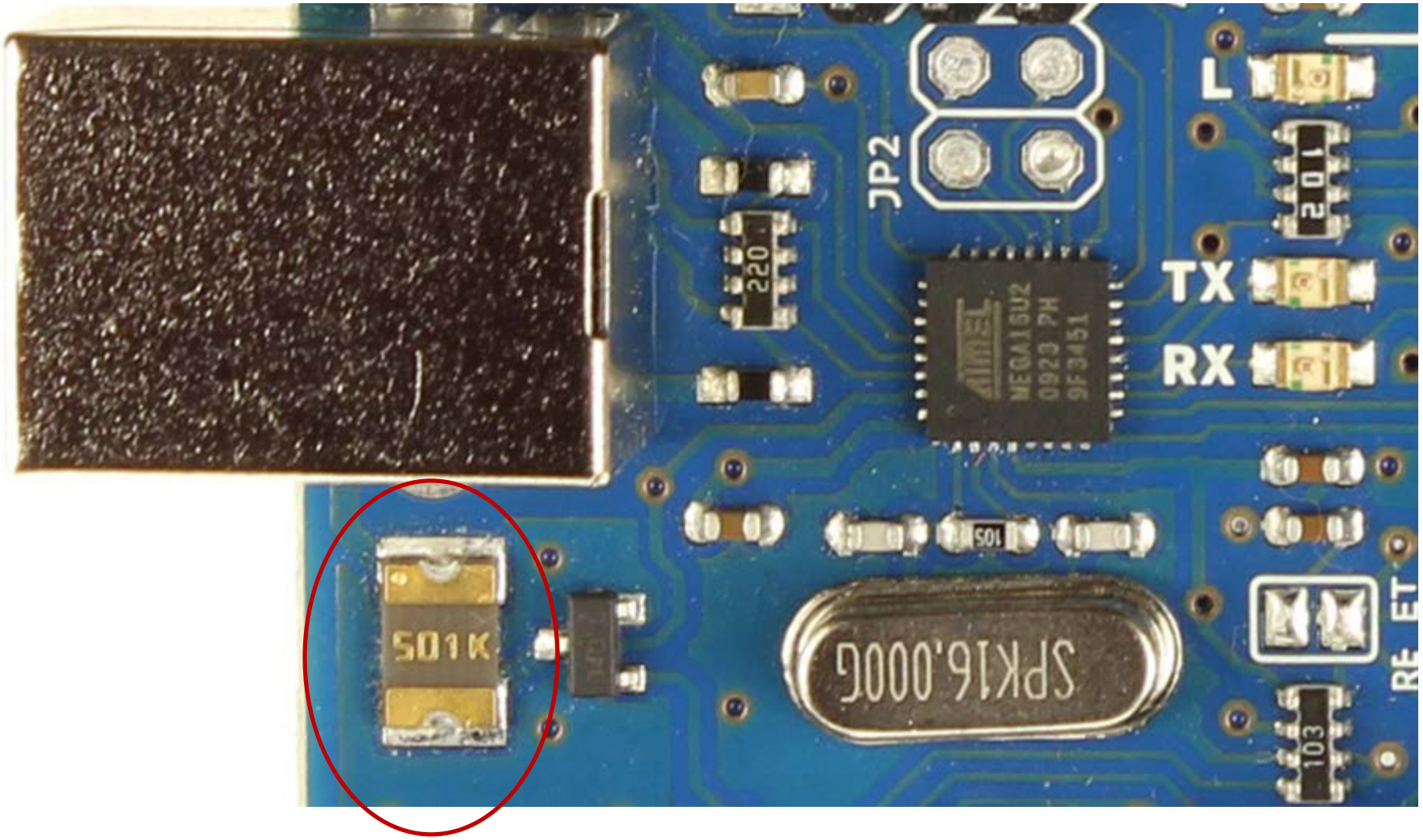
Asleep

ANALOGY ALERT


SMOKE ALARM


ARDUINO
UNO


119
119



Polyfuse (500mA)

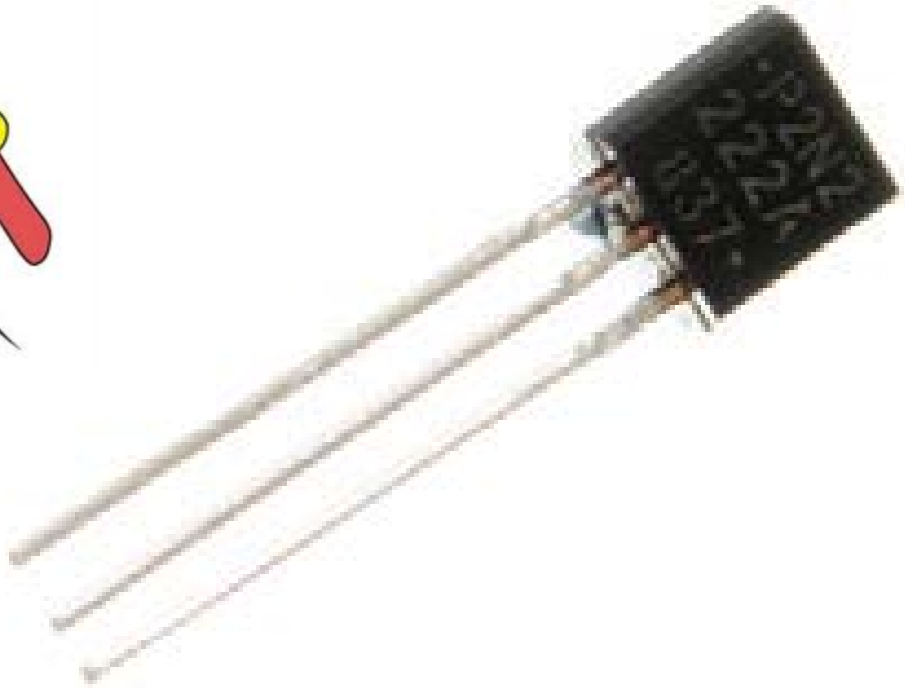
Ways to Kill an Arduino

Easily Possible

Shorting I/O Pins to Ground
Apply Overvoltage to I/O Pins
Shorting I/O Pins to Each Other
Exceed Total Microcontroller Current (200mA)

NO SMOKING

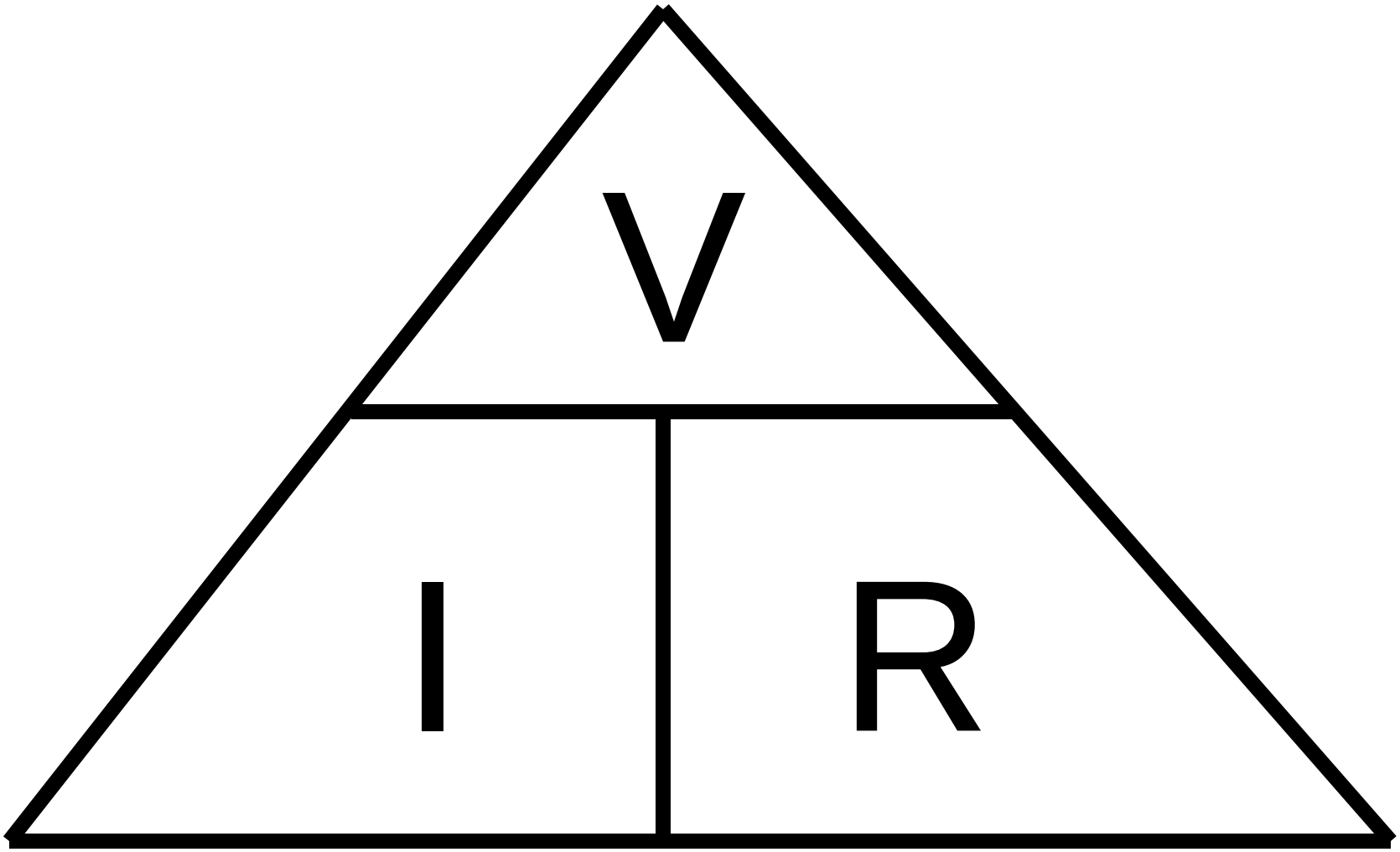
“the most important invention of the 20th century”



Transistor

ANALOGY ALERT

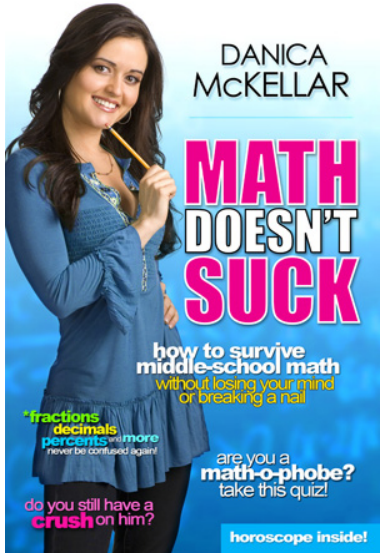
Valve

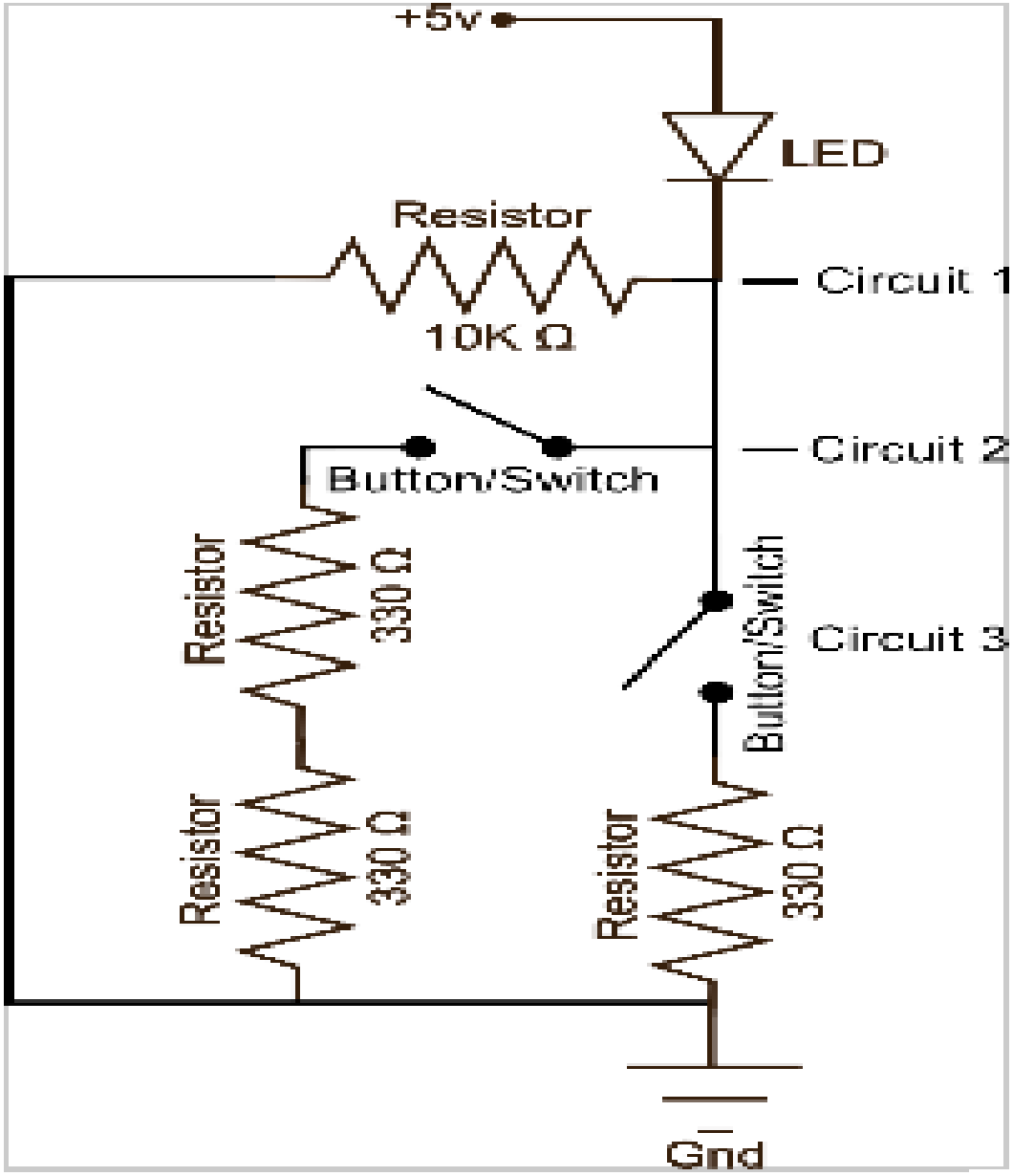

V
I
R

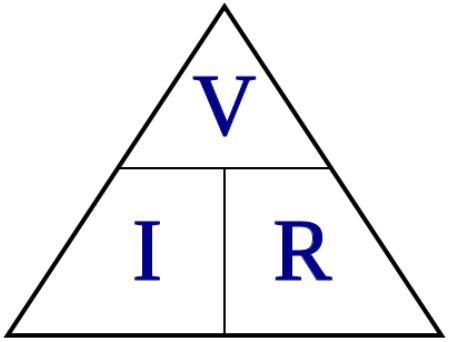
To calculate:

$$\frac{\text{VOLTS}}{\text{AMPS}} = \text{I} \times \text{R}$$

OHMS




+5v
LED
Resistor
10K Ω
Circuit 1
Circuit 2
Button/Switch
Resistor
330 Ω
Circuit 3
Button/Switch
Resistor
330 Ω
Resistor
330 Ω
Gnd


V
I
R

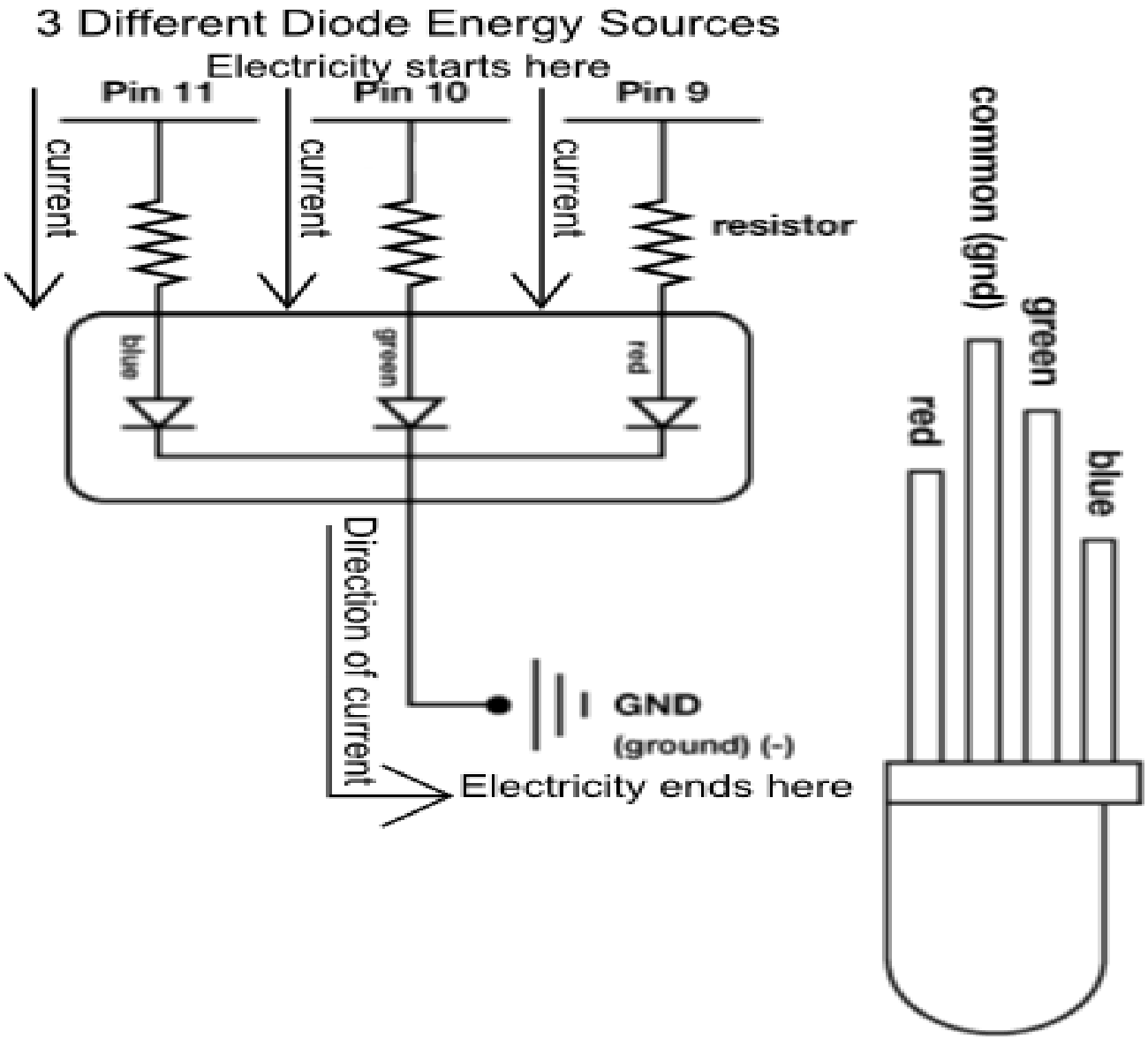
Piezo-Electric Buzzer

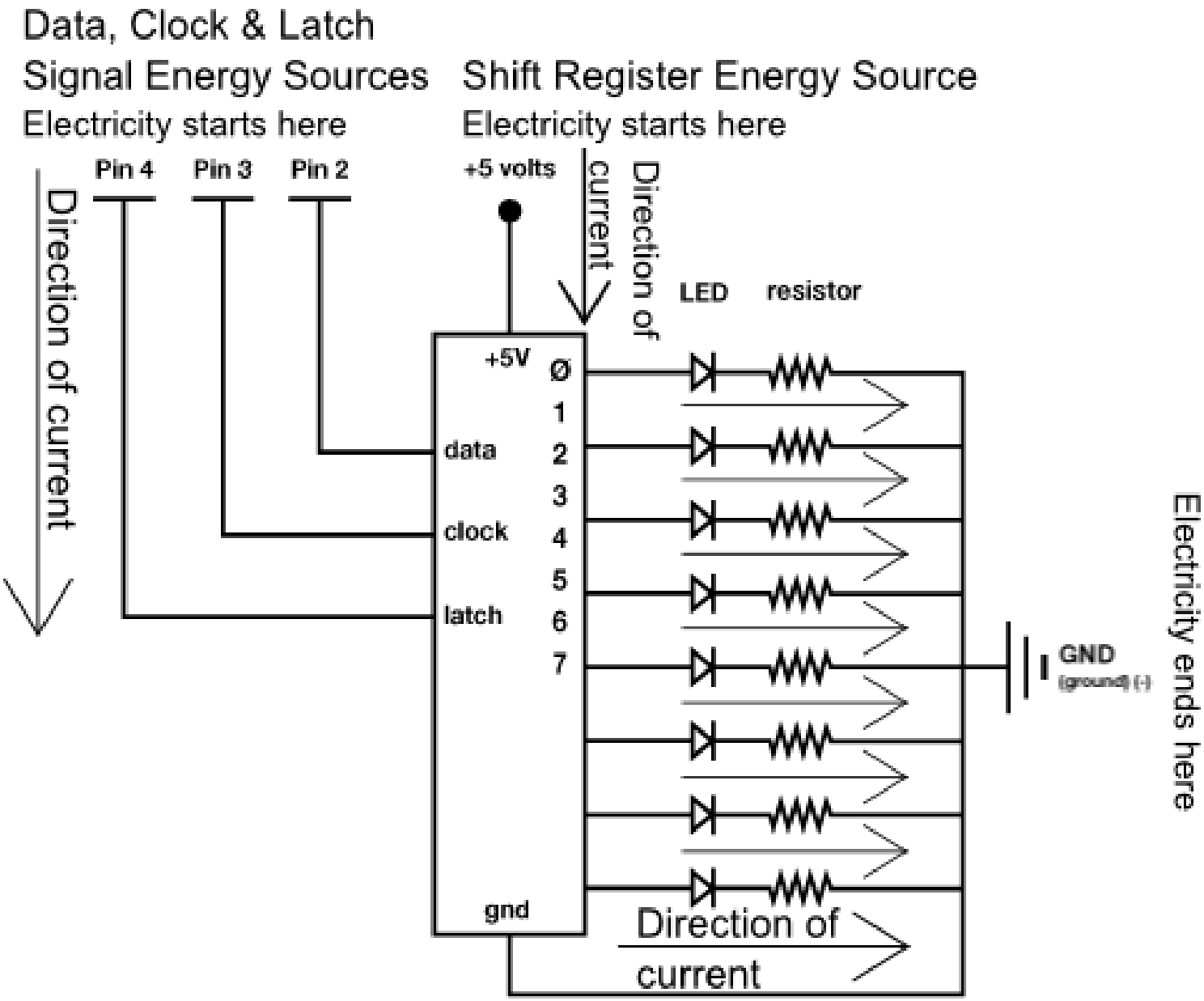
What it Does:

A pulse of current will cause it to click. A stream of pulses will cause it to emit a tone.

Identifying:

In this kit it comes in a little black barrel, but sometimes they are just a gold disc.


3 Different Diode Energy Sources
Electricity starts here
Pin 11
Pin 10
Pin 9
current
current
current
resistor
blue
green
red
Direction of current
GND
(ground) (-)
Electricity ends here
common (gnd)
green
red
blue


Data, Clock & Latch
Signal Energy Sources
Electricity starts here
Shift Register Energy Source
Electricity starts here
Pin 4
Pin 3
Pin 2
+5 volts
Direction of current
Direction of current
LED
resistor
+5V
data
clock
latch
gnd
Ø
1
2
3
4
5
6
7
GND
(ground) (-)
Direction of current
Electricity ends here

Did you finish building the Shift Register Circuit?

Data Types
&
Serial Communications
&
Why Buttons are more complicated than you might think….

Serial Communications


Cereal

Serial /**seer**-ee-*uhl*/: *adj.*

Computers

a. of or pertaining to the apparent or actual performance of data-processing operations ***one at a time*** (distinguished from parallel).

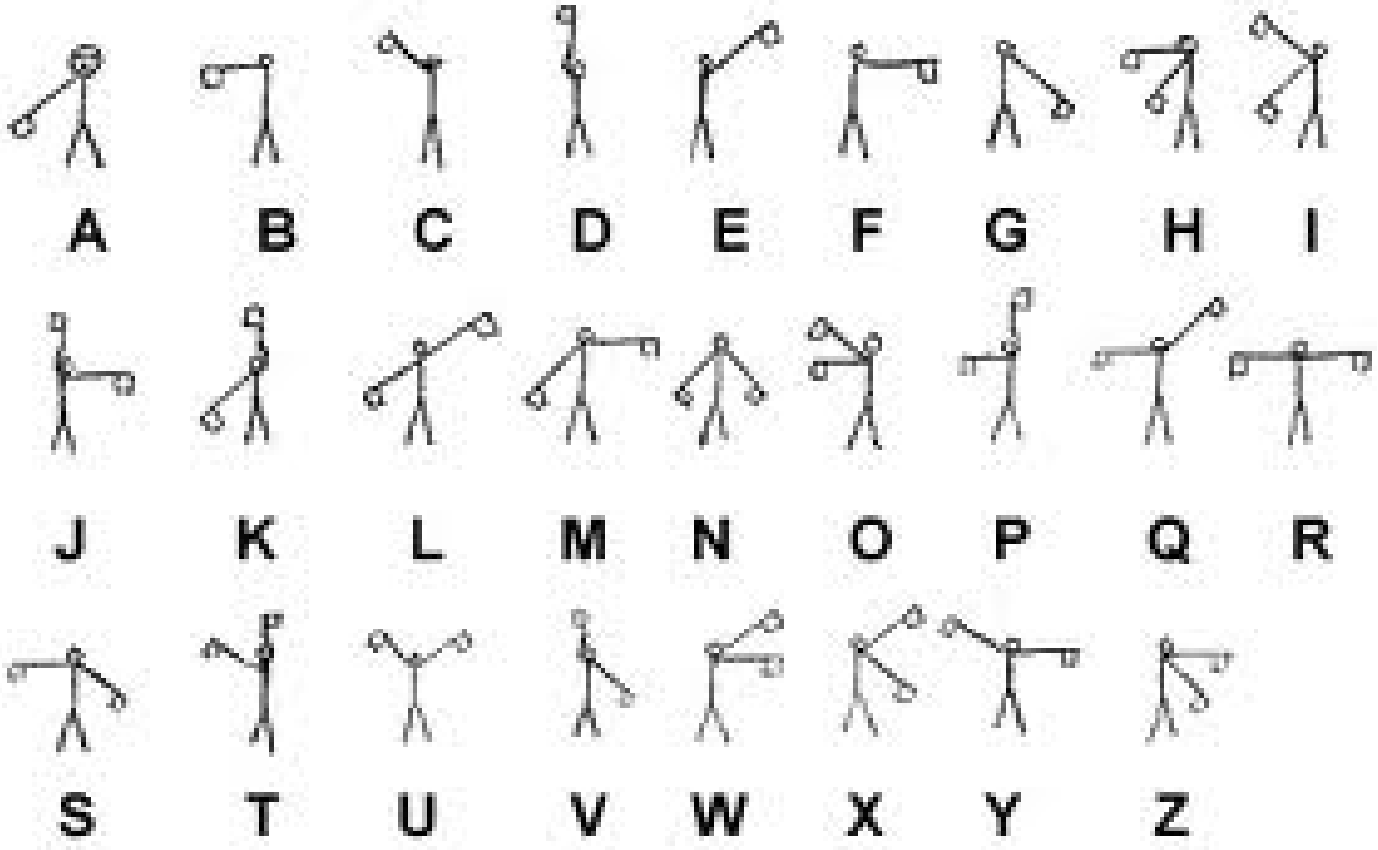
b. of or pertaining to the ***transmission*** or processing of each part of a whole in sequence, **as each bit of a byte** or each byte of a computer word (distinguished from parallel).

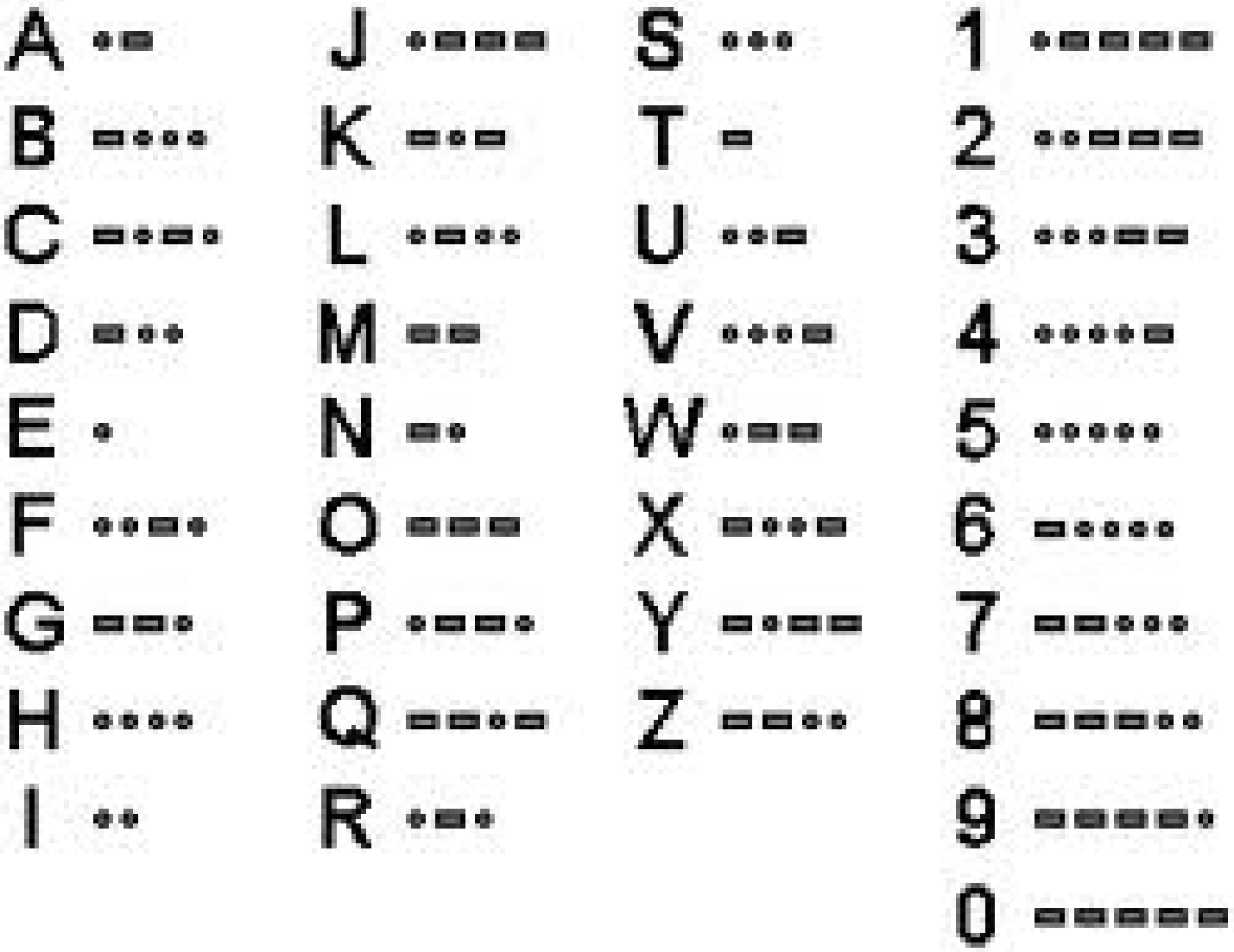
Serial Port (RS-232)

Serial Port (RS-232)

Universal ***Serial*** Bus (USB)

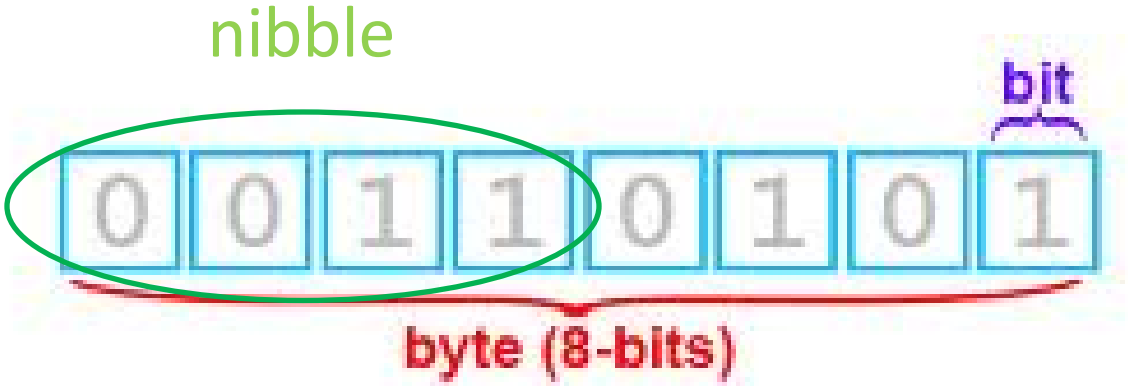

Ha! Look, he's spelt with a !
search ID: amus4
© Original Artist
Reproduction rights obtainable from
www.CartoonStock.com

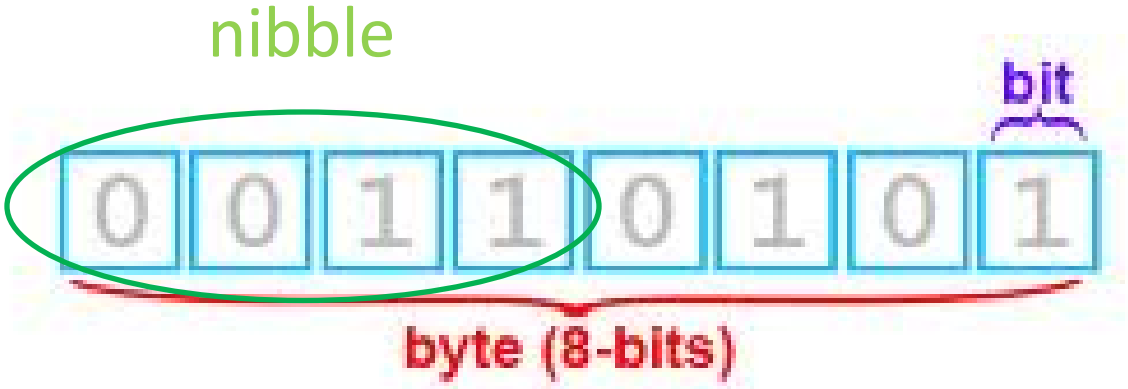

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z


A
J
S
1
B
K
T
2
C
L
U
3
D
M
V
4
E
N
W
5
F
O
X
6
G
P
Y
7
H
Q
Z
8
I
R
9
0


0 0 1 1 0 1 0 1
byte (8-bits)

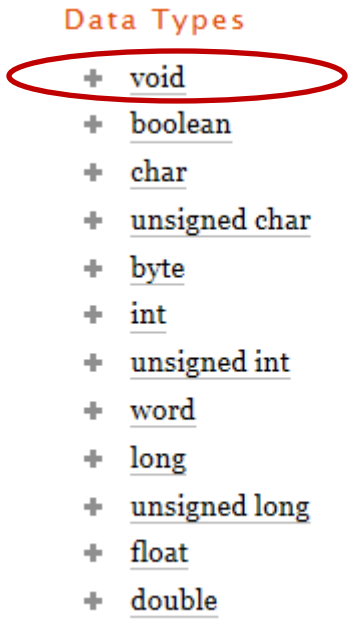

bit
0 0 1 1 0 1 0 1
byte (8-bits)

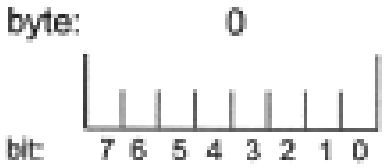

nibble
bit
0 0 1 1 0 1 0 1
byte (8-bits)


nibble
bit
0 0 1 1 0 1 0 1
byte (8-bits)

{TAKES NO SPACE}

{HAS NO VALUE}



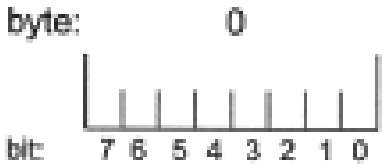


0 (false) or !0 (true)

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double

‘a’ = 0110 0001



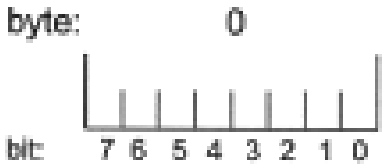
characters in ASCII (8- bits)

ASCII Code: Character to Binary

0	0011 0000	@	0100 0000	`	0110 0000
1	0011 0001	A	0100 0001	a	0110 0001
2	0011 0010	B	0100 0010	b	0110 0010
3	0011 0011	C	0100 0011	c	0110 0011
4	0011 0100	D	0100 0100	d	0110 0100
5	0011 0101	E	0100 0101	e	0110 0101
6	0011 0110	F	0100 0110	f	0110 0110
7	0011 0111	G	0100 0111	g	0110 0111
8	0011 1000	H	0100 1000	h	0110 1000
9	0011 1001	I	0100 1001	i	0110 1001
A	0100 0001	J	0100 1010	j	0110 1010
B	0100 0010	K	0100 1011	k	0110 1011
C	0100 0011	L	0100 1100	l	0110 1100
D	0100 0100	M	0100 1101	m	0110 1101
E	0100 0101	N	0100 1110	n	0110 1110
F	0100 0110	O	0100 1111	o	0110 1111
G	0100 0111	P	0101 0000	p	0111 0000
H	0100 1000	Q	0101 0001	q	0111 0001
I	0100 1001	R	0101 0010	r	0111 0010
J	0100 1010	S	0101 0011	s	0111 0011
K	0100 1011	T	0101 0100	t	0111 0100
L	0100 1100	U	0101 0101	u	0111 0101
M	0100 1101	V	0101 0110	v	0111 0110
N	0100 1110	W	0101 0111	w	0111 0111
		X	0101 1000	x	0111 1000
		Y	0101 1001	y	0111 1001
		Z	0101 1010	z	0111 1010
		[	0101 1011	{	0111 1011
		\	0101 1100	|	0111 1100
		]	0101 1101	}	0111 1101
		^	0101 1110	~	0111 1110
		_	0101 1111		
		`	0110 0000		
		a	0110 0001		
		b	0110 0010		
		c	0110 0011		
		d	0110 0100		
		e	0110 0101		
		f	0110 0110		
		g	0110 0111		
		h	0110 1000		
		i	0110 1001		
		j	0110 1010		
		k	0110 1011		
		l	0110 1100		
		m	0110 1101		
		n	0110 1110		
		o	0110 1111		
		p	0111 0000		
		q	0111 0001		
		r	0111 0010		
		s	0111 0011		
		t	0111 0100		
		u	0111 0101		
		v	0111 0110		
		w	0111 0111		
		x	0111 1000		
		y	0111 1001		
		z	0111 1010		
		{	0111 1011		
		|	0111 1100		
		}	0111 1101		
		~	0111 1110		
		space	0010 0000		

Data Types

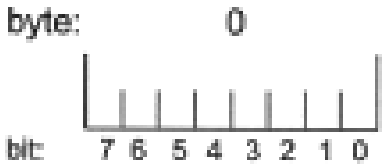
- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double



characters in ASCII (8- bits)
[Not very useful]

Data Types

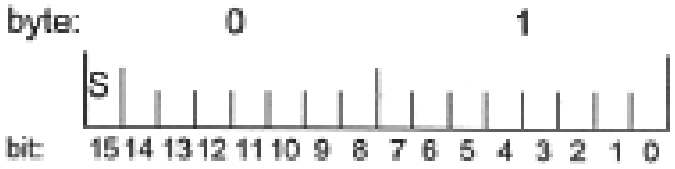
- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double



0 to 255

Data Types

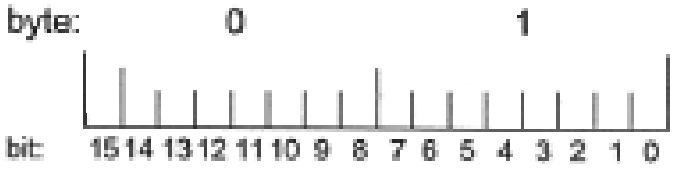
- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double



-32,767 to 32767

Data Types

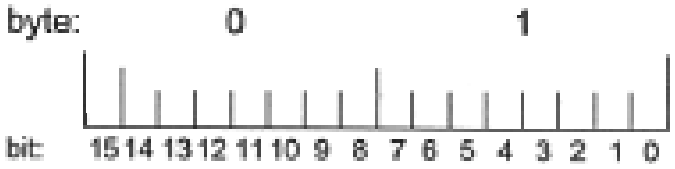
- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double



0 to 65535

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double



0 to 65535

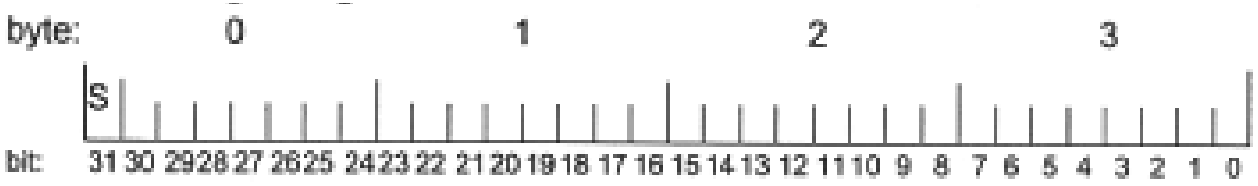
Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double

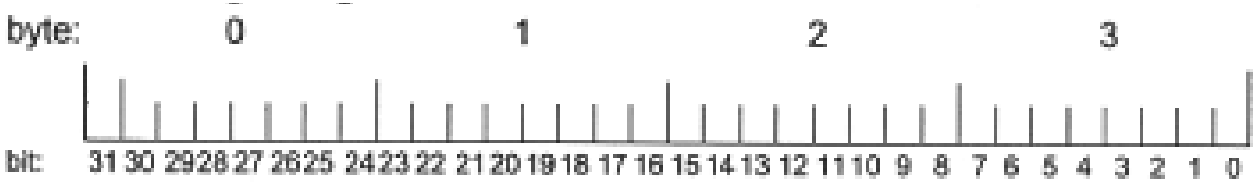
-2,147,483,647 to 2,147,483,647



Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double

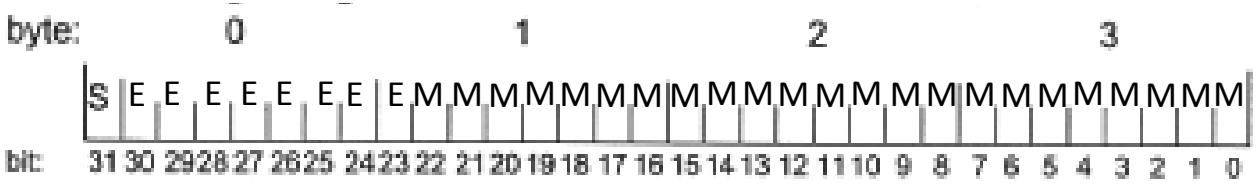
0 to 4,294,967,295



Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double

3.4E–38 to 3.4E+38

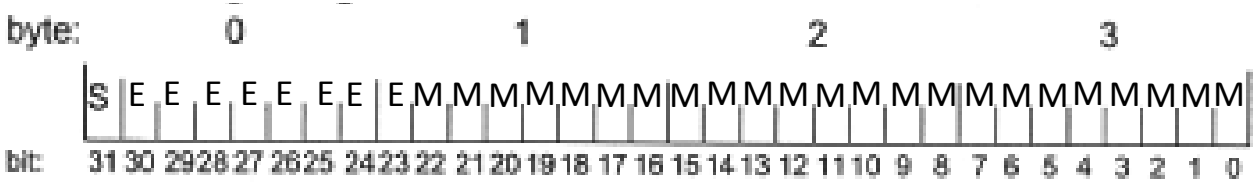


OOPS!

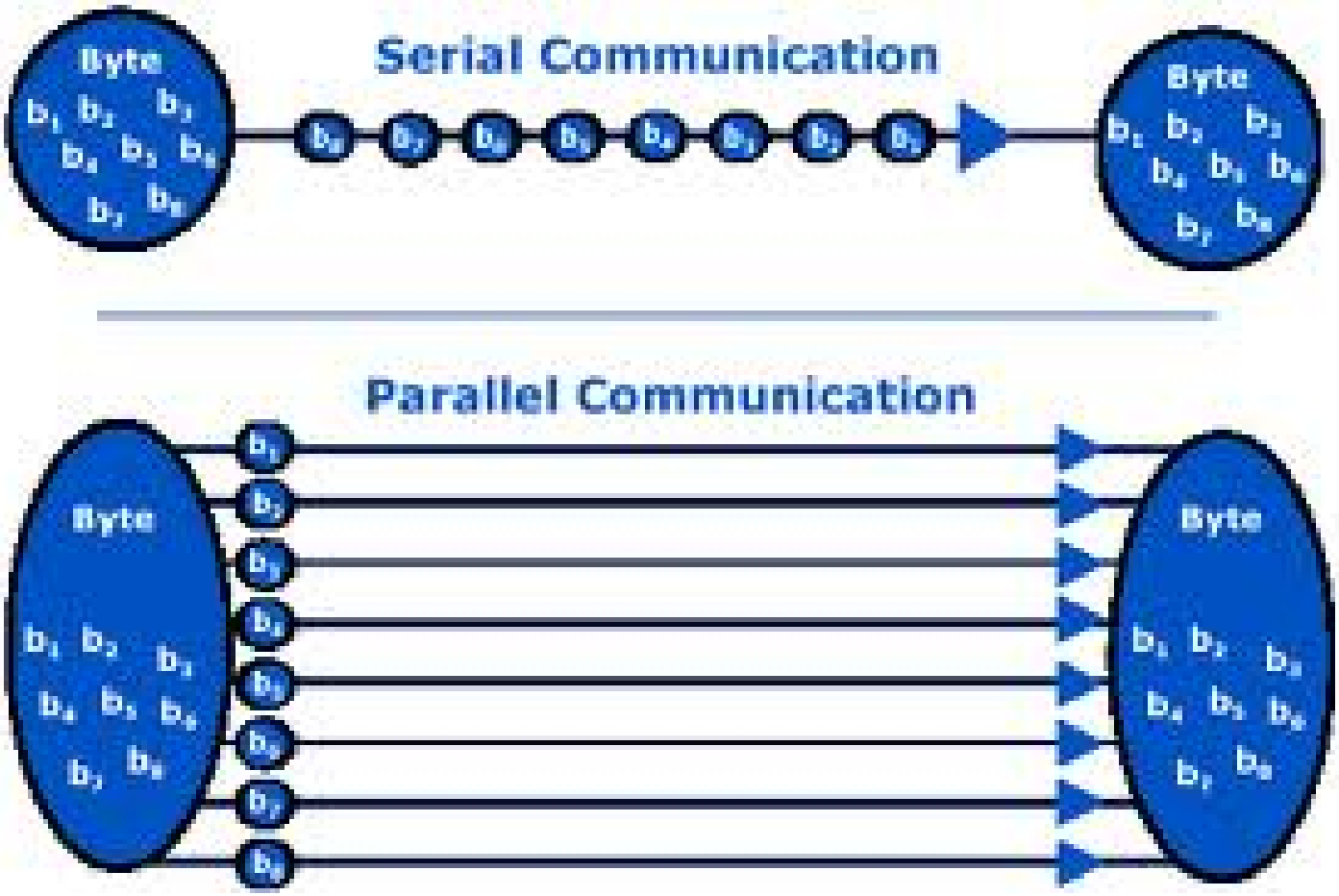
3.4E–38 to 3.4E+38

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double



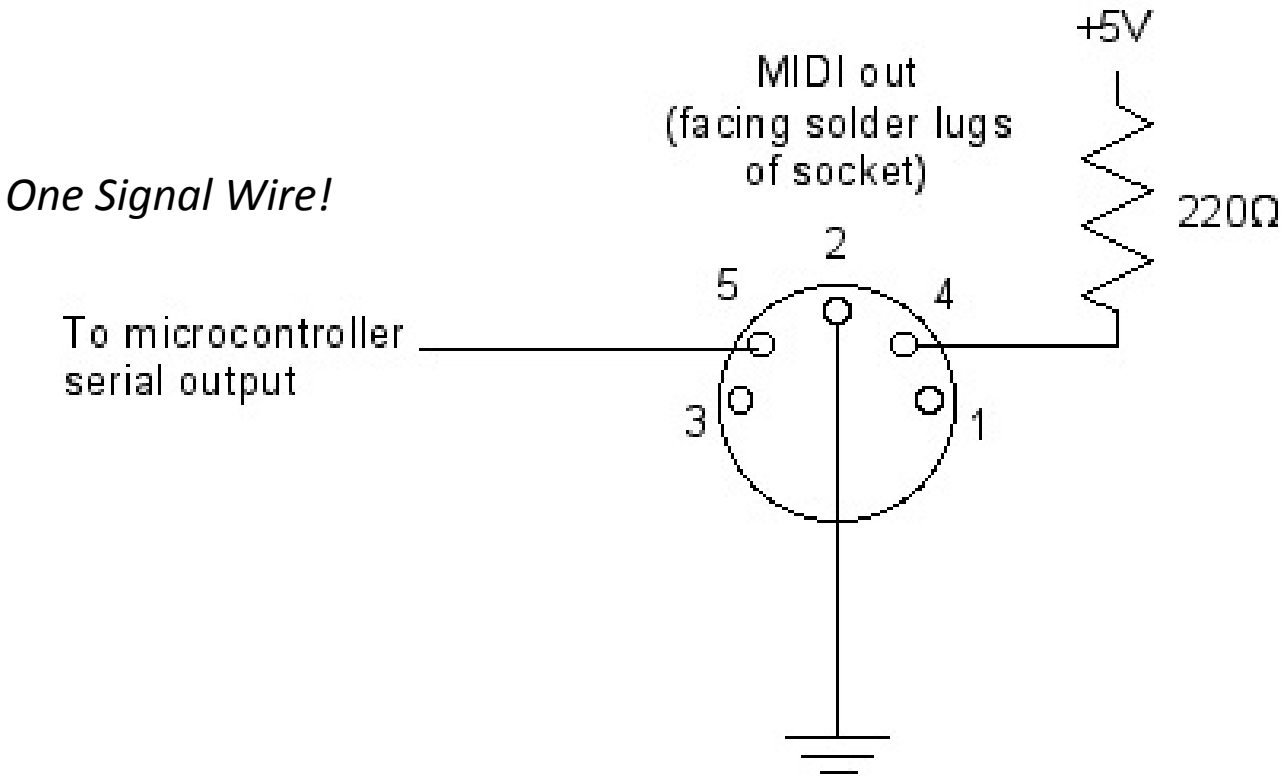

101010 010


Serial Communication
Byte
Byte
Parallel Communication
Byte
Byte

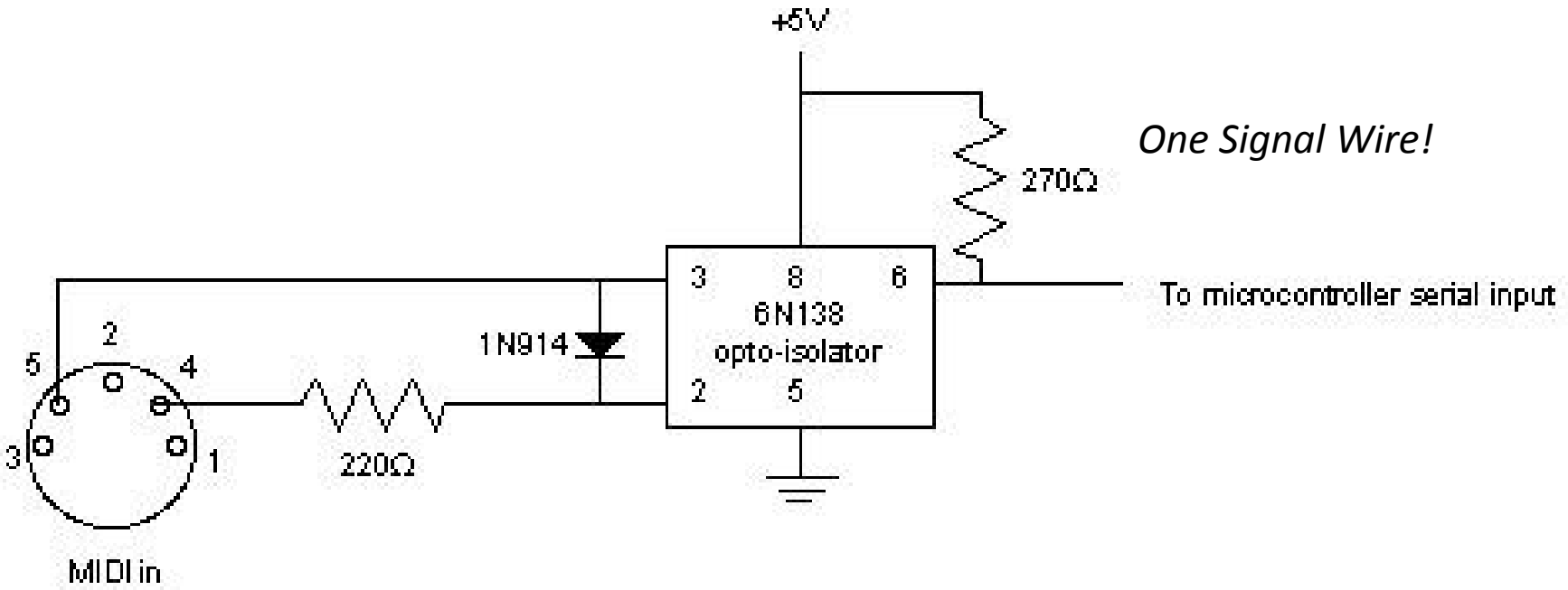
One Wire

MIDI

Kate *Charlie* *Jasmine*



Sending a MIDI message from a Synthesizer



Receiving a MIDI message to a Synthesizer

“play a middle C on the tenth MIDI channel, medium volume.”

“play a middle C on the tenth MIDI channel, medium volume.”

9A 45 45

“play a middle C on the tenth MIDI channel, medium volume.”

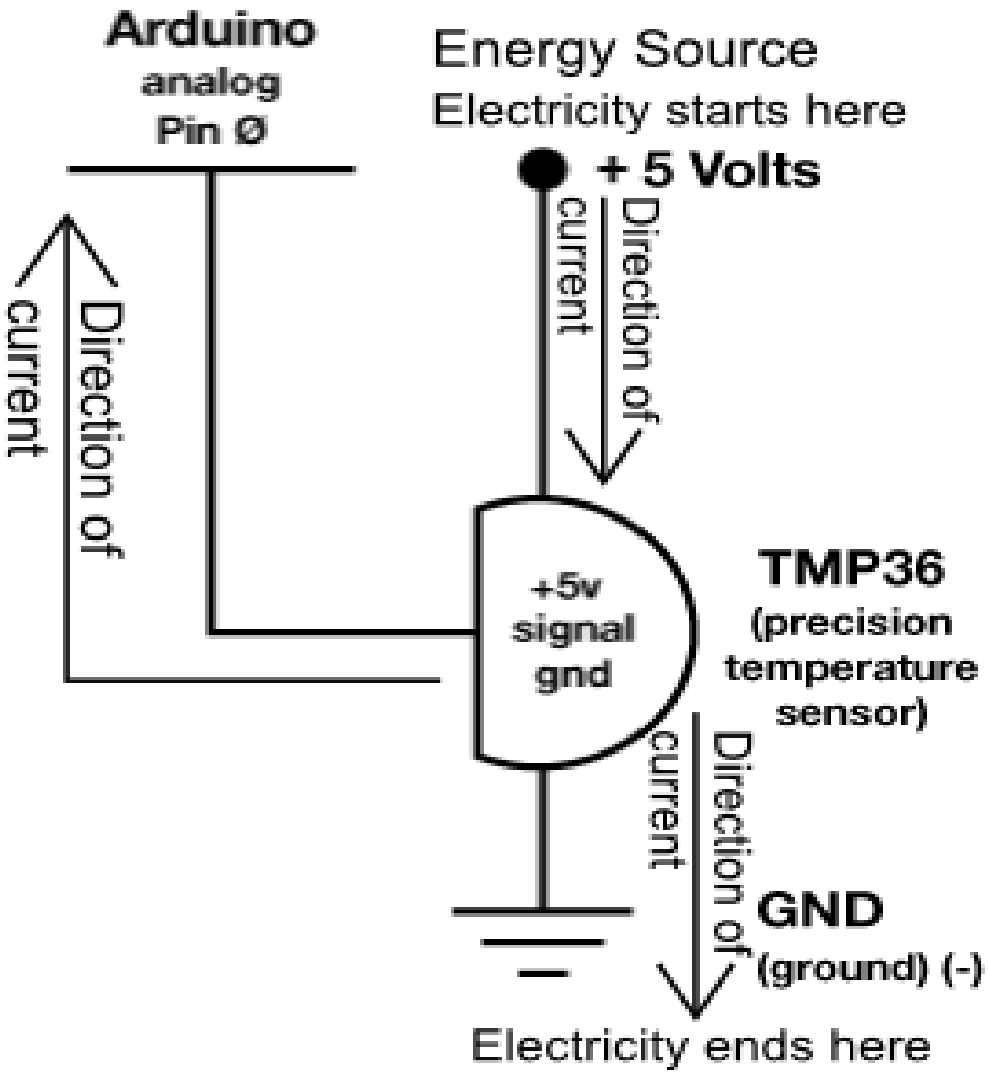
9A 45 45

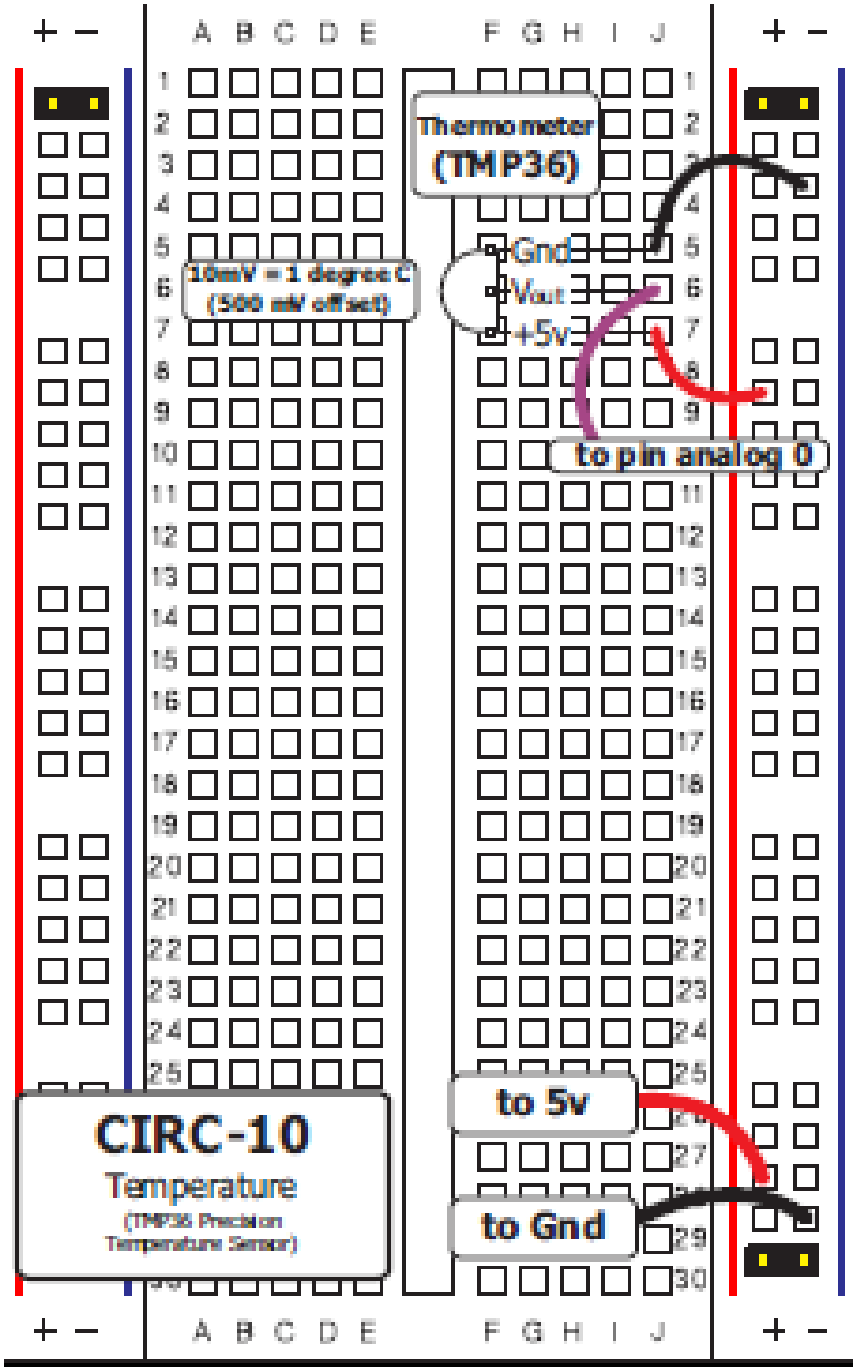
1001 1010 0100 0101 0100 0101

USB Serial

Lexa Jameyia Kate

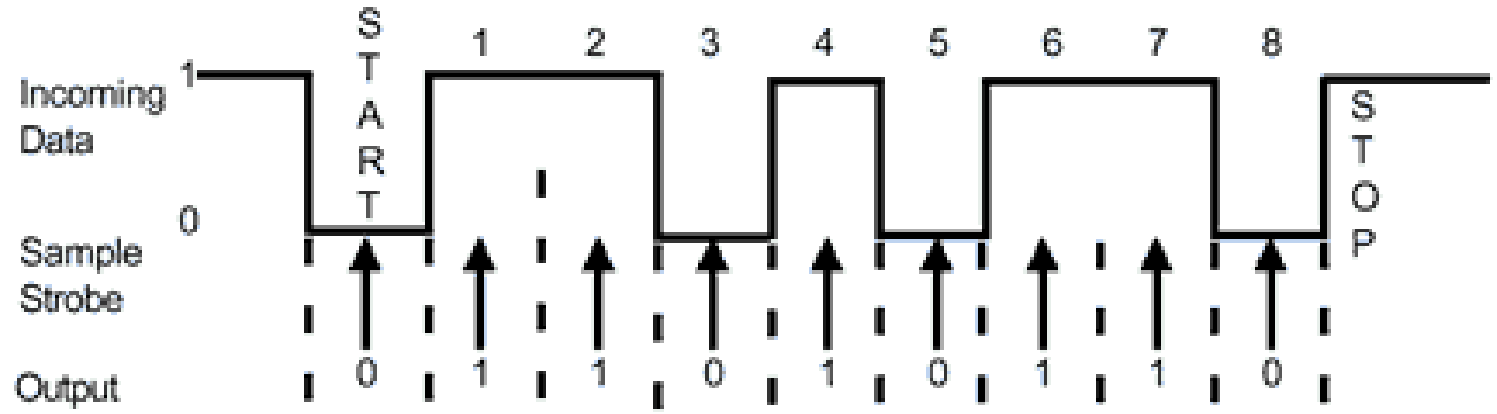
Everyone (for debugging)


Arduino
analog
Pin Ø
Energy Source
Electricity starts here
+ 5 Volts
Direction of current
Direction of current
+5v
signal
gnd
TMP36
(precision
temperature
sensor)
Direction of current
GND
(ground) (-)
Electricity ends here

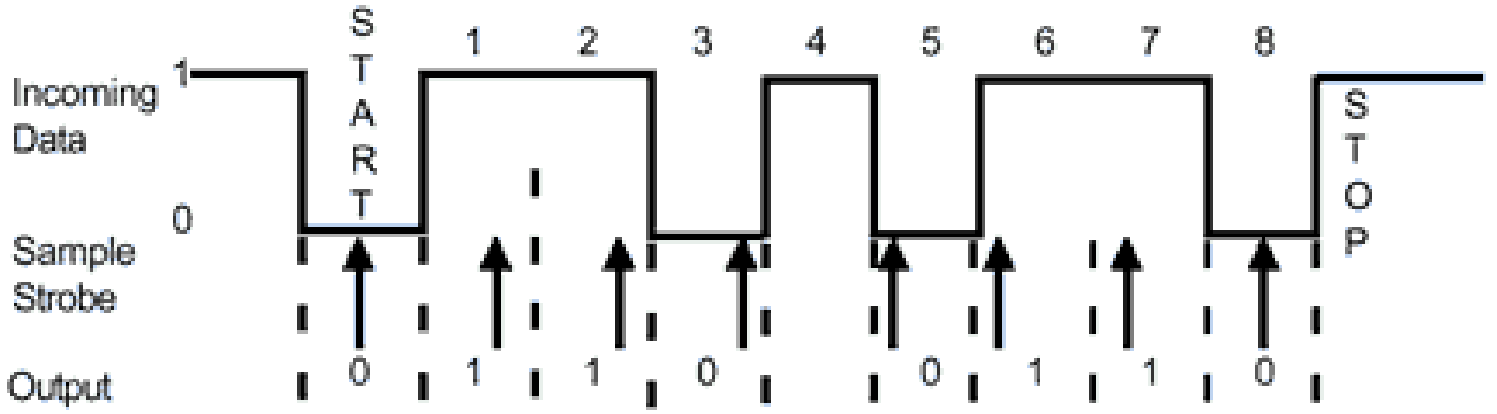

Thermometer
(TMP36)
Gnd
Vout
+5v
10mV = 1 degree C
(500 mV offset)
to pin analog 0
to 5v
to Gnd
CIRC-10
Temperature
(TMP36 Precision
Temperature Sensor)

```
int temperaturePin = 0;

void setup()
{
Serial.begin(9600);     //Serial comm. at a Baud Rate of 9600
}

void loop()
{
  float temp = getVoltage(temperaturePin);

  //Below is a line that compensates for an offset (see datasheet)
  temp = (temp - .5) * 100;

  Serial.println(temp);  // Send data to PC
  delay(1000);
}

float getVoltage(int pin)
{
  return (analogRead(pin) * .004882814);
}
```

One-Wire Communication can only go so fast before data is lost.



a. Best case, receiver samples at midpoint of each bit.



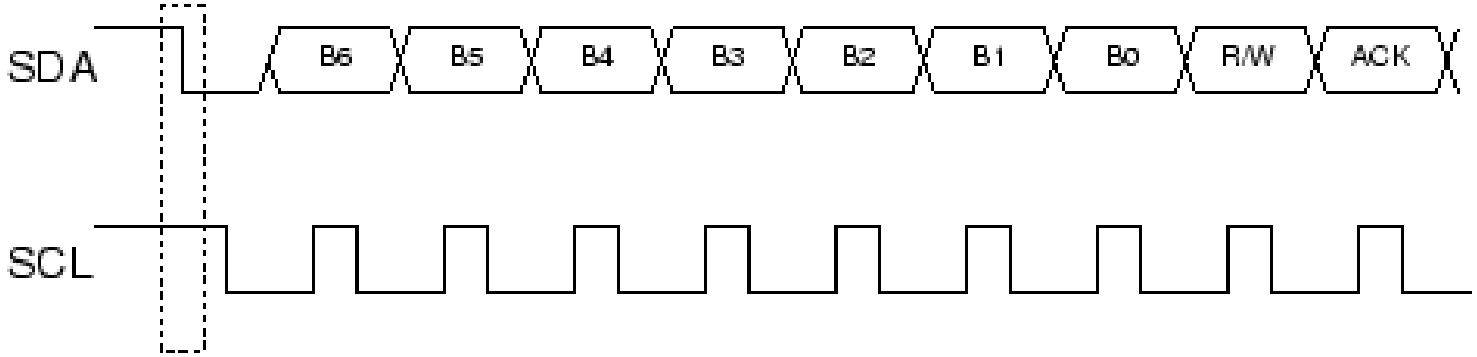
b. Receiving clock is too slow, causing bit 4 to be skipped and the data to be corrupted.

Ideal and corrupted asynchronous data sampling

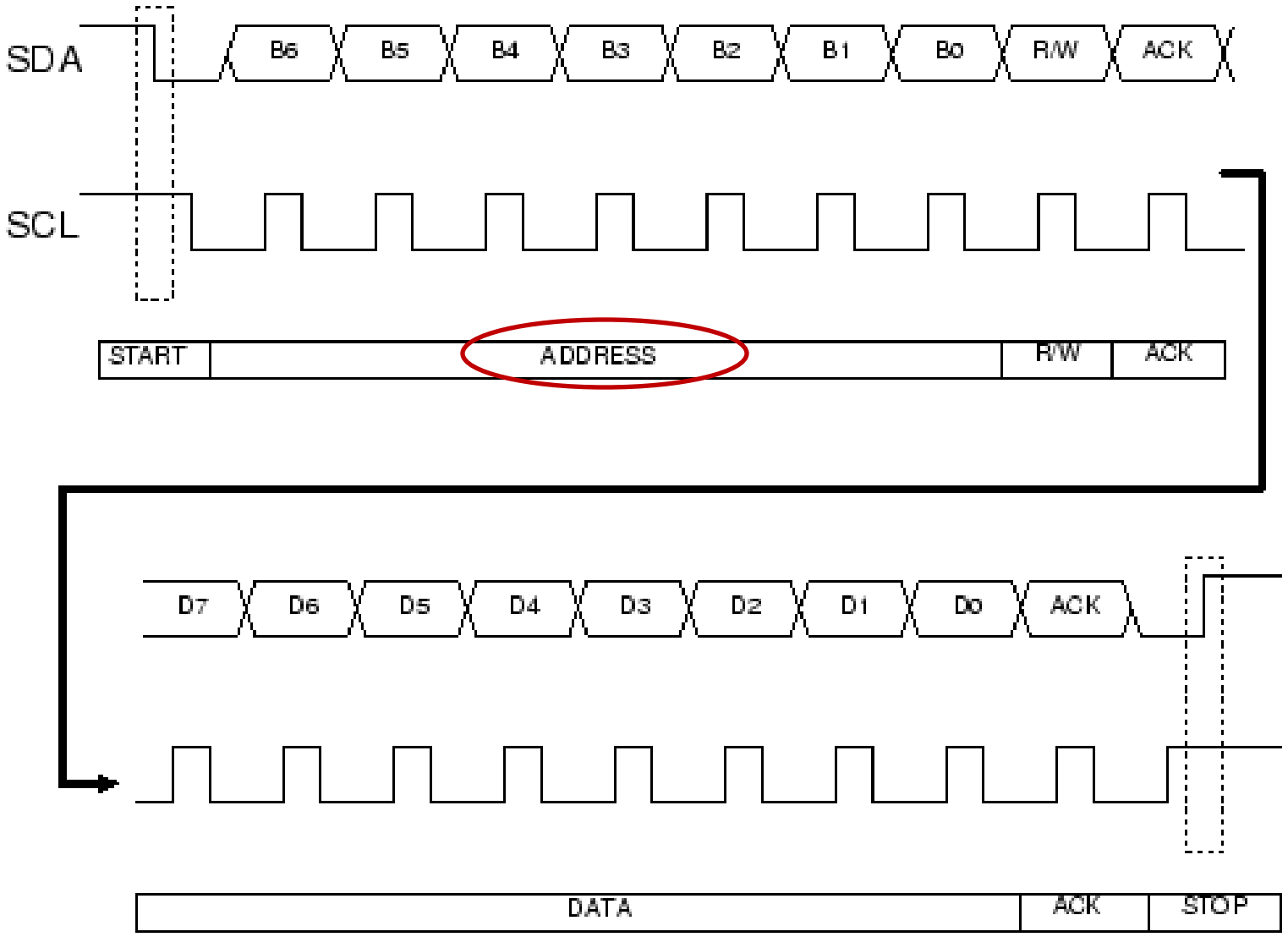
Two Wire

I2C

Scott


SDA
SCL
B6
B5
B4
B3
B2
B1
B0
R/W
ACK

Bus


SDA
B6
B5
B4
B3
B2
B1
B0
R/W
ACK
SCL
START
ADDRESS
R/W
ACK
D7
D6
D5
D4
D3
D2
D1
D0
ACK
DATA
ACK
STOP

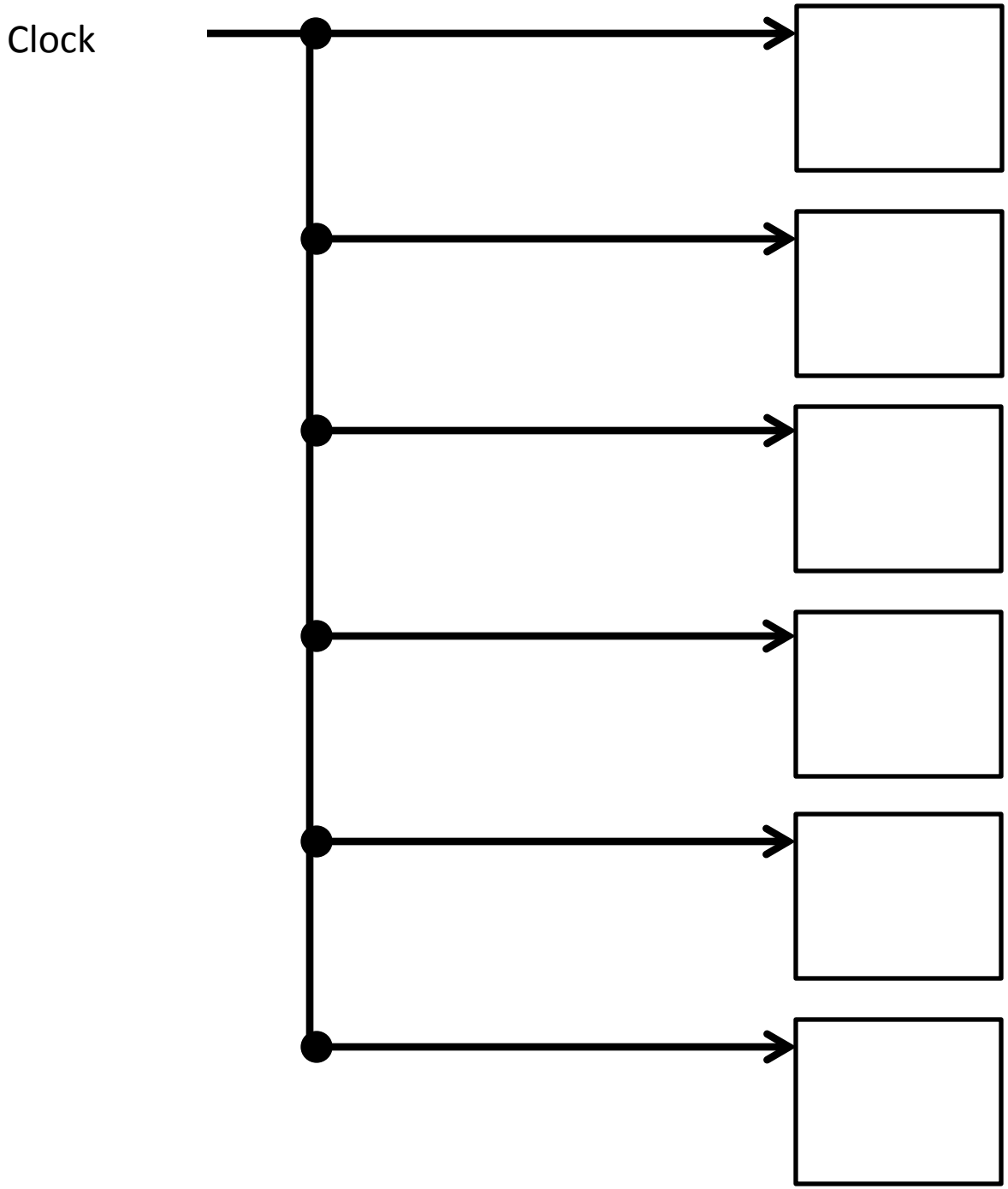
I2C Addressing is complicated; it’s possible multiple devices share the same address

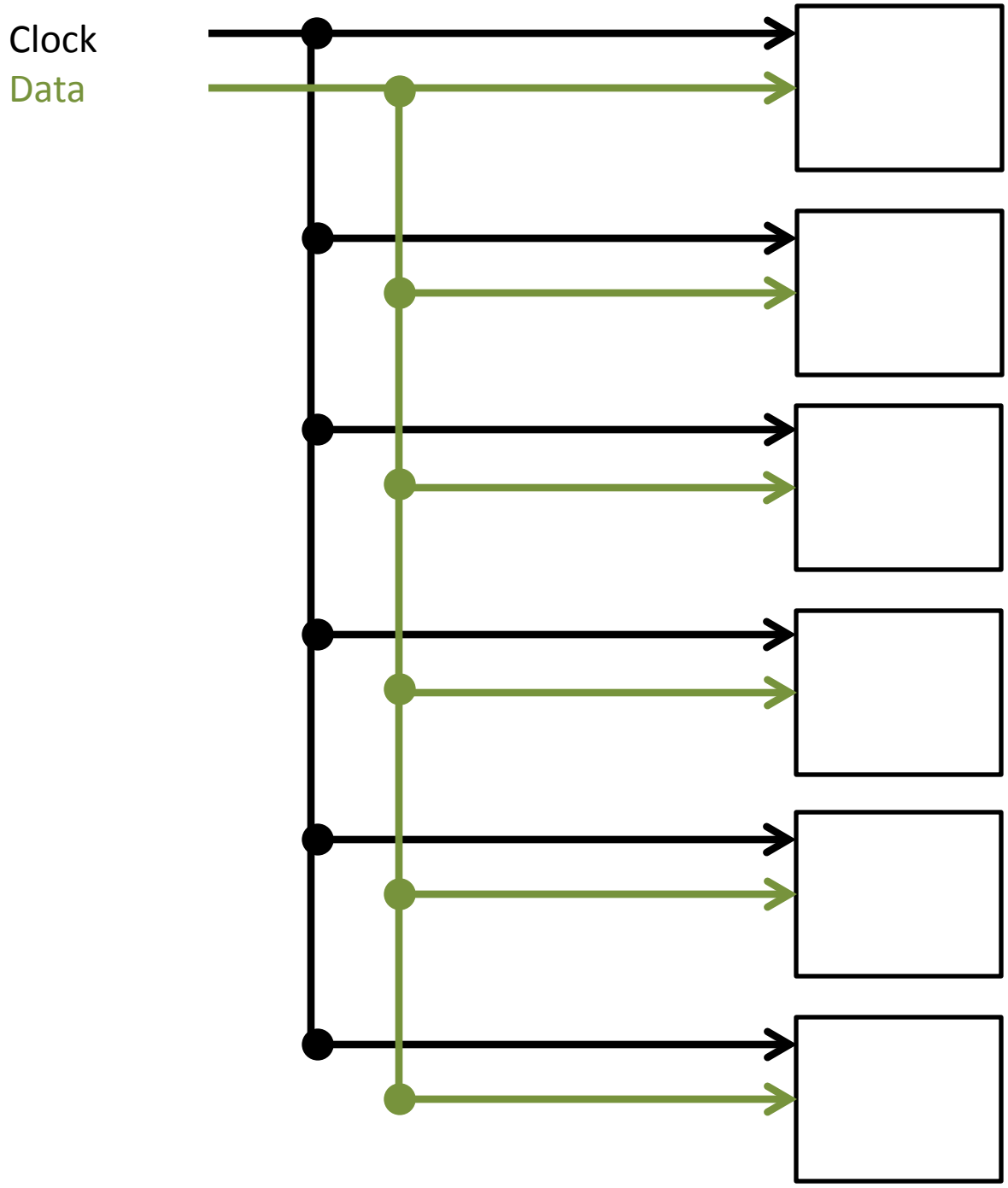
Three Wire

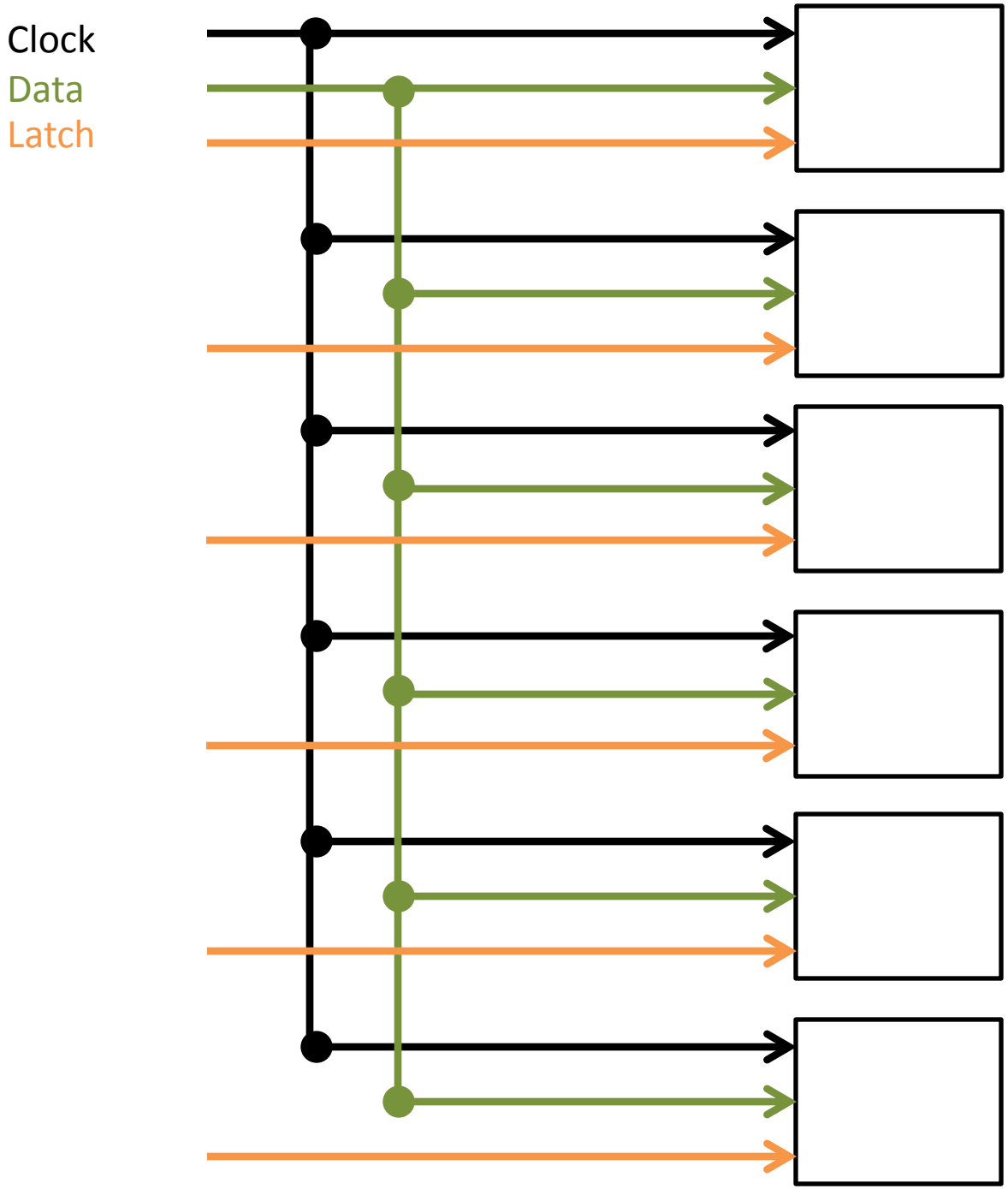
A third wire is added, which is used to "select" which target is being communicated with

Shift Register


CD4094 Shift Register/Latch
Clock
1
Data
1
Latch
0
Shift Register
0 1 1 0 0 0 0 1
Latch
0 0 0 0 0 0 0 0

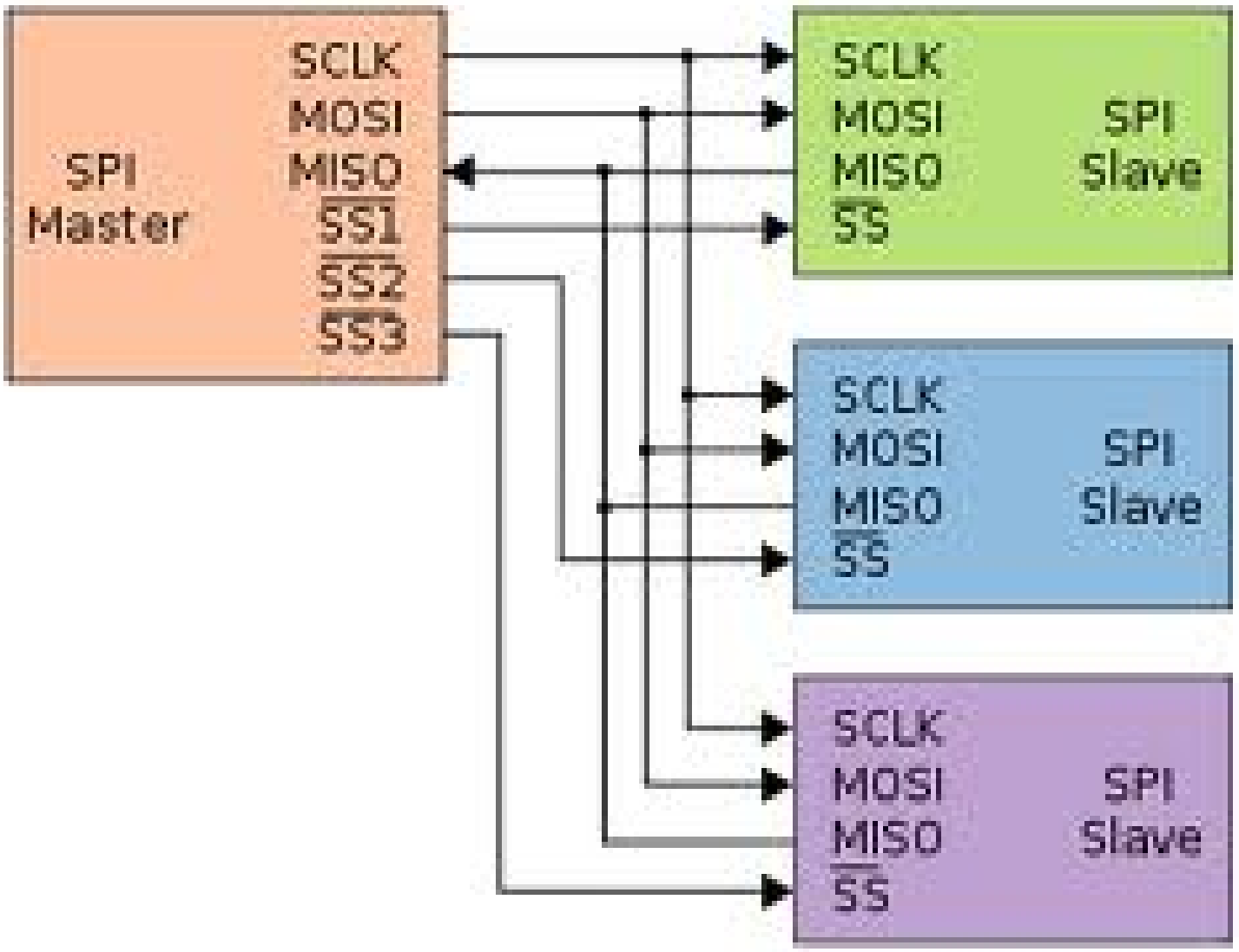

Clock


Clock
Data


Clock
Data
Latch

Serial Peripheral Interface (SPI)

Braden *Jasmine* *Quinn*


SPI
Master
SCLK
MOSI
MISO
SS1
SS2
SS3
SCLK
MOSI
MISO
SS
SPI
Slave
SCLK
MOSI
MISO
SS
SPI
Slave
SCLK
MOSI
MISO
SS
SPI
Slave

Master and Slave can communicate simultaneously

Faster than I2C (no addressing)

Fact: The delay() function isn't very accurate (+/- 1 ms)

But what if I want to communicate faster?

Fact: The delayMicroseconds() function is more accurate (+/- 1 us)

However: The delayMicroseconds() function disables interrupts

Interrupts

But what if I want to communicate faster?

Or what if I need interrupts enabled?

Fact: Once the communication reaches a certain speed, hardware assistance is required

Fact: Once the communication reaches a certain speed, hardware assistance is required

Libraries

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from **Sketch > Import Library**.

Standard Libraries

- EEPROM - reading and writing to "permanent" storage
- Ethernet - for connecting to the internet using the Arduino Ethernet Shield
- Firmata - for communicating with applications on the computer using a standard serial protocol.
- LiquidCrystal - for controlling liquid crystal displays (LCDs)
- SD - for reading and writing SD cards
- Servo - for controlling servo motors
- SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- SoftwareSerial - for serial communication on any digital pins
- Stepper - for controlling stepper motors
- Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

http://arduino.cc/en/Reference/Libraries

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from **Sketch > Import Library**.

Standard Libraries

- EEPROM - reading and writing to "permanent" storage
- Ethernet - for connecting to the internet using the Arduino Ethernet Shield
- Firmata - for communicating with applications on the computer using a standard serial protocol.
- LiquidCrystal - for controlling liquid crystal displays (LCDs)
- SD - for reading and writing SD cards
- Servo - for controlling servo motors
- SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- SoftwareSerial - for serial communication on any digital pins
- Stepper - for controlling stepper motors
- Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

http://arduino.cc/en/Reference/Libraries

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from **Sketch > Import Library**.

Standard Libraries

- EEPROM - reading and writing to "permanent" storage
- Ethernet - for connecting to the internet using the Arduino Ethernet Shield
- Firmata - for communicating with applications on the computer using a standard serial protocol.
- LiquidCrystal - for controlling liquid crystal displays (LCDs)
- SD - for reading and writing SD cards
- Servo - for controlling servo motors
- SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- SoftwareSerial - for serial communication on any digital pins
- Stepper - for controlling stepper motors
- Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

http://arduino.cc/en/Reference/Libraries

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from **Sketch > Import Library**.

Standard Libraries

- EEPROM - reading and writing to "permanent" storage
- Ethernet - for connecting to the internet using the Arduino Ethernet Shield
- Firmata - for communicating with applications on the computer using a standard serial protocol.
- LiquidCrystal - for controlling liquid crystal displays (LCDs)
- SD - for reading and writing SD cards
- Servo - for controlling servo motors
- SPI - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- SoftwareSerial - for serial communication on any digital pins
- Stepper - for controlling stepper motors
- Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

http://arduino.cc/en/Reference/Libraries

Why Buttons are more complicated than you might think….

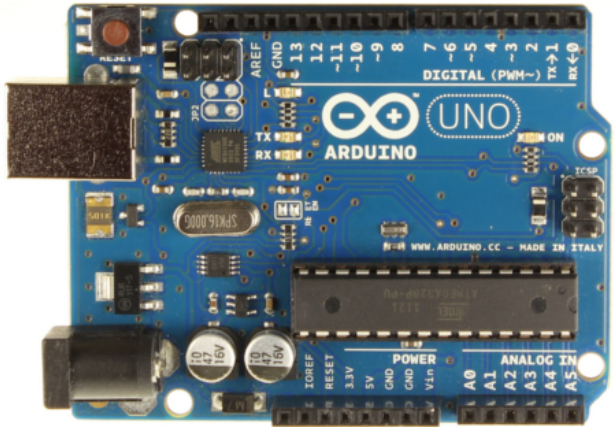
Fact: Buttons have no voltage, only a change in resistance

Fact: Buttons have no voltage, only a change in resistance

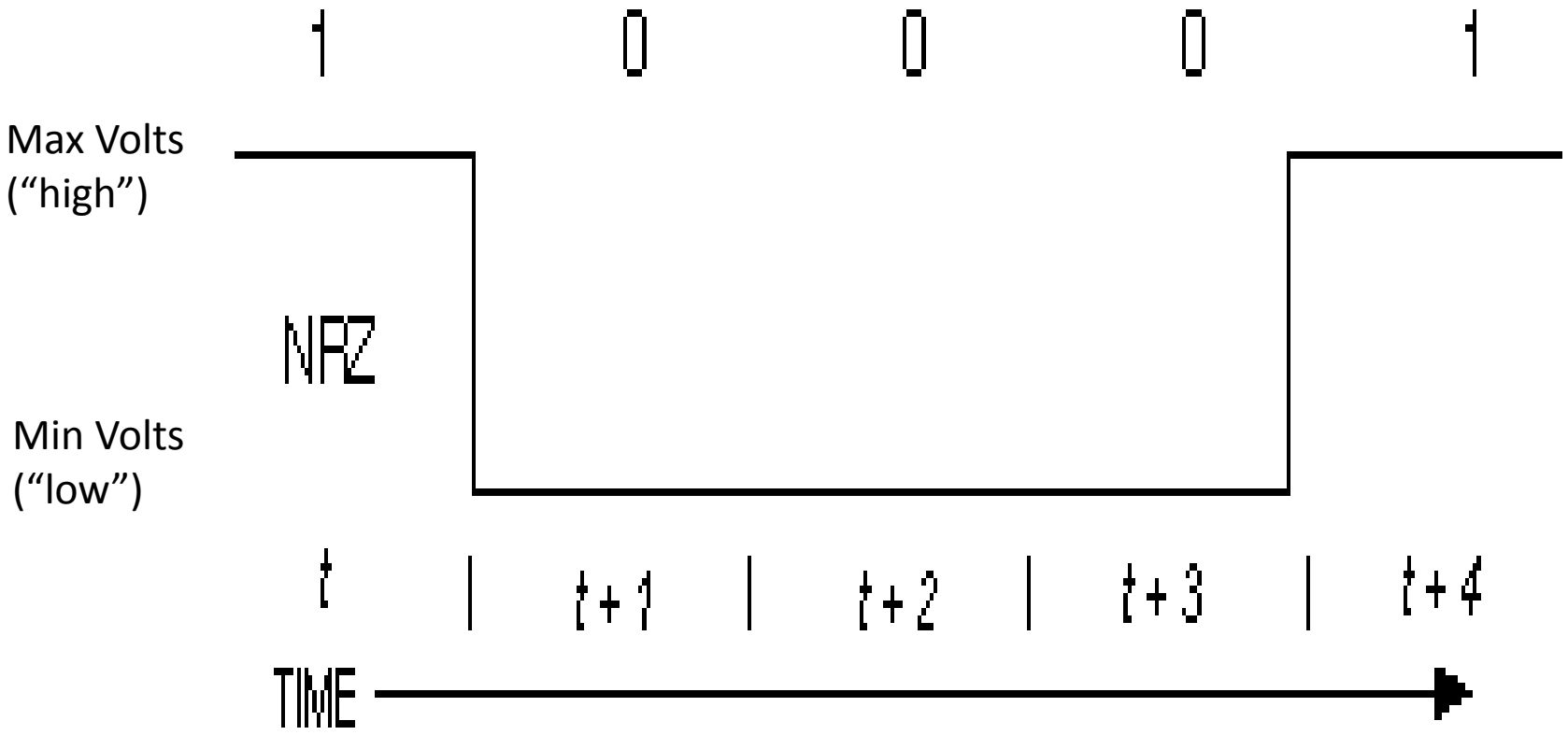
Infinite when Open

Fact: Buttons have no voltage, only a change in resistance

Zero when Closed

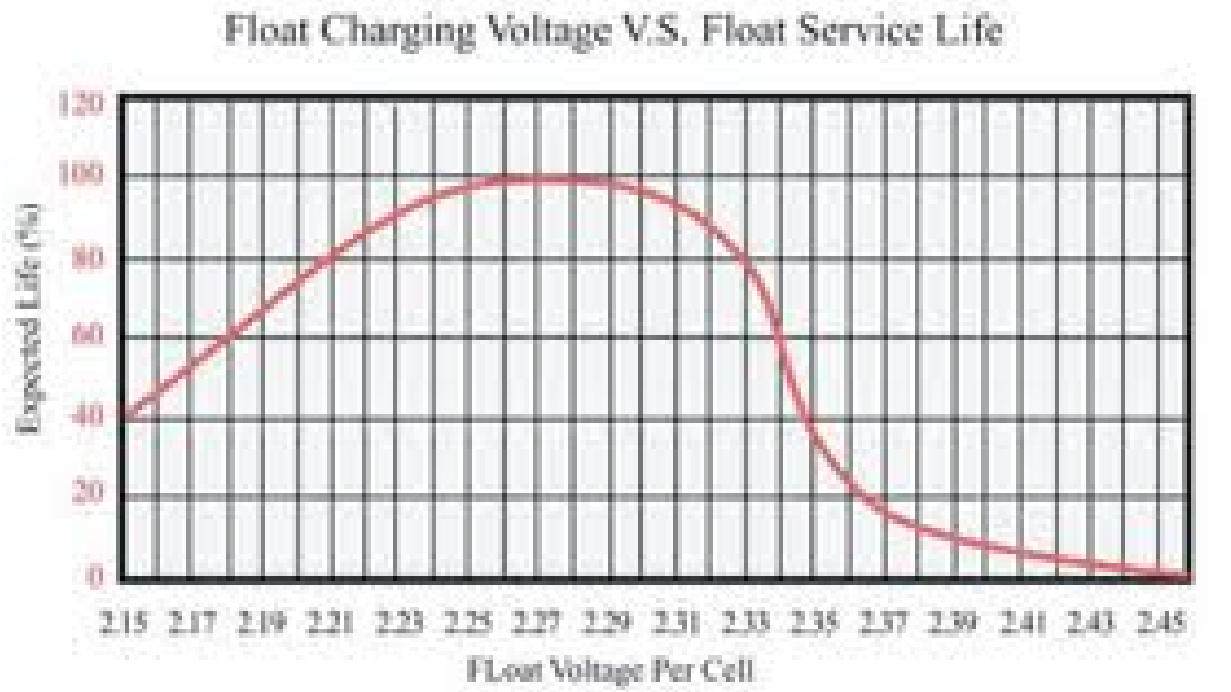


Fact: *Arduino inputs* ***only measures signals with voltage***



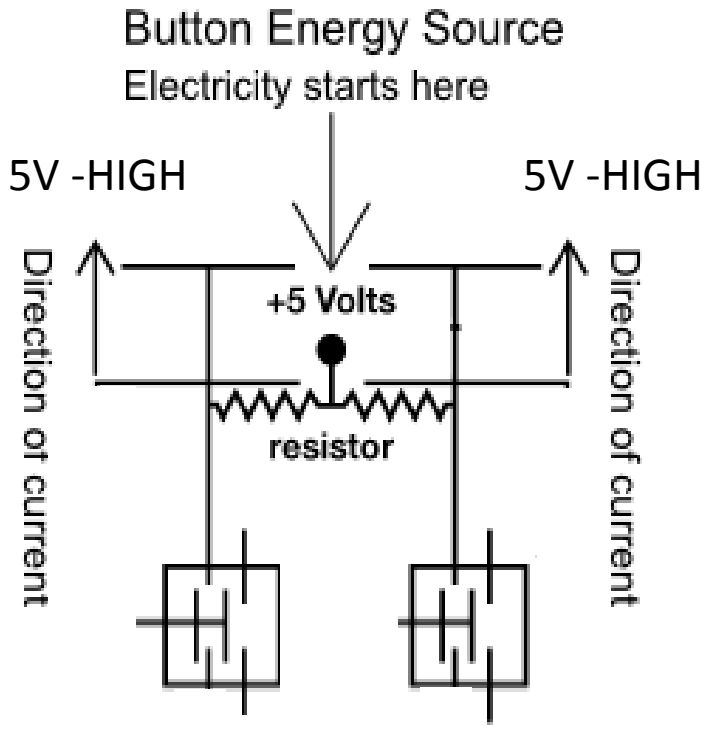
This is what we want a button press to look like!

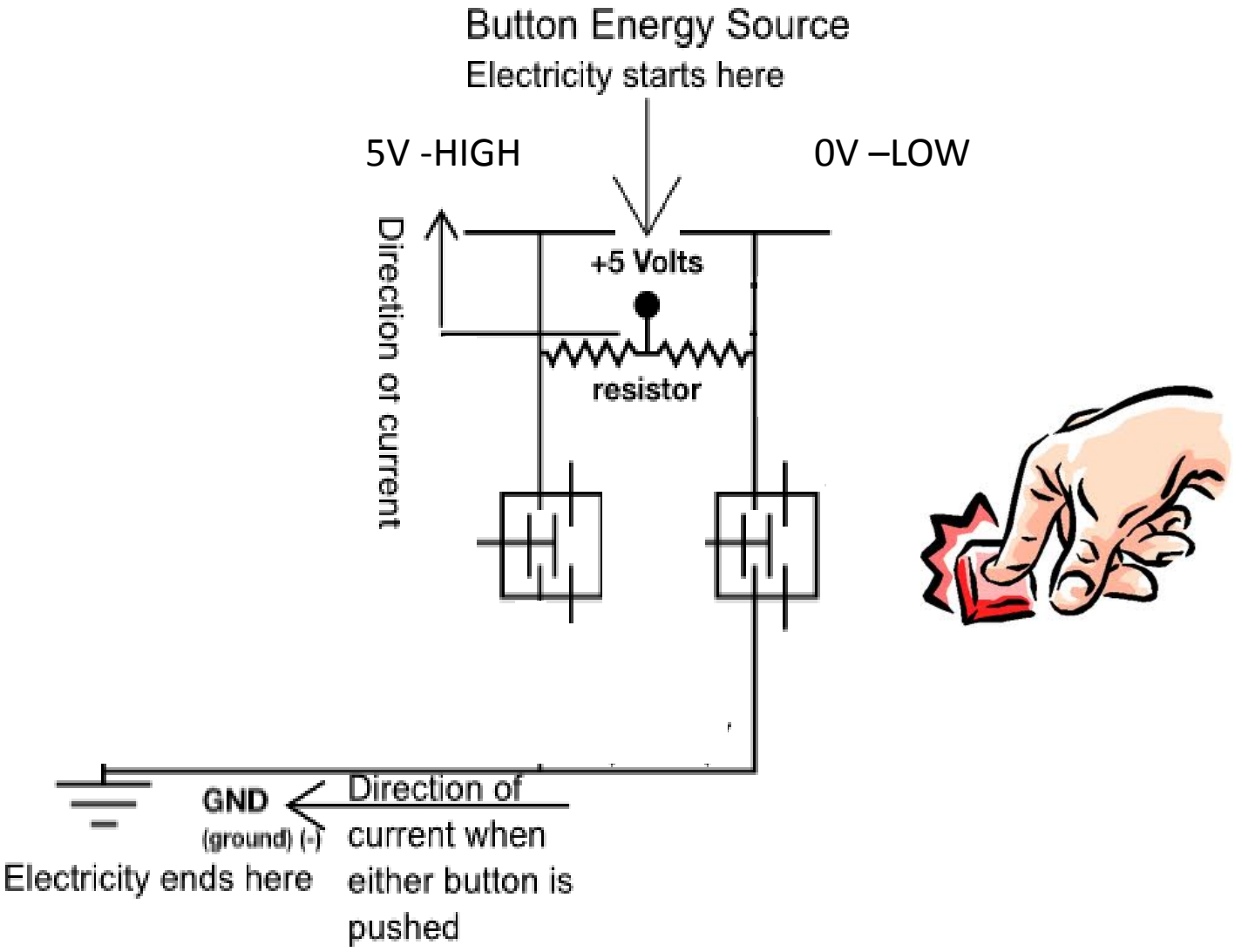
Fact: If we measure the voltage of a button, the value will “float”

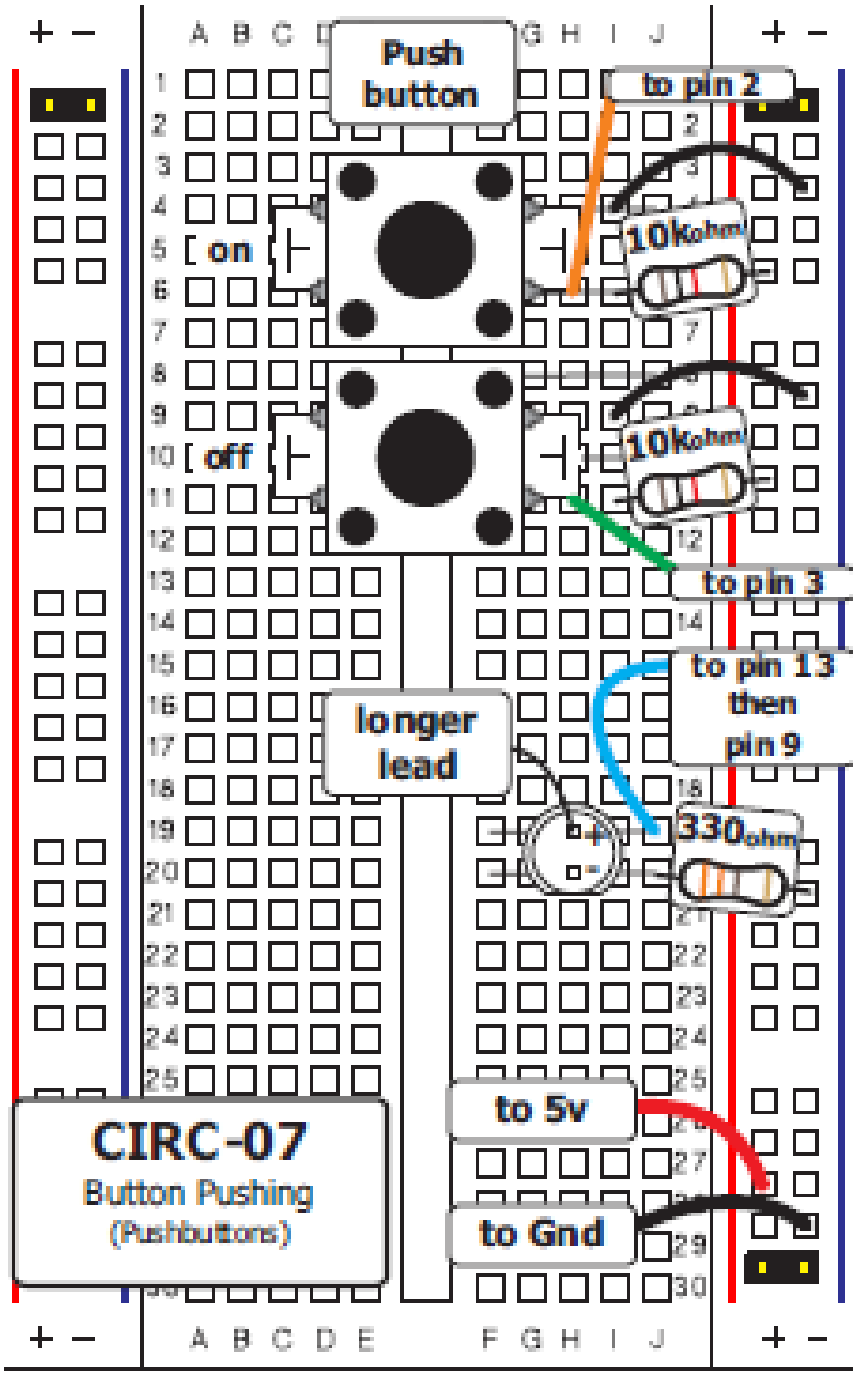


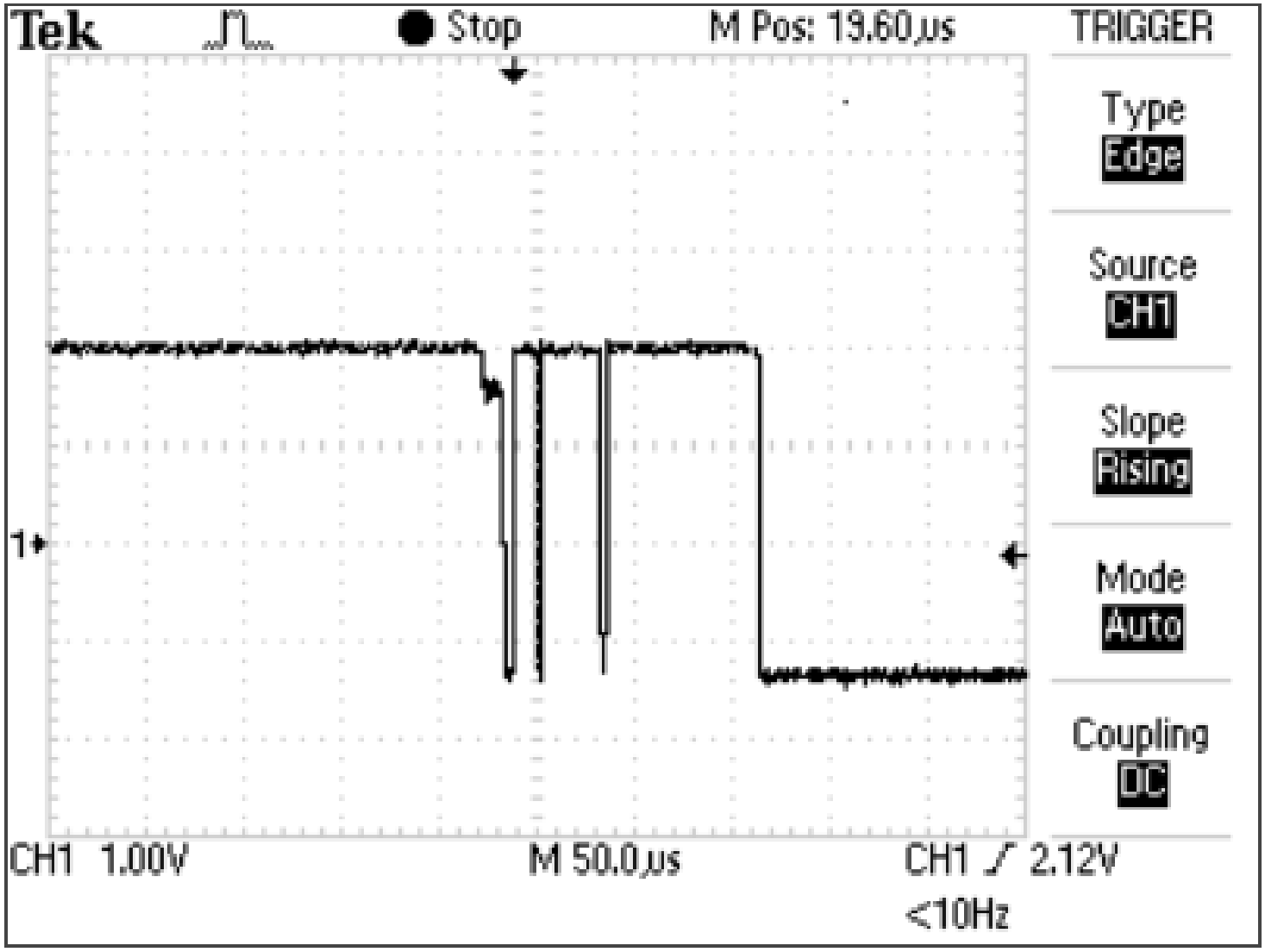
So how can we make the button appear like a digital signal?

Pull-up Resistors

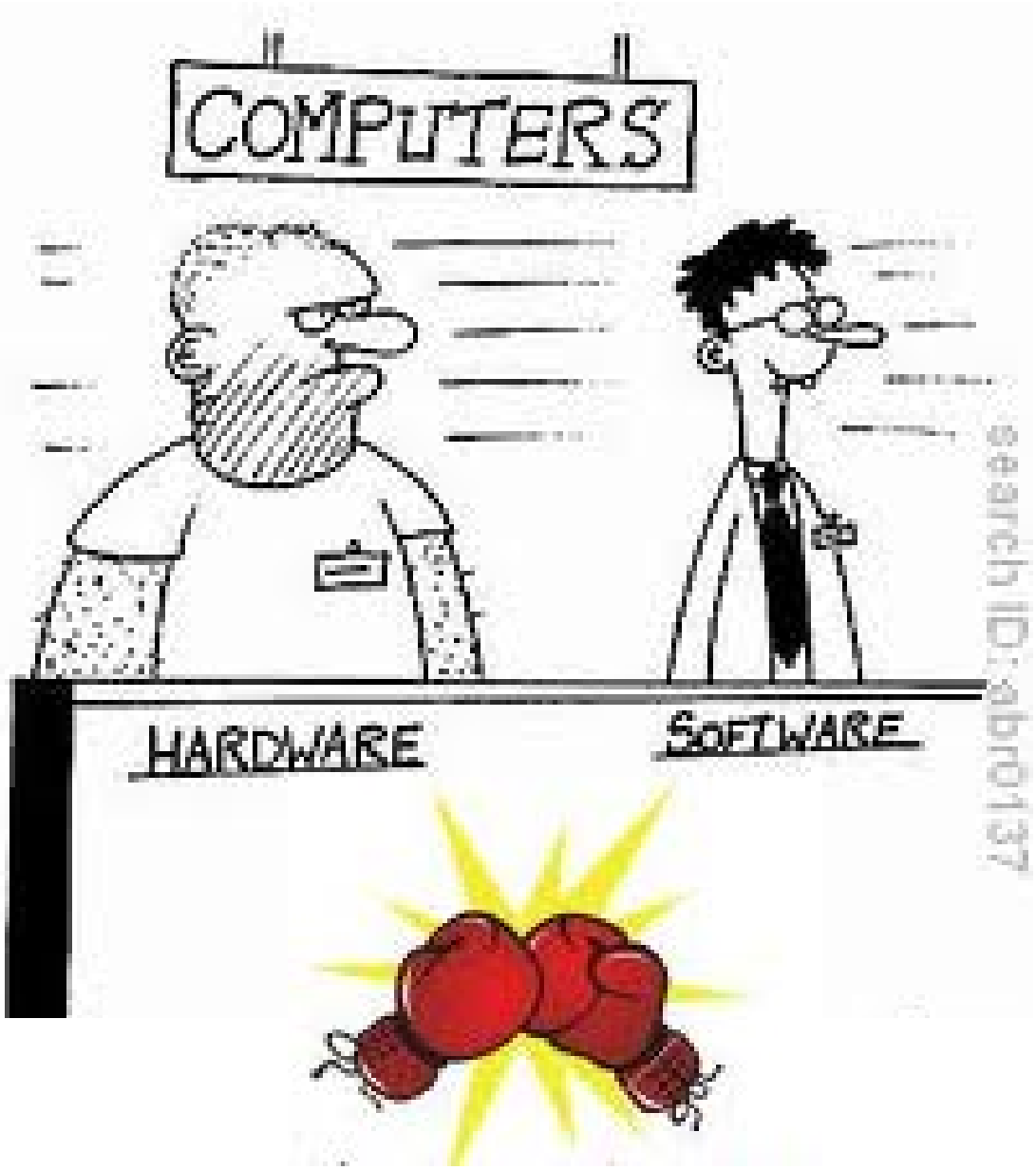

Button Energy Source
Electricity starts here
5V -HIGH
5V -HIGH
Direction of current
Direction of current
+5 Volts
resistor


Button Energy Source
Electricity starts here
5V -HIGH
0V –LOW
Direction of current
+5 Volts
resistor
GND
(ground) (-)
Electricity ends here
Direction of current when either button is pushed


Push button
to pin 2
10kohm
on
off
10kohm
to pin 3
to pin 13 then pin 9
longer lead
330ohm
to 5v
to Gnd
CIRC-07
Button Pushing
(Pushbuttons)



Switch bounce produced on switch press


COMPUTERS
HARDWARE
SOFTWARE
search ID: abr0137

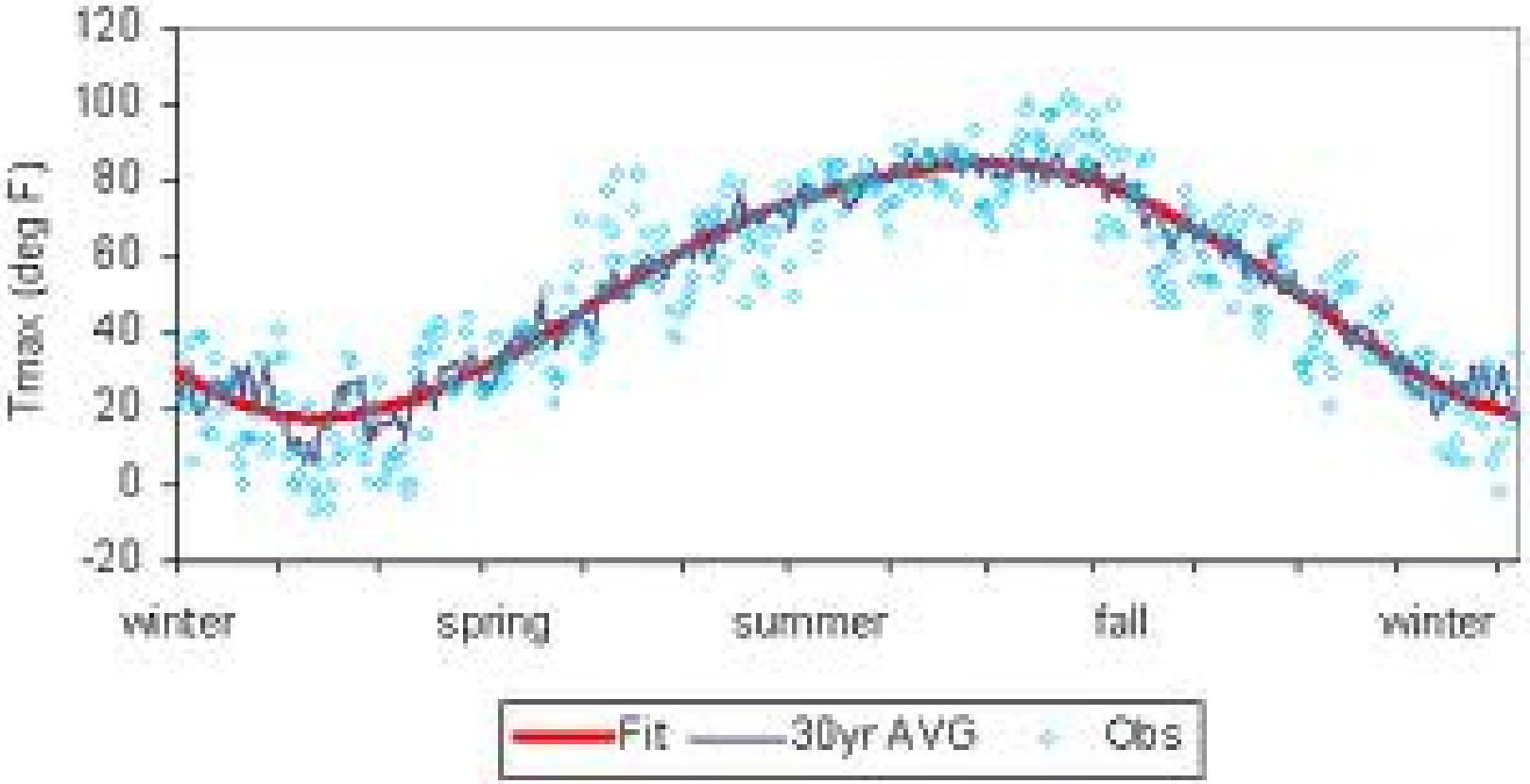
Low-Pass Filtering

Low-Pass Filtering (Smoothing)

Low-Pass Filtering (Averaging)

Low-Pass Filtering

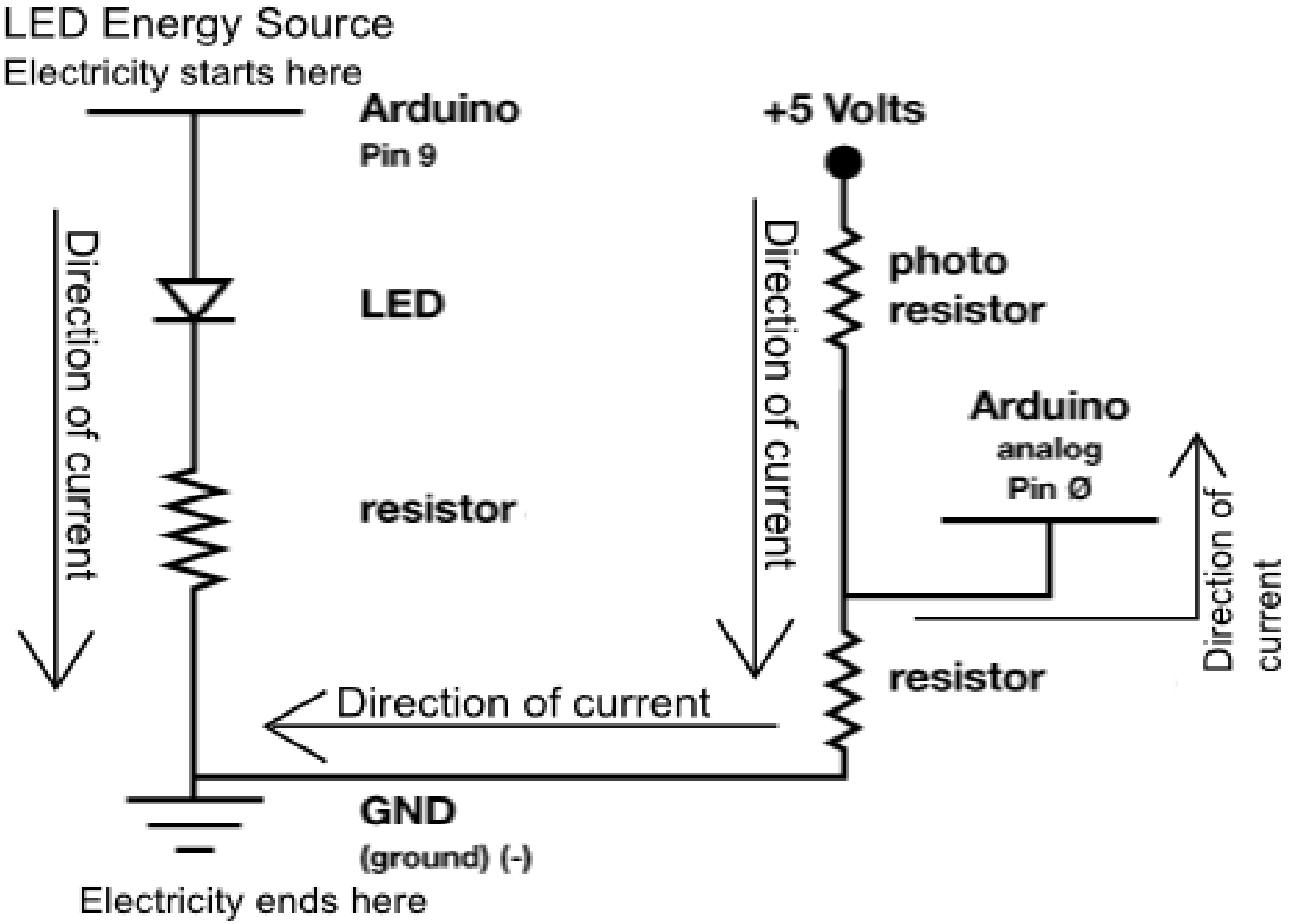
By averaging consecutive values, the rapidly-changing values are removed, revealing the underlying “trend”

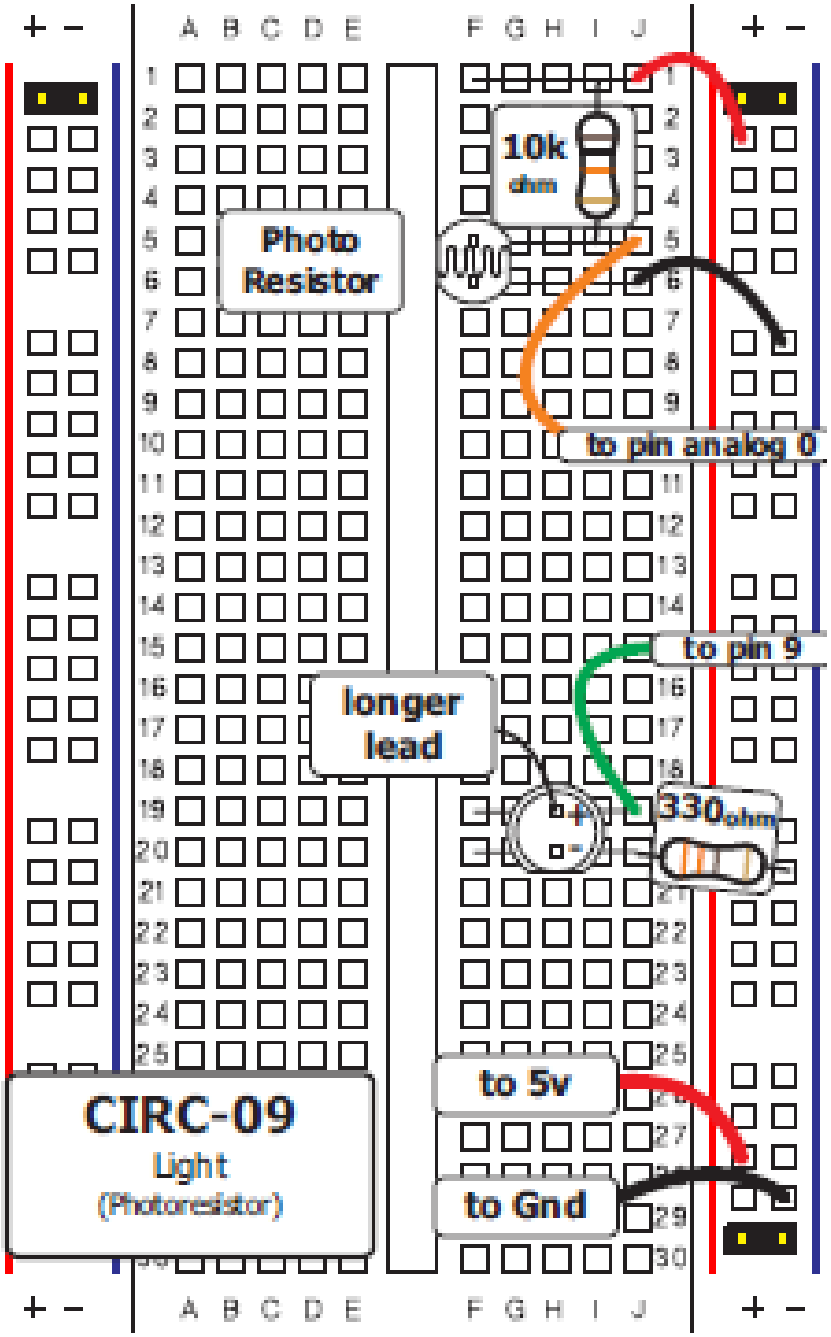

Tmax (deg F)
120
100
80
60
40
20
0
-20
winter
spring
summer
fall
winter
Fit
30yr AVG
Obs

```
boolean lastButtonValue;
boolean isButtonOn;

void setup()
{
  lastButtonValue = digitalRead(0);
  isButtonOn = false;
}

void loop()
{
 boolean currentButtonValue = digitalRead();
 if ( lastButtonValue == currentButtonValue )
 {
   if ( currentButtonValue != isButtonOn )
   {
     isButtonOn = currentButtonValue;              // Button value changed!
   }
 }
 delay(100);   // Maximum Debounce time
}
```

Regression

Voltage Dividers


LED Energy Source
Electricity starts here
Arduino
Pin 9
+5 Volts
Direction of current
LED
photo
resistor
Direction of current
Arduino
analog
Pin Ø
resistor
Direction of
current
resistor
Direction of current
GND
(ground) (-)
Electricity ends here


10k
ohm
Photo
Resistor
to pin analog 0
to pin 9
longer
lead
330ohm
to 5v
to Gnd
CIRC-09
Light
(Photoresistor)