IoT Product Design and Coding Bootcamp

Brian Rashap, Ph.D.

24-Feb-2020

Table of contents

IoT Fun


I THINK MY NEST SMOKE ALARM IS GOING OFF. GOOGLE ADWORDS JUST PITCHED ME A FIRE EXTINGUISHER AND AN OFFER FOR TEMPORARY HOUSING.
TOM FISH BURNE
©marketoonist.com

Brian Rashap, Ph.D.

- Proud husband of Krista and father of Shelby (21) and Ethan (18)
- Electrical Engineer with 25 years industrial experience
- High School track coach
- Hobbies: running, cycling, reading, spending time with family

Introductions

INTRODUCTIONS

Class Rules

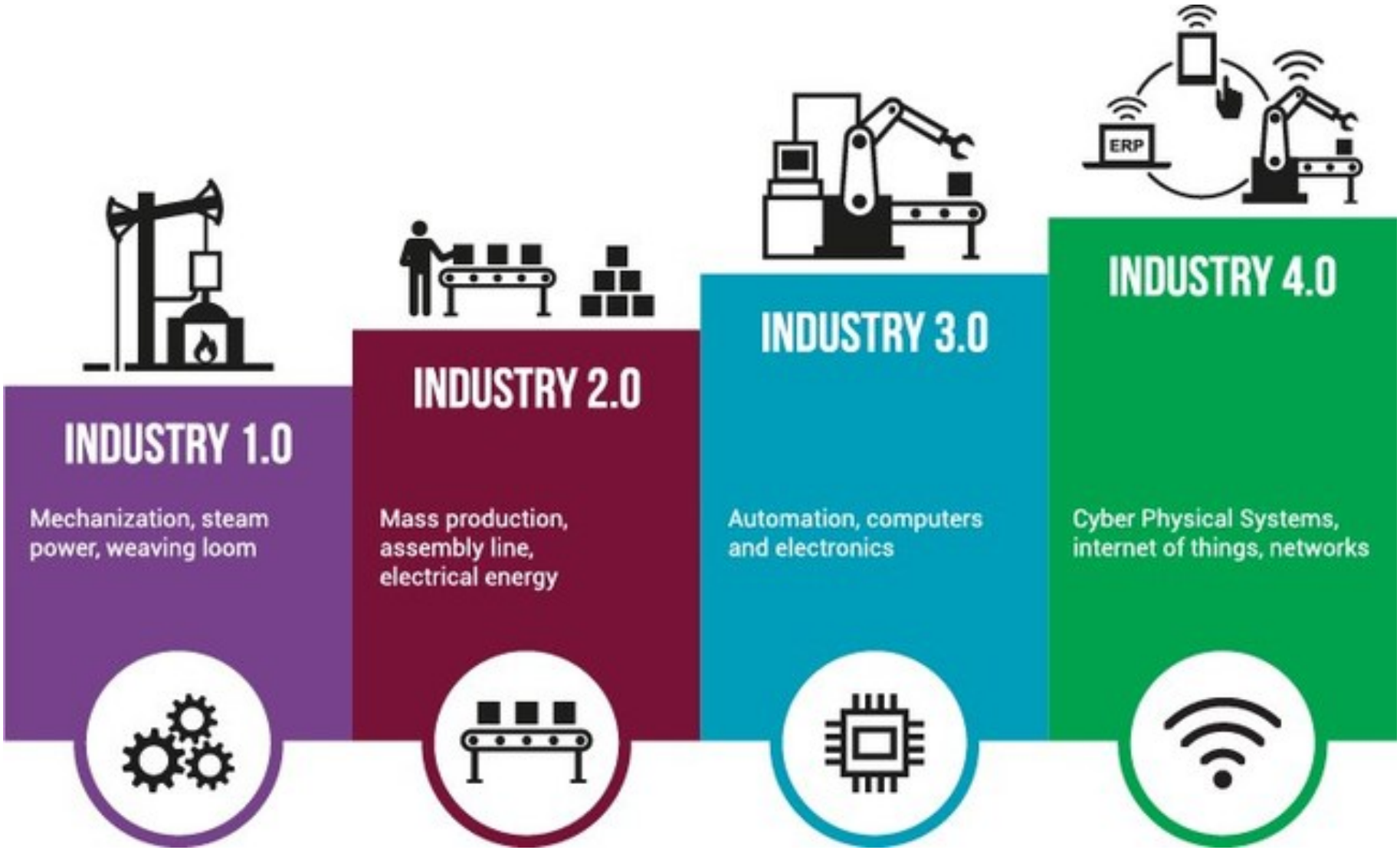
- Respect Each Other, Help Each Other
- Ask Questions
- Be On Time (let's us know via Slack if you won't be here)
- Keep Your Workspace and the Classroom Neat and Tidy
- If you are struggling, let myself, Susan, or Esteban know. We are here to HELP!

Grading

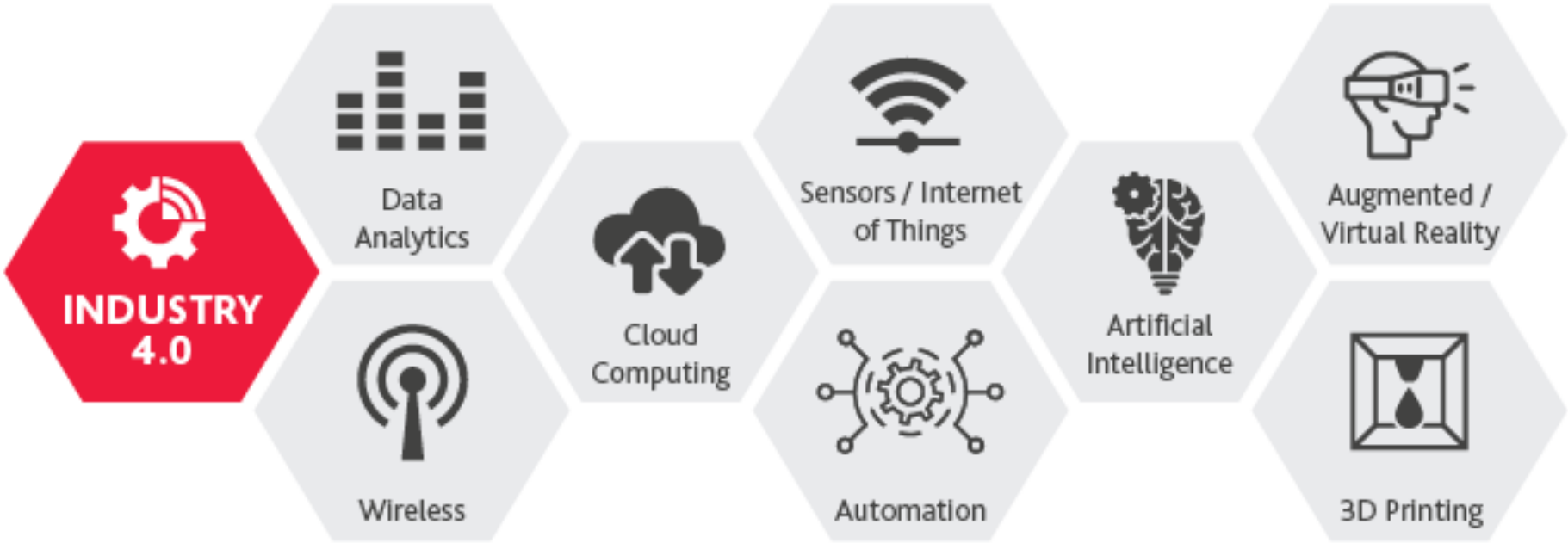
1. Weekly IoT assignments: 40
2. Lab Notebook: 10
3. Weekly quizzes: 10
4. Individual Midterm Project: Smart Room Controller: 20
5. Team Capstone Project: 20

More information later today on how assignments are turned in

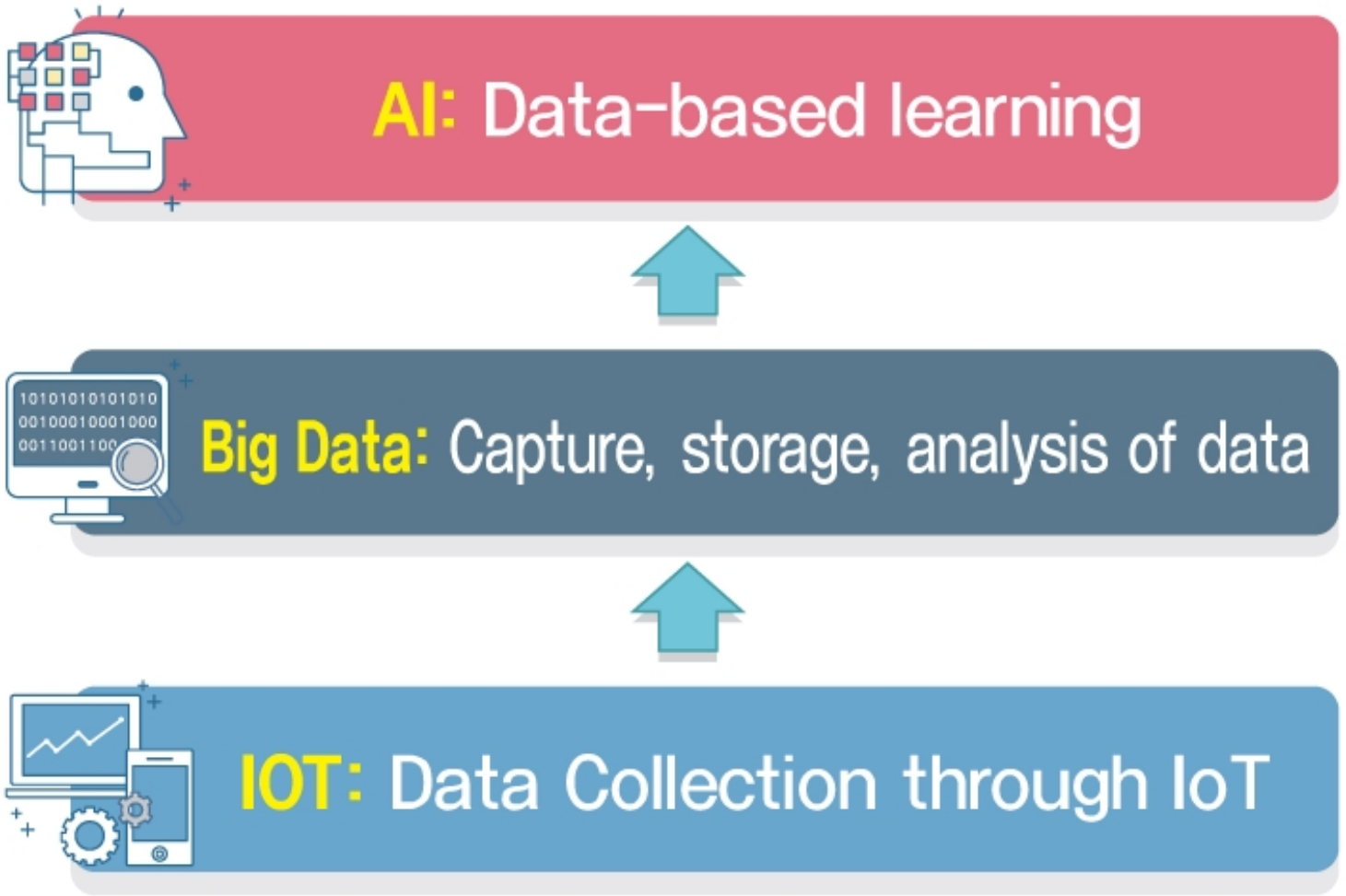
Evolution of Industry



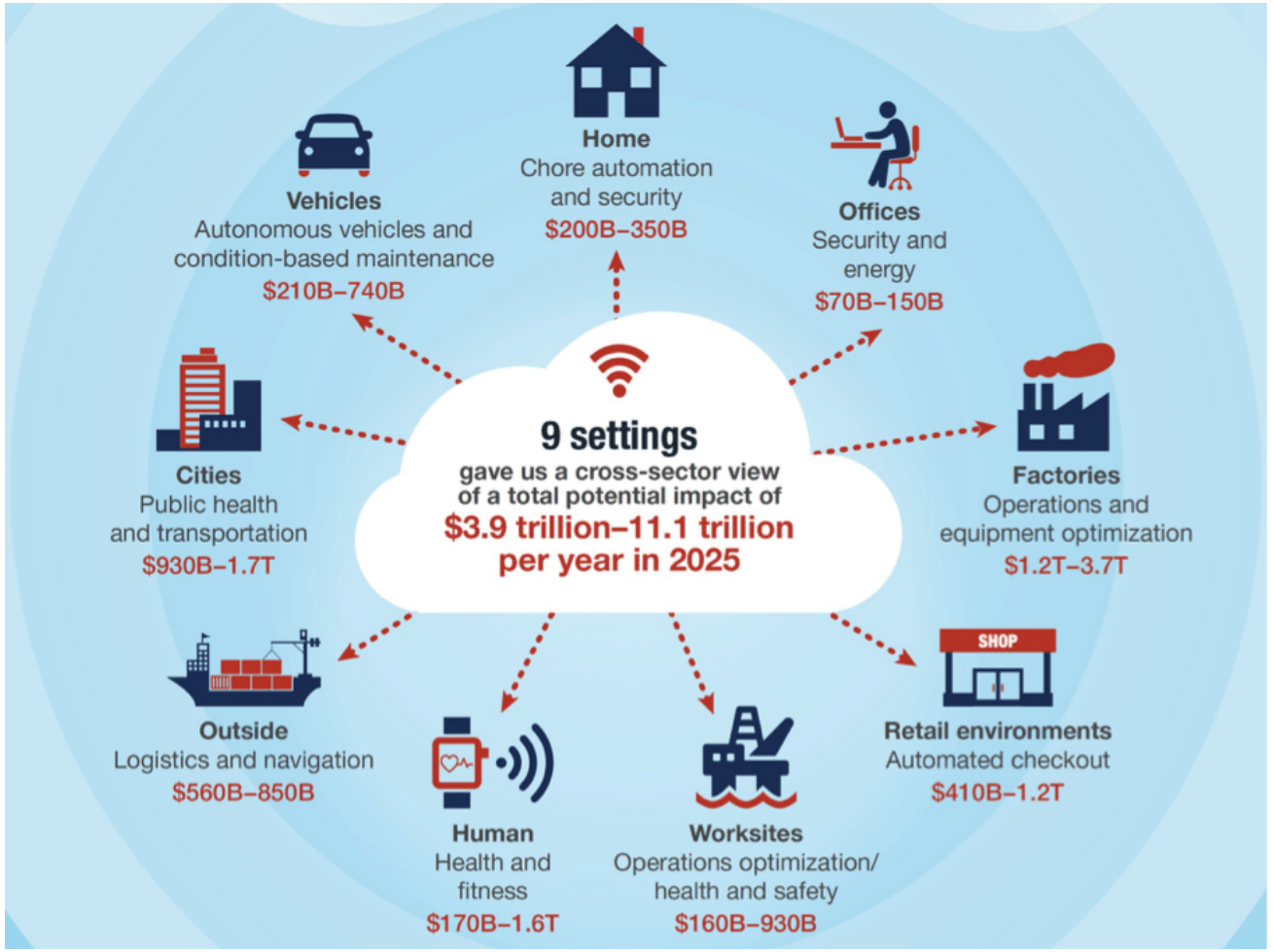
Components of Industry 4.0



IoT and Data Science

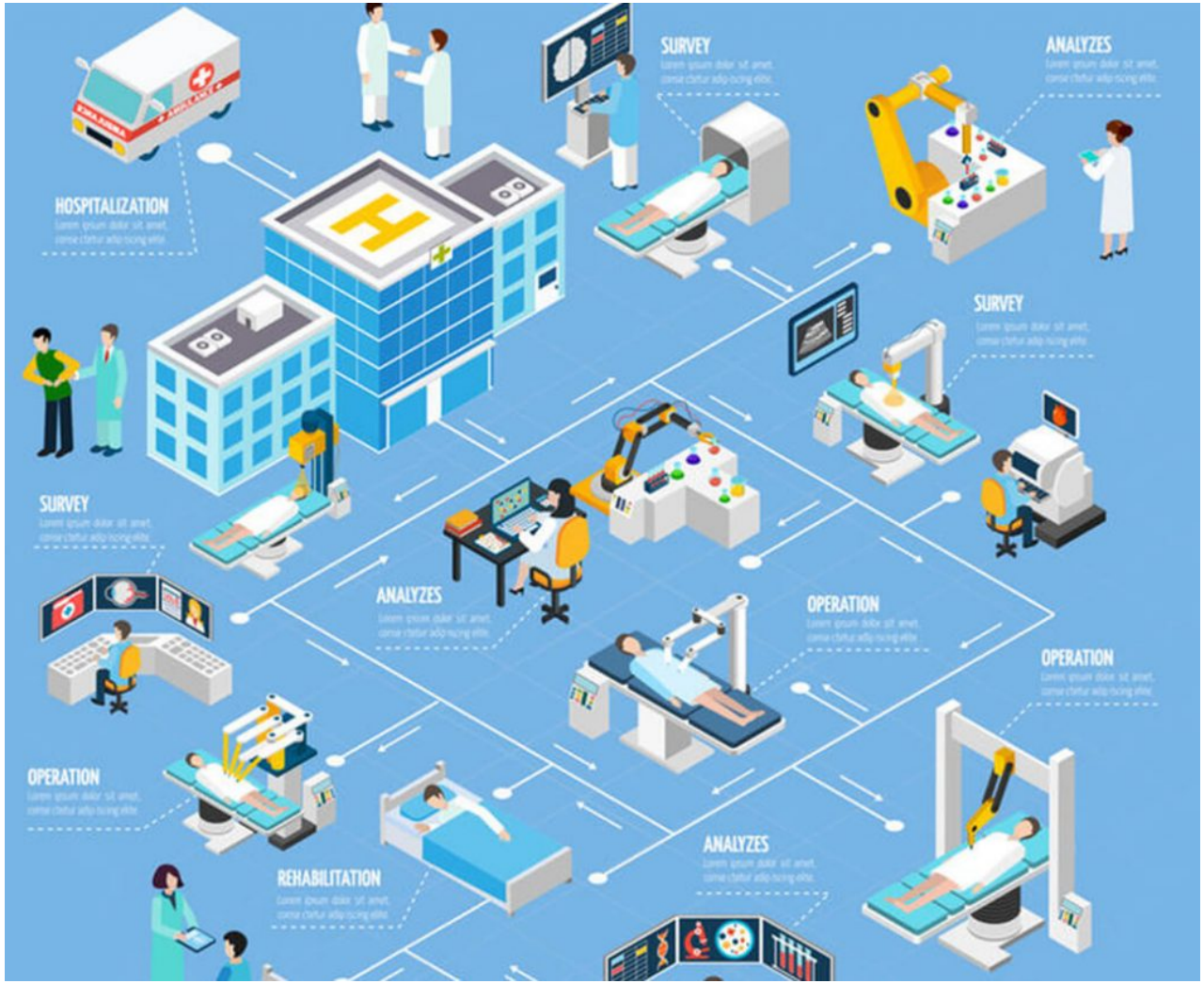


IoT 2025

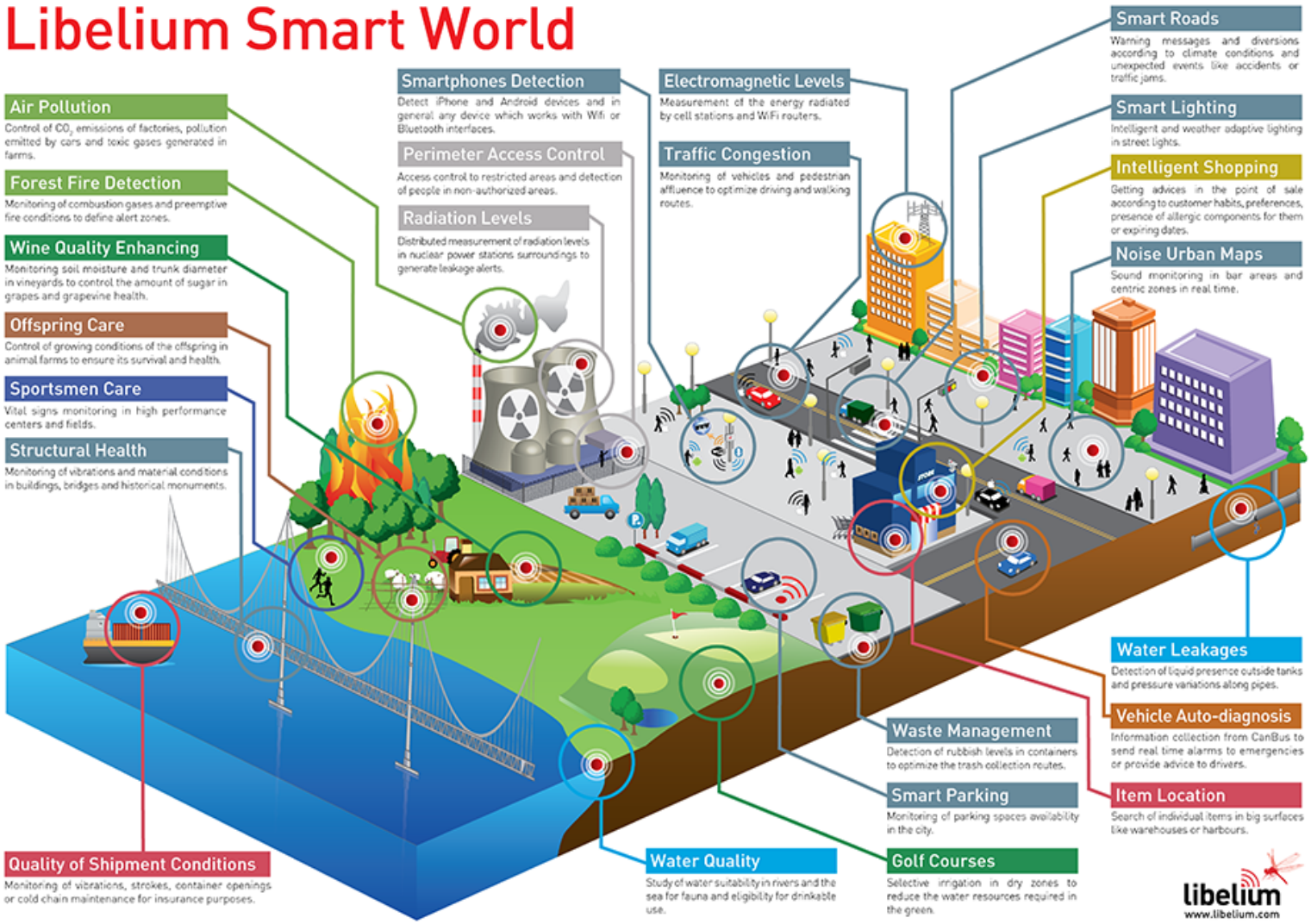

Home
Chore automation and security
$200B–350B
Vehicles
Autonomous vehicles and condition-based maintenance
$210B–740B
Offices
Security and energy
$70B–150B
9 settings
gave us a cross-sector view of a total potential impact of
$3.9 trillion–11.1 trillion per year in 2025
Cities
Public health and transportation
$930B–1.7T
Factories
Operations and equipment optimization
$1.2T–3.7T
SHOP
Outside
Logistics and navigation
$560B–850B
Retail environments
Automated checkout
$410B–1.2T
Human
Health and fitness
$170B–1.6T
Worksites
Operations optimization/ health and safety
$160B–930B

Smart Facilities

Healthcare 2025

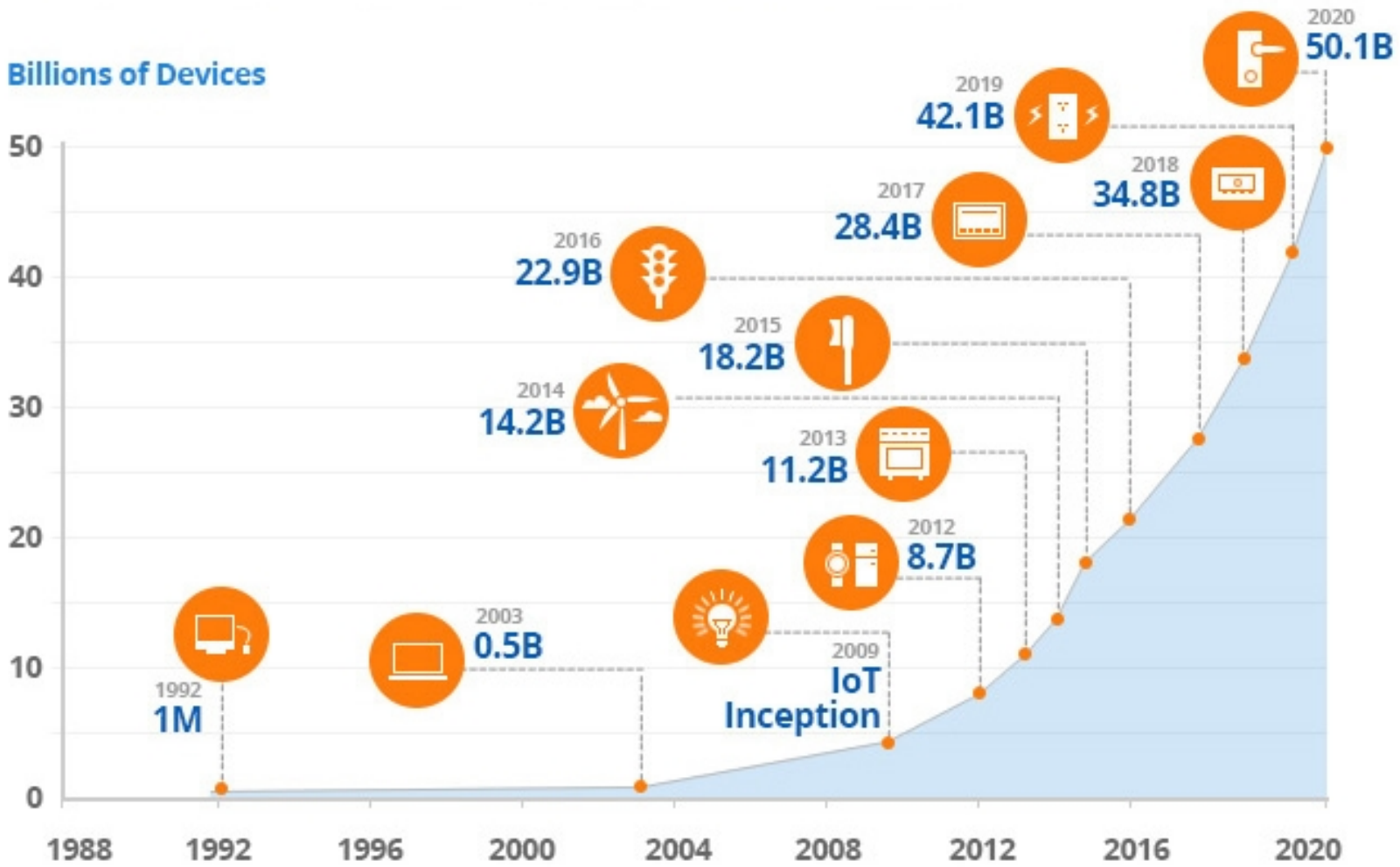


Smart World


Libelium Smart World
Air Pollution
Control of CO2 emissions of factories, pollution emitted by cars and toxic gases generated in farms.
Forest Fire Detection
Monitoring of combustion gases and preemptive fire conditions to define alert zones.
Wine Quality Enhancing
Monitoring soil moisture and trunk diameter in vineyards to control the amount of sugar in grapes and grapevine health.
Offspring Care
Control of growing conditions of the offspring in animal farms to ensure its survival and health.
Sportsmen Care
Vital signs monitoring in high performance centers and fields.
Structural Health
Monitoring of vibrations and material conditions in buildings, bridges and historical monuments.
Quality of Shipment Conditions
Monitoring of vibrations, strokes, container openings or cold chain maintenance for insurance purposes.
Smartphones Detection
Detect iPhone and Android devices and in general any device which works with Wifi or Bluetooth interfaces.
Perimeter Access Control
Access control to restricted areas and detection of people in non-authorized areas.
Radiation Levels
Distributed measurement of radiation levels in nuclear power stations surroundings to generate leakage alerts.
Electromagnetic Levels
Measurement of the energy radiated by cell stations and WiFi routers.
Traffic Congestion
Monitoring of vehicles and pedestrian affluence to optimize driving and walking routes.
Water Quality
Study of water suitability in rivers and the sea for fauna and eligibility for drinkable use.
Smart Roads
Warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.
Smart Lighting
Intelligent and weather adaptive lighting in street lights.
Intelligent Shopping
Getting advices in the point of sale according to customer habits, preferences, presence of allergic components for them or expiring dates.
Noise Urban Maps
Sound monitoring in bar areas and centric zones in real time.
Water Leakages
Detection of liquid presence outside tanks and pressure variations along pipes.
Vehicle Auto-diagnosis
Information collection from CanBus to send real time alarms to emergencies or provide advice to drivers.
Item Location
Search of individual items in big surfaces like warehouses or harbours.
Waste Management
Detection of rubbish levels in containers to optimize the trash collection routes.
Smart Parking
Monitoring of parking spaces availability in the city.
Golf Courses
Selective irrigation in dry zones to reduce the water resources required in the green.
libelium
www.libelium.com

And Out of This World

IoT Growth


Billions of Devices
50
40
30
20
10
0
1988
1992
1996
2000
2004
2008
2012
2016
2020
1992
1M
2003
0.5B
2009
IoT
Inception
2012
8.7B
2013
11.2B
2014
14.2B
2015
18.2B
2016
22.9B
2017
28.4B
2018
34.8B
2019
42.1B
2020
50.1B

Let's Begin Our Journey

Computer Languages

Mother Tongues

Tracing the roots of computer languages through the ages

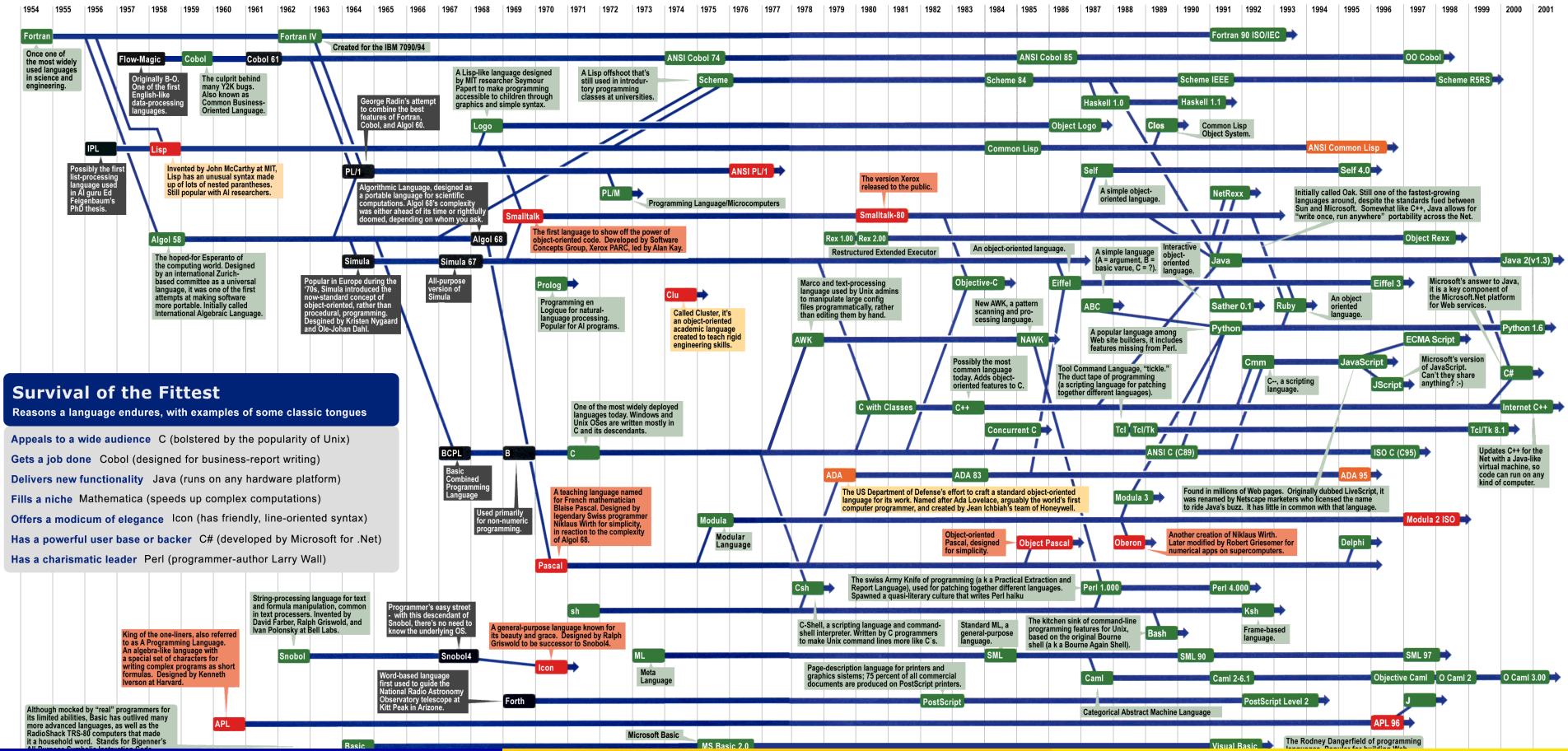
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Musuem in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at *HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html*. - Michael Mendeno

Key

- 1954 Year Introduced
- **Active:** thousands of users
- **Protected:** taught at universities; compilers available
- **Endangered:** usage dropping off
- **Extinct:** no known active users or up-to-date compilers
- **Lineage continues**

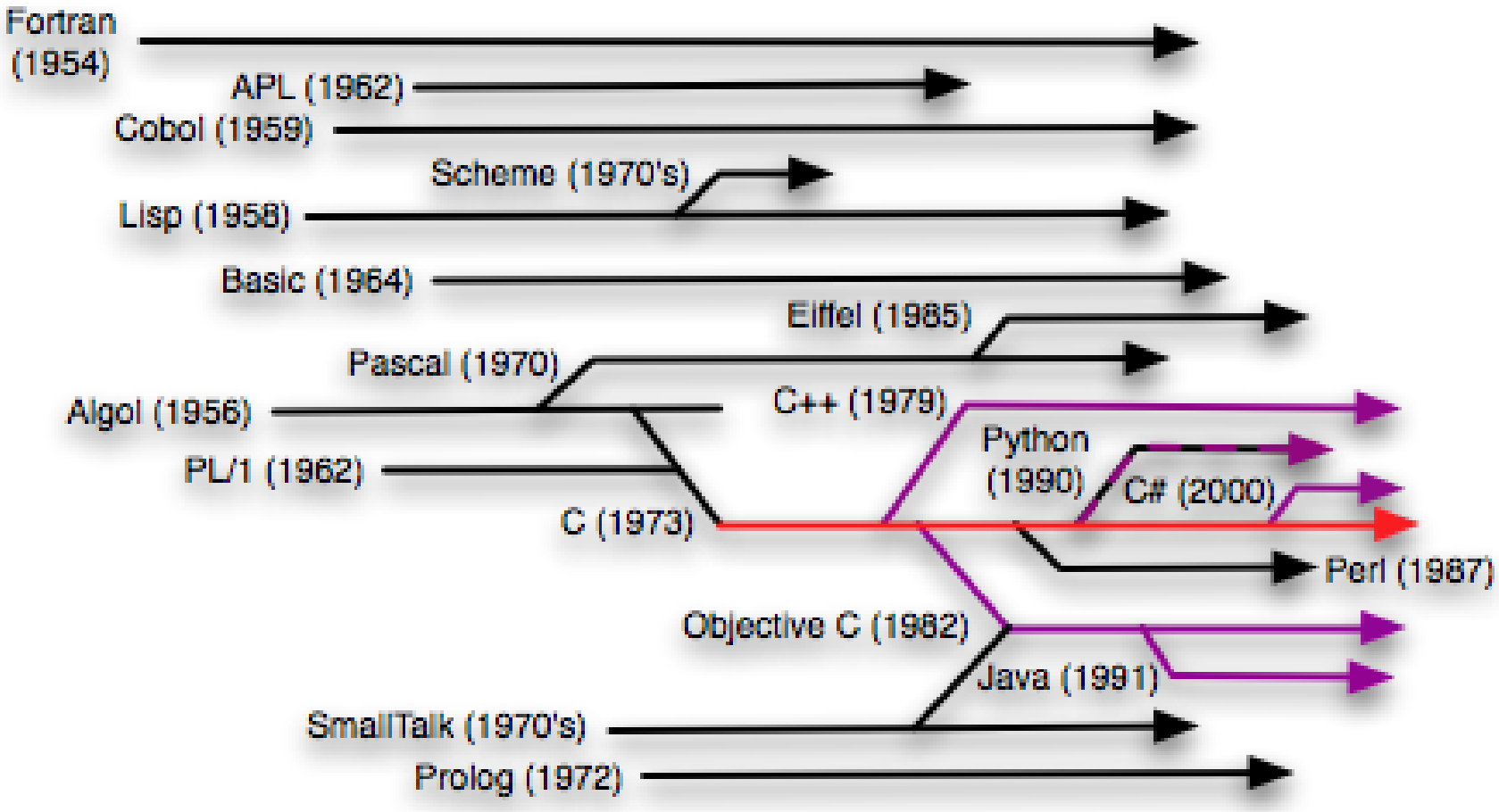


Survival of the Fittest

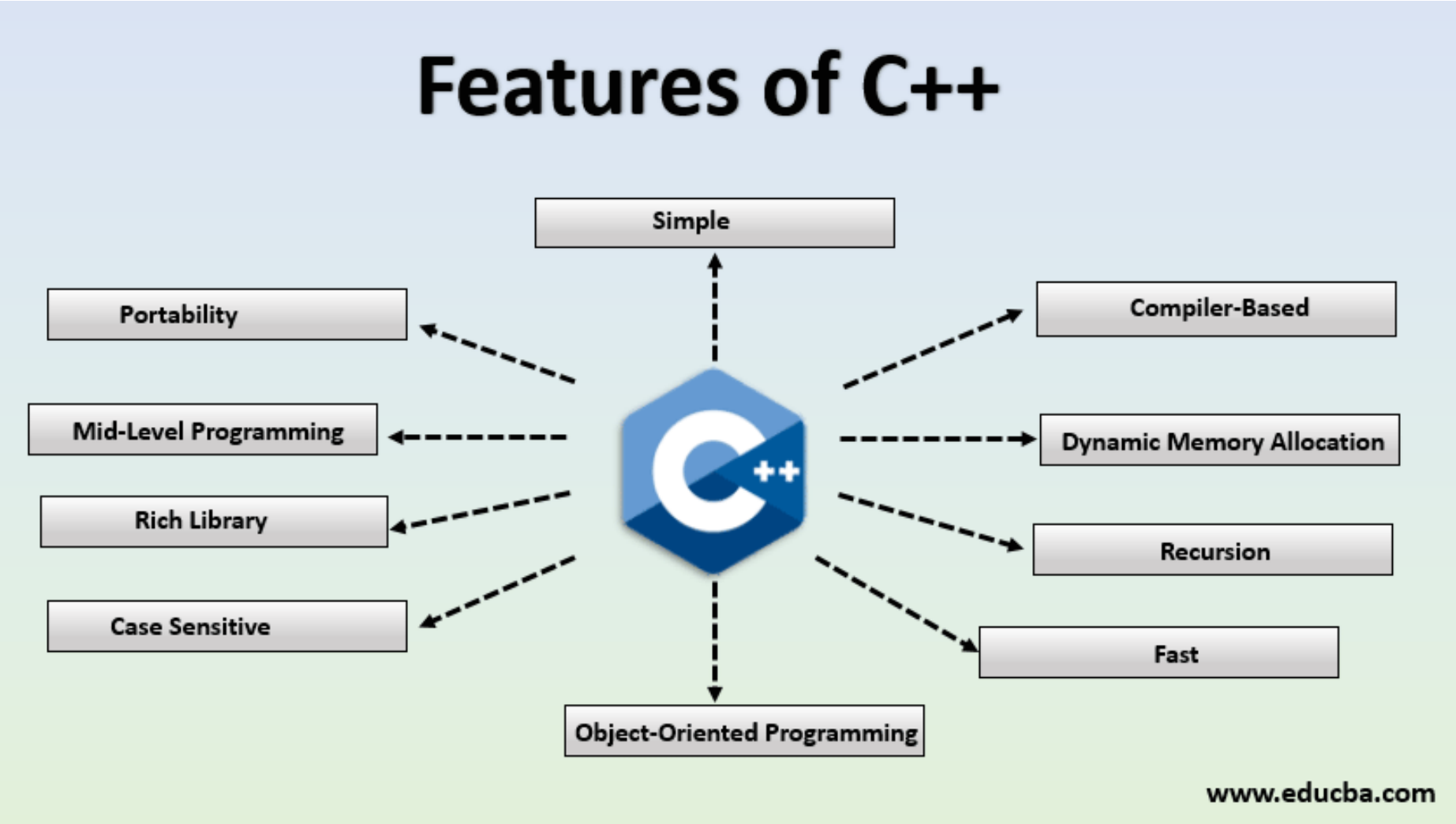
Reasons a language endures, with examples of some classic tongues

- **Appeals to a wide audience** C (bolstered by the popularity of Unix)
- **Gets a job done** Cobol (designed for business-report writing)
- **Delivers new functionality** Java (runs on any hardware platform)
- **Fills a niche** Mathematica (speeds up complex computations)
- **Offers a modicum of elegance** Icon (has friendly, line-oriented syntax)
- **Has a powerful user base or backer** C# (developed by Microsoft for .Net)
- **Has a charismatic leader** Perl (programmer-author Larry Wall)

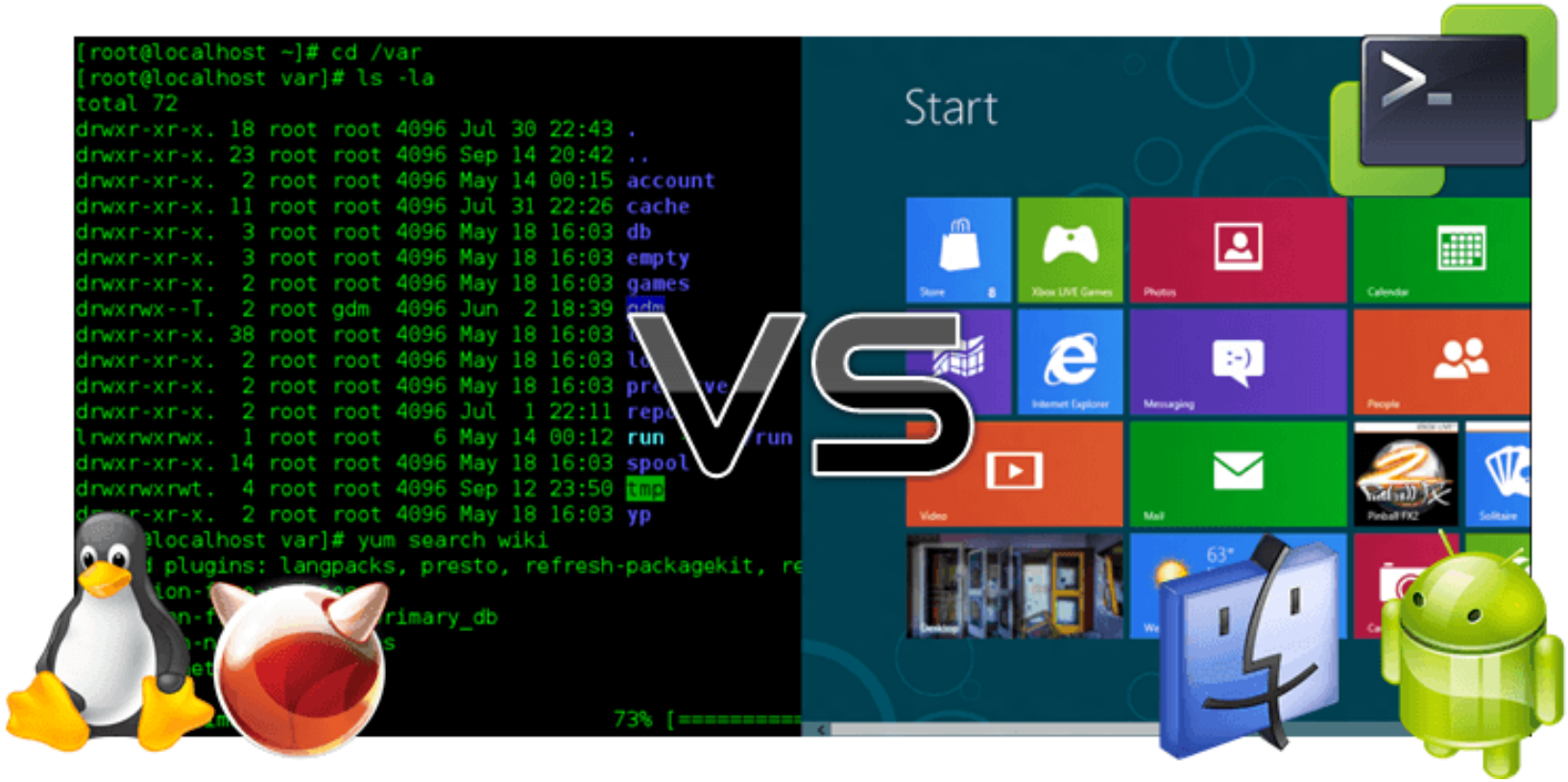
Computer Languages



Why C++


Features of C++
Simple
Portability
Compiler-Based
Mid-Level Programming
Dynamic Memory Allocation
Rich Library
Recursion
Case Sensitive
Fast
Object-Oriented Programming
www.educba.com

CLI vs GUI


[root@localhost ~]# cd /var
[root@localhost var]# ls -la
total 72
drwxr-xr-x. 18 root root 4096 Jul 30 22:43 .
drwxr-xr-x. 23 root root 4096 Sep 14 20:42 ..
drwxr-xr-x. 2 root root 4096 May 14 00:15 account
drwxr-xr-x. 11 root root 4096 Jul 31 22:26 cache
drwxr-xr-x. 3 root root 4096 May 18 16:03 db
drwxr-xr-x. 3 root root 4096 May 18 16:03 empty
drwxr-xr-x. 2 root root 4096 May 18 16:03 games
drwxrwx--T. 2 root gdm 4096 Jun 2 18:39 gdm
drwxr-xr-x. 38 root root 4096 May 18 16:03
drwxr-xr-x. 2 root root 4096 May 18 16:03
drwxr-xr-x. 2 root root 4096 May 18 16:03
drwxr-xr-x. 2 root root 4096 Jul 1 22:11
lrwxrwxrwx. 1 root root 6 May 14 00:12 run
drwxr-xr-x. 14 root root 4096 May 18 16:03 spool
drwxrwxrwt. 4 root root 4096 Sep 12 23:50 tmp
drwxr-xr-x. 2 root root 4096 May 18 16:03 yp
localhost var]# yum search wiki
plugins: langpacks, presto, refresh-packagekit,
rimary_db
73% [
VS
Start

Command Line Interface - Basic Navigation

Many of the PowerShell commands have been aliases to have corresponding commands that match Linux. These are the commands that we use below, as it will allow us to have a common set across platforms. If there are exceptions, they will be explicitly noted below.

- pwd: Show the present working directory.
- ls: To get the list of all the files or folders.
- cd: Used to change the directory.
- du: Show disk usage.
- man: Used to show the manual of any command present in Linux.

Command Line Interface - File and Directory Manipulation

- mkdir: Used to create a directory if not already exist. It accepts directory name as input parameter.
- rmdir: It is used to delete a directory if it is empty.
- cp: This command will copy the files and directories from source path to destination path. It can copy a file/directory with new name to the destination path. It accepts source file/directory and destination file/directory.
- mv: Used to move the files or directories. This command's working is almost similar to cp command but it deletes copy of file or directory from source path.
- rm: Used to remove files or directories.
- touch: Used to create or update a file.

Command Line Interface - Displaying the file contents

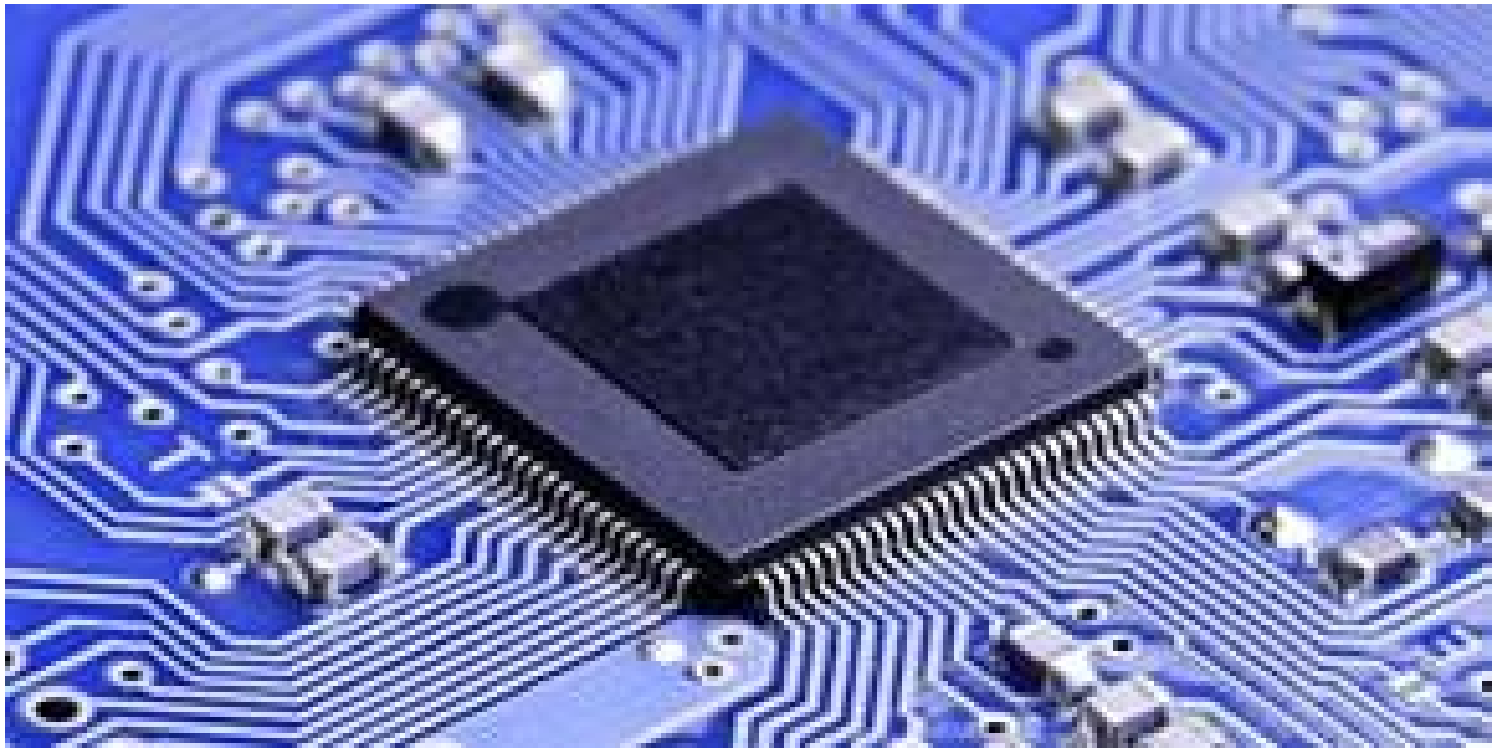
- cat: It is generally used to concatenate the files. It gives the output on the standard output.
- more: It is a filter for paging through text one screenful at a time.
- less: It is used to viewing the files instead of opening the file.Similar to more command but it allows backward as well as forward movement.
- head: Used to print the first N lines of a file. It accepts N as input and the default value of N is 10.
- tail: Used to print the last N-1 lines of a file. It accepts N as input and the default value of N is 10.

Commands can be "piped" together: ls | more

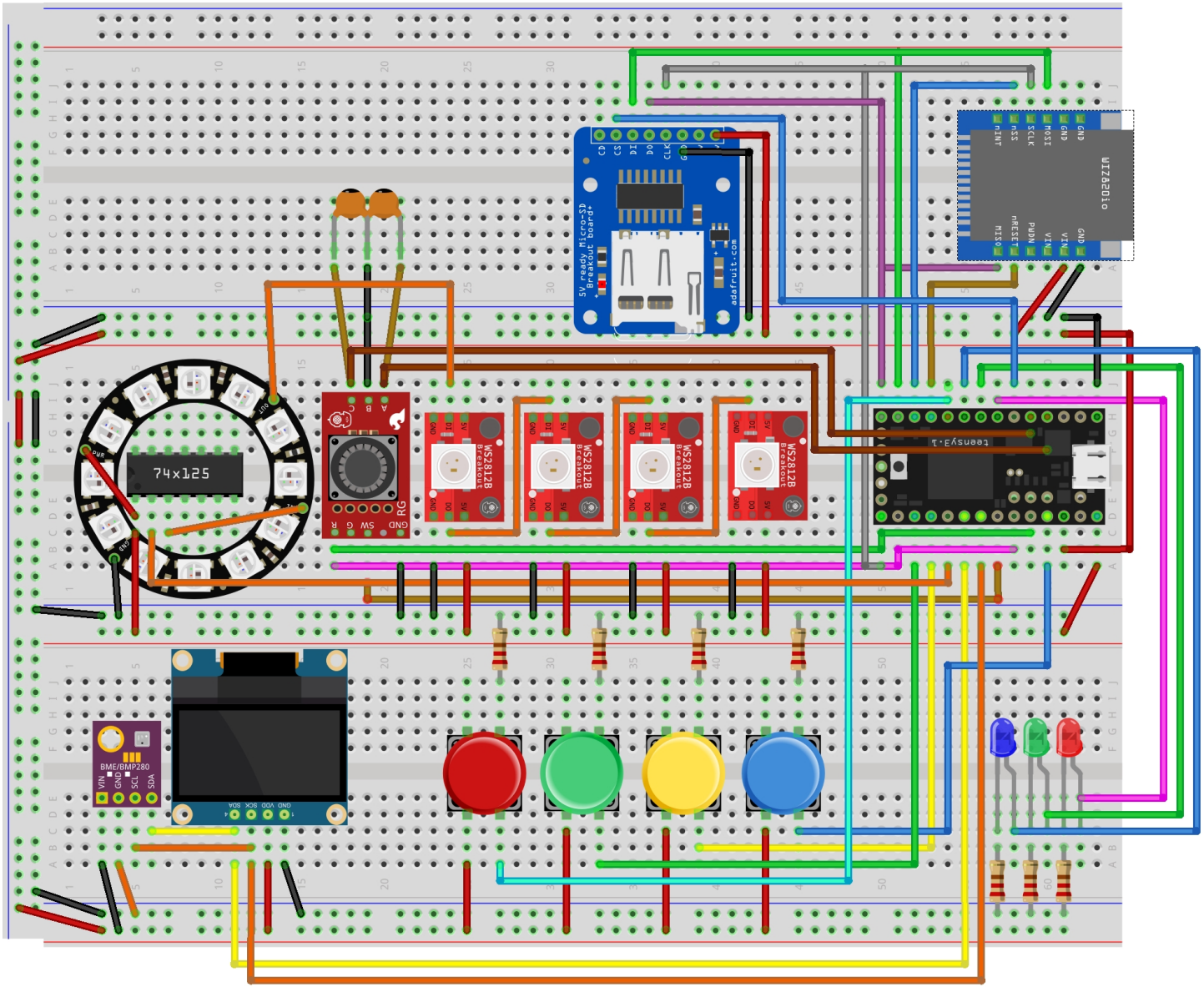
Powershell Commands Simplified

```
mkdir <directory name>  //make a directory
rmdir <directory name> //remove directory
ls //list contents of current directory
pwd //print path of current directory
cd <directory name> // change directory
cd .. //change directory to the directory above
new-item <file name> //create a file
cp <file name> <new file name> //copy file
mv <file name> <directory> //move file to new dir
rm <file name> //delete file
cat <file name> //display contents of file
```

Our First Microcontroller

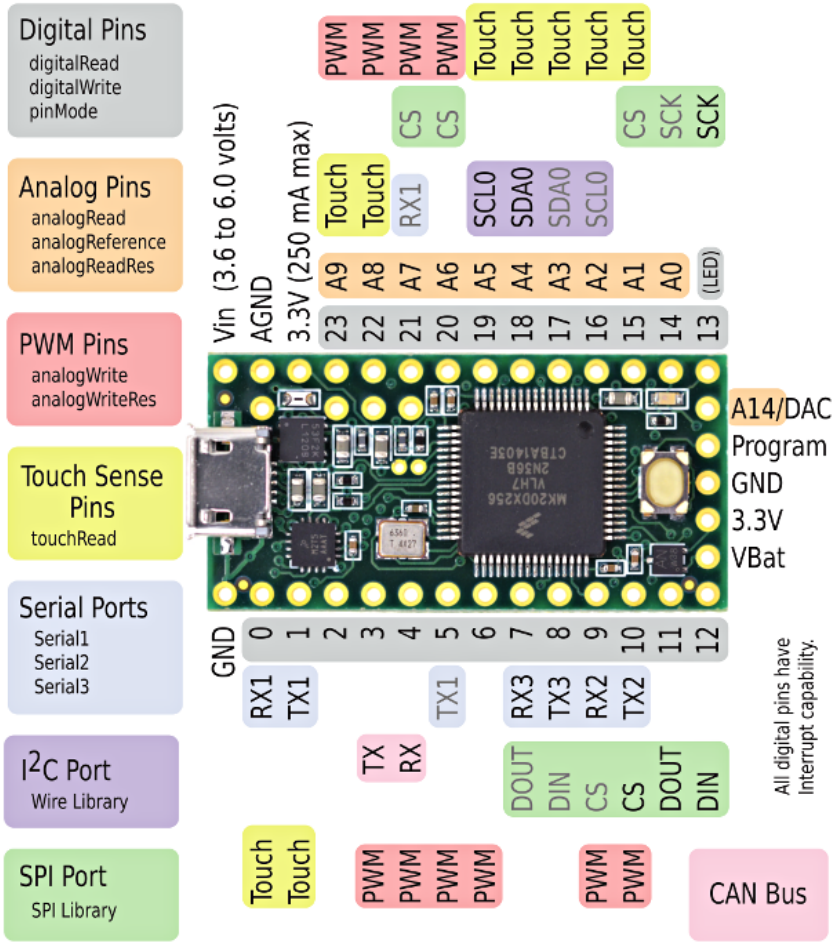


Smart Room Controller



Teensy 3.2

- Cortex-M4 72MHz (overclocked to 96 MHz)
- 34 GPIO pins
- 3.3V and 5.0V operating voltages
- 500mA of available power with USB



Arduino IDE for Teensy

We are going to start off using the Arduino IDE[1]. The Arduino IDE is programmed essentailly using C++ code, but make the compiling and loading onto the microcontroller simplier.

We begin by installing the Arduino IDE:
https : *//www.arduino.cc/en/main/software*

Then, we install the Teensyduino add-on:
https : *//www.pjrc.com/teensy/td_download.html*

[1] An IDE, or Integrated Development Environment, enables programmers to consolidate the different aspects of writing a computer program.

Other Software

1. Fritzing
   - IoT Bootcamp Teams Site
2. Drawio
   - `https://app.diagrams.net/`
3. Fusion 360
   - Instructional Videos - Teams Site
4. Formlab's Preform
   - `https://formlabs.com/software/`
5. Ultimaker's Cura
   - `https://ultimaker.com/software/ultimaker-cura`
6. Git
   - `https://git-scm.com/downloads`

GITHUB Simplified

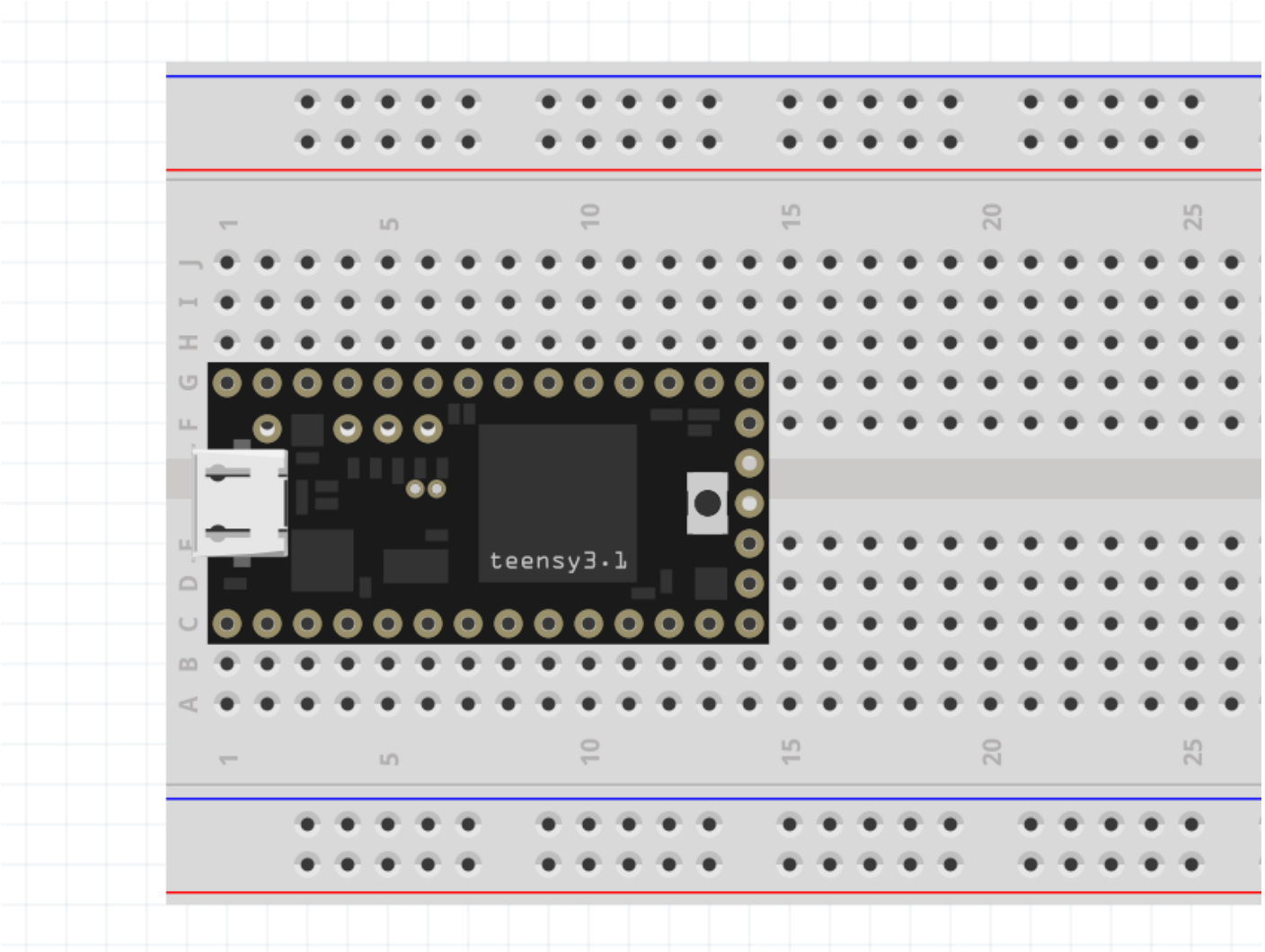
```
// In Powershell go to ./Documents/<yourname>
// Get a repository that already exists and pull
   it into your local machine
  git clone <URL of repository>

// From the repository directory, get updates
  git pull

// Send your changes up to the repository
  git add .
  git commit -m "<comment>"
  git push

// You may get asked to enter your GIT username
git config --global user.email "you@example.com"
```

Teensy on Breadboard



Basic Structure of Arduino Sketch

```
1  // the "header" is used for GLOBALS
2
3  void setup() {
4    // code in setup() runs once
5    // it is used to initialize objects,
6    // begin processes, and set variables
7    pinMode(13, OUTPUT) // set Pin 13 as an Output
8  }
9
10 void loop() {
11   // functionality of your code
12   // this loops indefinitely
13 }
```

Class Assignments

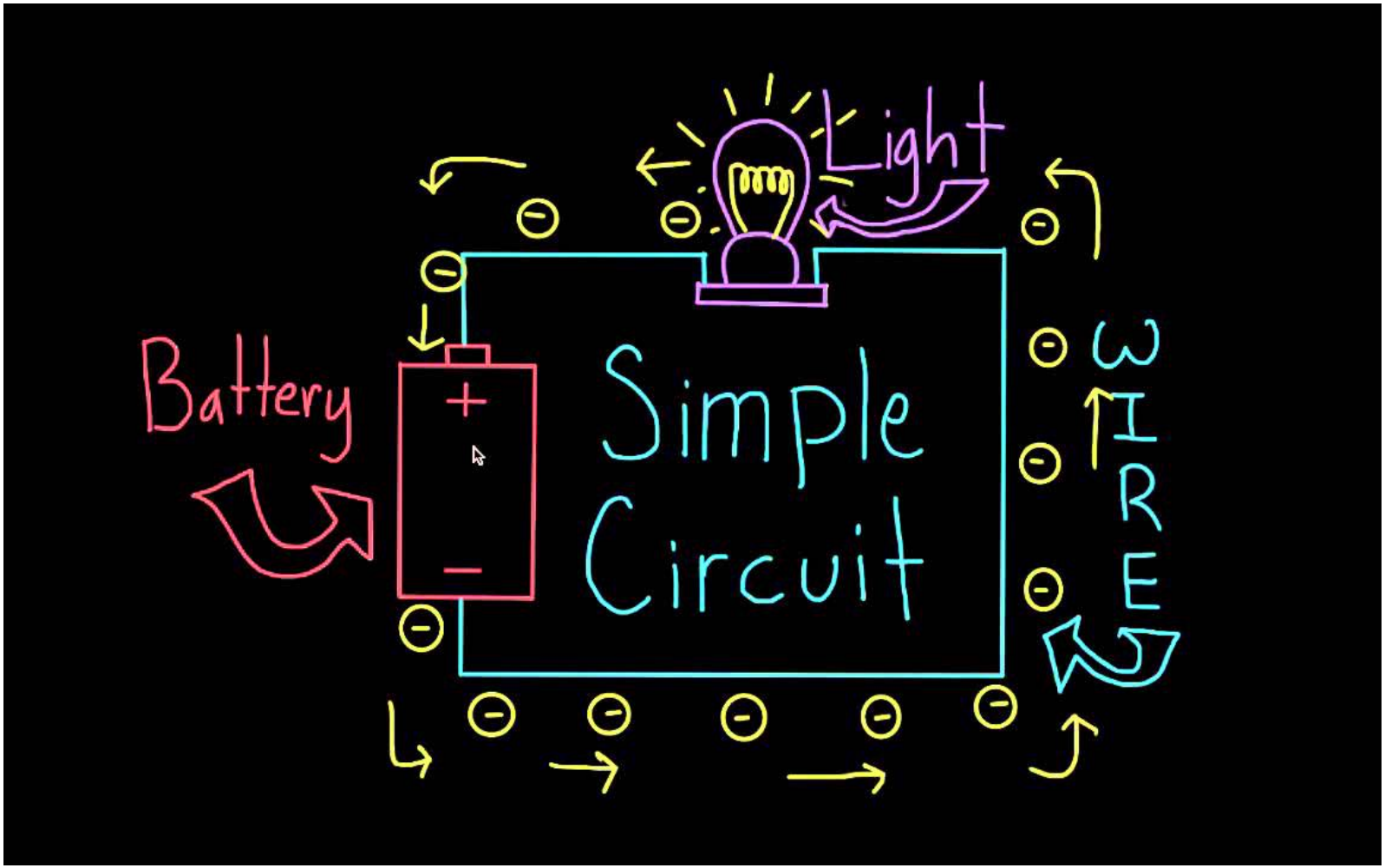
1. Labbook - flow chart
2. Labbook - schematic
3. Fritzing breadboad layout
4. Arduino code with comments

```
/*
 * Project:      Title of Project
 * Description:  Description of Project
 * Author:       Your Name
 * Date:         Today's Date
 */
```

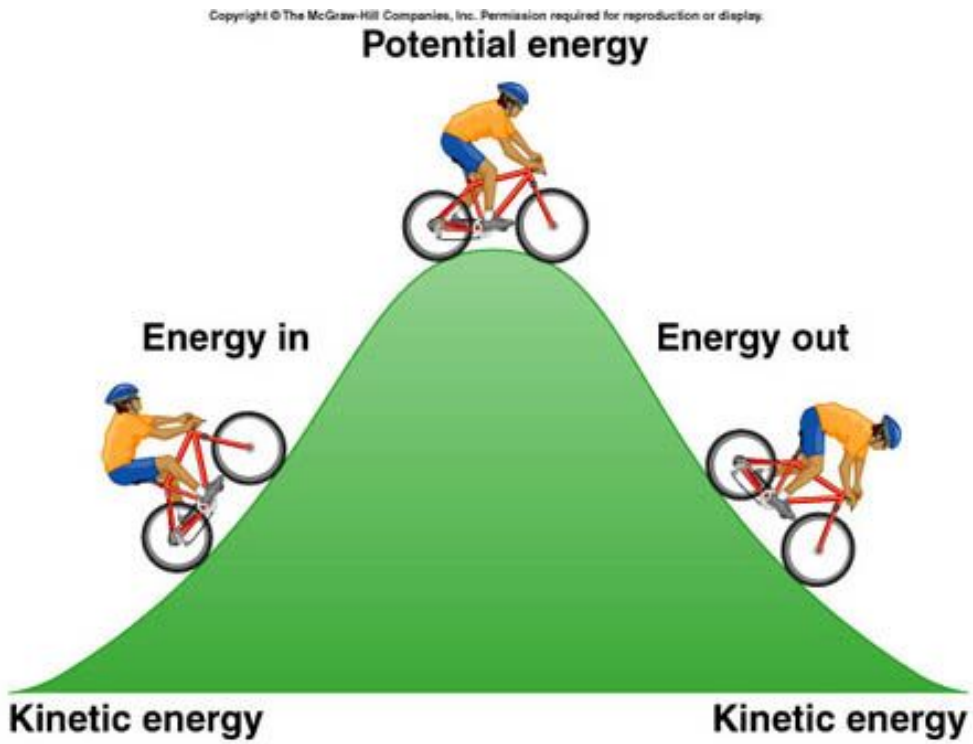
Assignment L01_01: Hello World

- Use pinMode
- Turn on LED with digitalWrite()
- Delays

Introduction to Electrical Circuits

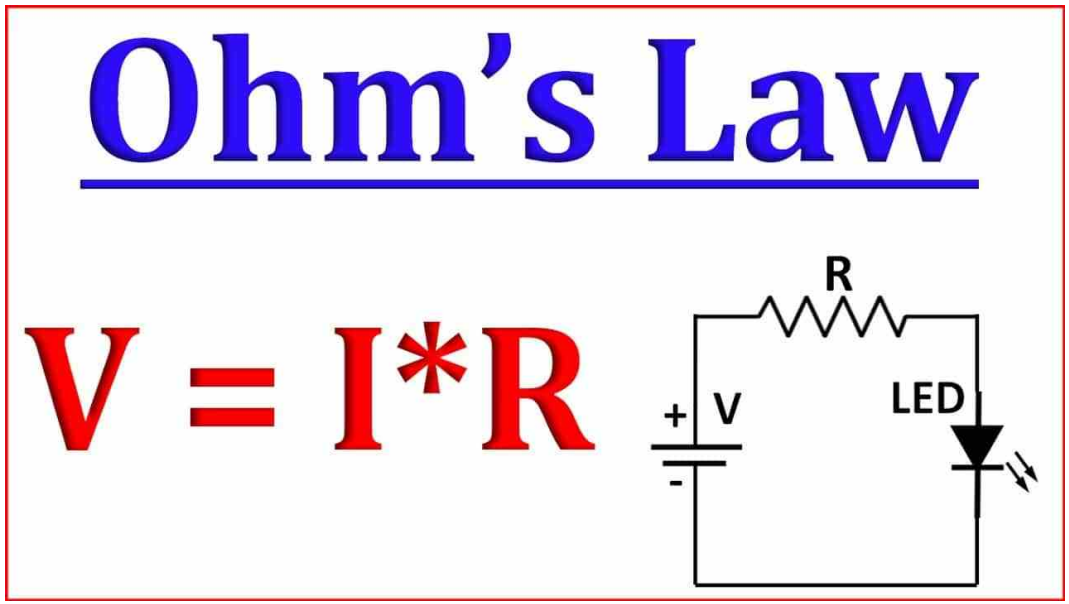


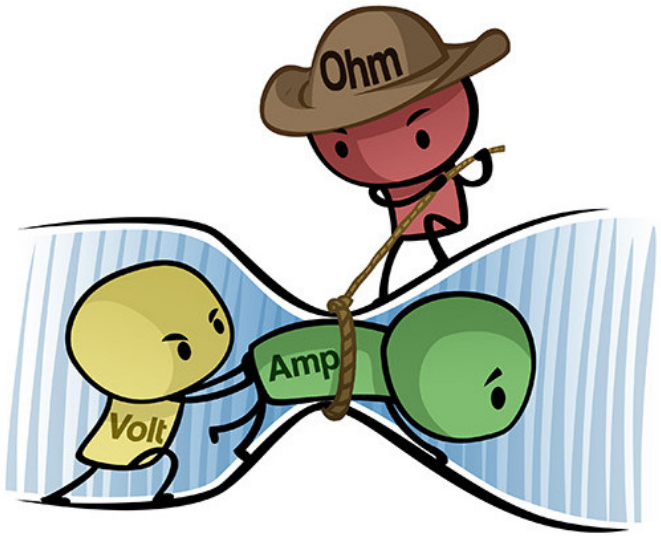
Energy


Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.
Potential energy
Energy in
Energy out
Kinetic energy
Kinetic energy

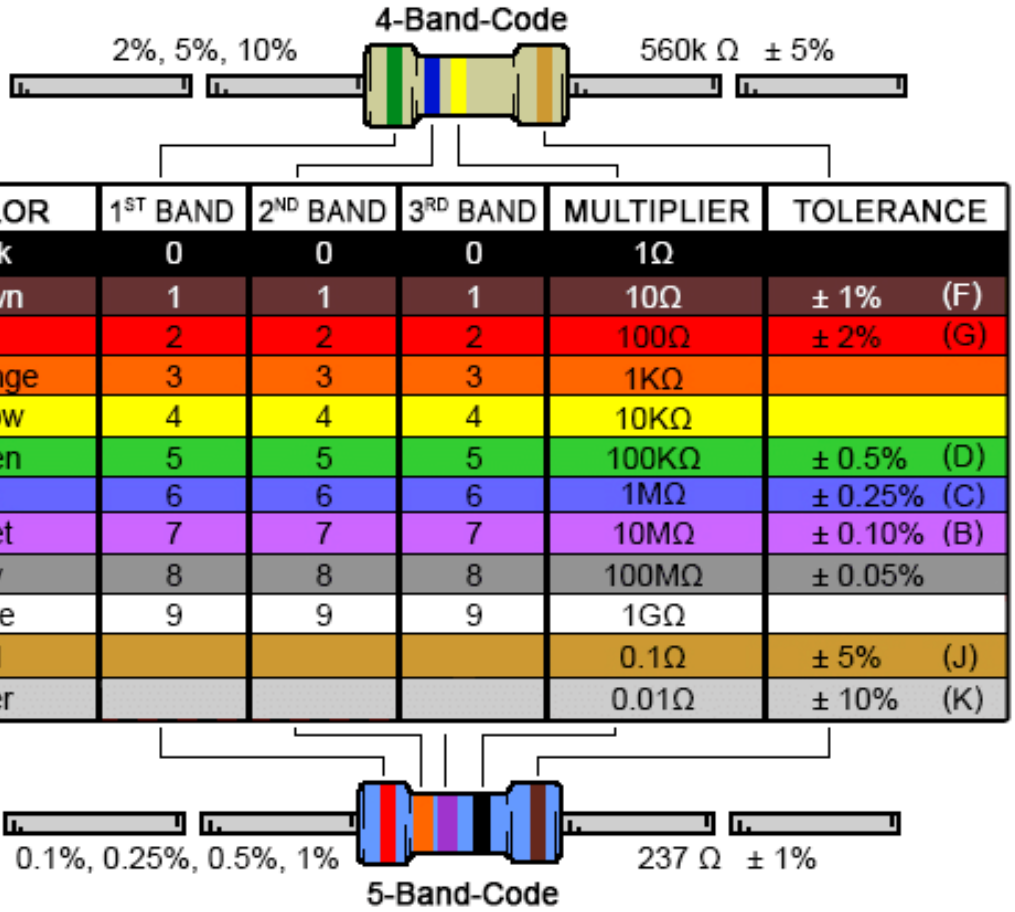
Ohm's Law

Georg Ohm (16 March 1789 – 6 July 1854) was a German physicist and mathematician. As a school teacher, Ohm began his research with the new electrochemical cell, invented by Italian scientist Alessandro Volta. Ohm found that there is a direct proportionality between the potential difference (voltage) applied across a conductor and the resultant electric current. This relationship is known as Ohm's law:



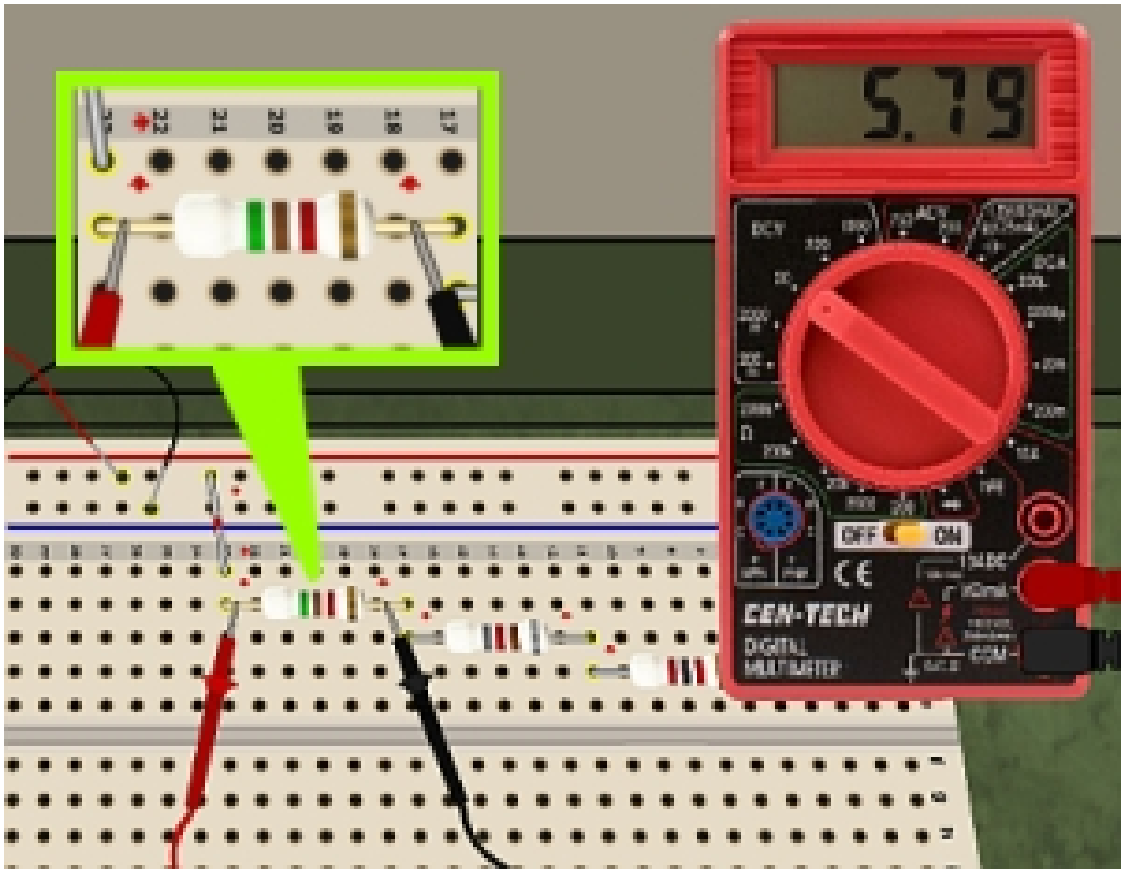


Resistor Color Bands

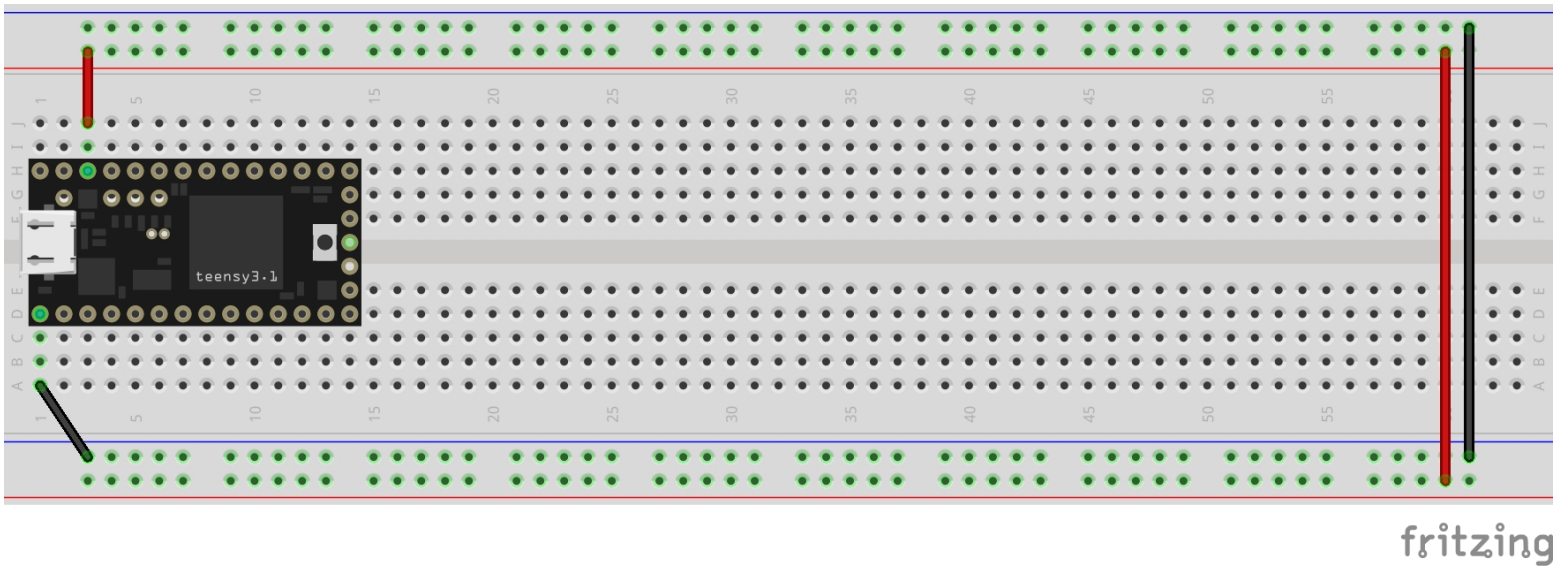


COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8	100MΩ	± 0.05%
White	9	9	9	1GΩ	
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

Measuring Voltage, Current, and Resistance



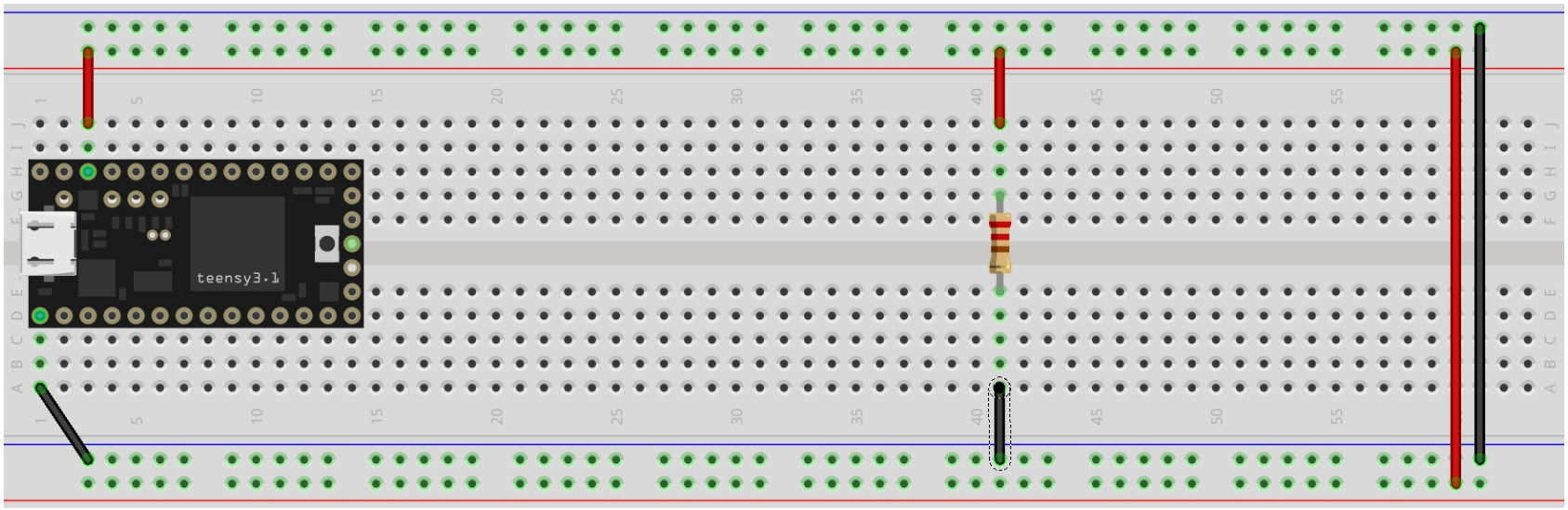
Power from the Teensy 3.2



The Teensy 3.2 has three pins related to power:

- 3.3V: 250mA of power to be used for most hardware
- V_{in}: 5V from the USB cable to power 5V hardware
- GND: The ground pin to close the electrical loop

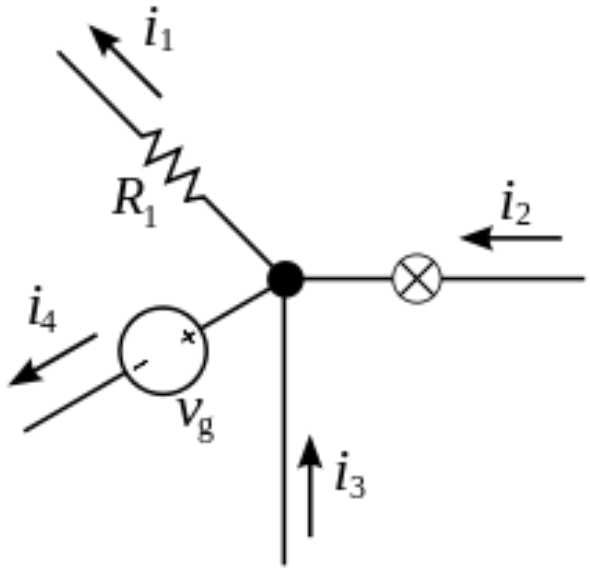
One Resistor



Kirchhoff's First Law

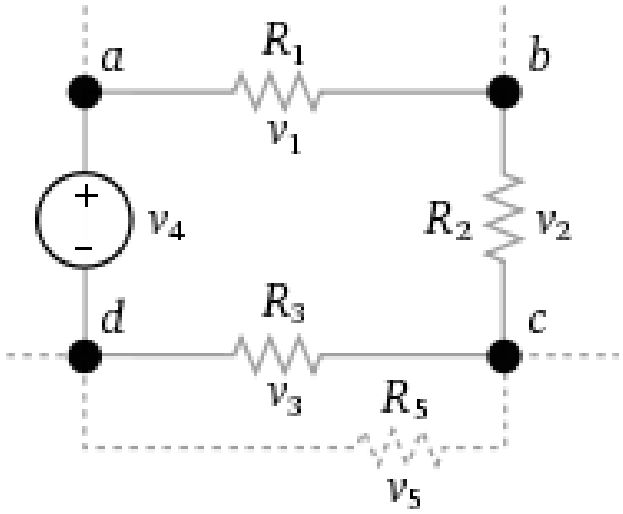
Gustav Robert Kirchhoff (12 March 1824 – 17 October 1887) was a German physicist who contributed to the fundamental understanding of electrical circuits. His first law:

> In an electrical circuit, the sum of currents flowing into that node is equal to the sum of currents flowing out of that node

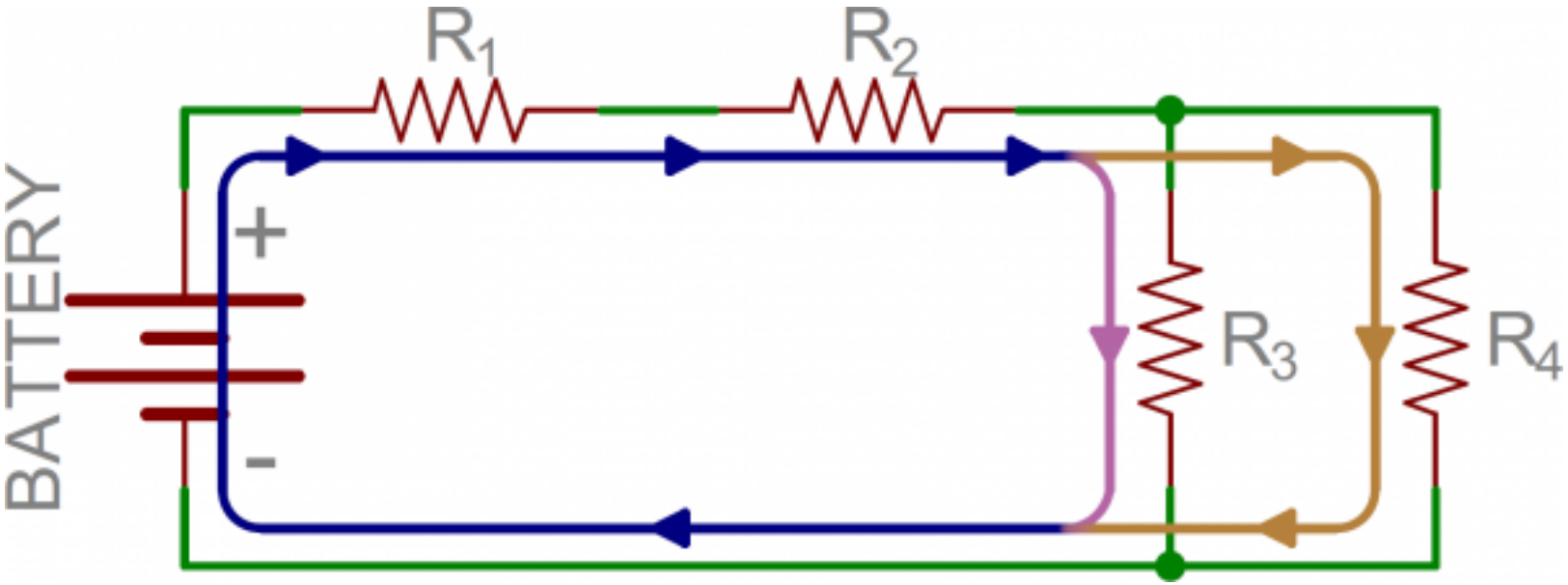


Kirchhoff's Second Law

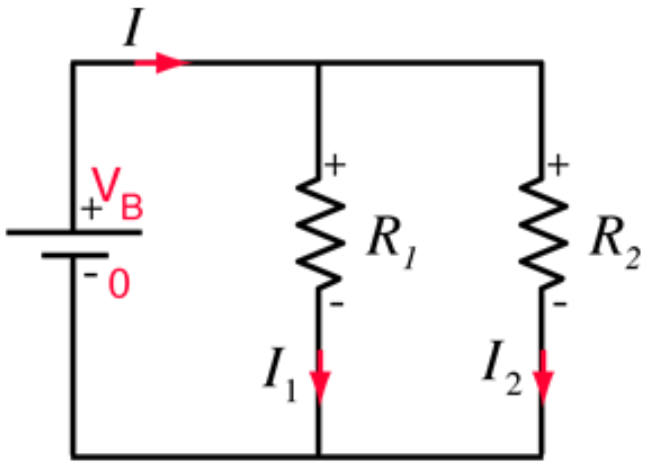
The directed sum of the potential differences (voltages) around any closed loop is zero.



Resistors in Series and Parallel

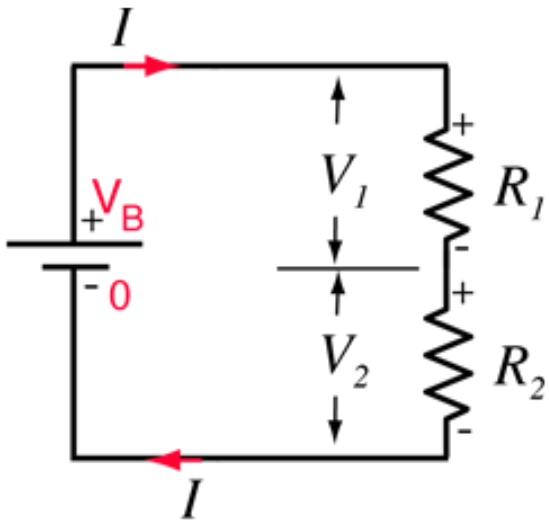


Resistors in Series and Parallel



Parallel resistors

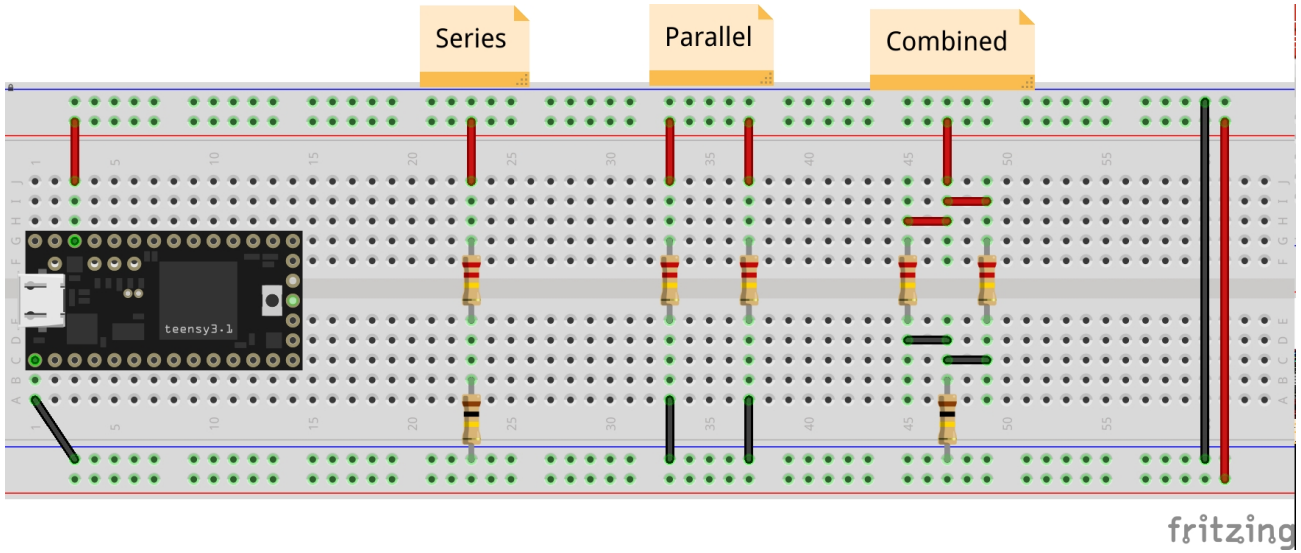
$$\frac{1}{R_{equivalent}} = \frac{1}{R_1} + \frac{1}{R_2}$$



Series resistors

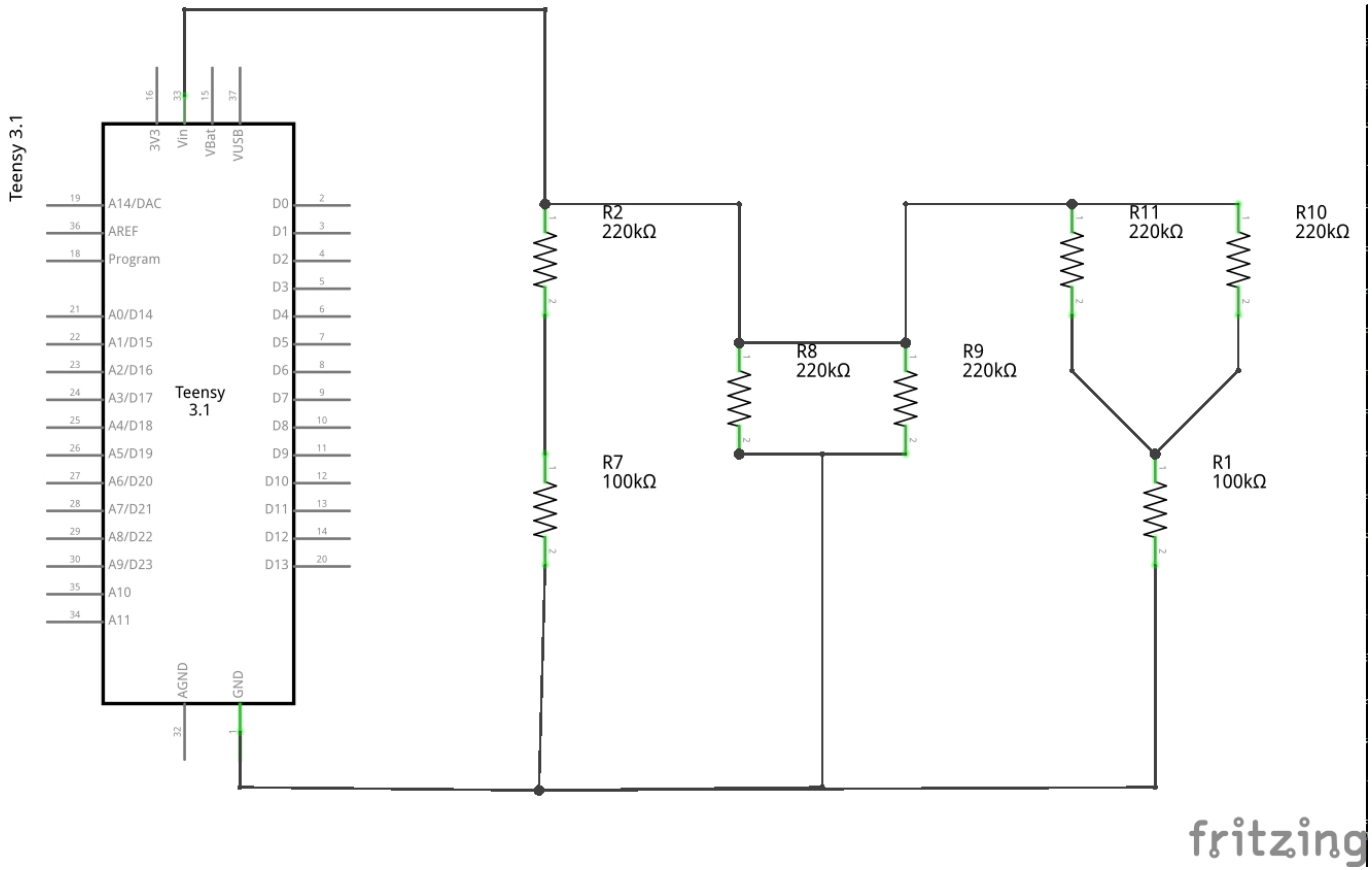
$$R_{equivalent} = R_1 + R_2$$

Resistor Lesson



- In your lab notebook, draw the circuit diagrams
  1. Series: $220k\Omega$ and $100k\Omega$
  2. Parallel: $220k\Omega$ and $220k\Omega$
  3. Combined: Two $220k\Omega$ in series with $100k\Omega$
- Calculate the combined resistance and the voltage at each point
- Create Fritzing diagram
- Build (one at a time*) on your breadboard and test with multimeter

Schematics in Fritzing



In Fritzing, go to the Schematic tab and layout your resistors.

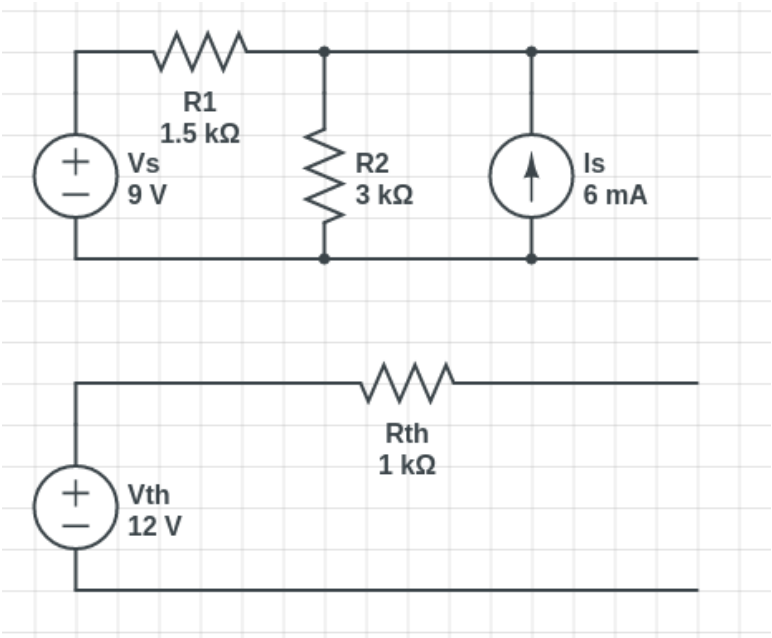
NOTE: The schematic for the Teensy does not match it's physical layout.

Thevenin Equivalent Circuits

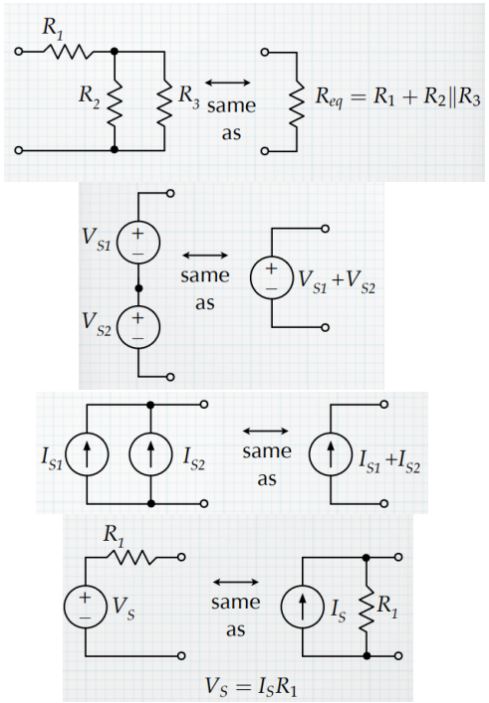
Léon Charles Thévenin (30 March 1857 – 21 September 1926) was a French telegraph engineer who extended Ohm's law to complex circuits.

Any combination of batteries and resistances with two terminals can be replaced by a single voltage source V_{th} and a single series resistor R_{th}.

Equivalent Circuit

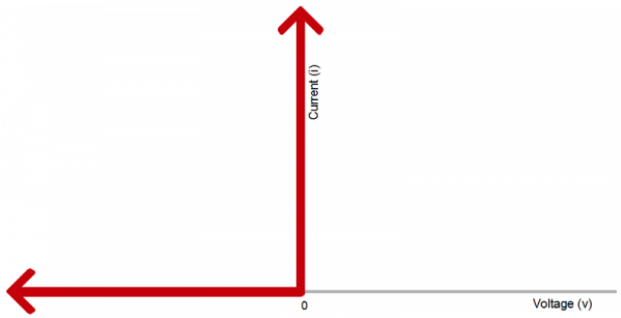


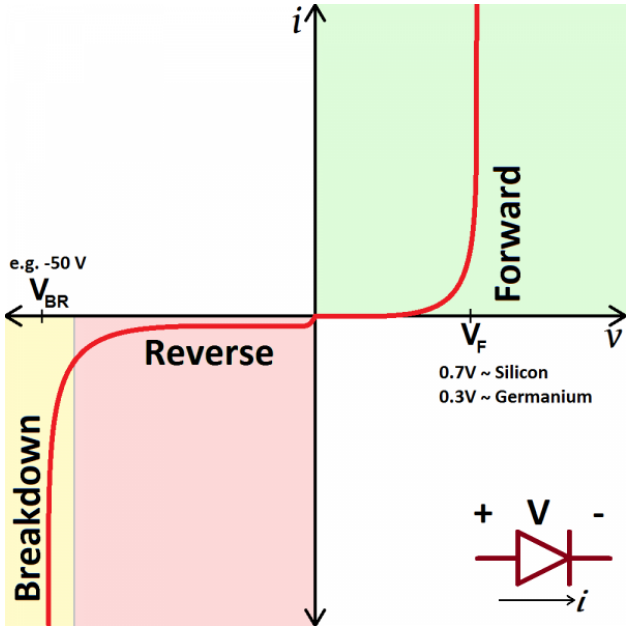
Useful relationships



Diodes

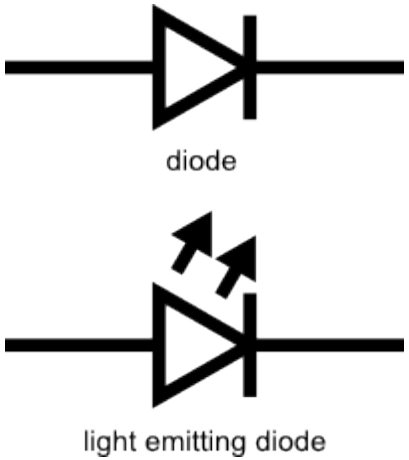
The key function of an diode is to control the direction of current-flow. Current passing through a diode can only go in one direction, called the forward direction. Current trying to flow the reverse direction is blocked.





Light Emitting Diodes

LEDs (that's "ell-ee-dees") are a particular type of diode that convert electrical energy into light.

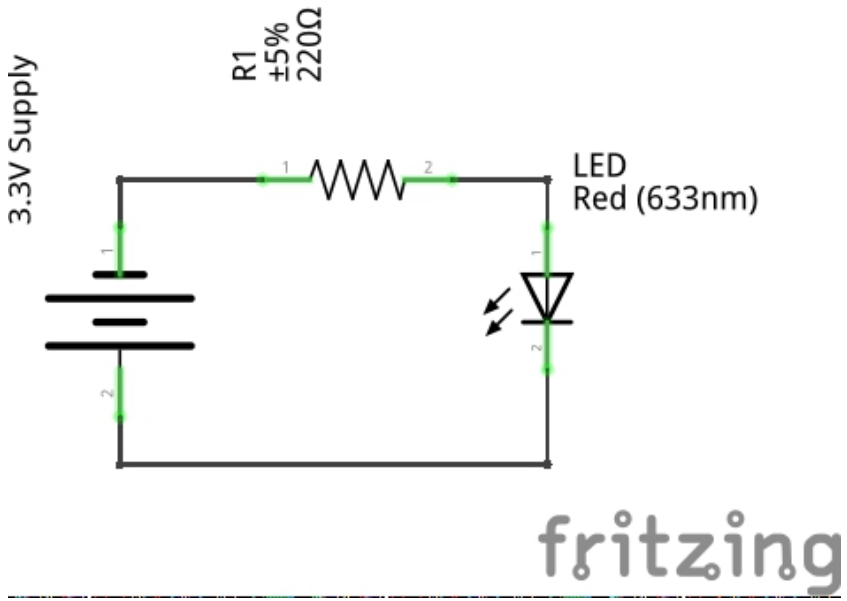


Current Limiting Resistors

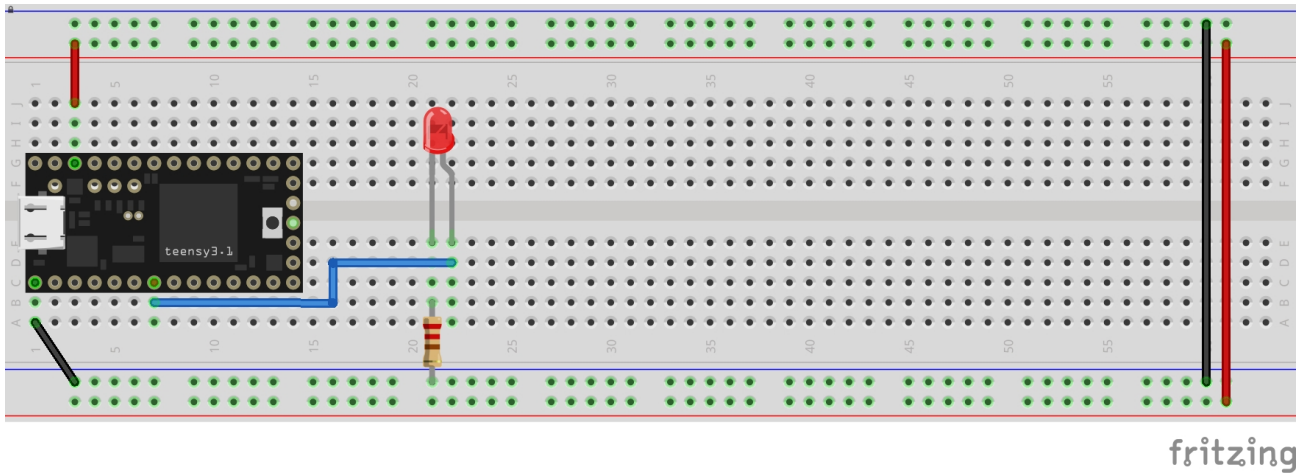
As a LED has very little resistance, when it is connected directly to a power supply, the current draw will exceed its specs and it will burn out.

$$V_{pp} - V_{LED} = IR \implies R >= \frac{V_{pp} - V_{LED}}{I_{max}}$$

For a 3.3V power supply, a 0.43V across the LED, and a max current of 100mA, the resistor needs to be greater than 29Ω.



Assignment L02_01_helloLED



- Using Pin 5 as an output and the appropriate current limiting resistor, blink the LED once per second.
- Change the resistor to $1k\Omega$ and then $10k\Omega$, what happens to the brightness?

REMEMBER: Labbook, Fritzing, breadboard, then code w/comments

Constants and Variables

It is often useful to give a name to something that will be used repeatedly in the code. Then can be constants or variables:

- Constant is a declaration that does not change throughout the code. For example a the pin that an LED is attached to.
- Variable is a declaration that changes as the code processes. For example, a counter or index.

The use of Constants and Variables has several advantages:

- It improves readability by assigning names to items
- Items can be changed by changing a single declaration
- It allows the code to do math

The first two Data Types that we will be using:

- **int**: an Integer between $\pm 675,376,778$
- **float**: a Floating point number with 7-digits precision

Constant and Variables Example

```
1  const int ledPin = 5;
2  const int ledDelay = 1000;
3  int i;
4
5  void setup() {
6    pinMode(ledPin, OUTPUT) // set ledPin as Output
7    i = 100;
8  }
9  void loop() {
10   digitalWrite(ledPin,HIGH);
11   delay(ledDelay);
12   digitalWrite(ledPin,LOW);
13   delay(ledDelay+i);
14   i = i + 100;
15 }
```

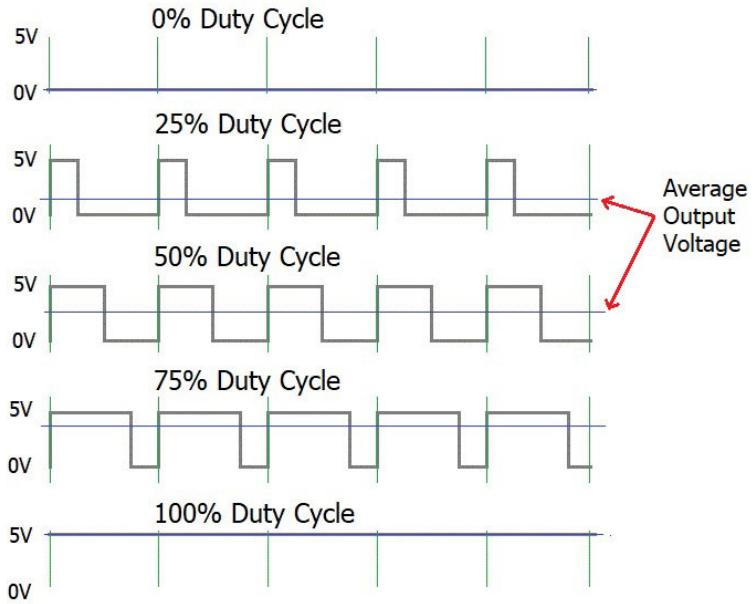
Assignment L02_02_helloLEDvar

- Convert L02_01_helloLED with constants and/or variables.

Pulse Width Modulation

Software Configurable:

- Digital Input: High/Low (3.3V/0V)
- Digital Output: High/Low (3.3V/0V)
- Analog Input: 0V to 3.3V
- Analog Output: 0V to 3.3V PWM



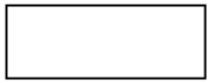
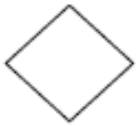
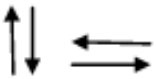
Assignment L02_03_helloLEDanalog

Use analogWrite to change the brightness of the LED, using values:

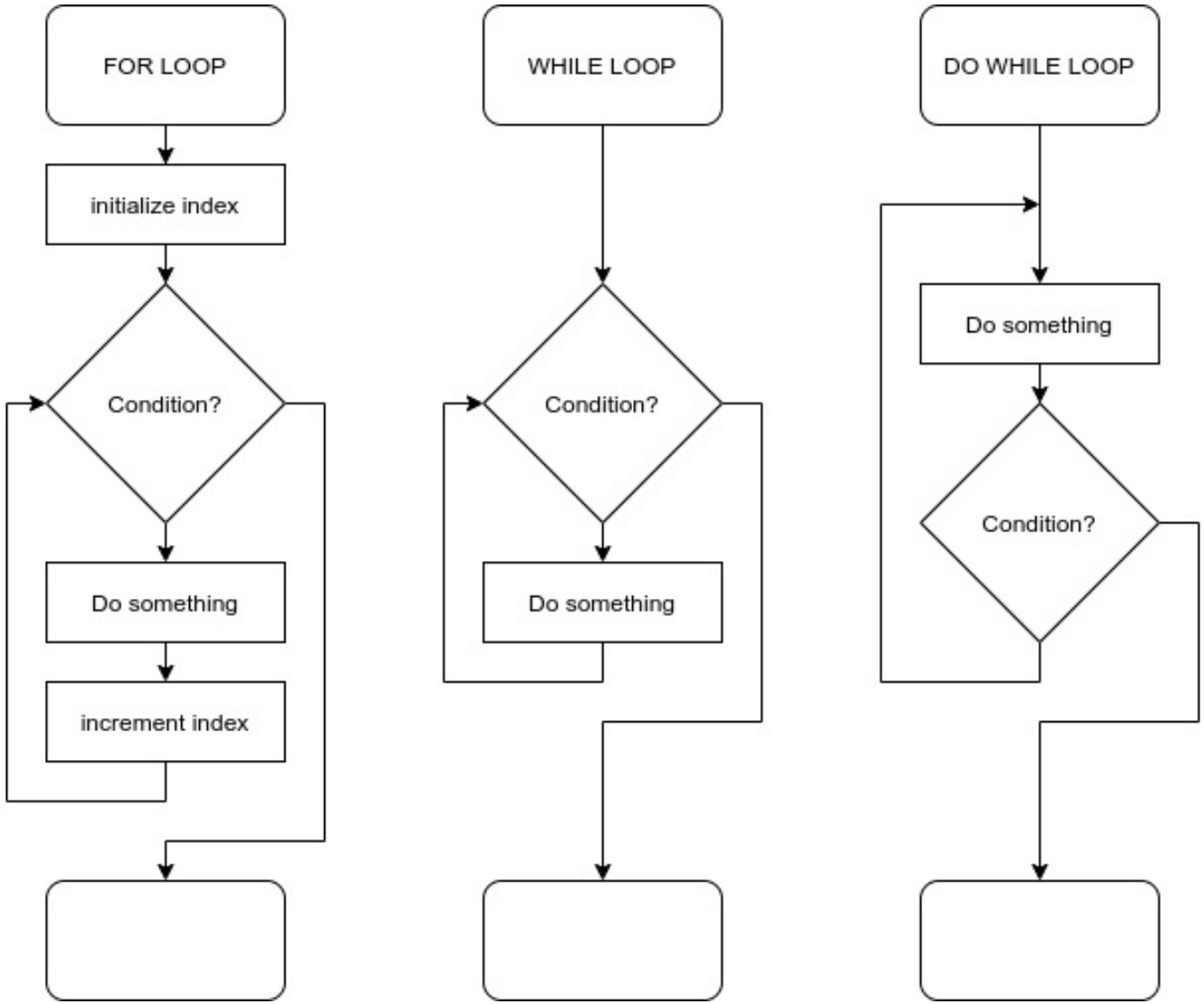
- 255
- 63
- 171
- 16

Measure the voltage with your multimeter at each value.

Flowcharts

Symbol	Name	Function
	Process	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.

Loops


FOR LOOP
initialize index
Condition?
Do something
increment index
WHILE LOOP
Condition?
Do something
DO WHILE LOOP
Do something
Condition?

FOR Loop syntax

```
1 // FOR loop syntax
2 for (initialization; condition; increment) {
3   // statement(s);
4 }
5
6 // EXAMPLE
7 for (j=0; j <= 255, j++) {
8   analogWrite(ledPin, j);
9   }
```

WHILE loop syntax

```
1  // WHILE loop syntax
2  while (condition) {
3    // statement(s)
4  }
5
6
7  // EXAMPLE
8  while (button == HIGH) {
9    digitalWrite(ledPin, HIGH);
10 } //continue this loop until button is released
```

For vs While Loops

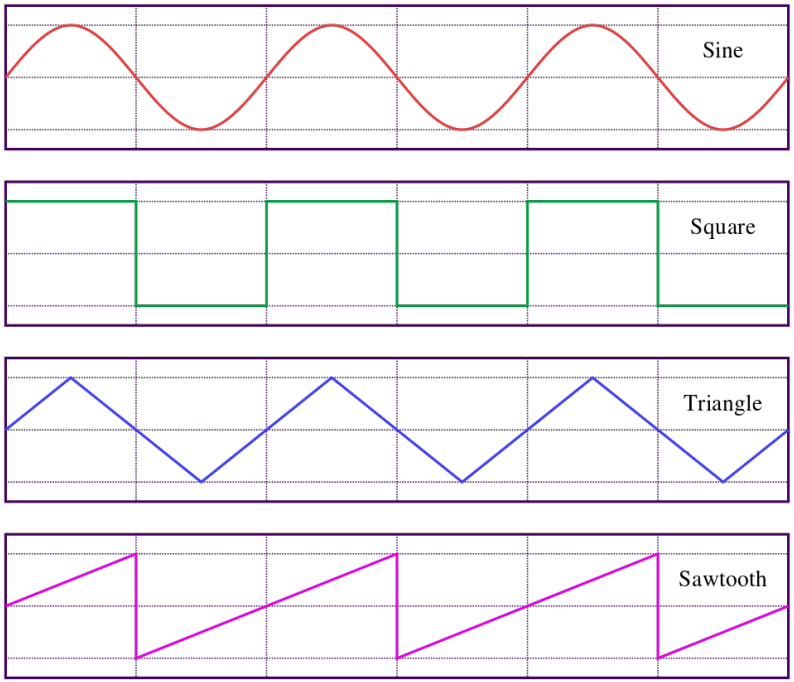
For vs While Loop

Comparison Chart

For Loop	While Loop
The for loop is used for definite loops when the number of iterations is known.	The while loop is used when the number of iterations is not known.
For loops can have their counter variables declared in the declaration itself.	There is no built-in loop control variable with a while loop.
This is preferable when we know exactly how many times the loop will be repeated.	The while loop will continue to run infinite number of times until the condition is met.
The loop iterates infinite number of times if the condition is not specified.	If the condition is not specified, it shows a compilation error.

DB Difference Between.net

Assignment L02_04_helloLEDtri



Using a FOR Loop, have the LEDs follow a Triangle Wave function from off to full brightness with a period of 10 seconds.

Header Files

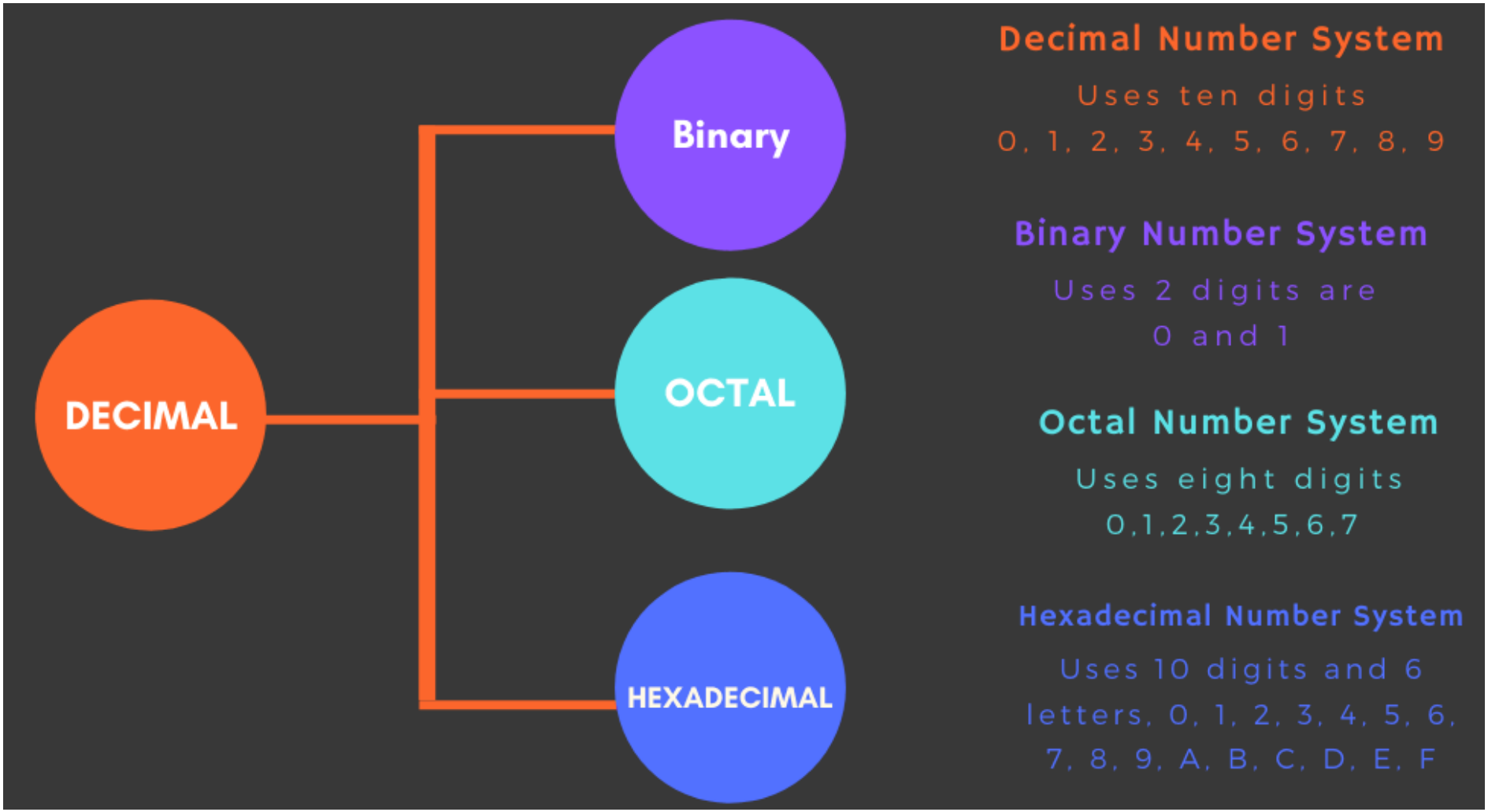
A header file is a file with the extension .h which contains C function declarations and macro definitions to be shared between several source files. There are two types of header files: those that the programmer writes and those that come with the compiler.

Both the user and system header files are included using the preprocessing directive #include. It has the following two forms:

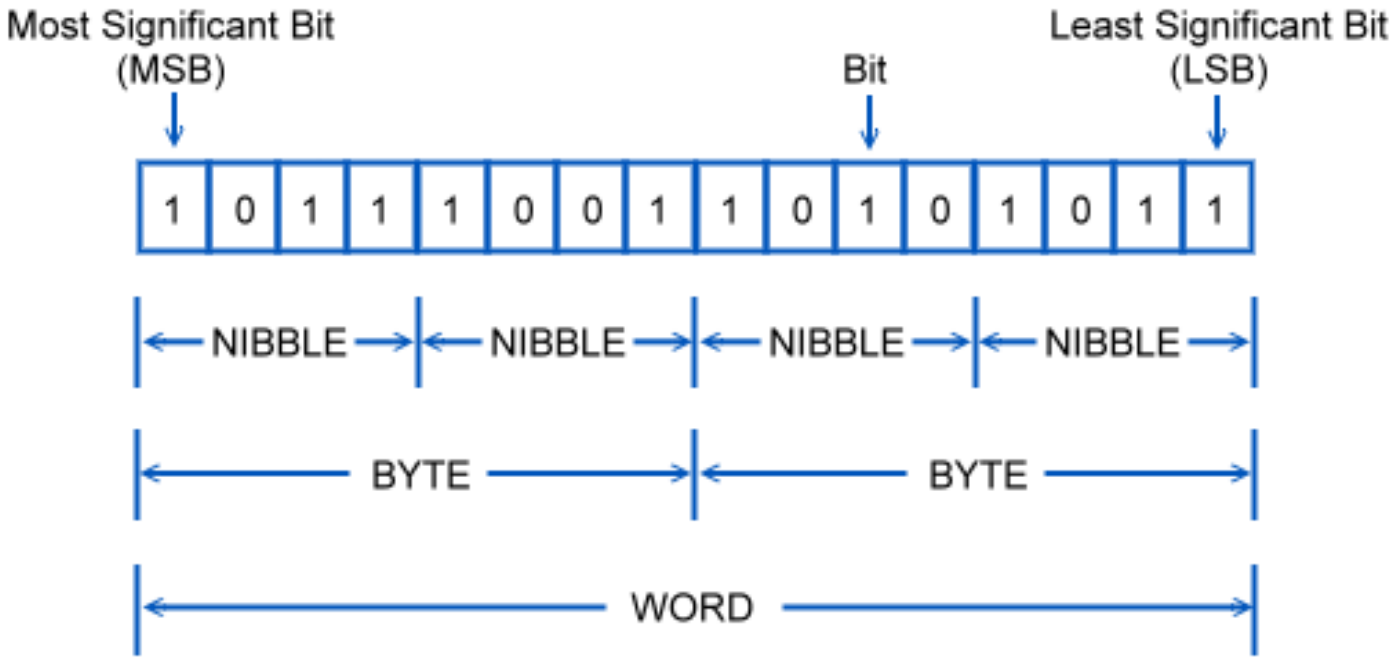
- #include $<$ *file.h* $>$ for system header files.
- #include "*file.h*" for user created header files in the directory that contains the current code.

An example is the math.h header that defines various mathematical functions.

Number Systems



Bits, Nibbles, Bytes, and Words



Data Types: Numbers

Data Type	8-bit AVR systems (Arduino Uno)			32-bit ARM systems (Teensy 3.2)		
	bytes	range (signed)	range (unsigned)	bytes	range (signed)	range (unsigned)
char	1	-128 to 127	0 to 255	1	-128 to 127	0 to 255
short	2	+/- 32,767	0 to 65,353	2	+/- 32,767	0 to 65,353
int	2	+/- 32,767	0 to 65,353	4	+/- 2,147,483,648	0 - 4,294,967,295
long	4	+/- 2,147,483,648	0 - 4,294,967,295	4	+/- 2,147,483,648	0 - 4,294,967,295
long long	8	+/- 9,223,372,036,854,770,000	0 to 18,446,744,073,709,551,615	8	+/- 9,223,372,036,854,770,000	0 to 18,446,744,073,709,551,615
float	4	3.4E +/- 38 (7 digits)	n/a	4	3.4E +/- 38 (7 digits)	n/a
double	4	3.4E +/- 38 (7 digits)	n/a	8	1.7E +/- 308 (15 digits)	n/a
long double	8	1.7E +/- 308 (15 digits)	n/a	8	1.7E +/- 308 (15 digits)	n/a
Unambiguous						
uint08_t	1	n/a	0 to 255	1	n/a	0 to 255
int08_t	1	-128 to 127	n/a	1	-128 to 127	n/a
uint16_t	2	n/a	0 to 65,353	2	n/a	0 to 65,353
int16_t	2	+/- 32,767	n/a	2	+/- 32,767	n/a
uint32_t	4	n/a	0 - 4,294,967,295	4	n/a	0 - 4,294,967,295
int32_t	4	+/- 2,147,483,648	n/a	4	+/- 2,147,483,648	n/a

There are $7.5x10^{18}$ grains of sand on Earth. A long long integer and the floating point numbers are larger than this.

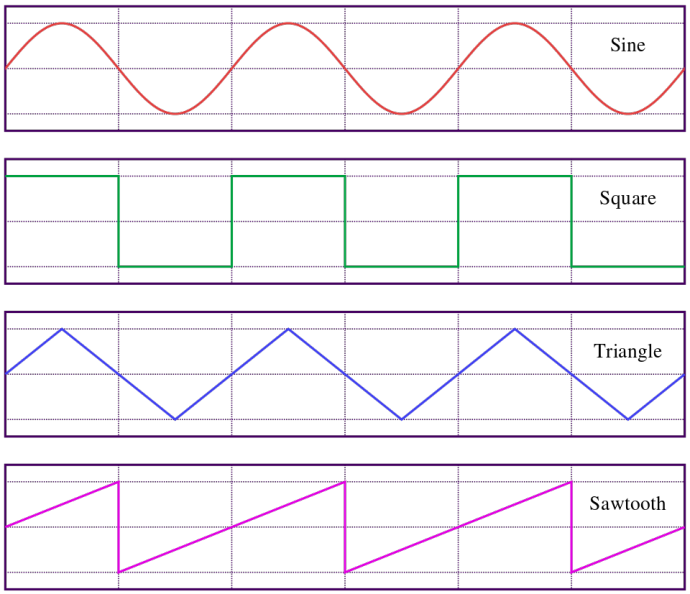
Basic Structure of Arduino Sketch Revisited

```
#include <file.h>       // include header files
const int ledPin = 5; // declare constants
float Vout;             // declare variables
float n;

void setup() {            // runs once
  pinMode(13, OUTPUT) // system settings
  n = 0;                // set variables
}

void loop() {             // loops indefinitely
  Vout = sin(2*PI*n);
  n = n+0.01;
}
```

Assignment L02_05_helloLEDsin

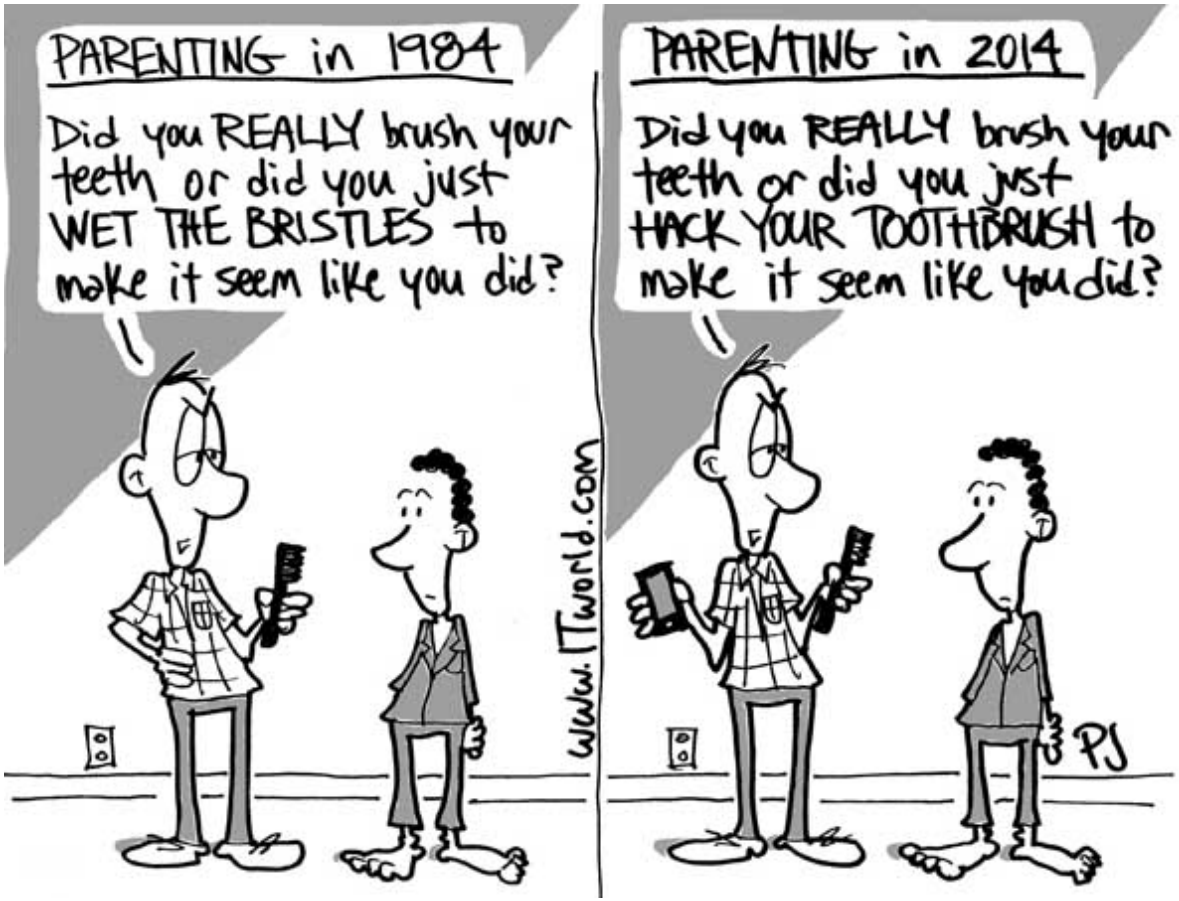


Use a sin() function to vary brightness of your LED

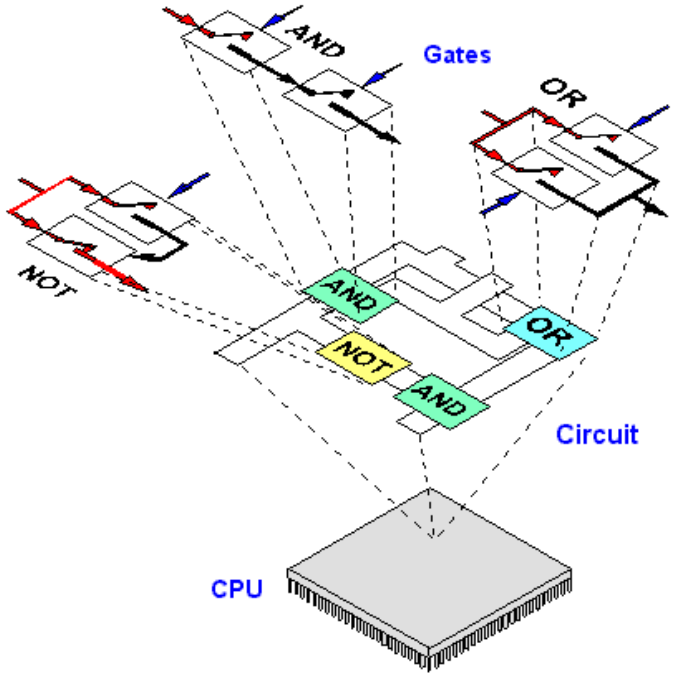
- Use math.h
- Function sin take a double as an input and returns at double.
- Set the period to 5 seconds. One cycle of $sin(2\pi n)$ for each integer value of n.

The function millis() returns milliseconds since Teensy has been powered on. n = (millis()/5000) / period in seconds.

IoT Fun


PARENTING in 1984
Did you REALLY brush your teeth or did you just WET THE BRISTLES to make it seem like you did?
PARENTING in 2014
Did you REALLY brush your teeth or did you just HACK YOUR TOOTHBRUSH to make it seem like you did?
www.ITworld.com
PJ

Data Types: Boolean



Boolean datatype (bool) holds either a TRUE or FALSE

Boolean Logic Operations (condition statements)

1. NOT (!): true if operand is false and visa-versa
   - $x = !x$
2. AND (&&): true if both operands are true
   - $z = x \&\& y$
3. OR ($\|$): true if either operand is true
   - $z = x \| y$

Boolean In Action

```
x = !x;          // invert x

if (!x) {        // if x is false
  // statements
  }

// if both pins read HIGH
if ((digitalRead(pin1) == HIGH) && (digitalRead(
    pin2) == HIGH) {
  //statements
  }

// if either value is greater than zero
if (x > 0 || y > 0) {
  // statements
  }
```

Displaying to the Screen: The Serial Monitor

```
void setup() {

// Enable Serial Monitor
  Serial.begin (9600);
  while (!Serial);  // wait for Serial monitor
  Serial.println ("Ready to Go");
}

void loop() {
  for (i=0; i <=13; i++)
  Serial.print(i);
  delay(printDelay);
}
```

Print Statements

1. Serial.print() prints data to the monitor through the serial port as human-readable text:
   - Serial.print('N') prints: N
   - Serial.print("Hello World") prints: Hello World
   - Serial.print(78) prints: 78
   - Serial.print(3.141592) prints 3.14
   - Serial.print(3.141592,5) prints 3.14159
2. Serial.println() displays the print() followed by a carriage return (\r) or newline (\n).
3. Serial.printf() displays a formatted print.

Format Specifiers Statements

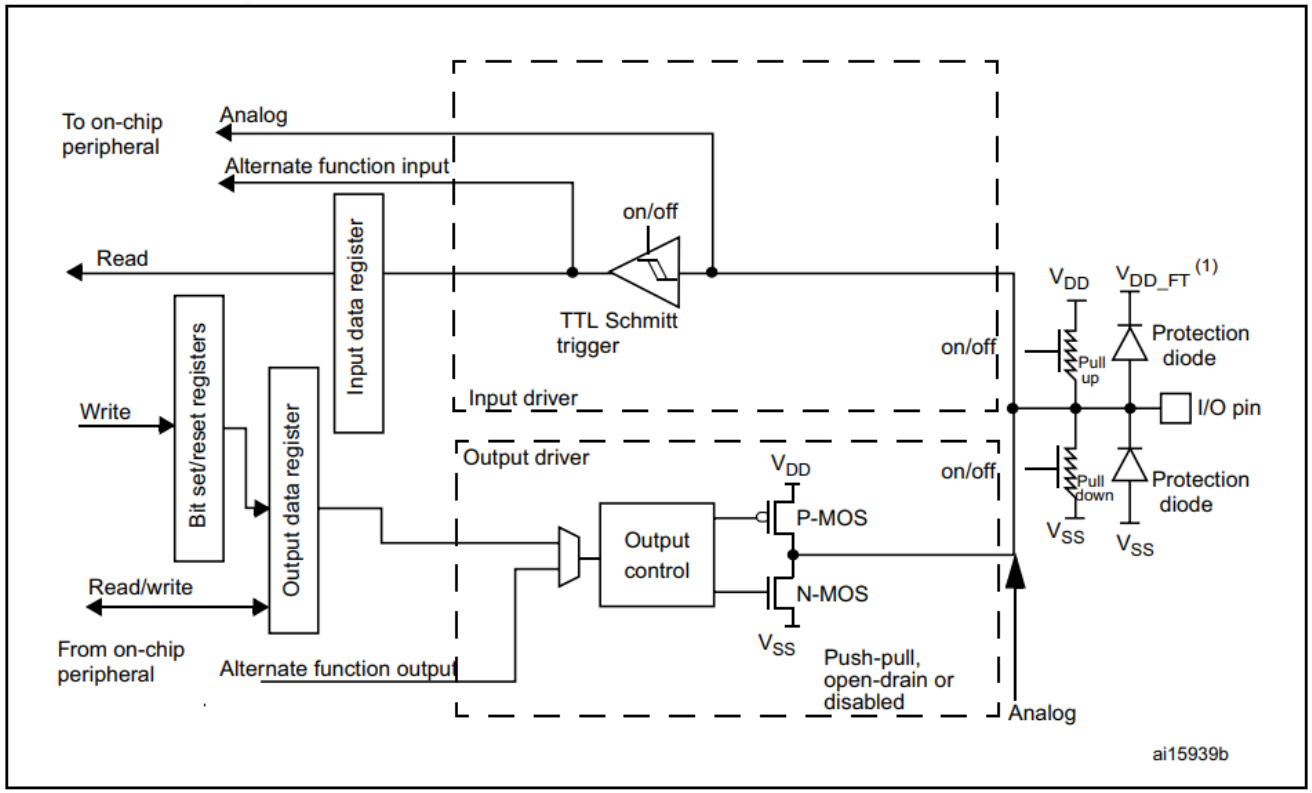
specifier	Output	Example
d or i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

Serial.printf("Let's print an integer %i and a float %0.4f \n", int, float);

Assignment L03_00_SerialMonitor

1. Print Hello World to your monitor screen.
2. Next, display to the screen a count from 0 to 13, separated by commas, by using:
   - Serial.print();
   - Serial.println();
   - Serial.printf();

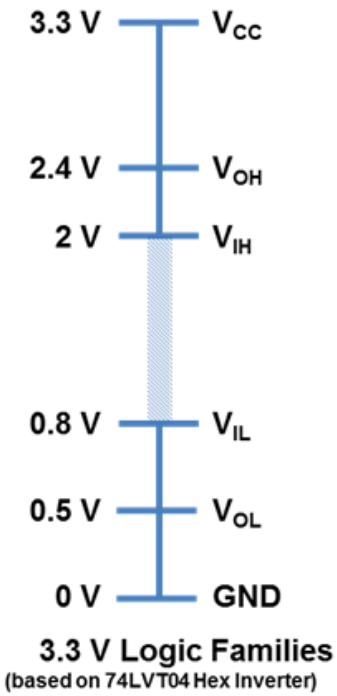
One Pin - Many Functions



Software Programmable: Input or Output and Digital or Analog.

Digital Input/Output

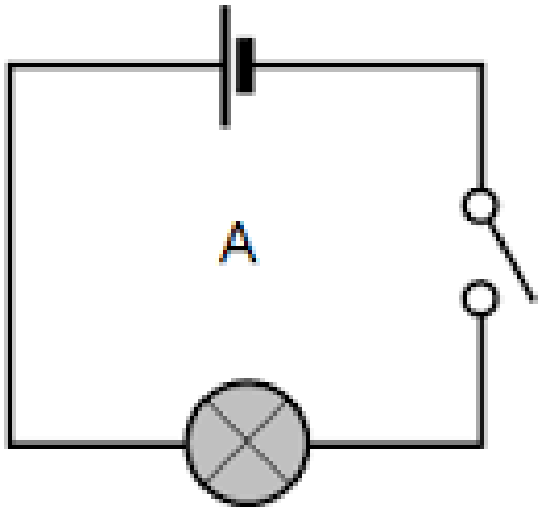
Digital electronics rely on binary logic to store, process, and transmit data or information. Binary Logic refers to one of two states – ON or OFF. This is commonly translated as a binary 1 or binary 0. A binary 1 is also referred to as a HIGH signal and a binary 0 is referred to as a LOW signal.

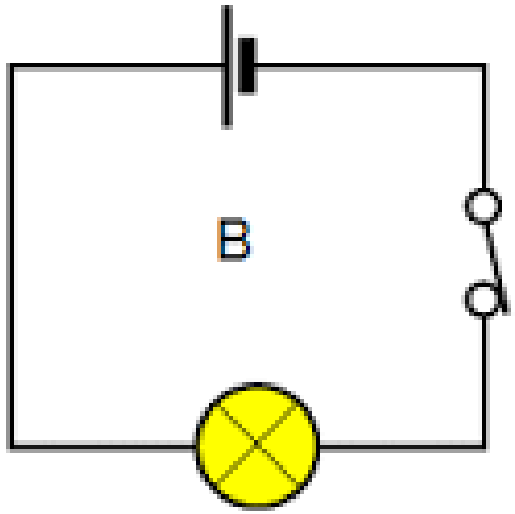


- digitalWrite(pin,value);
- inputValue = digitalRead(pin);

where, value equals HIGH or LOW.

Switches





Types of Switches

SPST

Single-Pole
Single-Throw
Single-Break

SPST

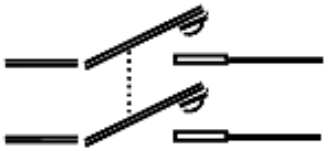
Single-Pole
Single-Throw
Double-Break

SPDT

Single-Pole
Double-Throw
Single-Break

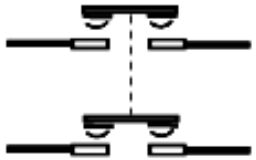
SPDT

Single-Pole
Double-Throw
Double-Break



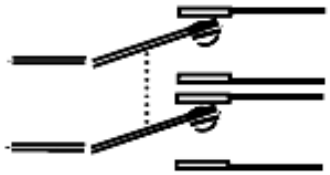
DPST

Double-Pole
Single-Throw
Single-Break



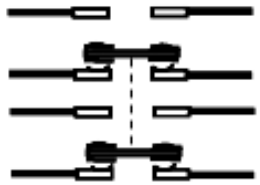
DPST

Double-Pole
Single-Throw
Double-Break



DPDT

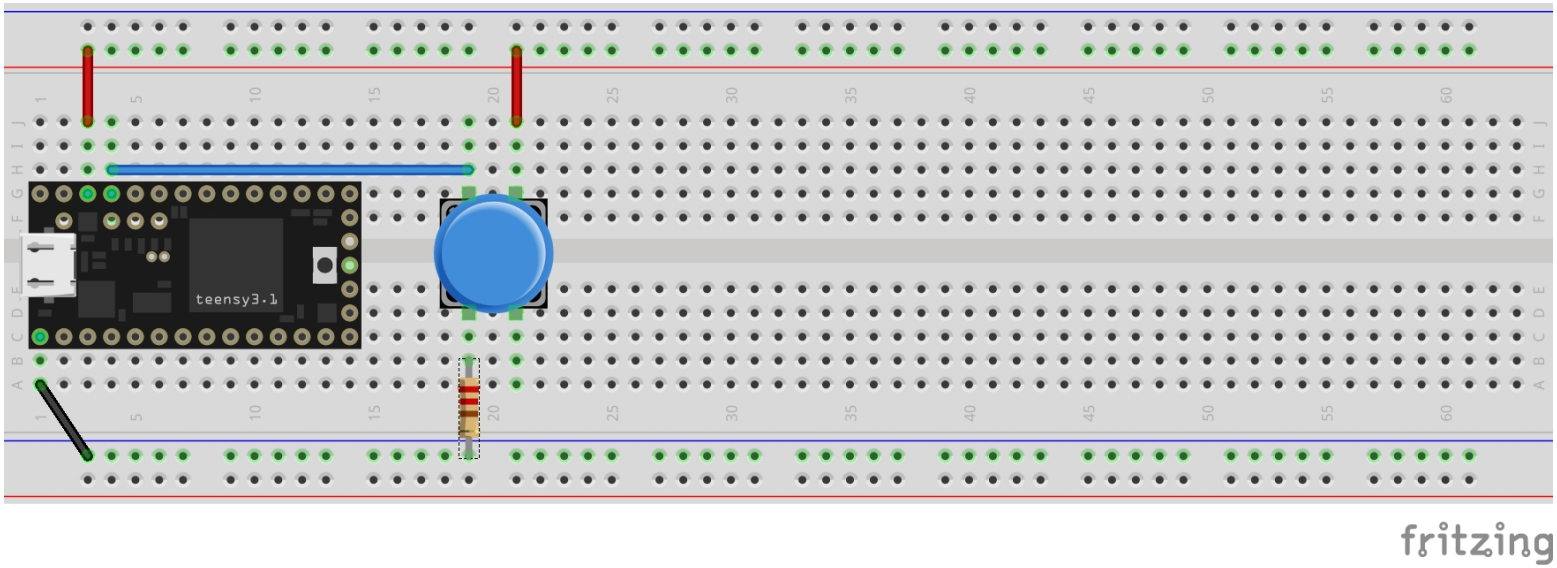
Double-Pole
Double-Throw
Single-Break



DPDT

Double-Pole
Double-Throw
Double-Break

Our First Button and Pull Down Resistors



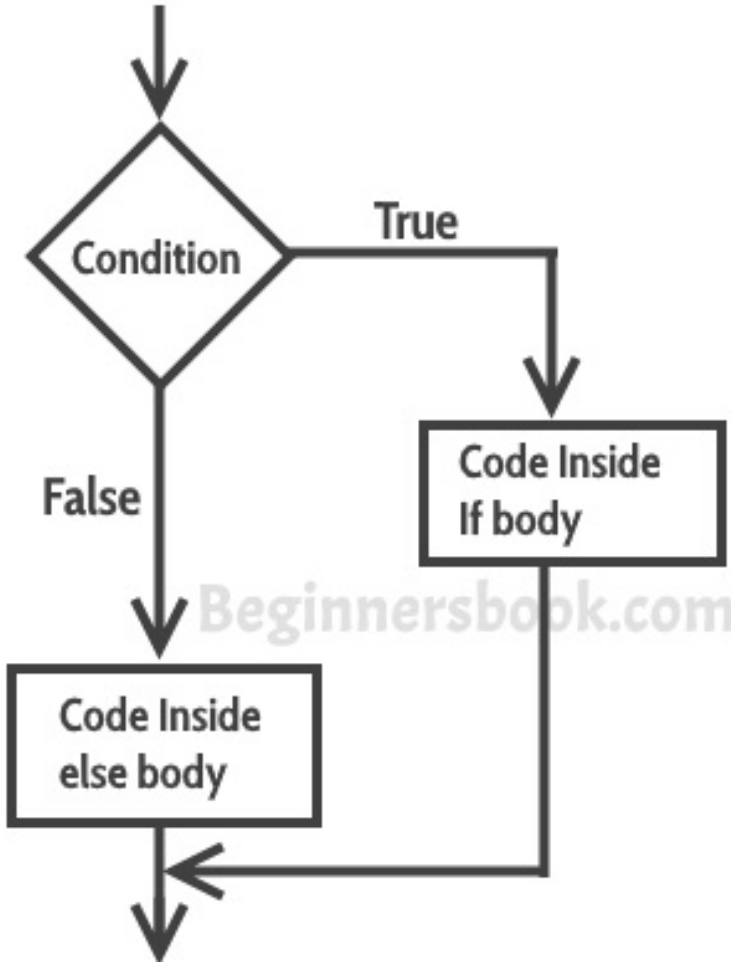
Wait - why is there a resistor connected to ground?

Assignment: Buttons

- Labbook: draw circuit
- Fritzing diagram
- Wire your circuit
- Write the code

1. L03_01_button
   - Connect button (Pin 23)
   - Print button state to the screen.
   - Now, remove the pull-down resistor and restart your code.
2. L03_02_button_pullup
   - Replace the pull-down resistor with a pull-up resistor.
     - Not pressed: 3.3V
     - Pressed: GND
     - How does the logic change?
3. L03_03_button_input_pullup
   - Remove the pull-up resistor
   - Implement: pinMode(pin,INPUT_PULLUP);

IF-ELSE Statements

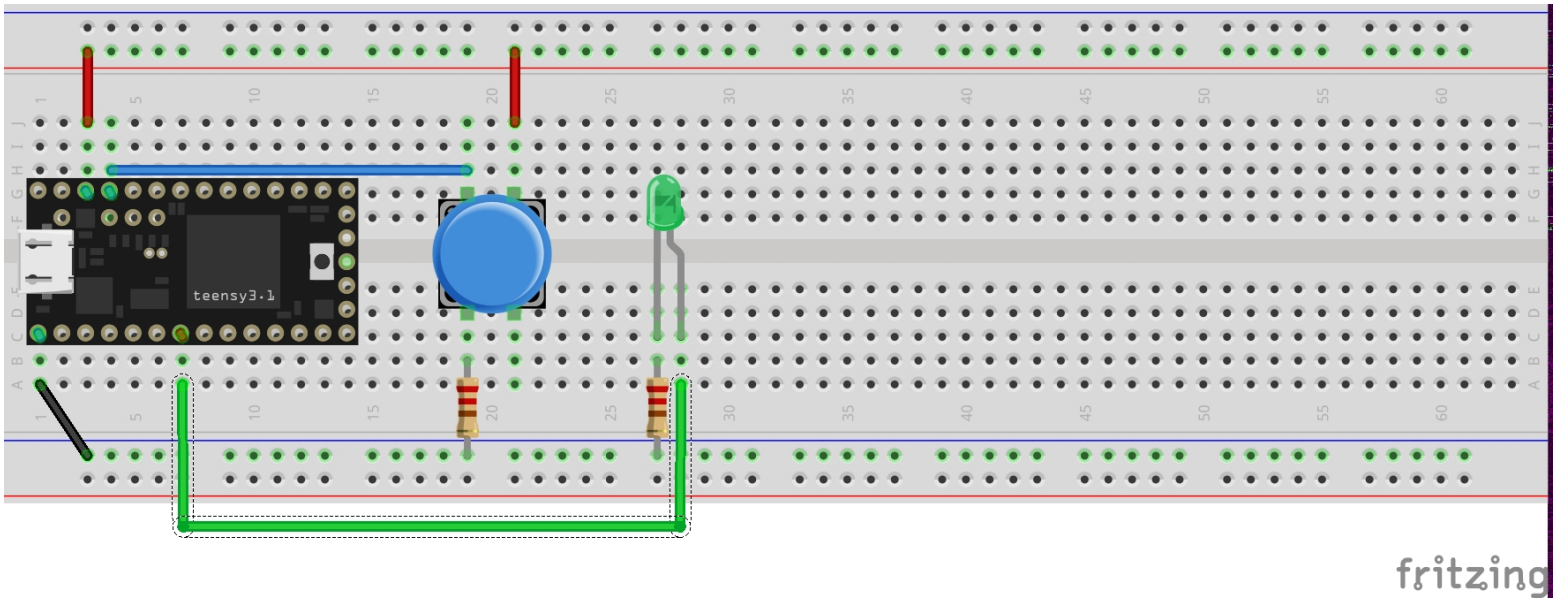


IF-ELSE Statements

```
// IF statement SYNTAX
if (condition) {
  //statement(s)
}
else {
  // else statement(s)
  }

// EXAMPLE
if (button == HIGH) {
  Serial.printf("Button is not pressed \n");
  }
else {
  Serial.printf("Button is pressed \n");
}
```

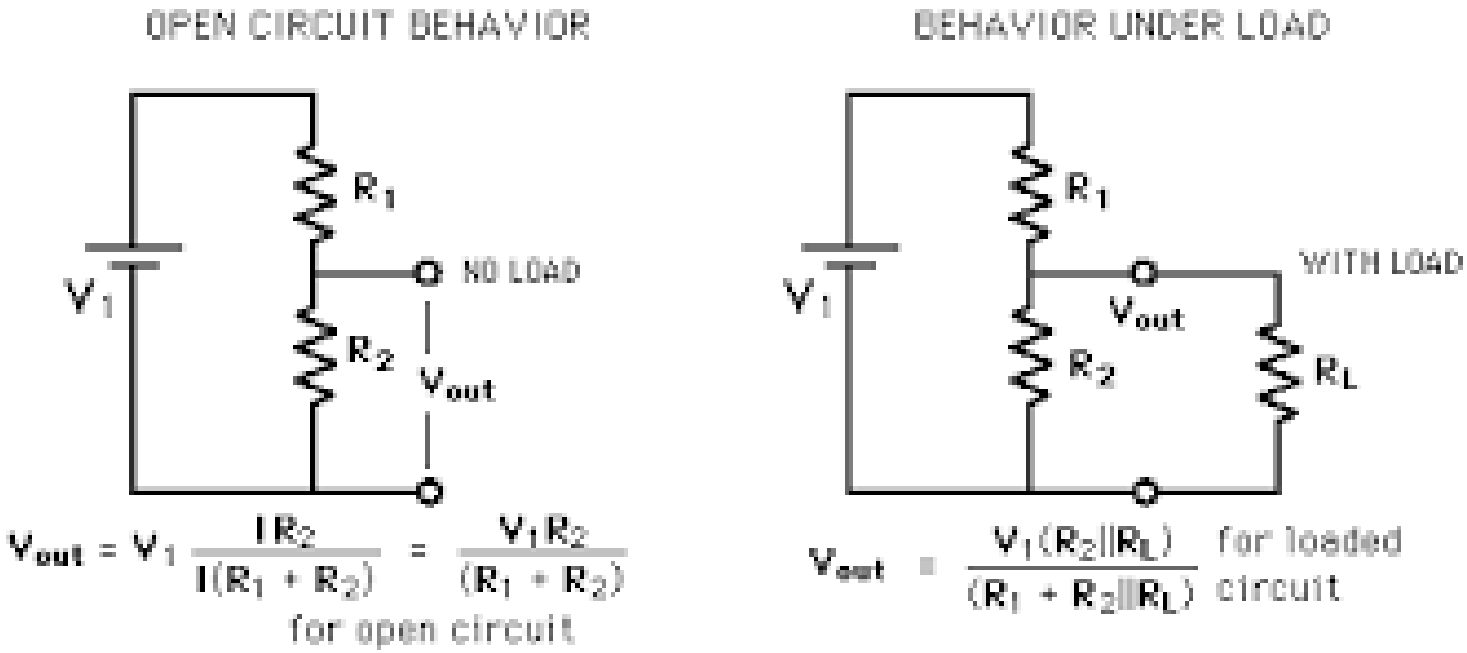
Button and LED



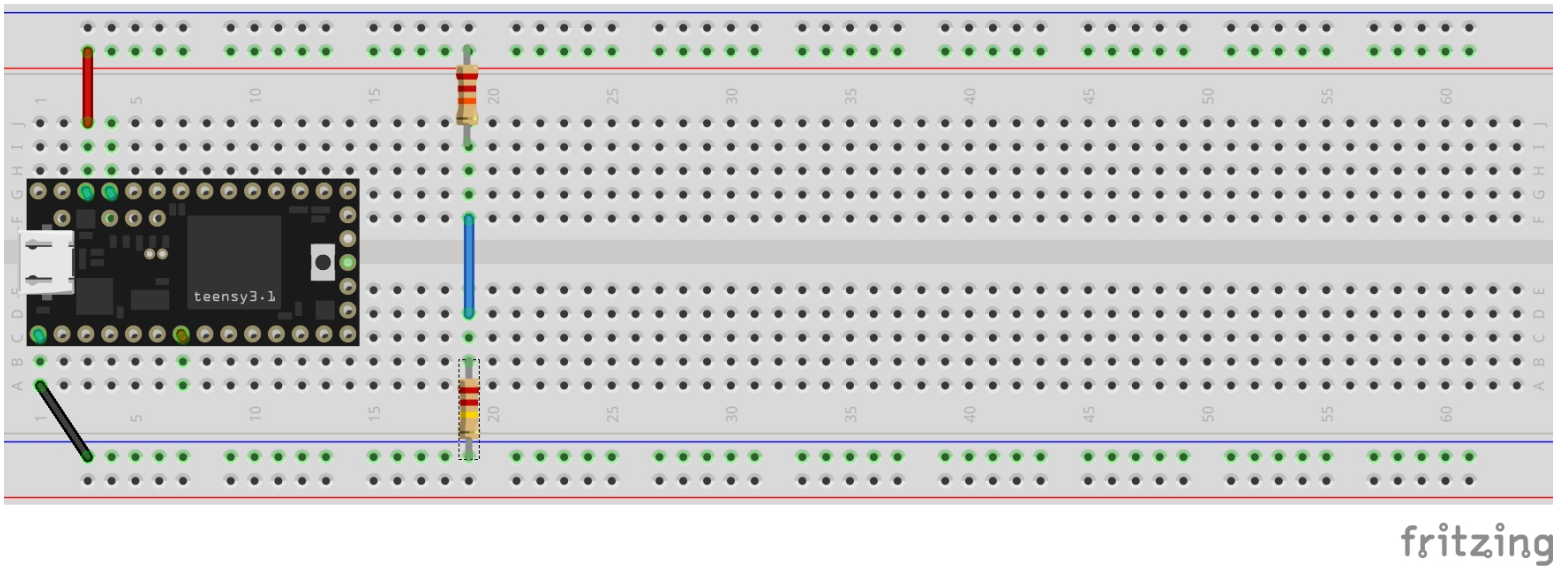
Assignment: Buttons and LEDs

1. L03_04_buttonLED
   - Add an LED to Pin 5 and use the button to turn the LED on and off.
   - Also, print button state to the screen
2. L03_05_twobuttonLED
   - Add a second button (Pin 16) and LED (Pin 6)
   - Have each button control one LED
   - Also, print button states to the screen

Voltage Divider



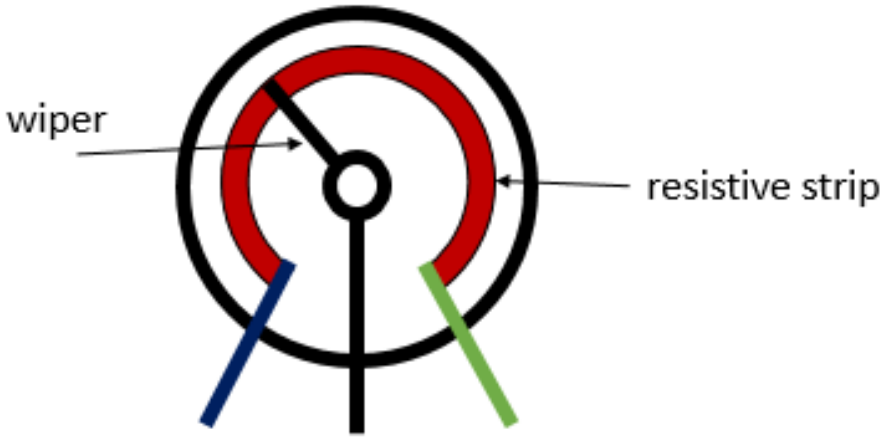
Voltage Dividing



We are just using the Teensy to provide Power and GND.

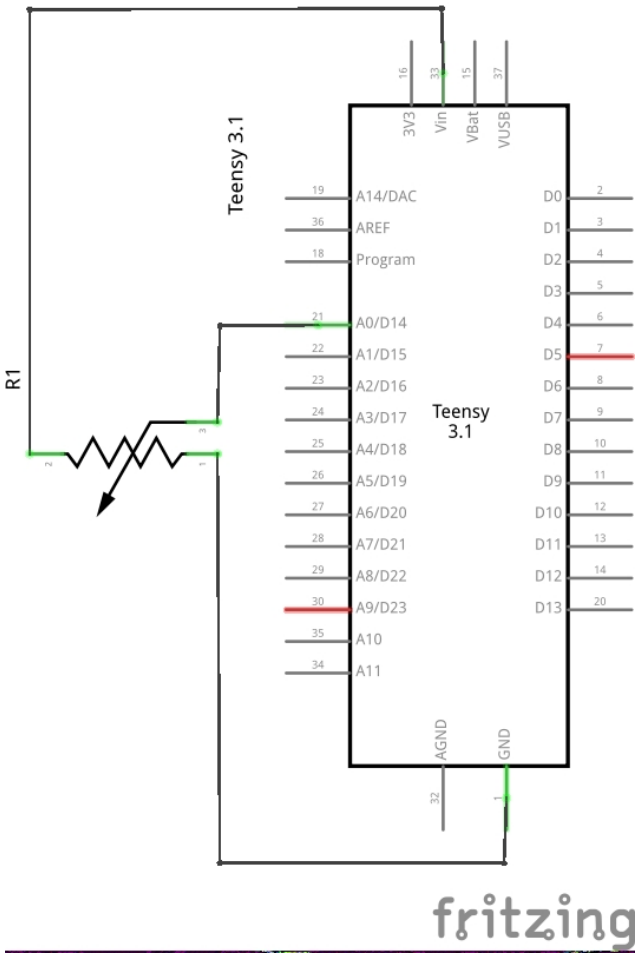
- Use various combinations of $22k\Omega$, $47k\Omega$, $100k\Omega$, $220k\Omega$ resistors.
- Calculate the Series resistance and the voltage between the two resistors in your Lab Notebook.
- Measure with your multimeter and compare.

Potentiometer - Variable Resistor



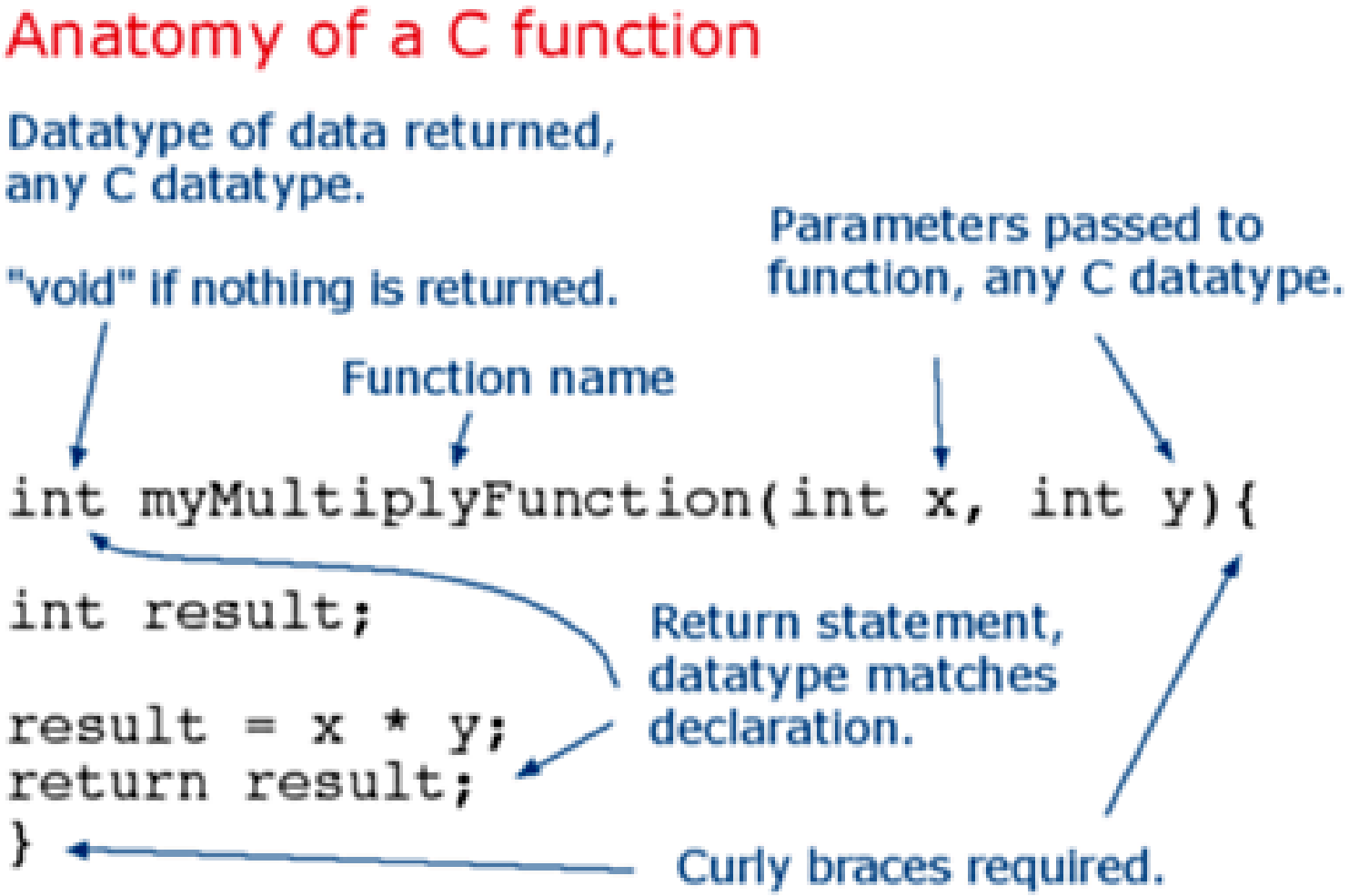
A potentiometer has 3 pins. Two terminals (the blue and green) are connected to a resistive element and the third terminal (the black one) is connected to an adjustable wiper.

Assignment L03_06_AnalogInput



1. Utilize analogRead() to measure analog input across potentiometer (voltage divider) using Pin 14.
2. Determine the range of the analogRead across the entire range of the potentiometer.

Anatomy of a Function



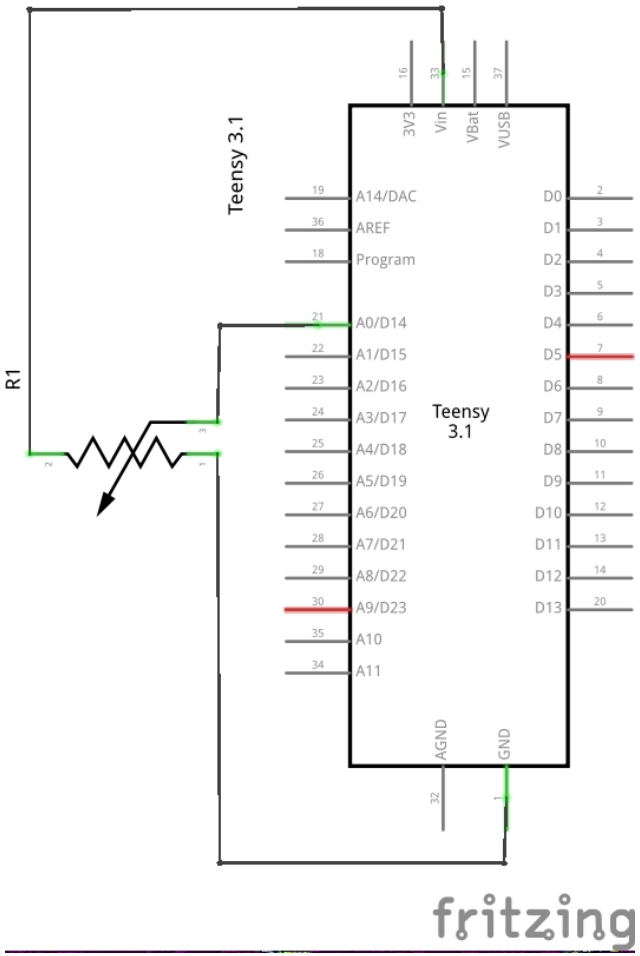
Basic Structure of Arduino Sketch Revisited

```
// include header files, declare variables
// declare global variables
void setup() {                  // runs once
  //initialize variables, begin processes
}

void loop() {                   // loops indefinitely
  doubleNum = twotimes(Num)
}

int twotimes(int Num) {
  int answer;    // answer is a local variable
  answer = 2 * Num;
  return answer;
}
```

Assignment L03_06_AnalogInput

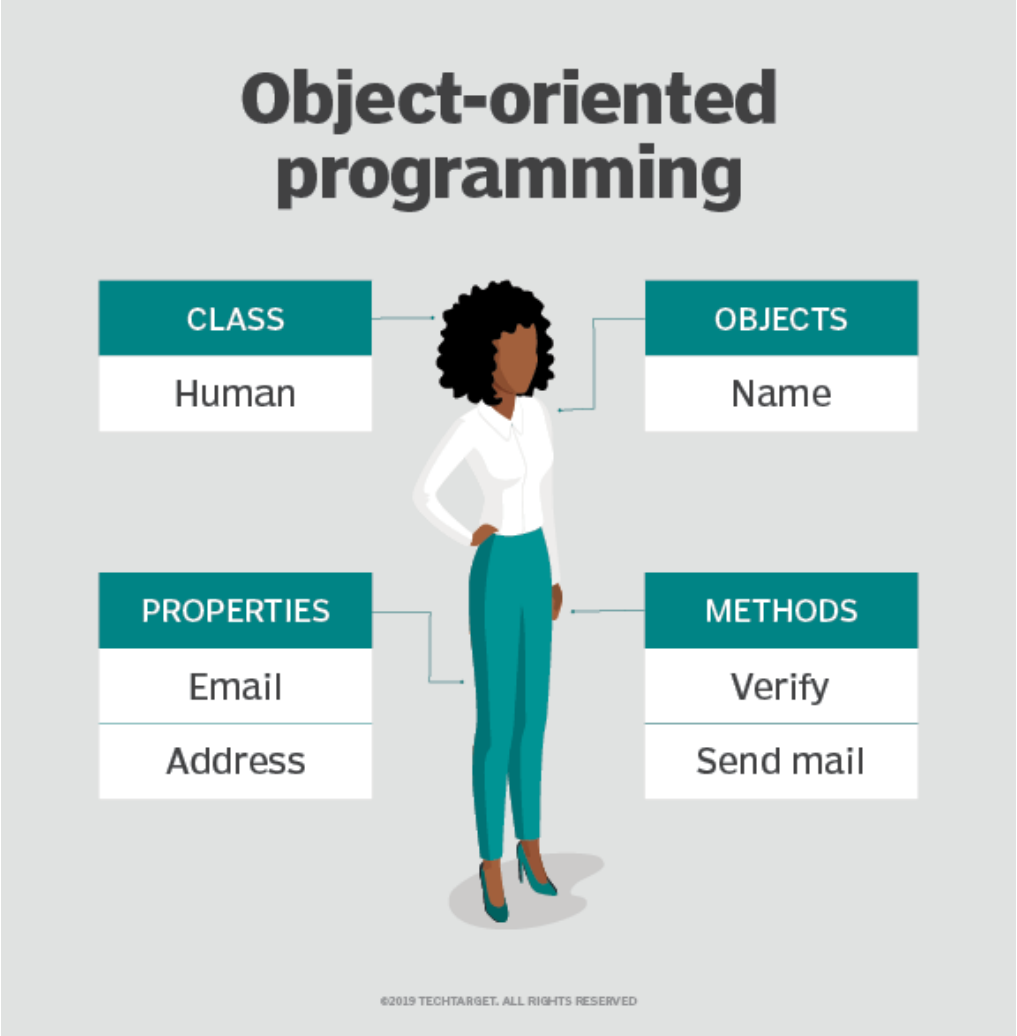


1. Utilize analogRead() to measure analog input across potentiometer (voltage divider) using Pin 14.
2. Determine the range of the analogRead across the entire range of the potentiometer.
3. Write a function, in2volts(), that converts the analog input value to voltage.

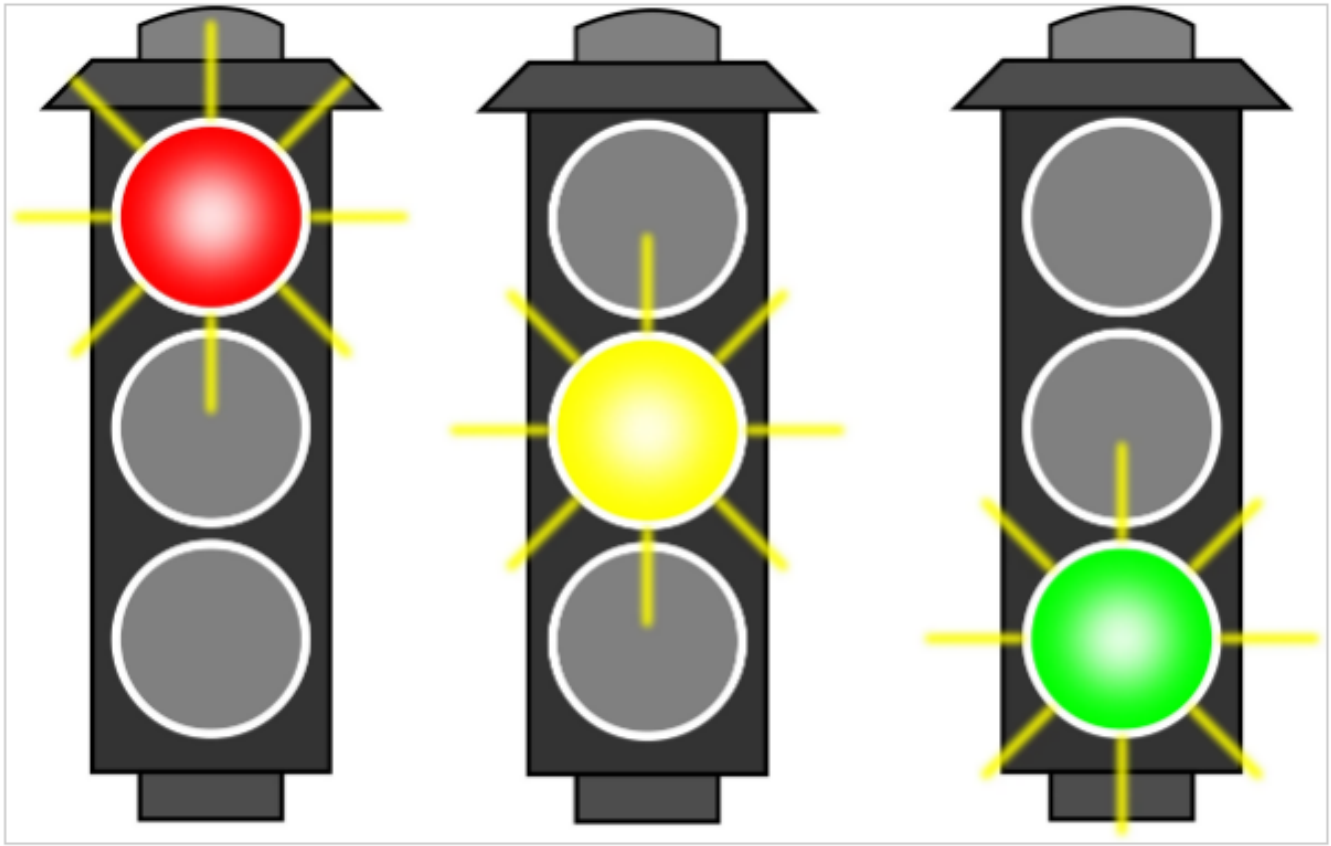
IoT Humor

"I remember when you could only lose a chess game to a supercomputer."

Objects

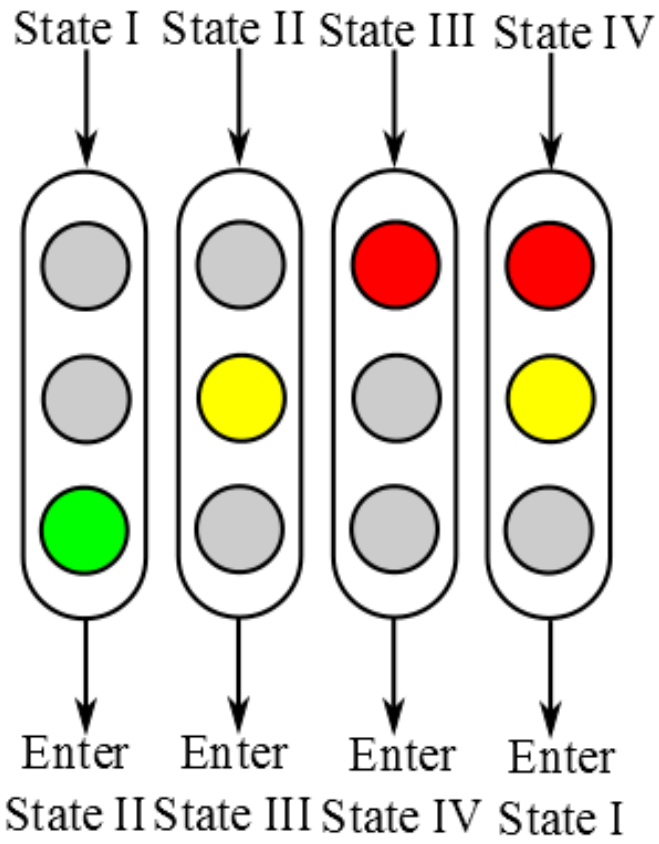

Object-oriented programming
CLASS
Human
OBJECTS
Name
PROPERTIES
Email
Address
METHODS
Verify
Send mail
©2019 TECHTARGET. ALL RIGHTS RESERVED

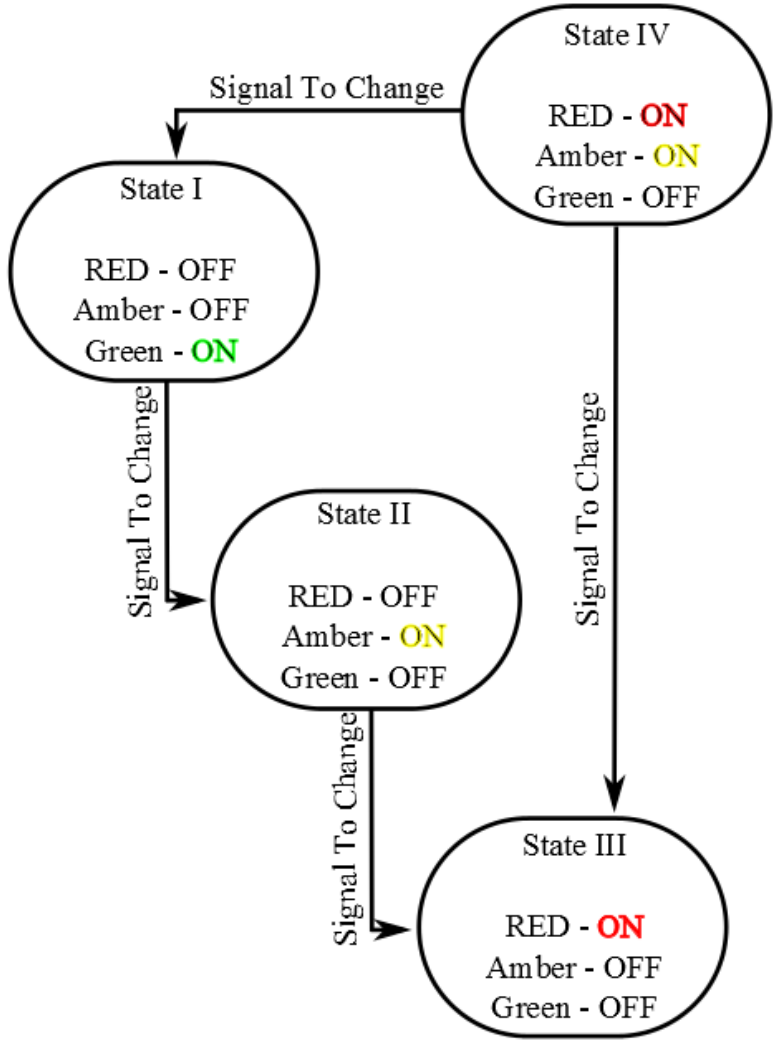
Traffic Light



Let's use the traffic light to build our own Objects

State Machine - Traffic Lights, British Style





SWITCH...CASE syntax - multiple IFs

```
1  // SWITCH...CASE syntax
2  switch (var) {
3    case label1:
4      // statements
5      break;
6    case label2:
7      // statements
8      break;
9    default:
10     // statements
11     break;
12 }
```

Where:

- var: variable whose value is compared to the case values
- label1, label2 are the case values (int or char)

Enumeration (enum)

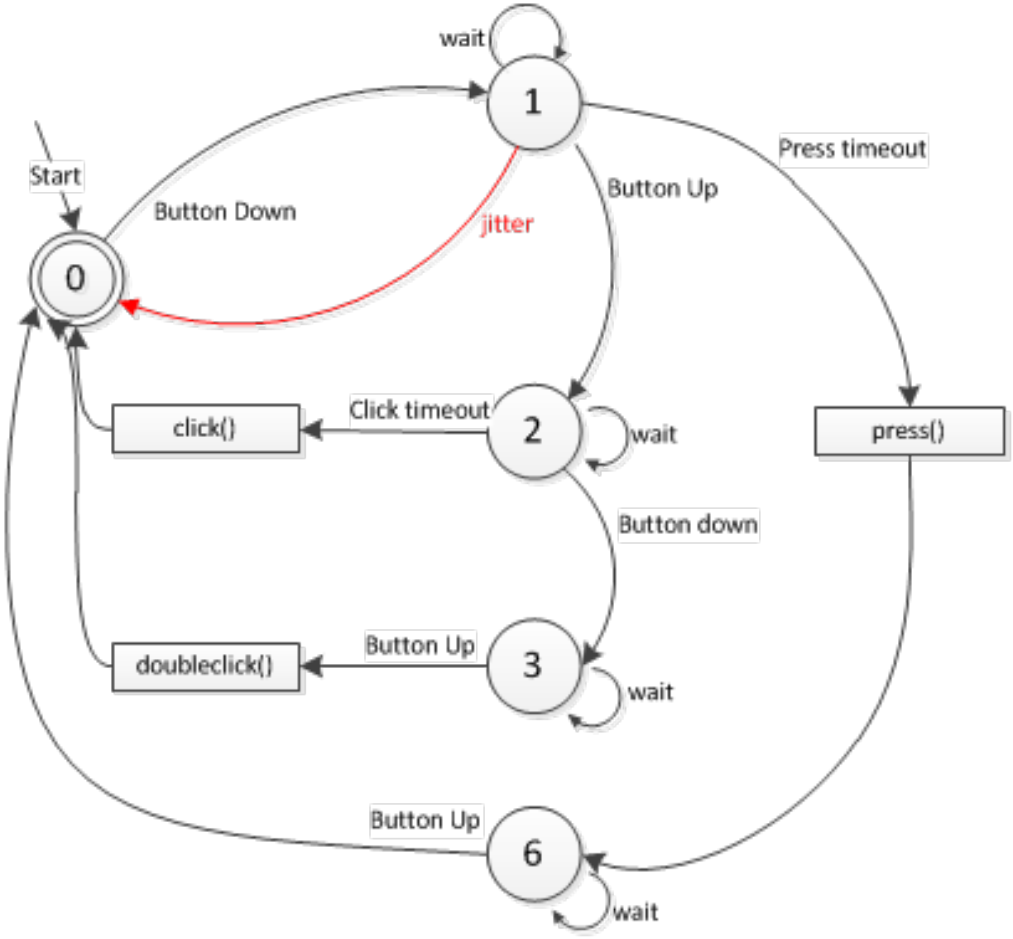
The C-language has a declaration type, enum, which allows for multiple states.

- Within the enum declaration descriptive tags are
- Then the compiler assigns the tags an integer value.

```
1 // ENUM example:
2 // A variable State with four states
3 enum State {
4   GREEN,
5   YELLOW,
6   RED,
7   RED_YELLOW
8 };
```

The compiler treats enum as your personal variable type. For example, the emum variable (e.g., State) can now be used within switch...case statements.

OneButton Library



The tick() method checks the input pin for a single click, double click or long press situation.

Basic Structure of Arduino Sketch - Revisited

```
// the "header" is used for GLOBALS
#include <library.h> // library files
int class_size;      // declare global variables
Adafruit_BME280 bme; // name object in class

void setup() {
  oled.init();              // initialize objects
  bme.begin(0x76)           // begin processes
  arraySize = sizeof(array)4; // set variables
}

void loop() {
  // functionality of your code
  // this loops indefinately
}
```

OneButton Declarations

```
#include "OneButton.h"
OneButton button1(pin, activeLOW, pullUP);

void setup() {
  button1.attachClick(click1);
  button1.attachDoubleClick(doubleclick1);
  button1.attachLongPressStart(longPressStart1);
  button1.attachLongPressStop(longPressStop1);
  button1.attachDuringLongPress(longPress1);
  button1.setClickTicks(250);
  button1.setPressTicks(2000);
}
```

- pin: the pin the button is connected to.
- activeLOW: "true" means input LOW when button pressed.
- pullUP: "true" is INPUT_PULLUP pinMode.

Assignment: OneButton

- Notebook: flowchart
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L04_01_oneButton
   - Use OneButton libary and button on Pin 23.
   - Click() - toggle bool variable buttonState.
   - doubleClick() - toggle bool variable flash.
2. L04_02_oneButtonLED
   - Toggle LED on/off with buttonState.
3. L04_03_oneButtonLEDblink
   - When ON, toggle from solid to blinking with flash.

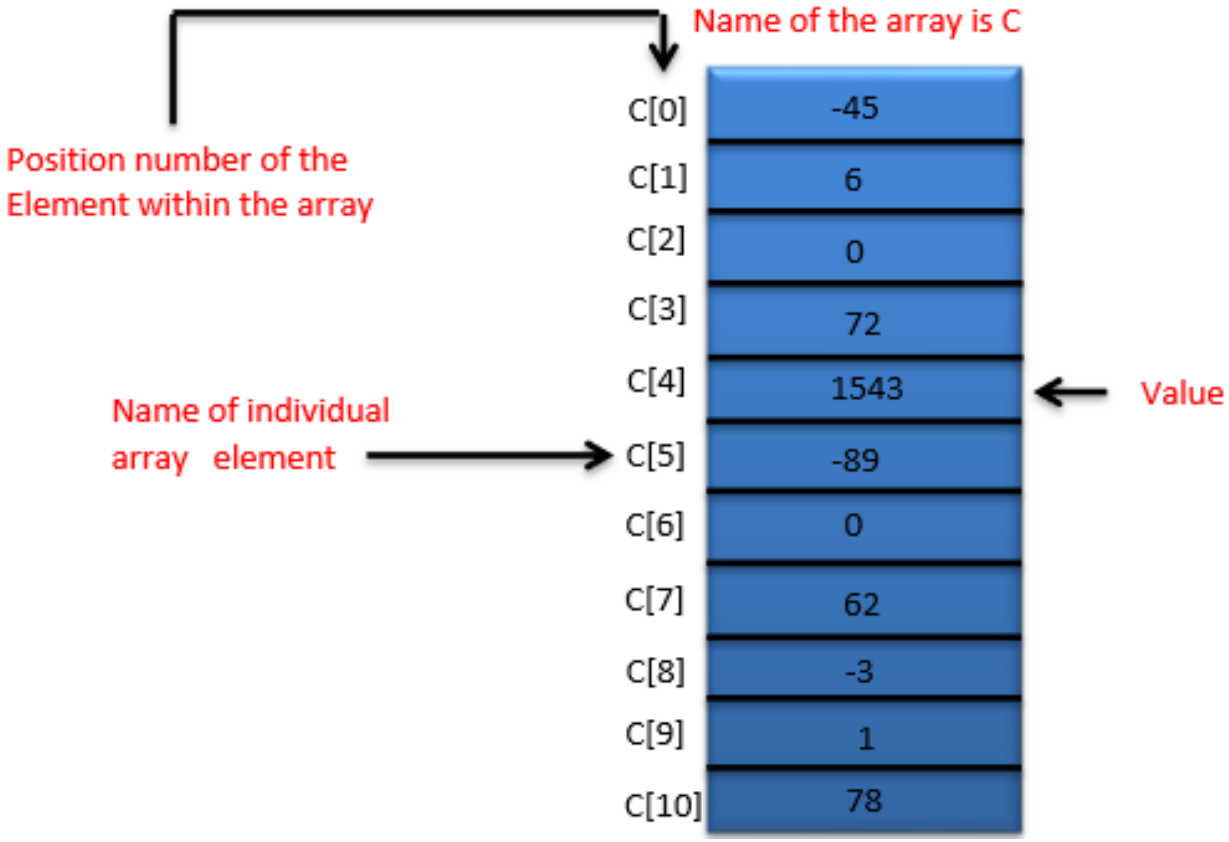
Avoiding Delays

```
void loop() {
  //run constantly
  currentTime = millis();

  //run once per second
  if((currentTime-lastSecond)>1000) {
    Serial.print(".");
    lastSecond = millis();
  }

  //run once per minute
  if((currentTime-lastMinute)>60000) {
    Serial.println();
    Serial.println("Minute");
    lastMinute = millis();
  }
```

Arrays



- Syntax: datatype var[ ] = {element 1, element 2, element 3};
- Example: int ledArray[ ] = {greenPin, yellowPin, redPin};

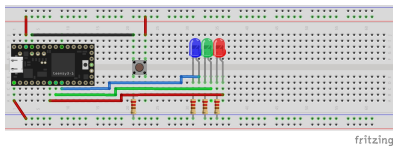
Using Arrays

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
char message[6] = "hello";

void loop() {
  mySensVals[0] = 10; //assign value to array
  x = mySensVals[4]; //retrieve value from array
  for (i = 0; i < 5; i = i + 1) {
      Serial.println(myPins[i]);
  }
}
```

NOTE: The array index starts at 0 (not 1).

Assignment: OneButton



- Notebook: flowchart
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

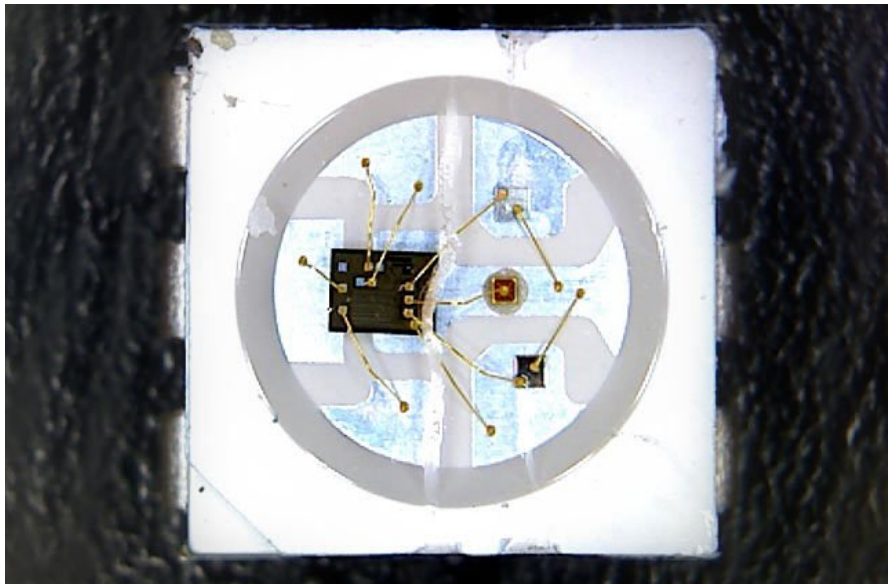
1. L04_04_oneButtonArray
   - Use 3 LEDs and one Button
   - Click() - toggle current LED on/off
   - doubleClick() - using an array, cycle to next LED
   - longPressStart() - light up the three LEDs in sequence
   - longPressStop() - light up the three LEDs in reverse order

Assignment: Serial Read

- Serial Read Example

1. L04_05_timer - Create a Stop Watch and Countdown Timer
   - Click for start and stop
   - Double Click to switch between Stop Watch and Timer. In Timer mode, prompt the user on the Serial Monitor to enter the time to countdown from.
   - Long Press for reset

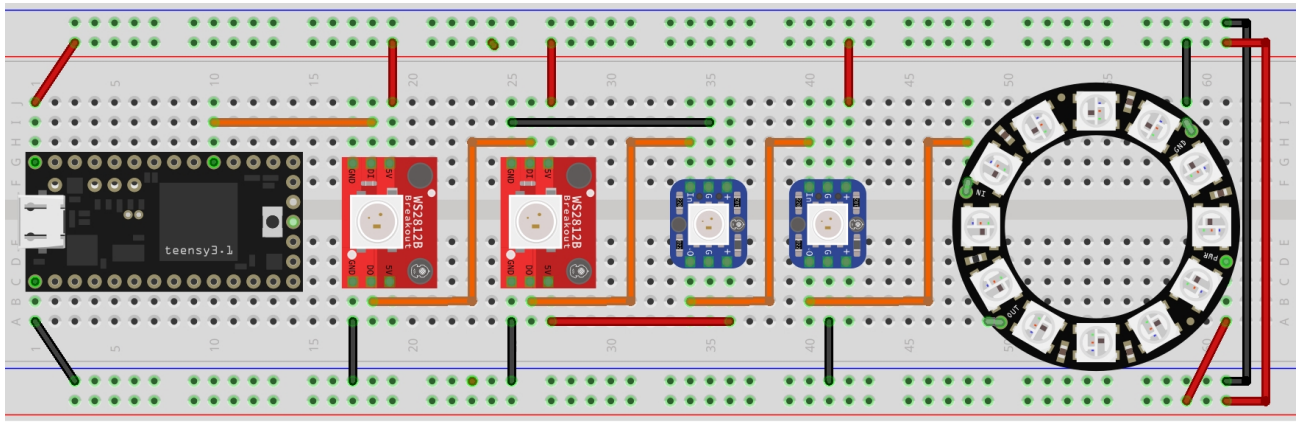
NeoPixels



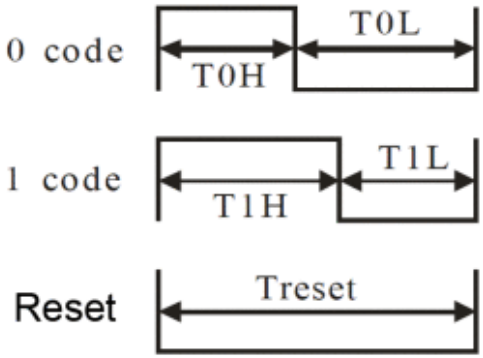
NeoPixels are:

- Addressable RGB LEDs based on the WS2812 (or WS2811) LED/drivers.
- They come as individual pixels, in strips, in matrices, rings, etc.
- They can be programmed via your microcontroller to create a wide array of effects and animations.

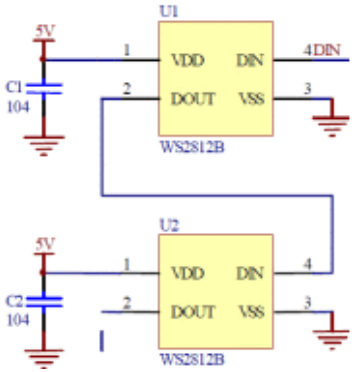
NeoPixel Programming



WS2812 Protocol



LED-Chain



NeoPixel Declaration

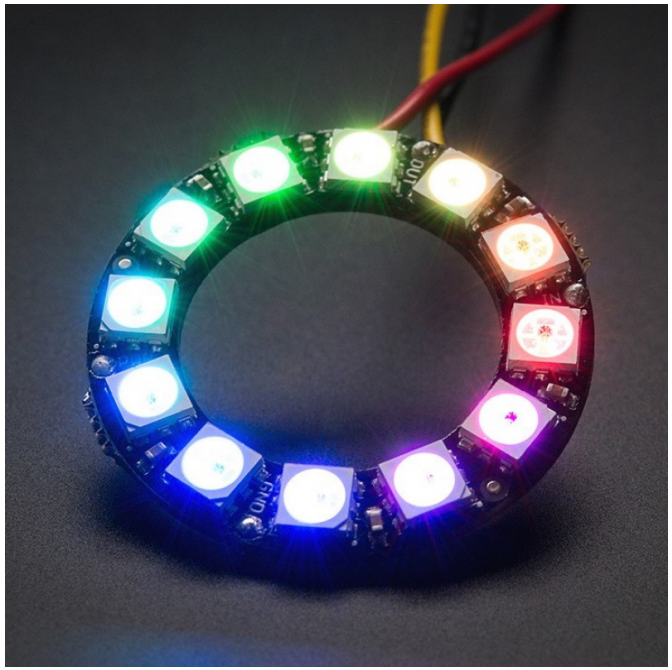
```
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel strip(LED_COUNT, LED_PIN,
    NEO_GRB + NEO_KHZ800);
/* Argument 1 = Number of pixels
 * Argument 2 = GPIO pin number
 * Argument 3 = Pixel type flags, add together:
 * NEO_KHZ800  800 KHz bitstream (WS2812)
 * NEO_KHZ400  400 KHz (WS2811)
 * NEO_GRB     Pixels are wired for GRB bitstream
     (most NeoPixel products)
 * NEO_RGB     Pixels are wired for RGB bitstream
     (v1
 * NEO_RGBW    Pixels are wired for RGBW
    bitstream
 */
```

Using NeoPixel Methods

```
1  void setup() {
2    pixel.begin();
3    pixel.show(); //initalize all off
4  }
5
6  void loop() {
7    pixel.setPixelColor(n, red, green, blue);
8    pixel.setPixelColor(n, color); \\hex code
9    pixel.fill(color, first, count);
10   pixel.setBightness(bri) \\ 0 - 255
11   pixel.show(); \\nothing changes until show()
12   pixel.clear();
13   pixel.show();
14 }
```

Assignment: NeoPixels



- Notebook: flowchart
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L05_01_neoPixel
   - Light up the 4 pixels and the ring using a FOR-loop.
2. L05_02_colorHeader
   - Implement a header file that contains the pixel colors.
3. L05_03_pixelStrip, using FOR-loop, implement functions that:
   - Send a pixel of a random color down and back on the strip
   - Light the strip up as a rainbow
   - Send a pair of Maize and Blue lights down the strip.
4. L05_04_pixelFill
   - Light up 6 segments of different colors using the fill() method

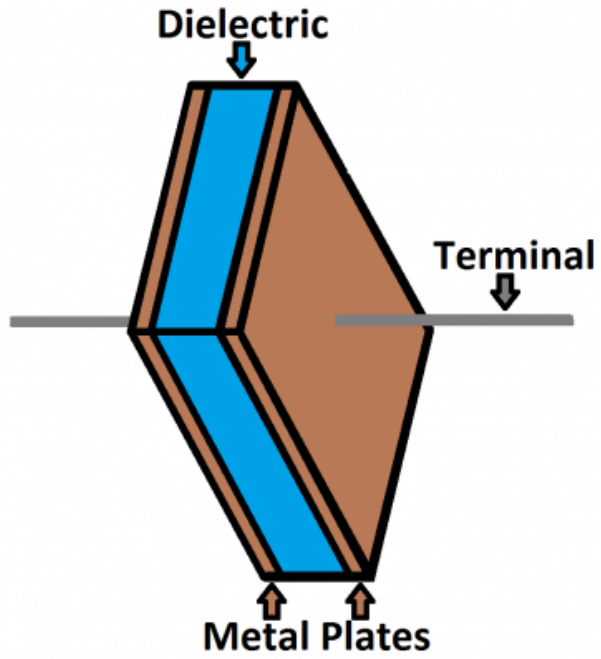
IoT Humor


"WE HAVE TO GO OUT FOR DINNER. THE REFRIGERATOR ISN'T SPEAKING TO THE STOVE."

Capacitors

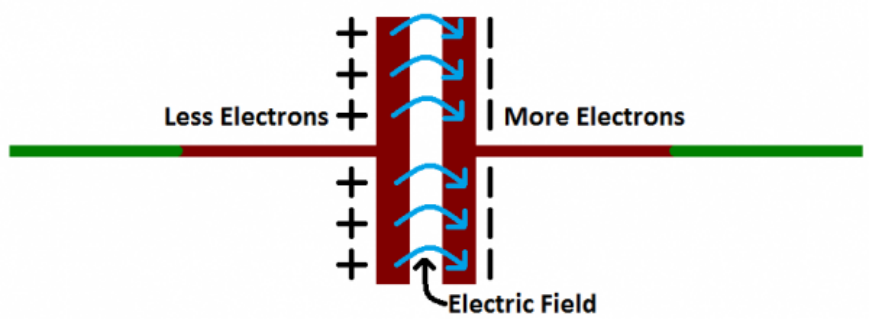
A capacitor is created out of two metal plates and an insulating material called a dielectric. The metal plates are placed very close to each other, in parallel, but the dielectric sits between them to make sure they don't touch.

- The dielectric can be made out of all sorts of insulating materials: paper, glass, rubber, ceramic, plastic, or anything that will impede the flow of current.
- The plates are made of a conductive material: aluminum, tantalum, silver, or other metals.



Capacitors

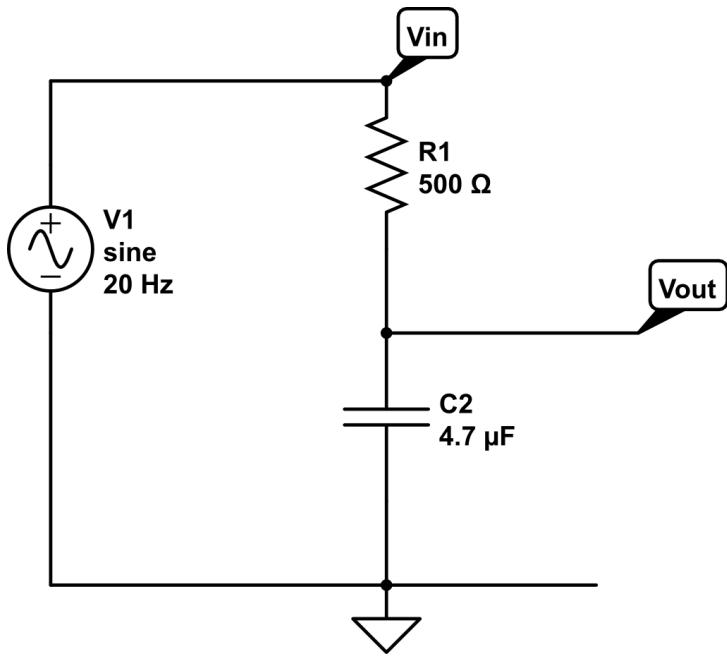
When current flows into a capacitor, the charges get "stuck" on the plates because it can not get past the insulating dielectric. Electrons build up on one of the plates, and it becomes overall negatively charged. The large amount of negative charges pushes away like charges on the other plate, making it positively charged.



The stationary charges on these plates create an electric field, which influence electric potential energy and voltage. When charges group together on a capacitor like this, the cap is storing electric energy just as a battery might store chemical energy.

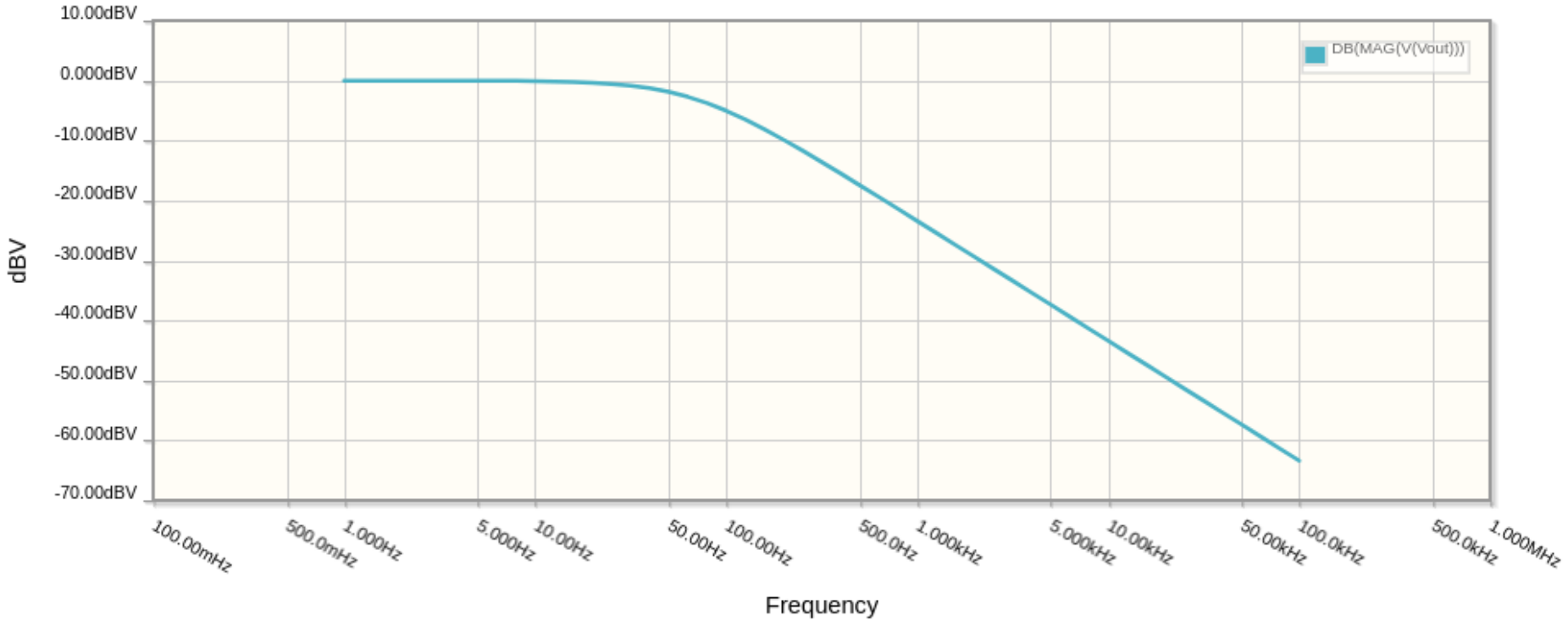
Low Pass Filter - cutoff frequency f_c

- At low frequencies, there is plenty of time for the capacitor to charge up to practically the same voltage as the input voltage.
- At high frequencies, the capacitor only has time to charge up a small amount before the input switches direction. The output goes up and down only a small fraction of the amount the input goes up and down. At double the frequency, there's only time for it to charge up half the amount.



$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC}$$

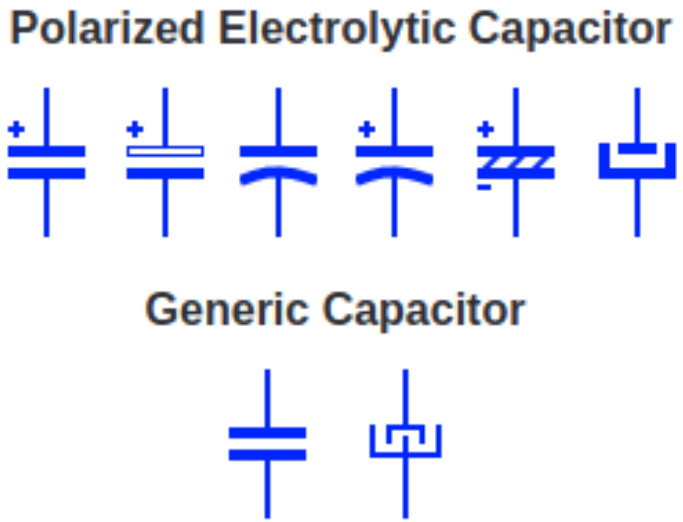
Low Pass Filter Response



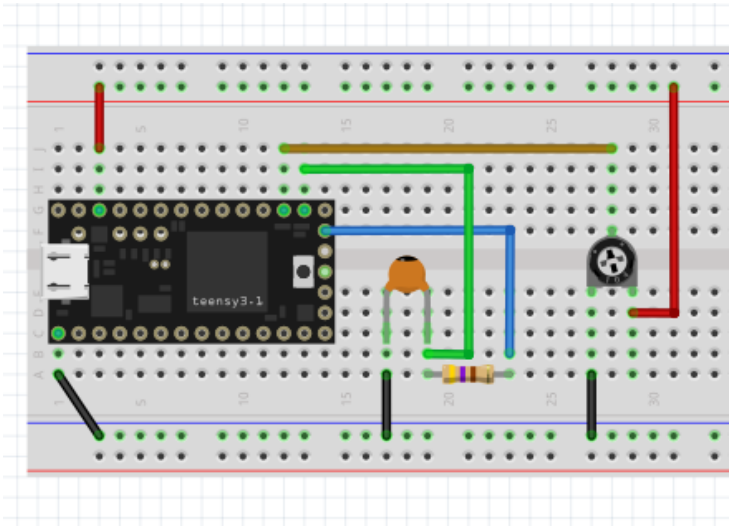
$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi(500)(4.7x10^{-6})} = 67.5678Hz$$

Capacitors - does it matter how they are placed

- Some types of capacitors (electrolytic and tantalum) are polarized (the have + and - terminals). This is due to how the dielectric film has been deposited, the reverse polarity leads to degradation of the dielectric.
- Other capacitors (ceramic and film) do not have a polarity and can be installed in either direction.



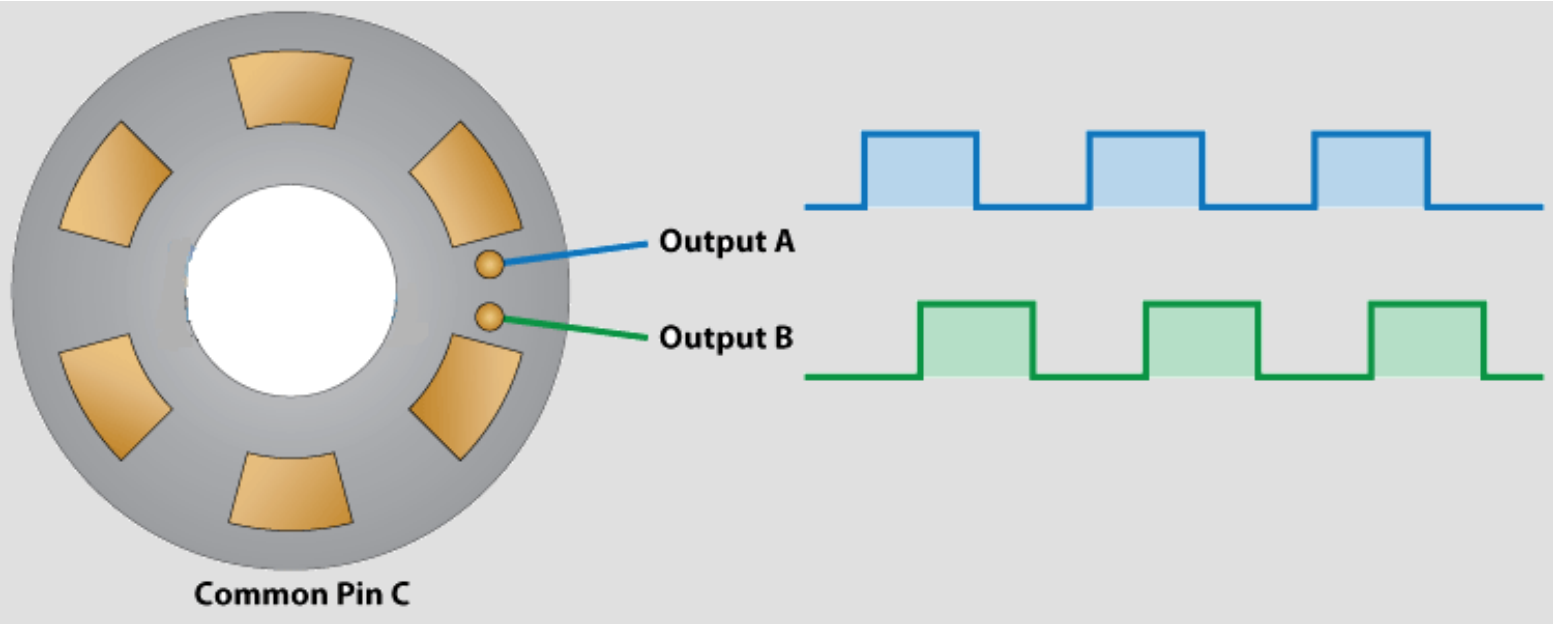
Assignment: Low Pass Fitlers



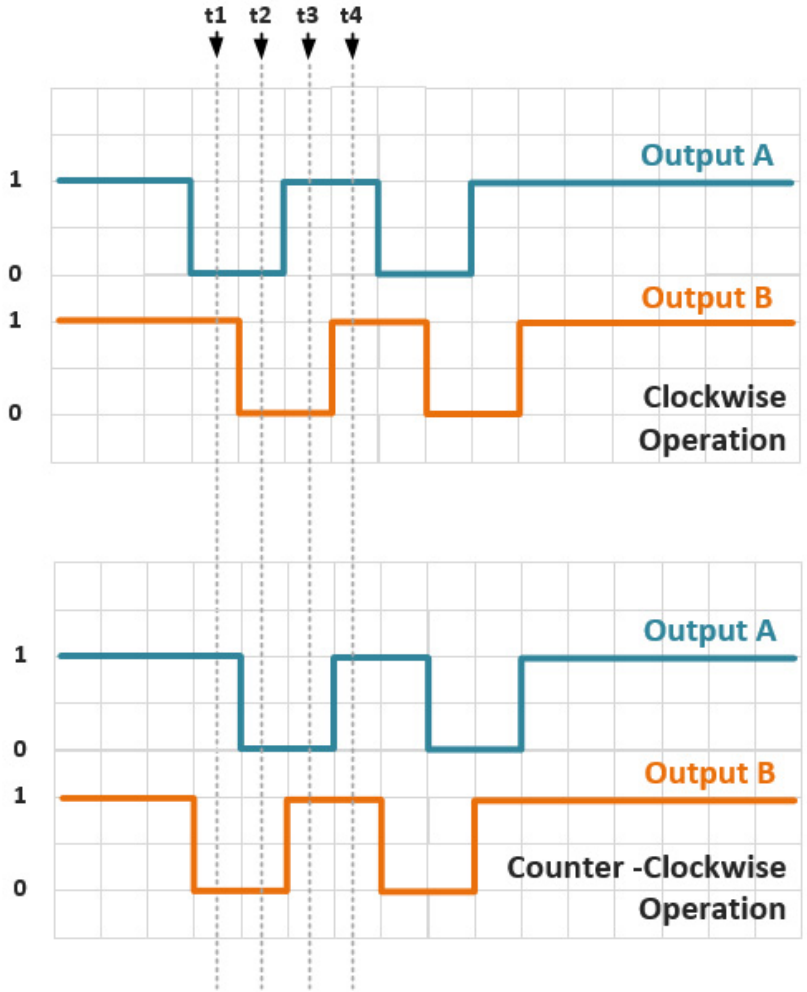
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L06_00_lowPass
   - Create code that generates a sine wave of frequency ν: $sin(2\pi\nu t)$
   - Connect the output to an input
   - Using the Serial Plotter, plot both the output and the input.
   - Create a low pass filter with $f_c \approx 67Hz$
   - Pass the output through the low pass filter before inputting back to the Teensy.
   - In code, vary the frequency and observe the difference between the two signals.
   - Use the potentiometer to now vary the frequency.

Encoders



Encoders



Clockwise Sequence

	A	B
t1	0	1
t2	0	0
t3	1	0
t4	1	1

Counter-Clockwise Sequence

	A	B
t1	1	0
t2	0	0
t3	0	1
t4	1	1

Encoder Class

```
#include <Encoder.h>
Encoder myEnc(pinA,pinB);

void setup() {
}

void loop() {
  // read encoder position
  position = myEnc.read();

  // set encoder to a position
  myEnc.write(maxPos);
}
```

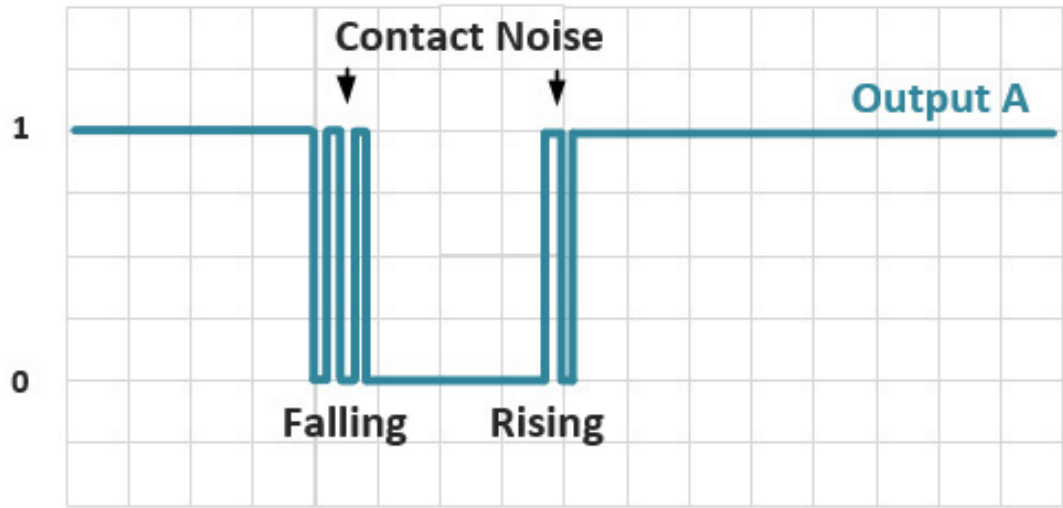
Assignment: Encoders

- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

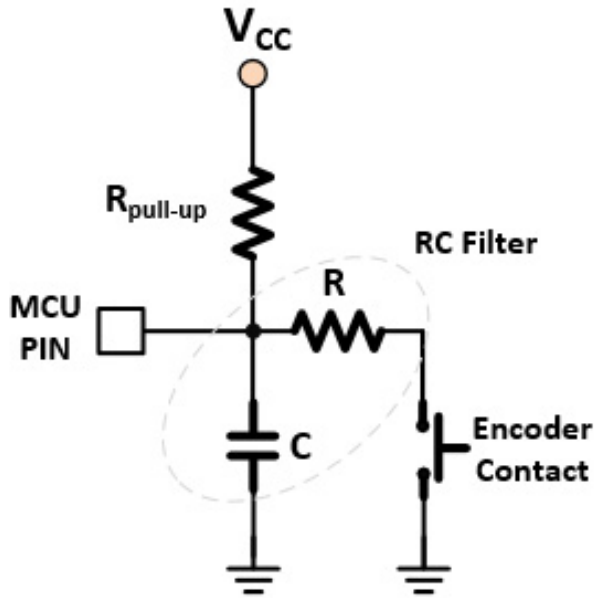
1. L06_01_encoder
   - Write encoder position to the screen
2. L06_02_encoderScaled
   - The encoder has 96 positions
   - Manually map the encoder to 12 pos (0-7 = 0,8 - 15 = 2, etc.)
   - Next, use the map() function:
3. L06_03_encoder_NeoPixel
   - Use the encoder to light up the pixel ring
4. L06_03_encoder_switch
   - Connect your microcontroller to the encoder switch and LEDs

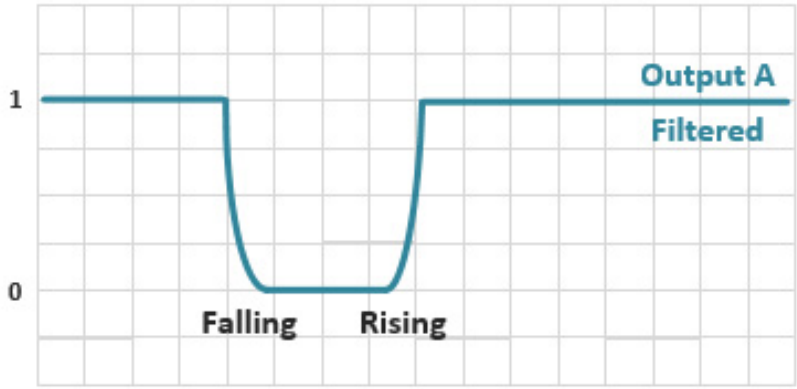
newVal = map(value, fromLow, fromHigh, toLow, toHigh)

Encoder Jitter



Encoder - Low Pass Filter



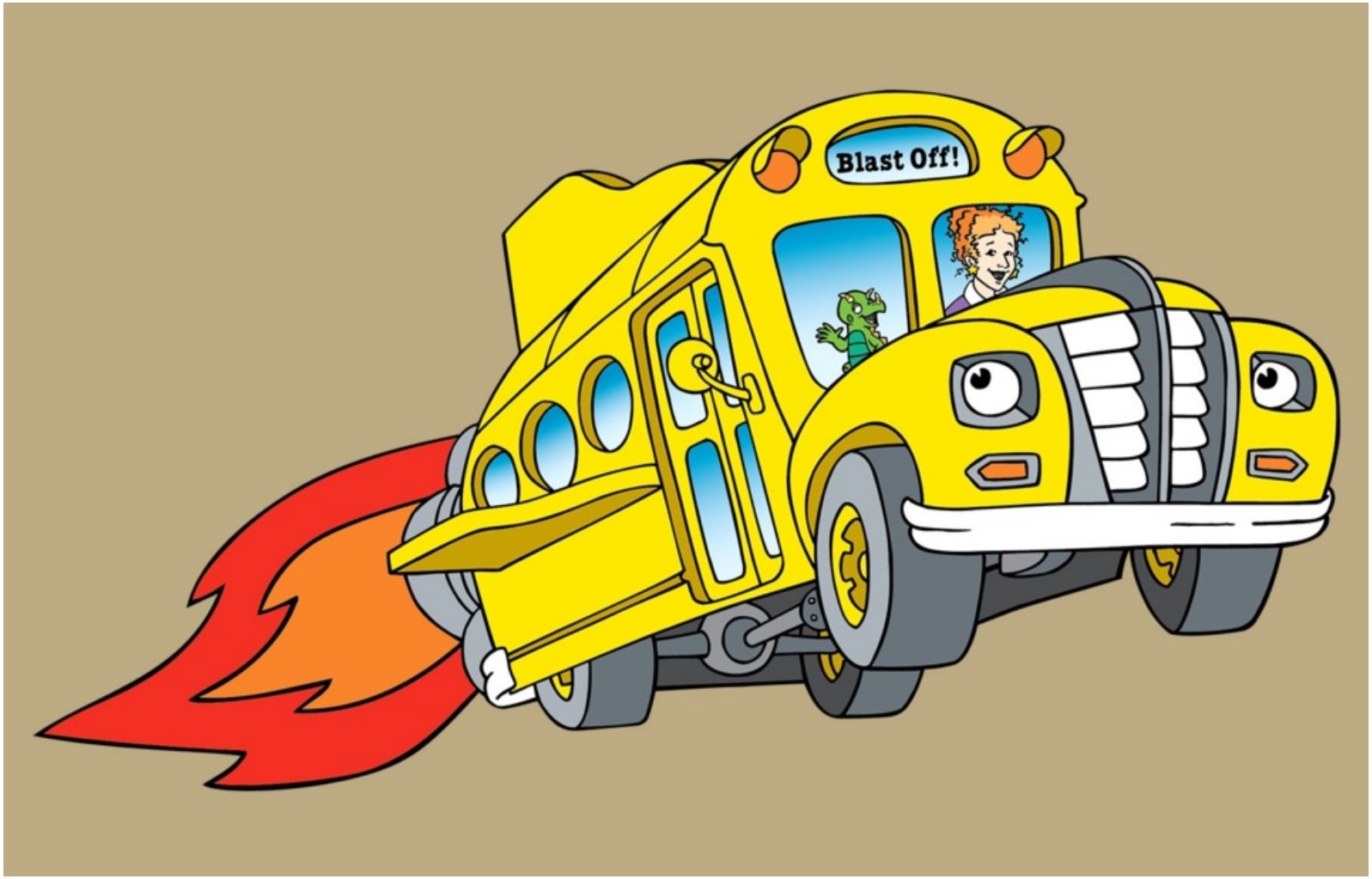


Assignment: Encoders

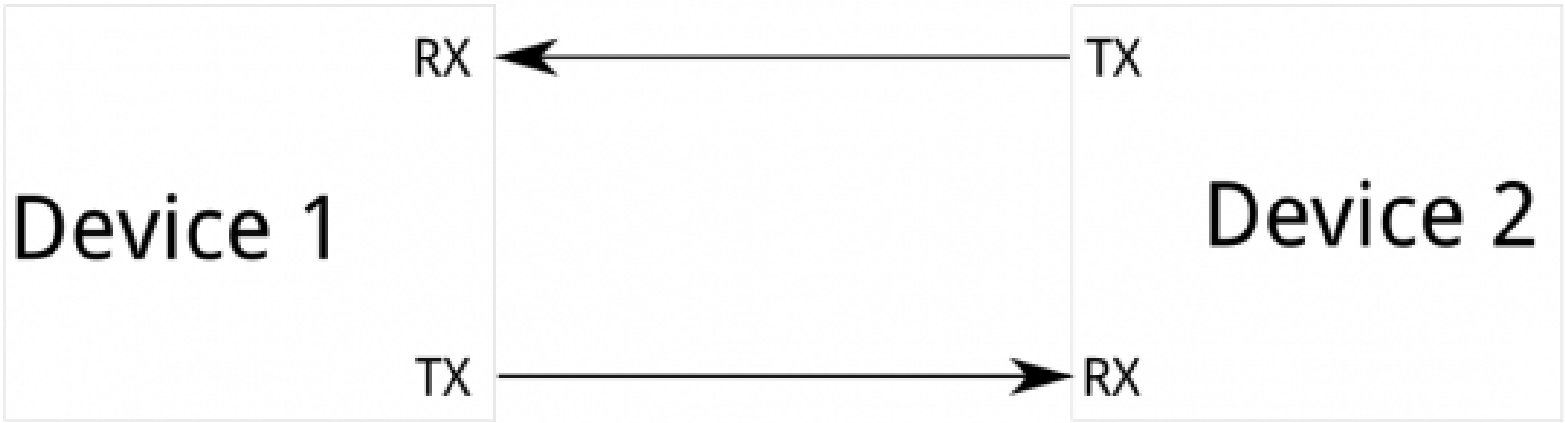
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L06_04_encoder_switch
   - Connect your microcontroller to the encoder switch and LEDs
   - Use the switch to turn on/off the NeoPixels.
   - Also, the encoder LED should be red for off and green for on.

Buses and Interfaces

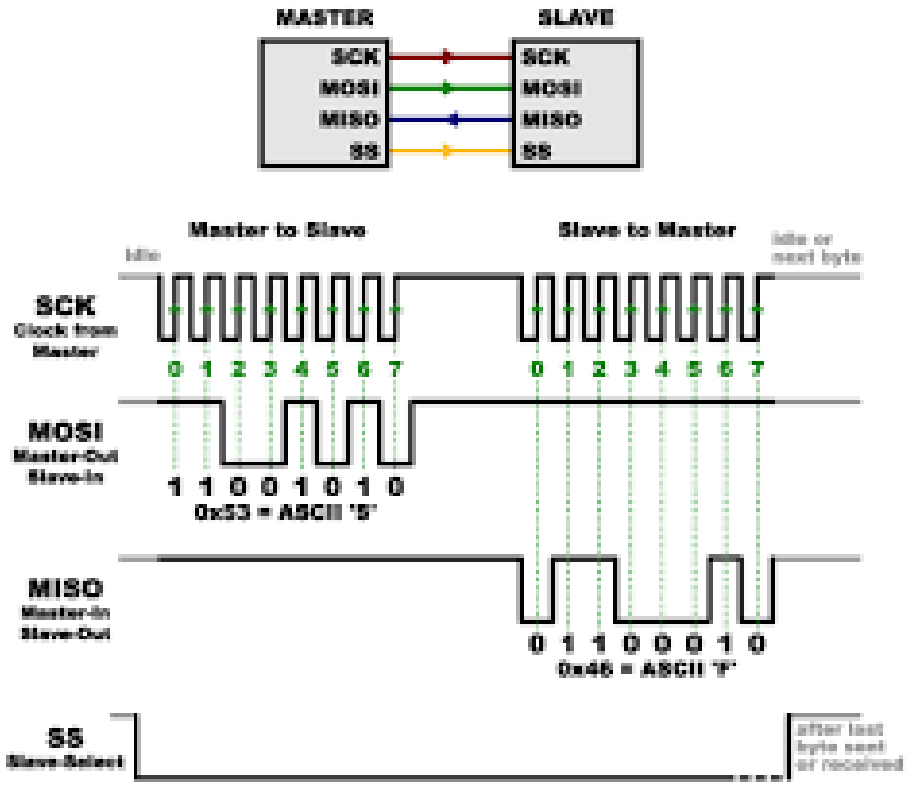


UART



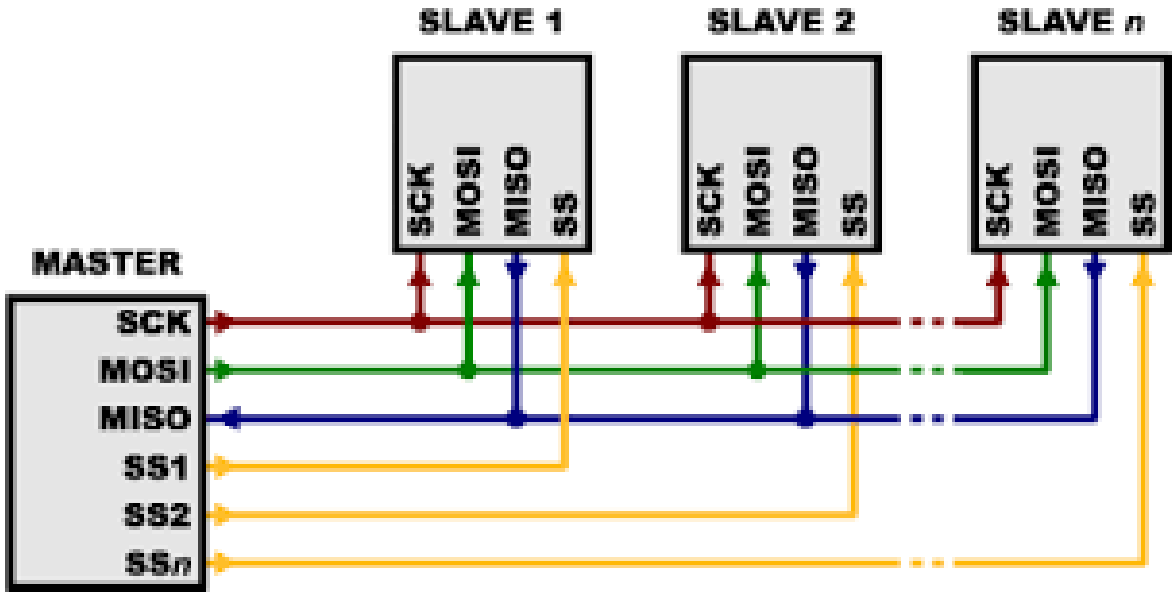
Universal Asynchronous Receiver/Transmitter

Serial Peripheral Interface



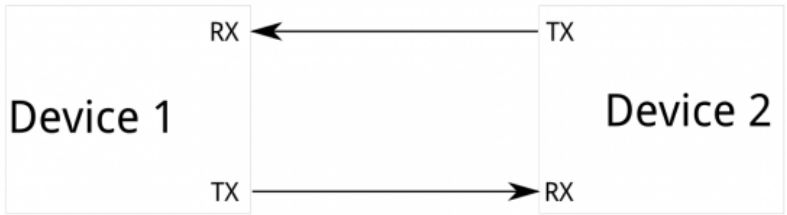
- Master Out, Slave In (MOSI) connects to Data In
- Master In, Slave Out (MISO) connects to Data Out

Serial Peripheral Interface

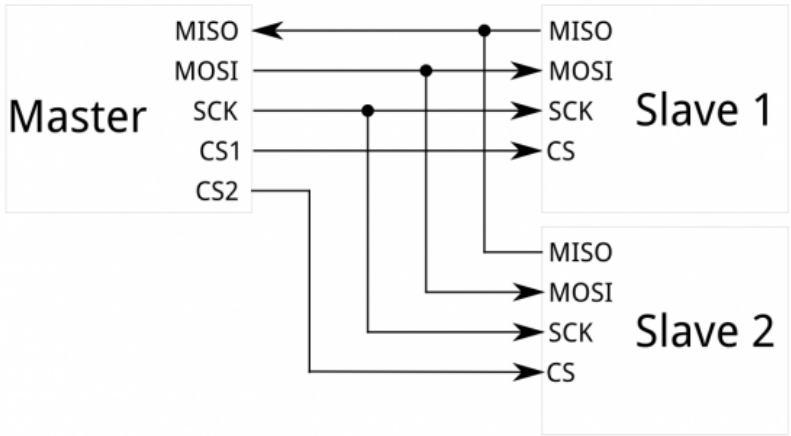


Serial +/−

UART



SPI



Assignment: Write to μSD Card

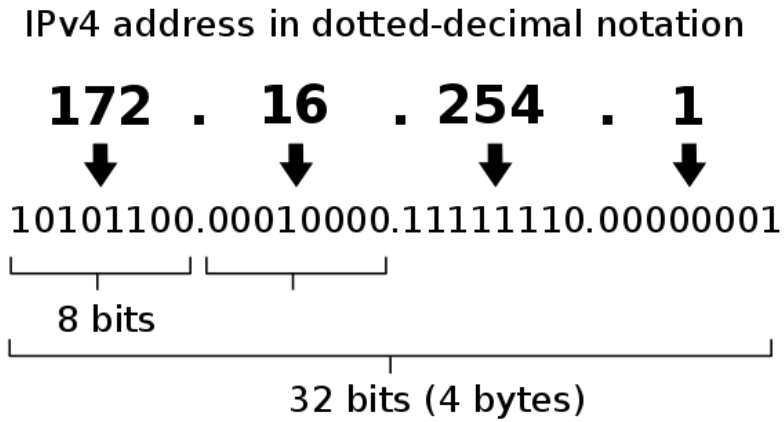
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L07_01_dataLogger
   - String datatype
   - SPI and SD libraries
   - File object
   - SD.begin()
   - dataFile.print()
   - Simulate 3 data streams and write to SD card

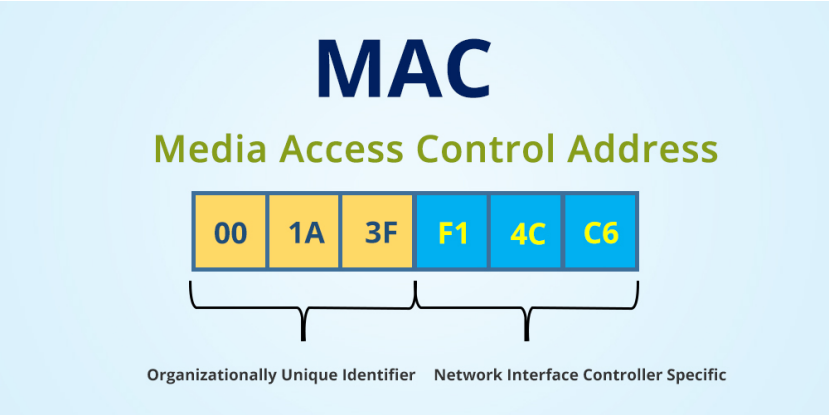
The Internet

IP Addresses

- When a device joins the network it is given an internet address.
  - static or dynamic
- IPv4 (32-bit) - 4.2 billion
- IPv6 (128-bit) - 340 quadrilliard
- In Powershell, try: *ipconfig /all*
- In Terminal (MAC), try: ipconfig getifaddr en0



MAC Address



A MAC Address is a unique 6-byte (48-bit) address that is usually permanently burned into a network interface card (NIC) and uniquely identifies the device on an Ethernet-based network. The uniqueness of MAC addresses is ensured by IEEE.

If you are creating your own MAC address, the 2's place bit of the first byte, the "locally administered bit" should be set. The 1's place bit, the "globally administered" bit must be off.

Therefore, xA-xx-xx-xx-xx-xx is valid, while x7-xx-xx-xx-xx-xx is not.

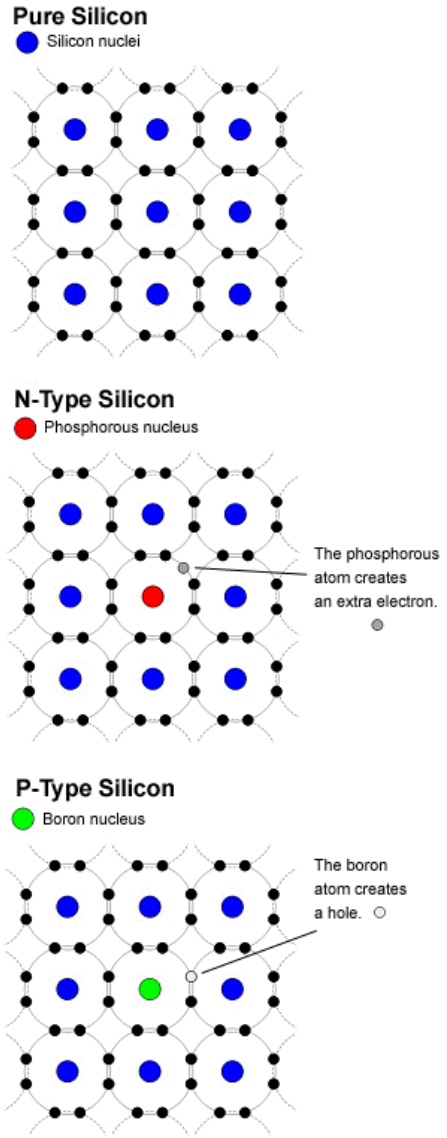
Assignment: Wemo

- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. Create a mac.h header file
2. L08_00_EthernetTest
   - Modify to use your mac.h file
3. L08_01_Wemo
   - Using the wemo.h library and WemoH_Example as a template, create code that turns on and off multiple Wemo Smart Outlets in the classroom.
4. L08_02_Wemo_Object
   - Modify the wemo.h library from being a function to being Class and Methods.
   - Modify your wemo code the create and use a wemo object.

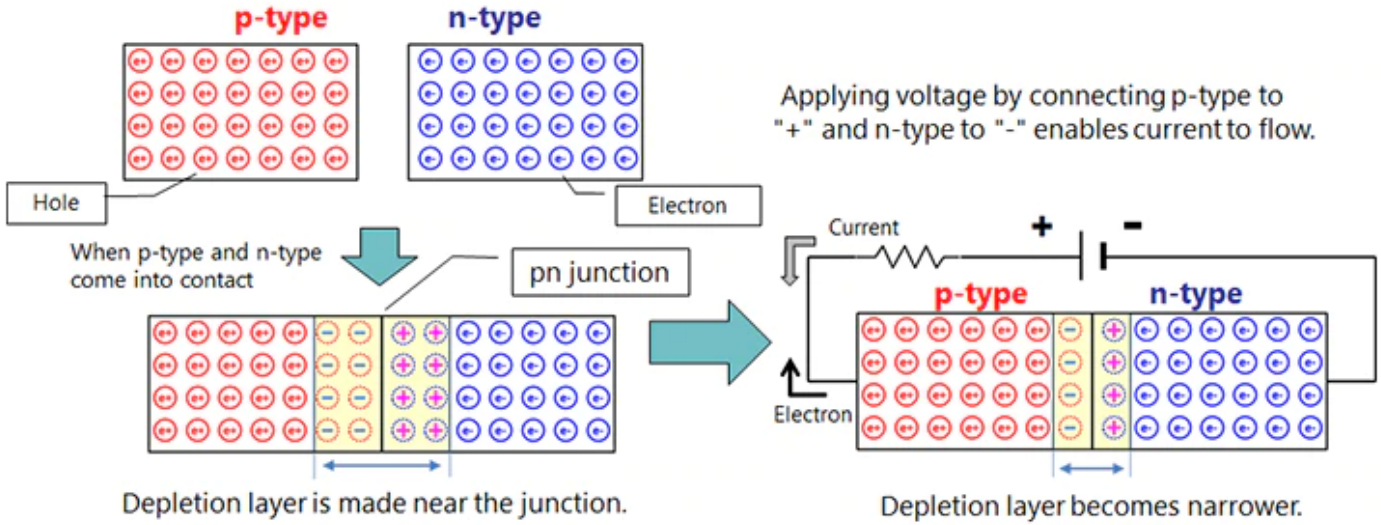
Semiconductor

- A silicon atom has four electrons in its outer shell and bonds tightly with four surrounding silicon atoms creating a crystal matrix with eight electrons in the outer shells. The tight bonds make pure silicon non-conducting.
- Phosphorus has five electrons, and when combined, the fifth electron becomes a "free" electron that moves easily within the crystal when a voltage is applied.
- Boron has only three electrons in its outer shell and can bond with only three of surrounding silicon atoms. Thus one silicon atom has a vacant location in its outer shell, called a "hole," that readily accepts an electron.



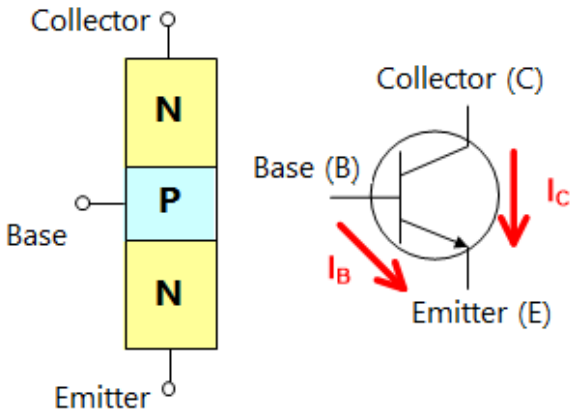
pn junction diode

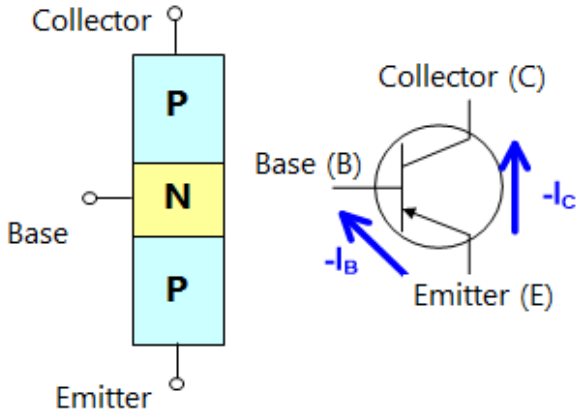
- When p-type and n-type semiconductors are bonded, holes and free electrons are attracted, combine, and disappear near the boundary. Since there are no carriers in this area, it is called a depletion layer and it is an insulator.
- A positive voltage applied to the p-type region causes electrons to flow sequentially from the n-type. The electrons will first disappear by combining with holes, but excess electrons move to the positive pole and current will flow.



Bipolar Junction Transistor

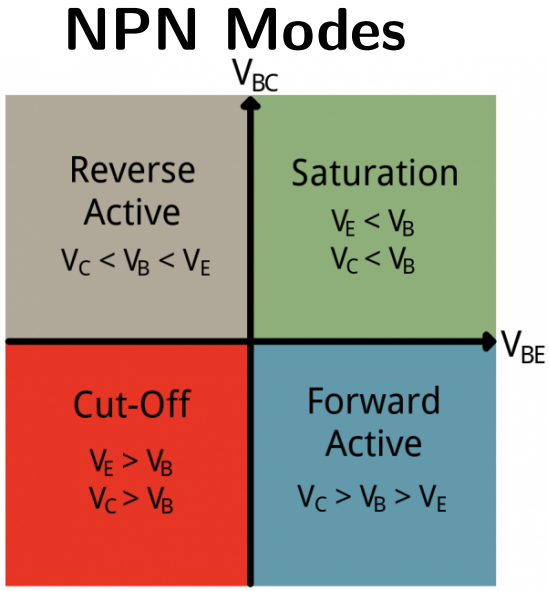
The transistor has three regions, namely base, emitter and collector. The emitter is a heavily doped terminal and emits electrons into the base. Base terminal is lightly doped and passes the emitter-injected electrons on to the collector. The collector terminal is intermediately doped and collects electrons from base. This collector is large as compared with other two regions so it dissipates more heat.





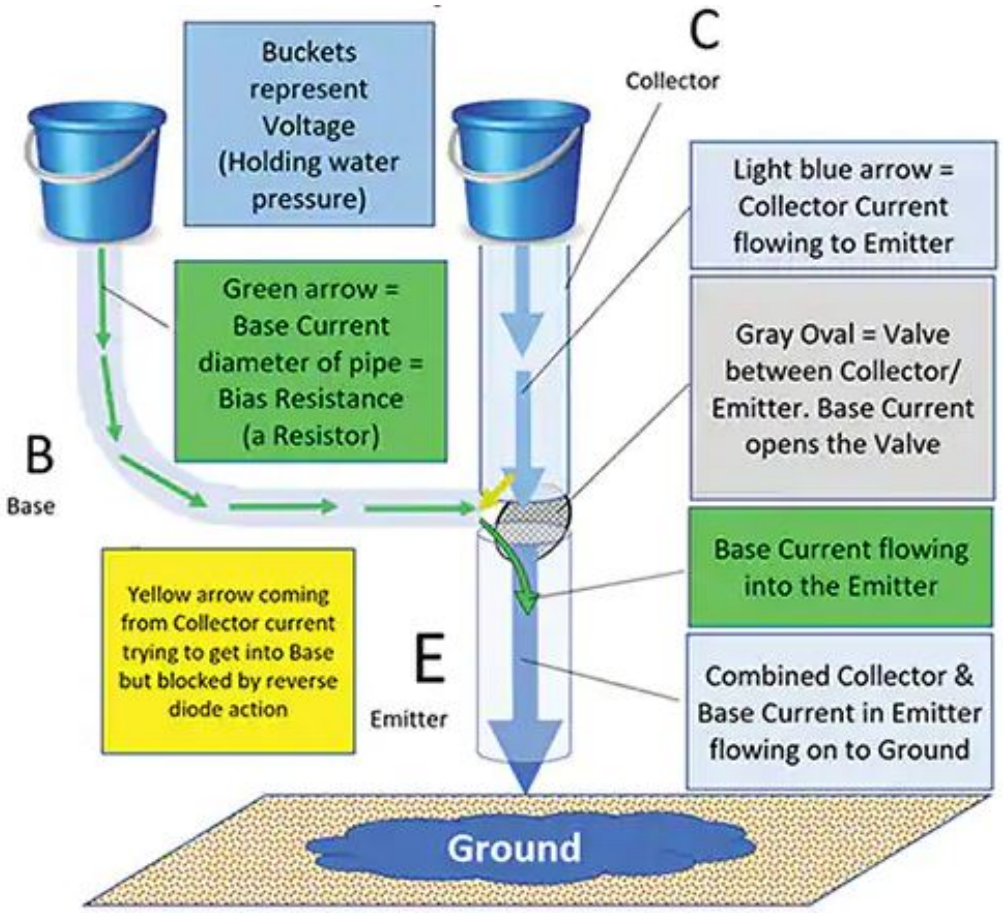
Bipolar Junction Transistor - Modes of Operation

- **Saturation:** Current freely flows from collector to emitter. (ON Switch)
- **Cut-off:** No current flows from collector to emitter. (OFF Switch)
- **Active:** The current from collector to emitter is proportional to the current flowing into the base. (Amplifier)
- **Reverse-Active:** Like active mode, the current is proportional to the base current, but it flows reverse from emitter to collector (not the purpose transistors were designed for).



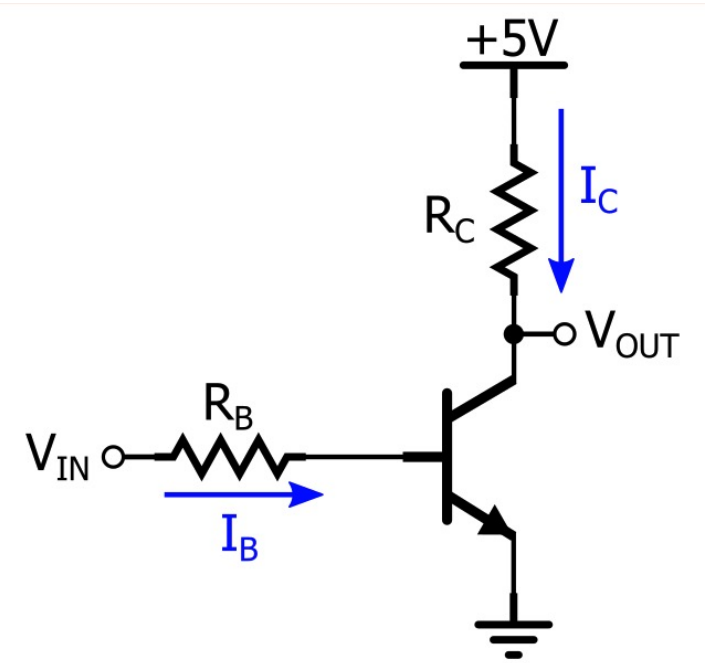
Voltage relations	NPN Mode	PNP Mode
$V_E < V_B < V_C$	Active	Reverse
$V_E < V_B > V_C$	Saturation	Cutoff
$V_E > V_B < V_C$	Cutoff	Saturation
$V_E > V_B > V_C$	Reverse	Active

Water Analogy



Active Mode NPN Transistor Circuit

If you apply a voltage V_{IN} that is high enough to forward-bias the base-to-emitter junction, current (I_B) will flow from the input terminal, through R_B, through the BE junction, to ground. Current (I_C) will also flow through R_C and the collector-to-emitter portion of the transistor.

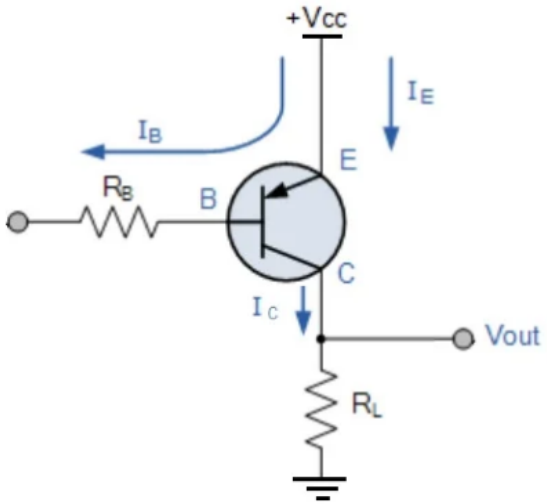


NOTE: V_{OUT} is an amplified but inverted signal of V_{IN}.
This simple circuit will step-up an 0 - 3.3V output from the microcontroller to 0 - 5.0V (inverted). The low impedance of the output will also provide sufficient current to drive a higher current device (e.g., a relay).

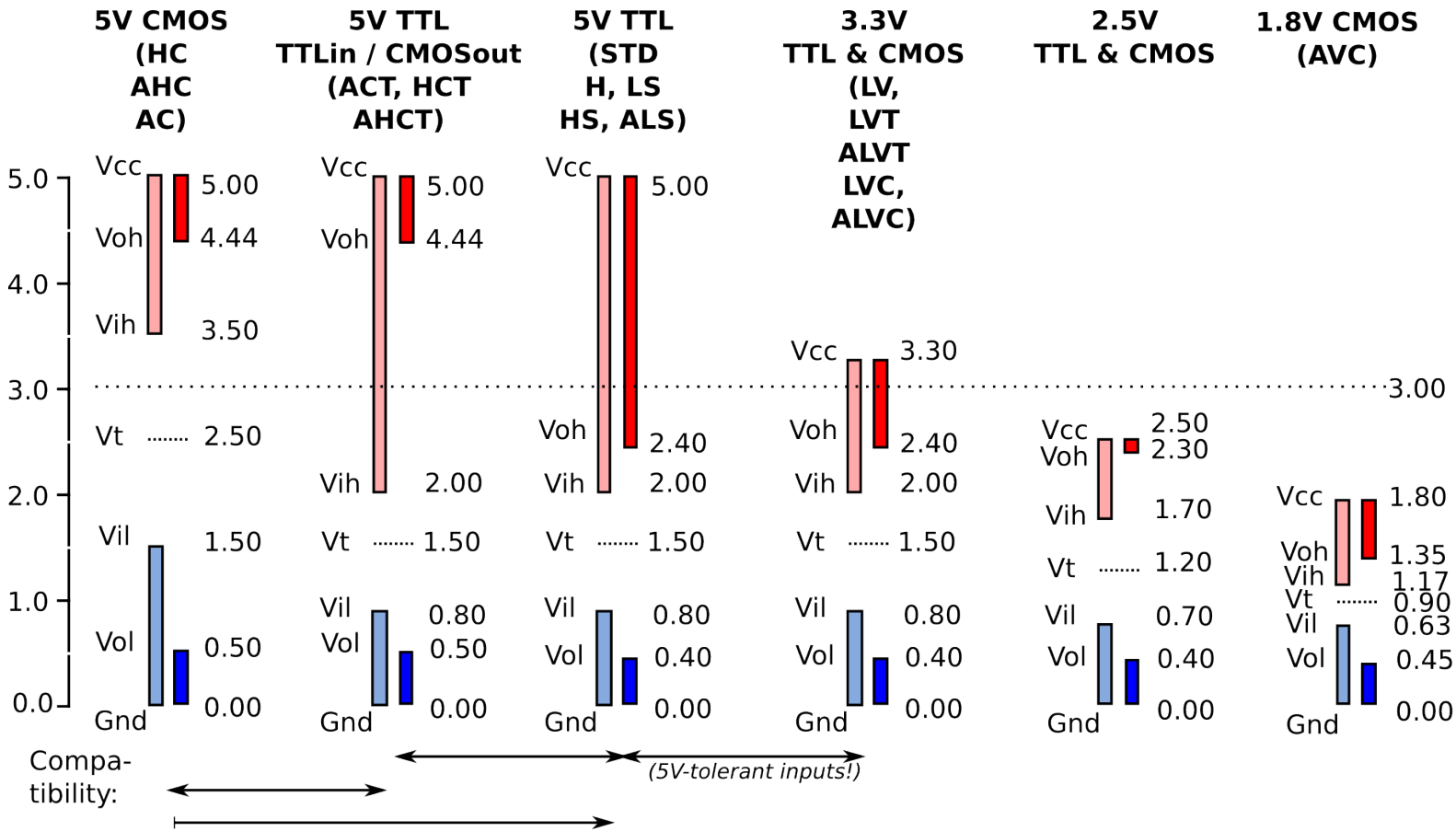
PNP Transistor

NPN Transistors are more common than PNP for a number of reasons:

- The voltage and current behavior of an NPN transistor is significantly more intuitive.
- When a switch or driver circuit is required, NPNs provide a more straightforward interface to digital output signals (such as a control signal generated by a microcontroller).
- NPNs are higher performance (faster switching speeds) due to higher mobility of electrons vs holes.



Logic Voltage Level Standards

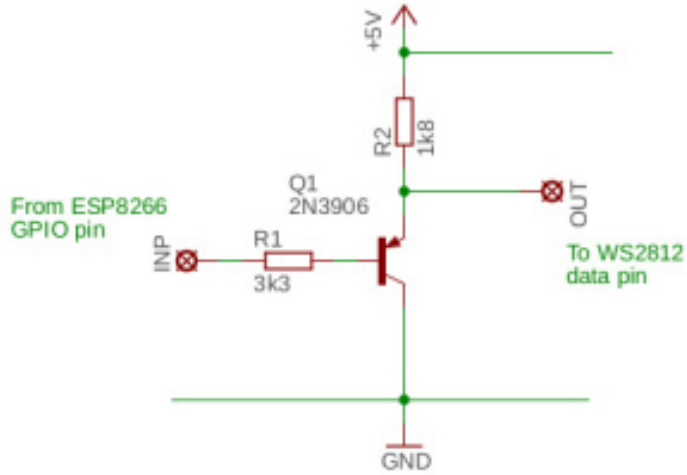


Data source: EETimes, A brief recap of popular logic standards (Mark Pearson, Maxim).

Or, what is wrong with the NeoPixels.

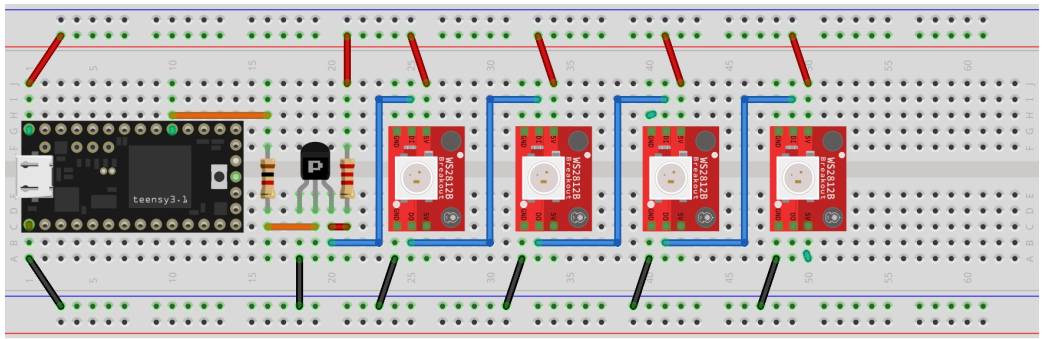
Emitter Follower

- NeoPixels are designed around 5V CMOS transistors
  - $V_{IH} > 3.5V$
  - 3.3V Microcontroller $V_{OH} = 3.3V$
- An Emitter Followers (i.e., a PNP transistor wired backwards) is a current amplifier, but will also produce a $V_{OUT} = 3.9V$.
- Alternatively, the first NeoPixel could be sacrificed by reducing its V_{cc} to 4.3V with a diode.

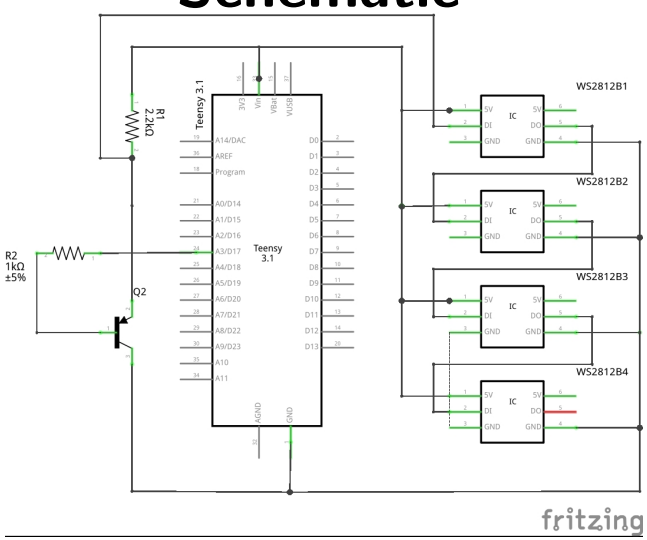


Emitter Follower Layout

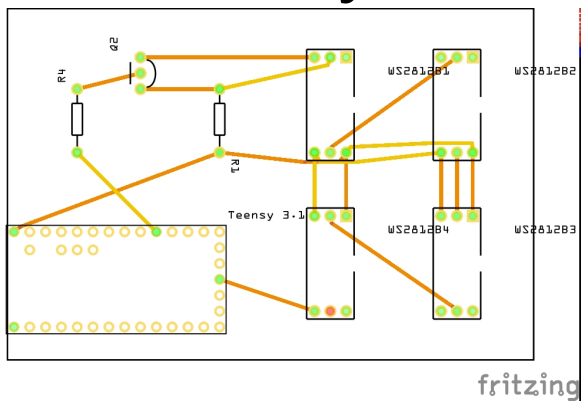
Fritzing



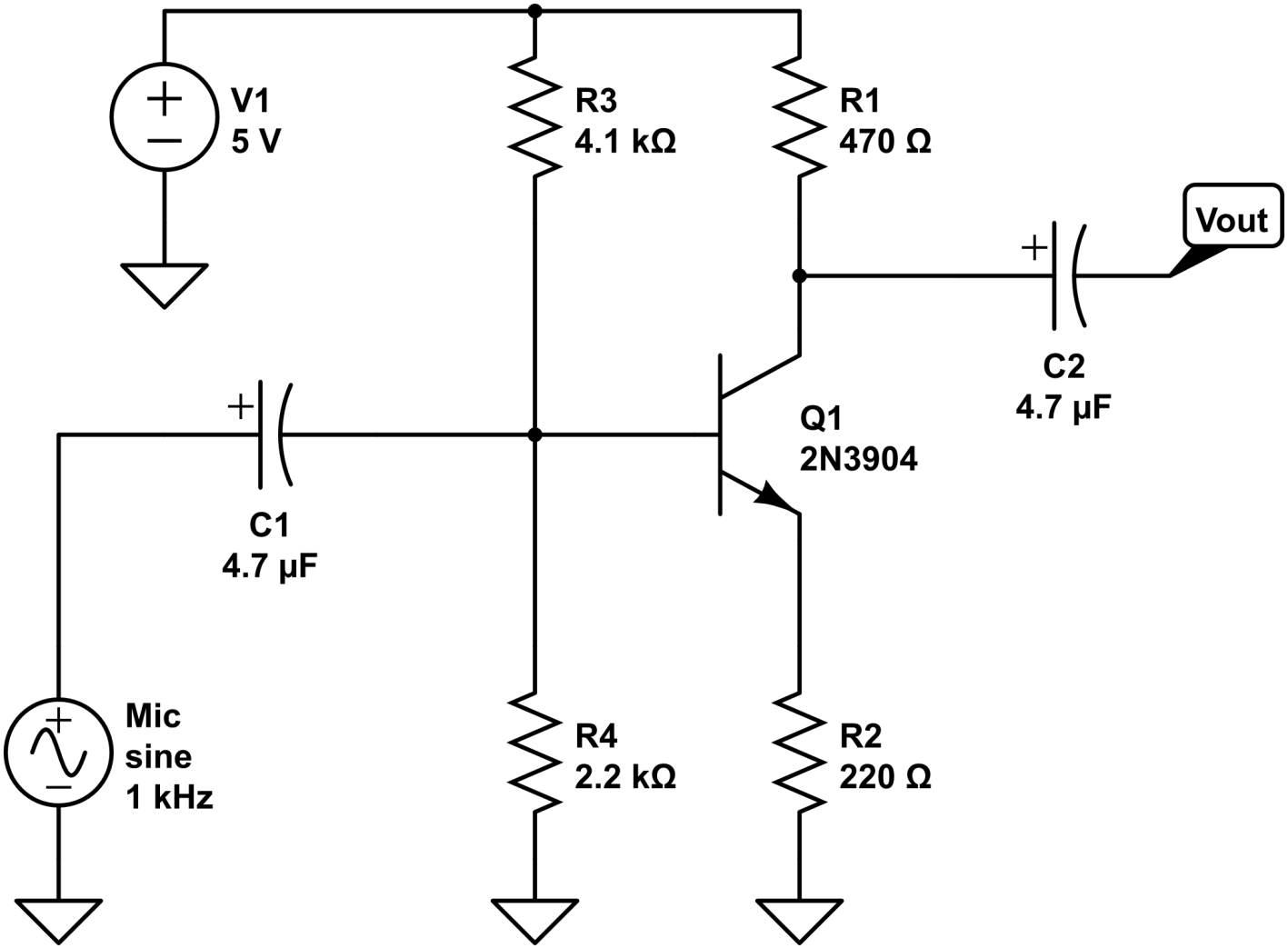
Schematic



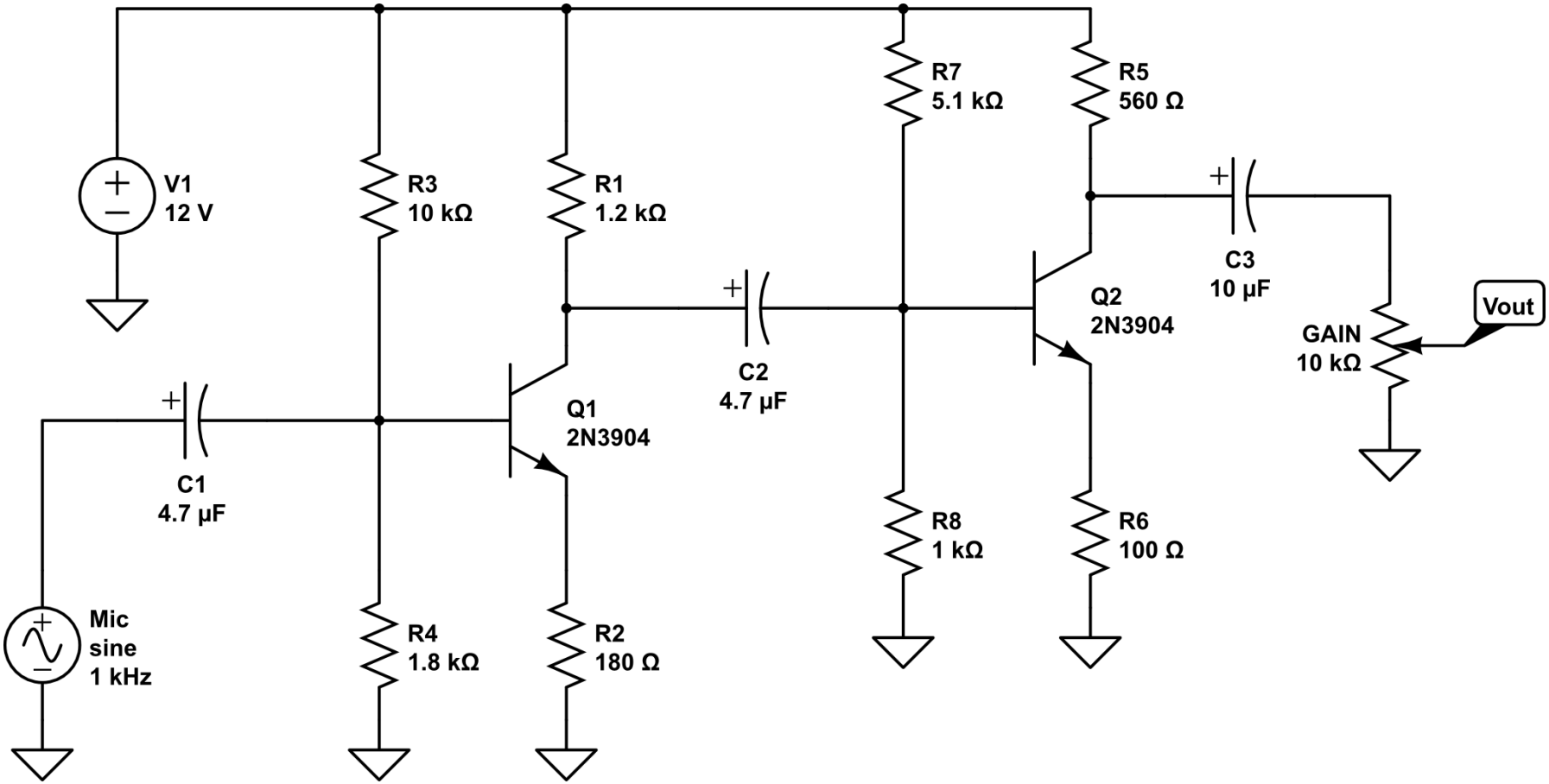
PCB Layout



NPN Pre-Amplifier Circuit

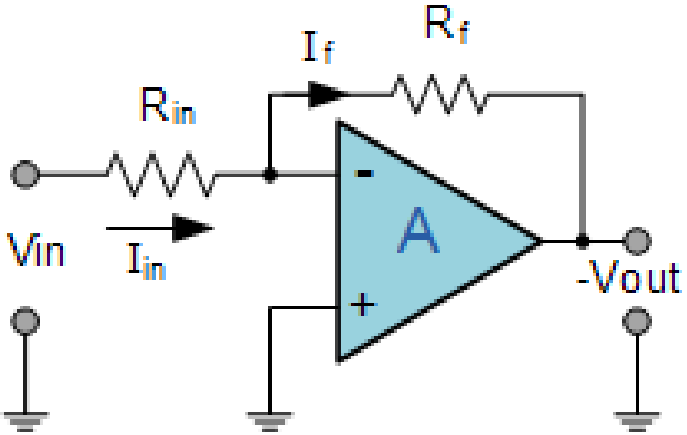


Two Stage Amplifier



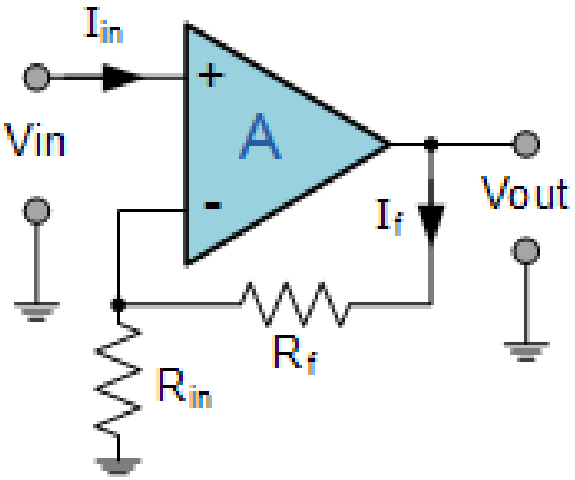
Op Amp Lesson - Under Construction

Inverting Op-amp



$$A = \frac{Vout}{Vin} = -\frac{Rf}{Rin}$$

Non-inverting Op-amp

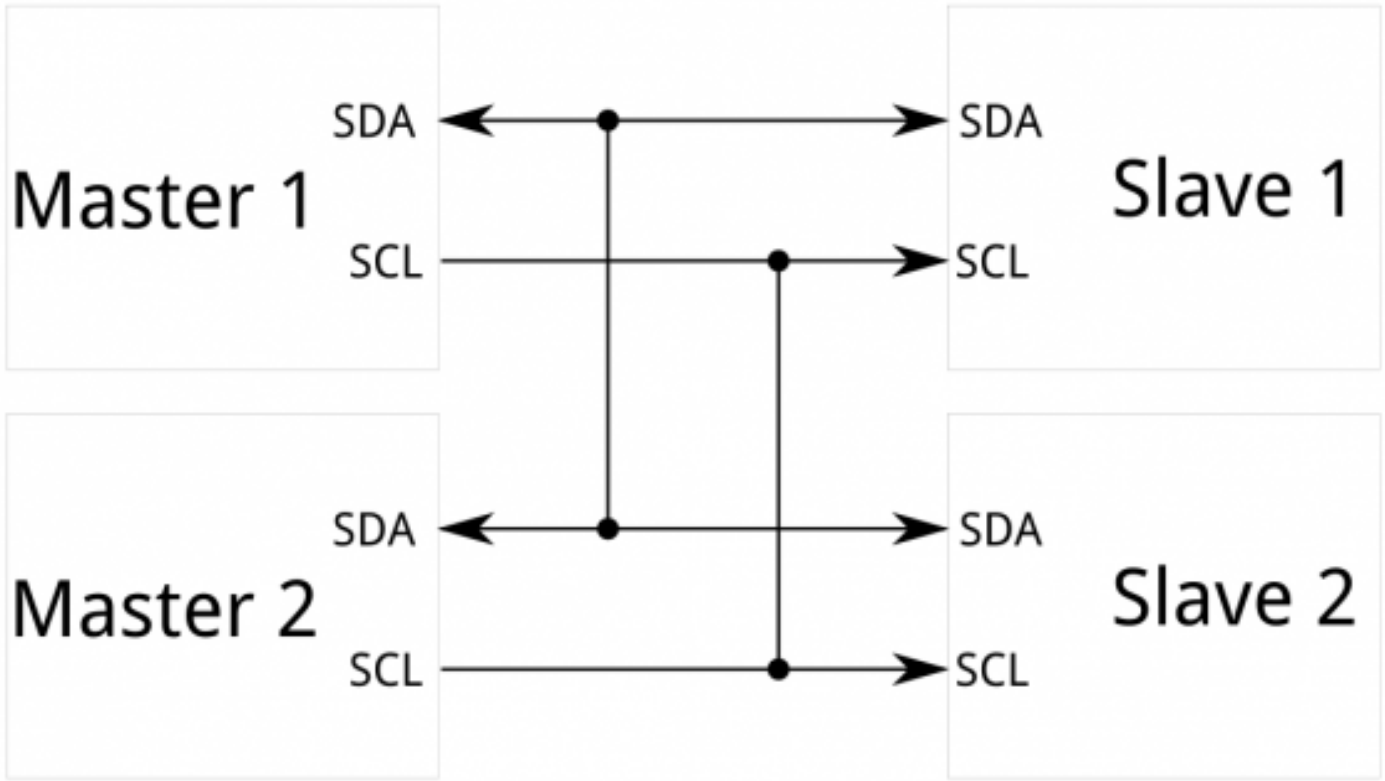


$$A = \frac{Vout}{Vin} = 1 + \frac{Rf}{Rin}$$

Assignment: Amplifiers

1. L09_01_NeoPixel
2. L09_02_NPNAmp
   - Make NPN Amp
3. L09_03_OpAmp
   - Op Amp

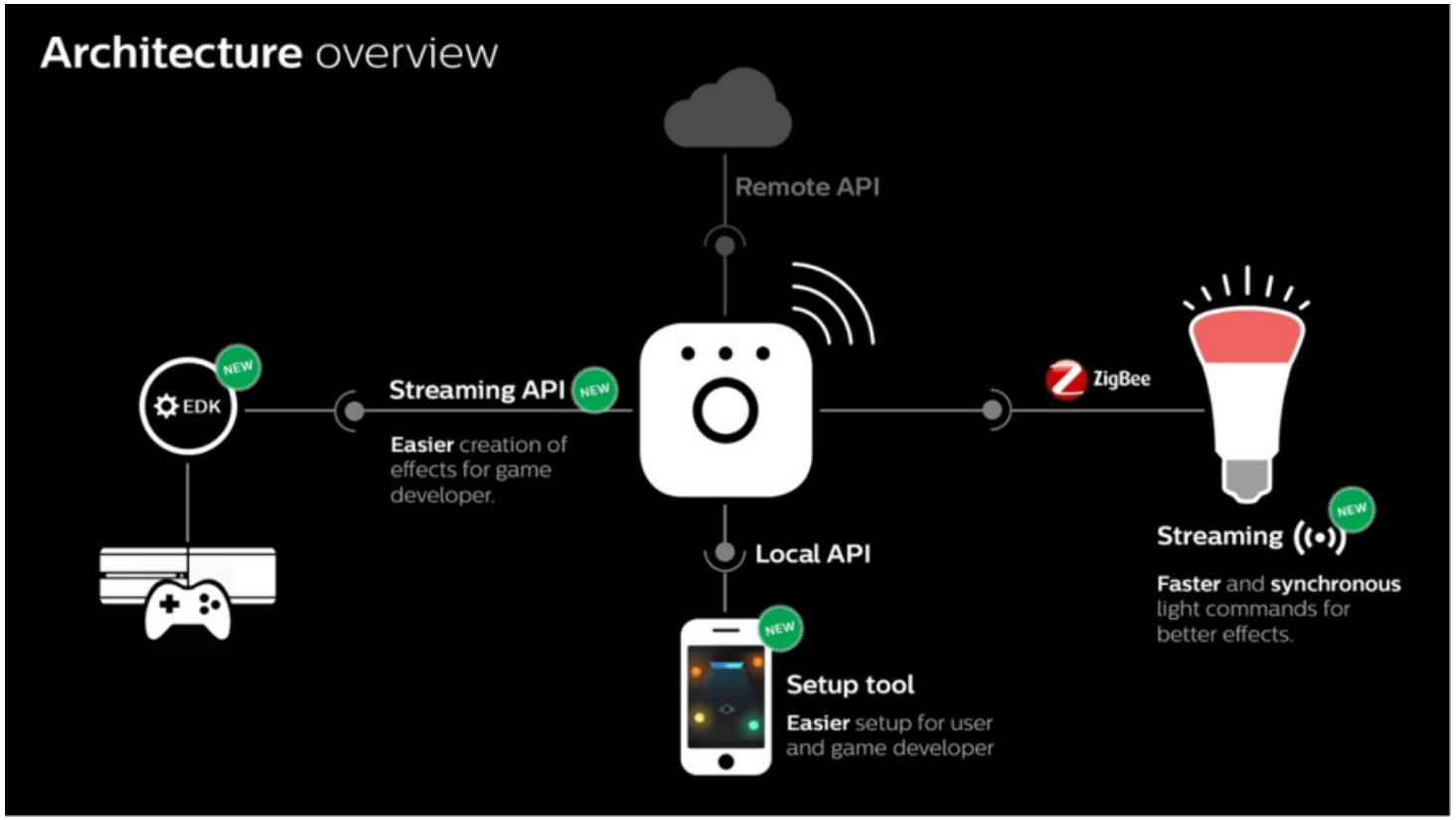
Inter-integrated Circuit (I^2C)



Assignment: I^2

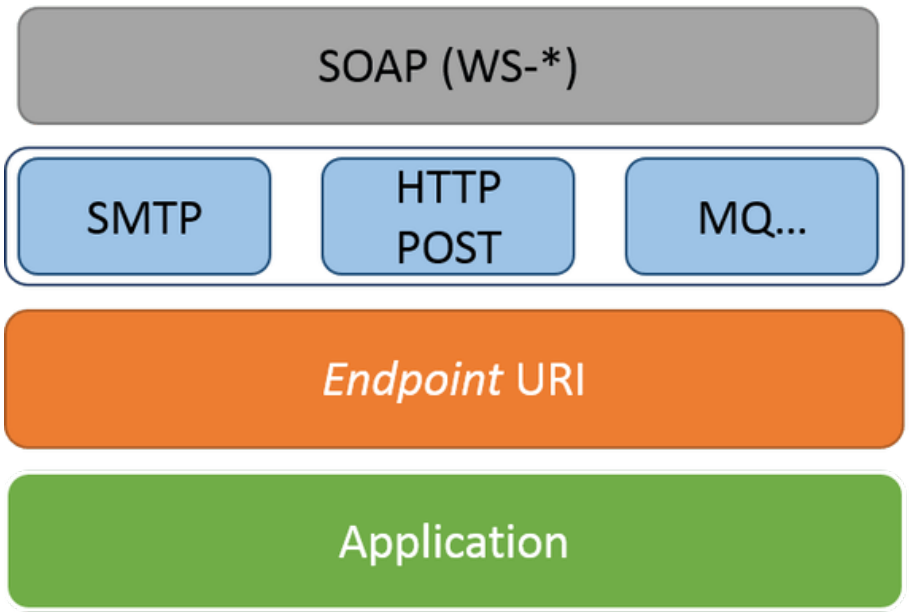
1. L10_01_OLEDWrite
   - TASKS
2. L10_02_BME280
   - TASKS
3. L10_03_BME280_OLED
   - TASKS
4. L10_04_BME280_SDMicro
   - TASKS

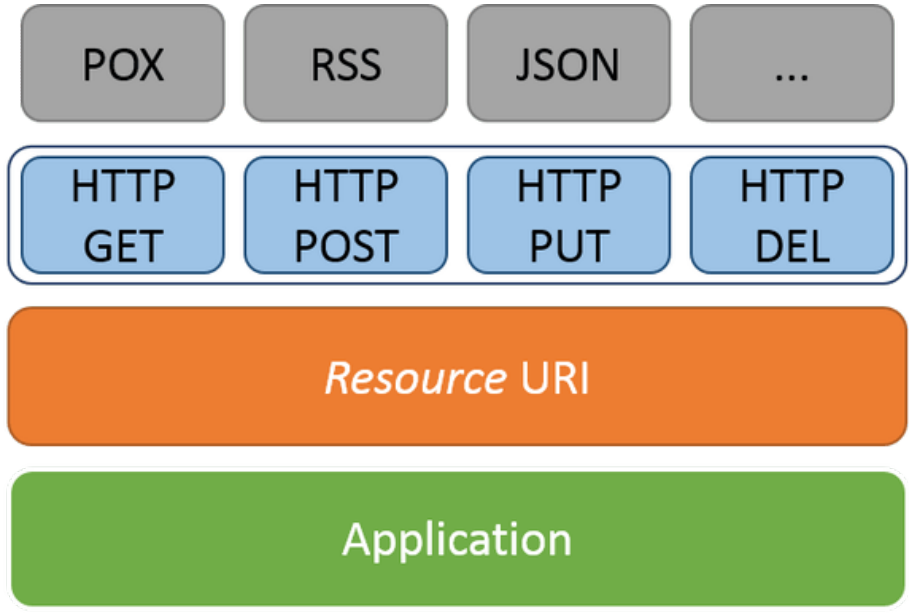
Phillips Hue API



Application Programming Interface: a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

SOAP vs REST


SOAP (WS-*)
SMTP
HTTP POST
MQ...
Endpoint URI
Application
SOAP


POX
RSS
JSON
...
HTTP GET
HTTP POST
HTTP PUT
HTTP DEL
Resource URI
Application
REST

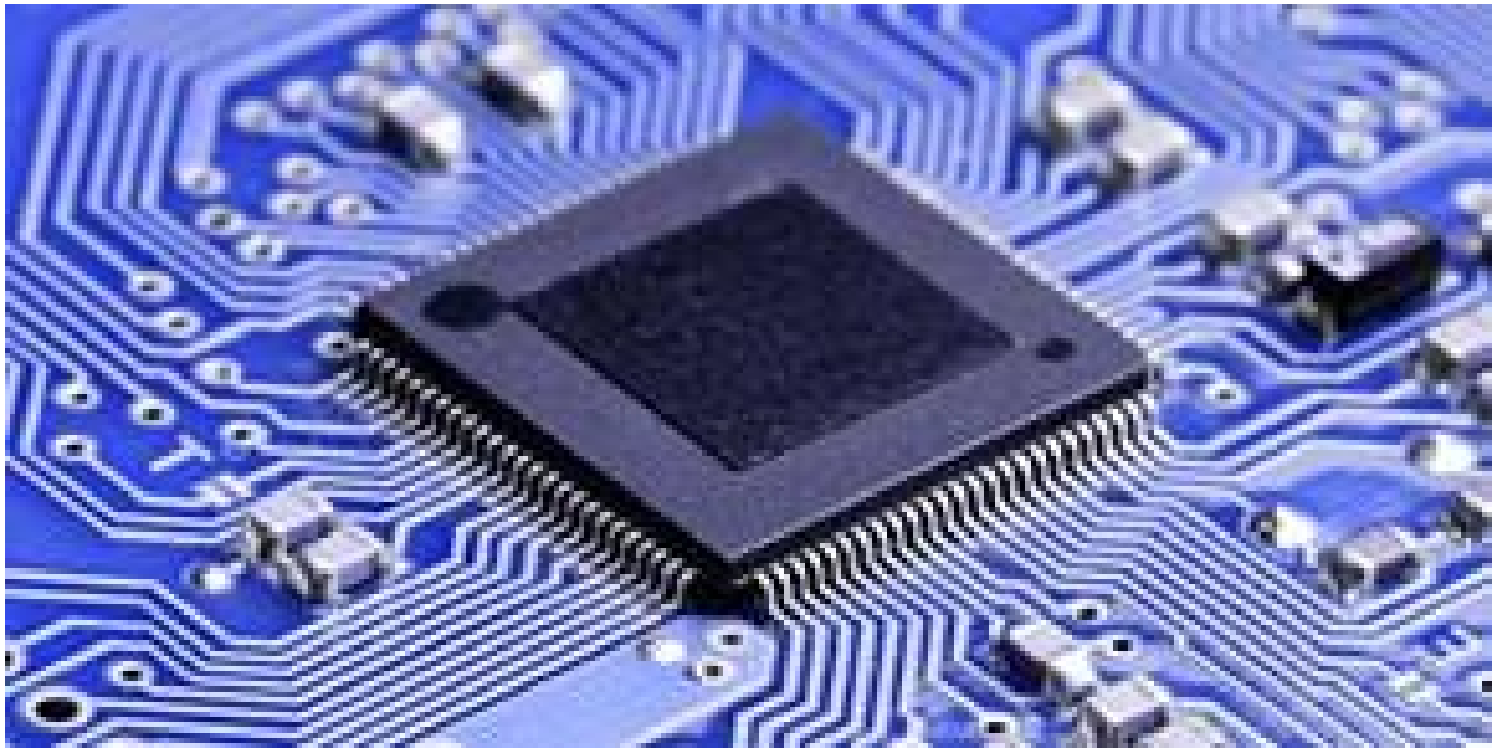
Assignment: L11_01_Hue

- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

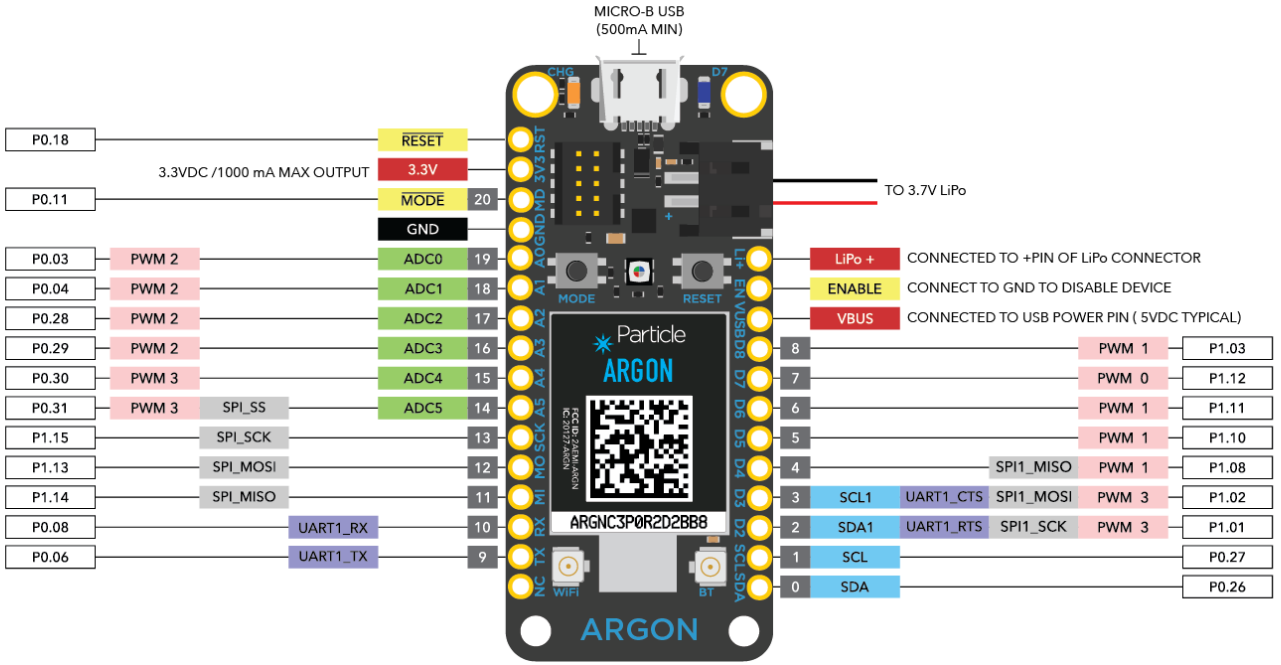
1. Using the hue.h library and HueH_Example as a template, create code that
   - button that turns on and off the Hue light at your pod
   - uses the encoder to change the brightness of the Hue bulb
   - has a method of cycling the Hue light through the colors of the rainbow.


WORK
IN PROGRESS

Our Second Microcontroller

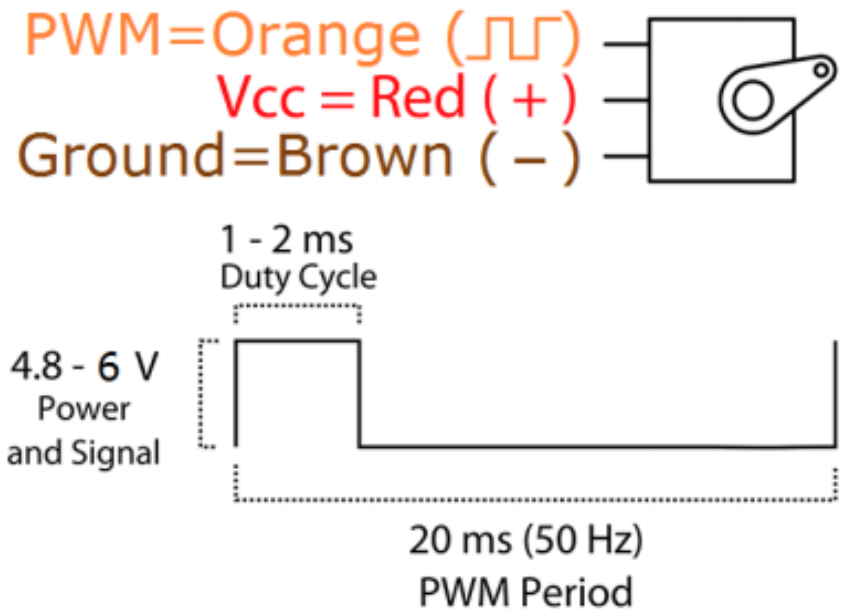


Particle Argon Pin Layout

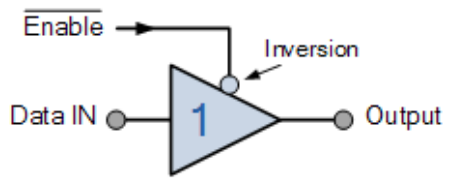


v1.0

Servo Motors



TriState Buffer

Symbol	Truth Table		
	Enable	IN	OUT
 Tri-state Buffer	0	0	0
	0	1	1
	1	0	Hi-Z
	1	1	Hi-Z
Read as Output = Input if Enable is **NOT** equal to "1"