IoT Product Design and Rapid Prototyping

Brian Rashap, Ph.D.

1-JUN-2022

IoT Fun

Introduction

Brian Rashap, Ph.D.

- Proud husband of Krista and father of Shelby (23) and Ethan (19)
- Electrical Engineer with 25 years industrial experience
- High School track coach
- Hobbies: running, cycling, reading, spending time with family

Christian Chavez, CEO, Vital Grow Inc

- Student from IoT Cohort 3
- CEO of Vital Grow, an IoT Agriculture Company

Introductions

INTRODUCTIONS

Class Rules

- Respect each other. Help each other.
- Ask questions.
- Be on time (let us know via Slack if you won't be here)
- Keep your workspace and the classroom neat and tidy.
- If you are struggling, let me, Susan, or Esteban know. We are here to HELP!
- Class hours
  - Mon-Th: 8am to 5pm [1]
  - Friday: 8am to 3pm [2]
  - Lunch Break: 1 hour near noon. Maybe combined with work time.
  - Please respect Brian and Christian's lunch break as well.

[1] Doors open at 7:50, please be in your seats ready to learn by 8:00

[2] Occasionally on Friday there will be optional activities from 3 to 5

Grading

Assignments total 1000 points. To graduate, you need to earn at least:

- 750 total points
- 200 points (80%) on the Capstone.
- 65 points (65%) on Quizzes, the two Midterms, and Solidworks.

Point distribution

1. IoT assignments + Lab Notebooks: 300 points [3]
2. 3D modeling (Solidworks) assignments: 100 points
3. Weekly quizzes: 100 points
4. Midterm Projects: Smart Room Controller/Plant Watering System: 100 points each (200 total)
5. Team Capstone Project: 250 points
6. Professional Development: 50 points

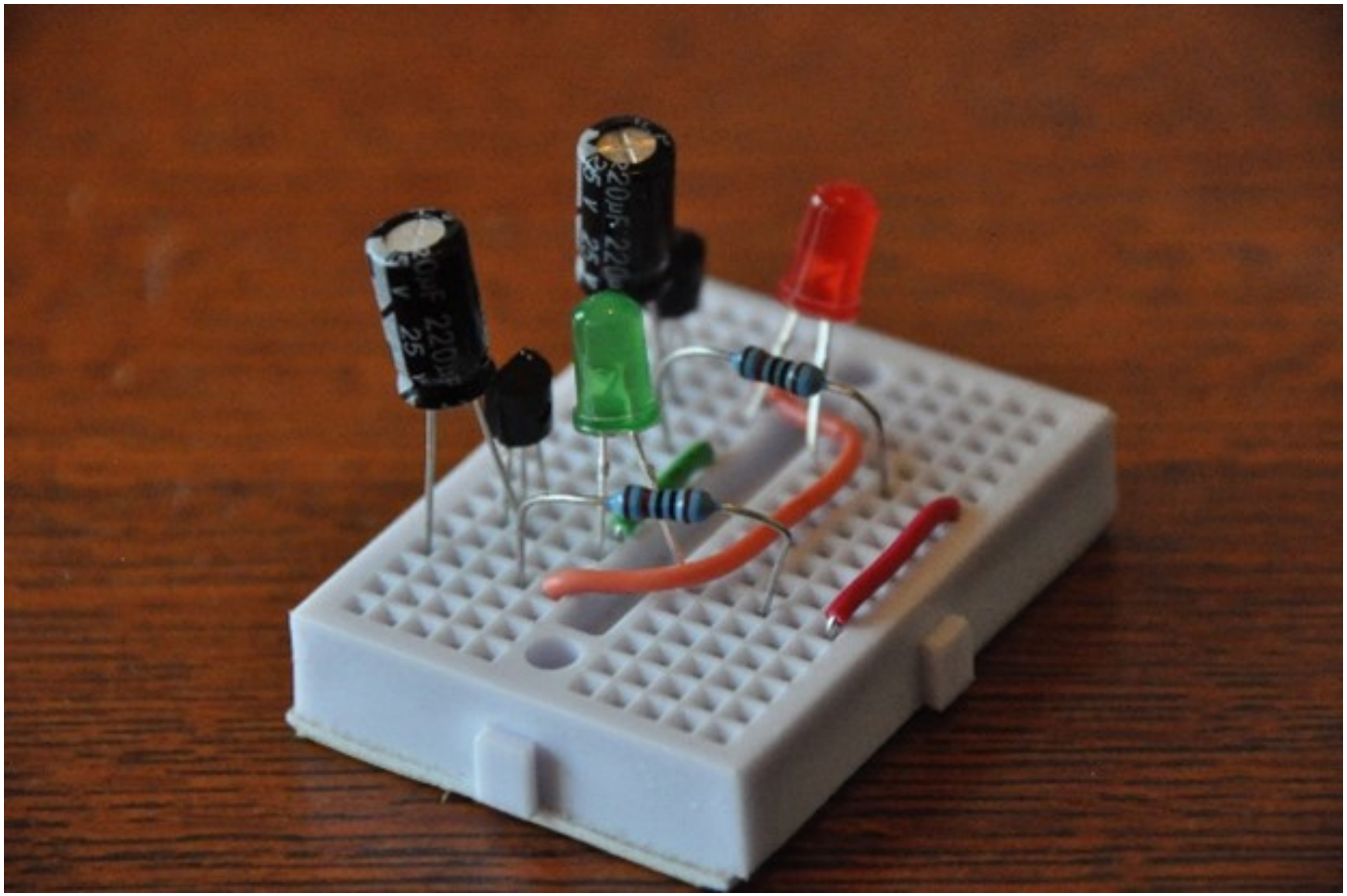
[3] All coding assignments must follow style-guide

Credit for Prior Learning (CPL)

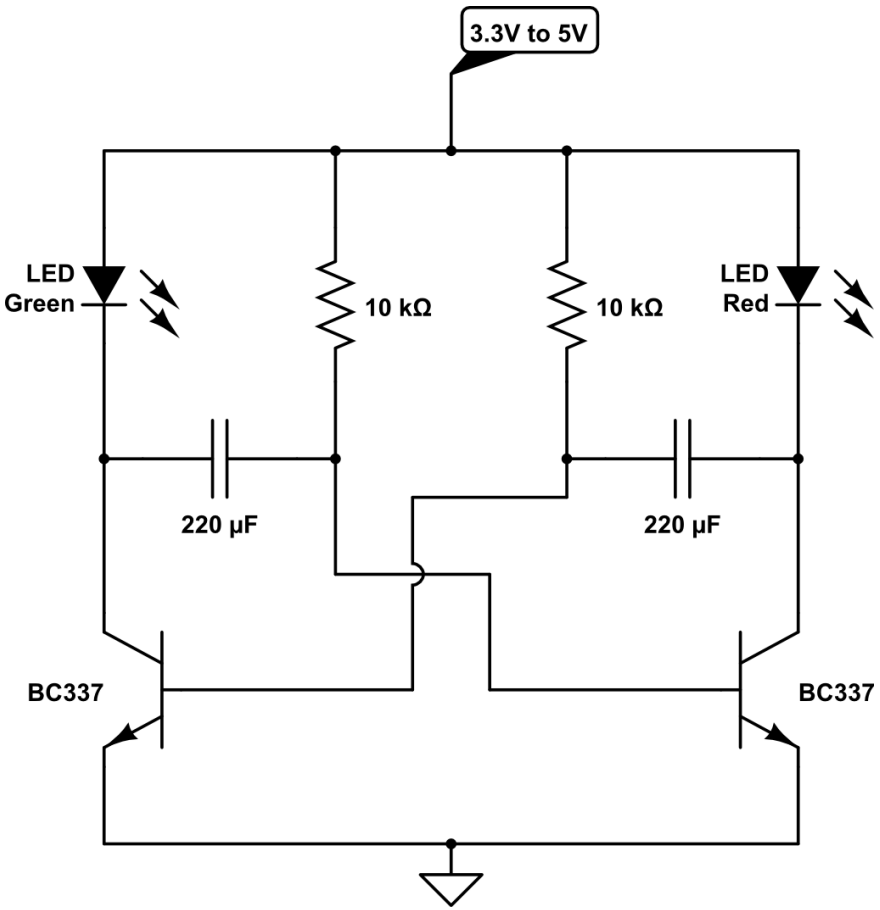
Approved for CPL	
CIS 1605	Internet of Things
CIS 1275	Introduction to C++
BCIS 1110	Fundamentals of Information Literacy and Systems
BUSA 1130	Business Professionalism
BUSA 1198	Project Management Fundamentals
CSIS 1151	Intro to Programming for Non-Majors of CS*

* CSIS 1151 credit requires appropriate math prerequisites

Let's Build Something



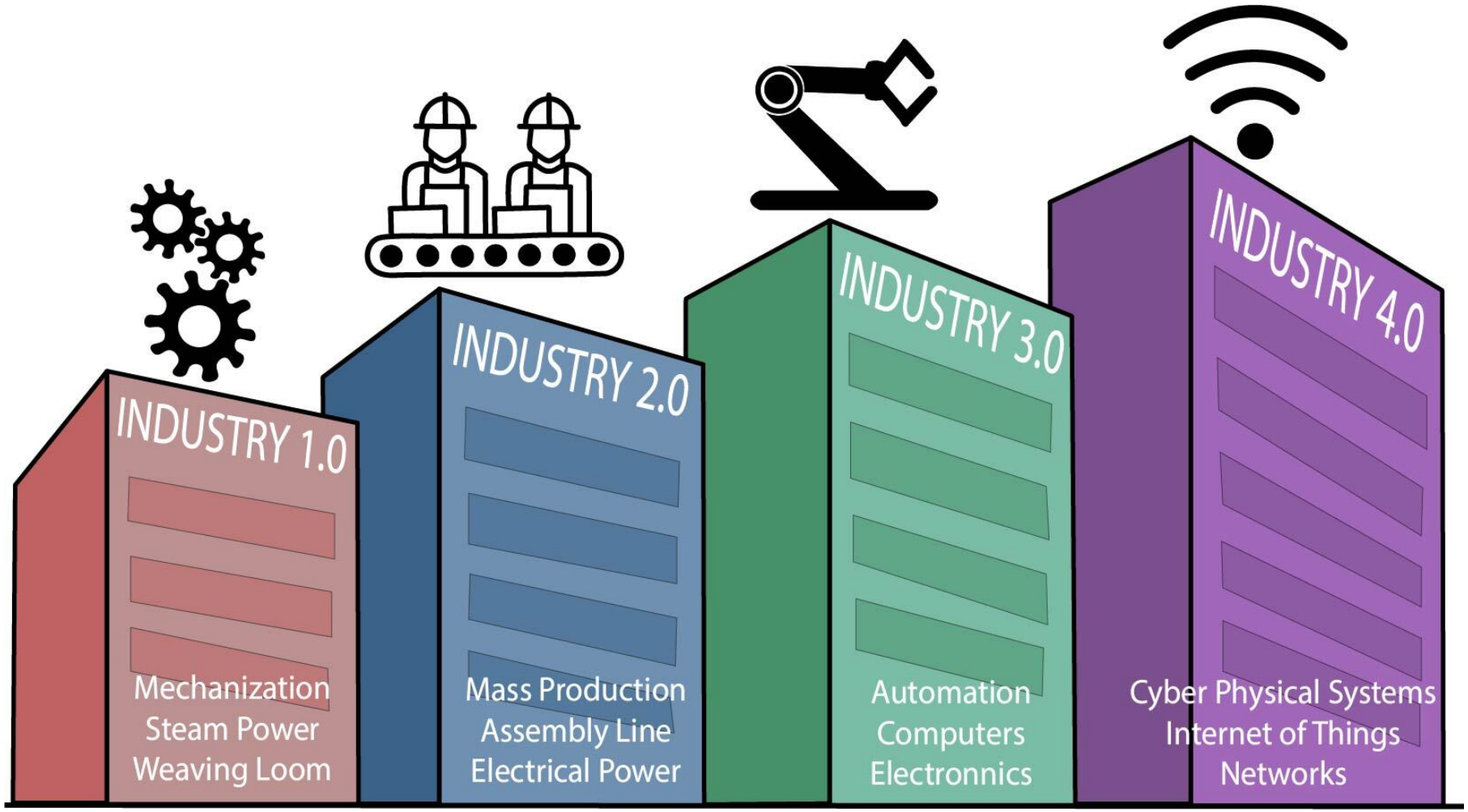
Oscillator: Flip Flop



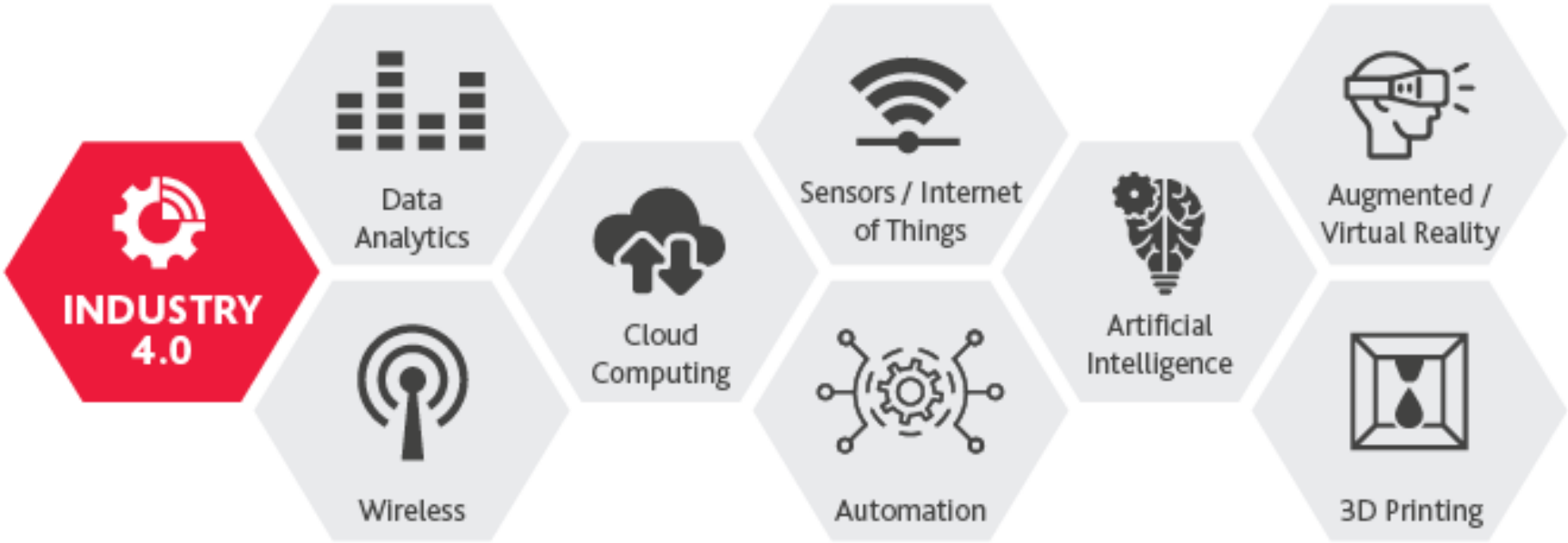
Components:

- breadboard
- light emitting diode (LED)
- wires
- resistors
- capacitors
- transistors
- battery charge circuit

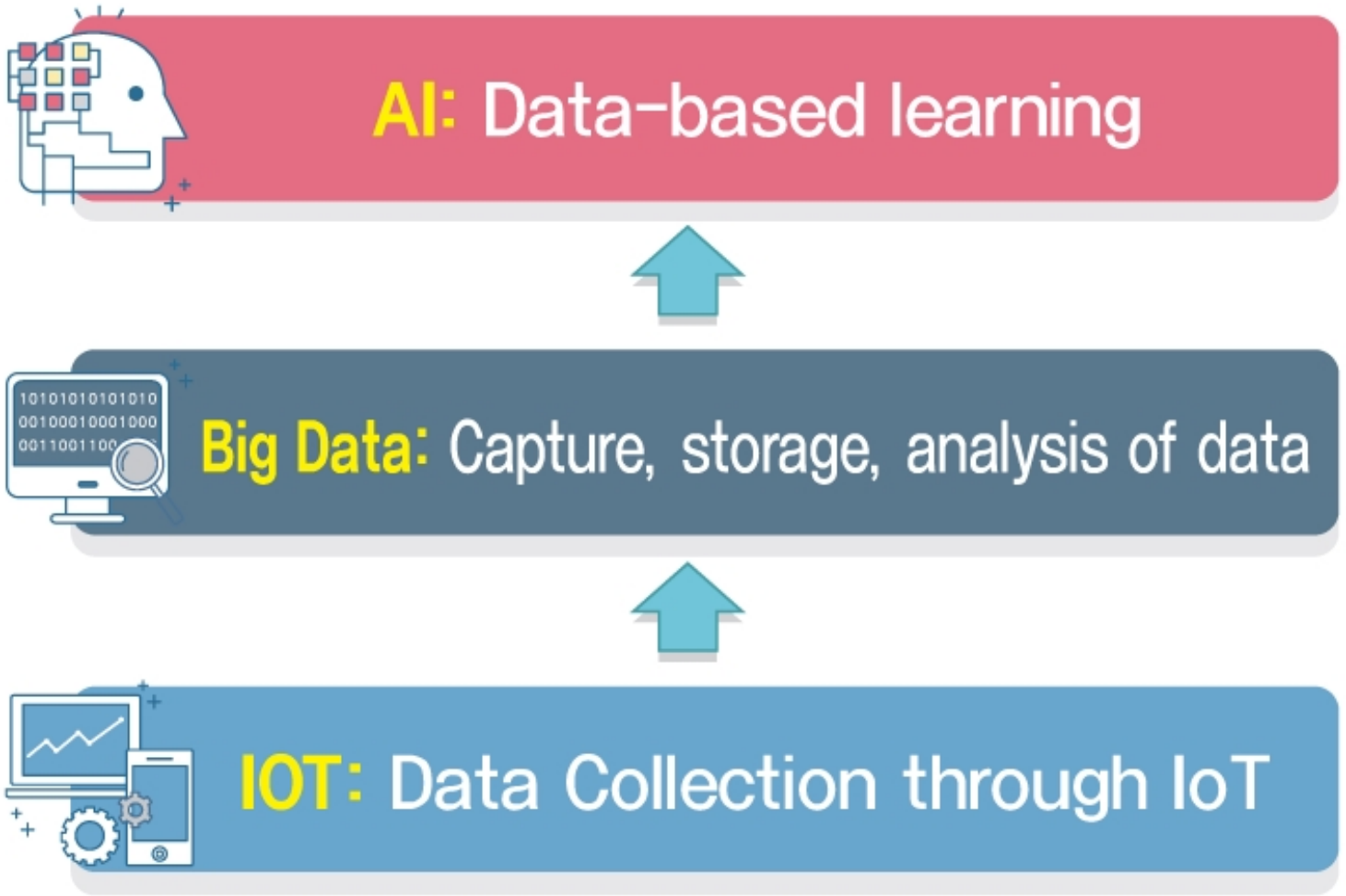
Evolution of Industry



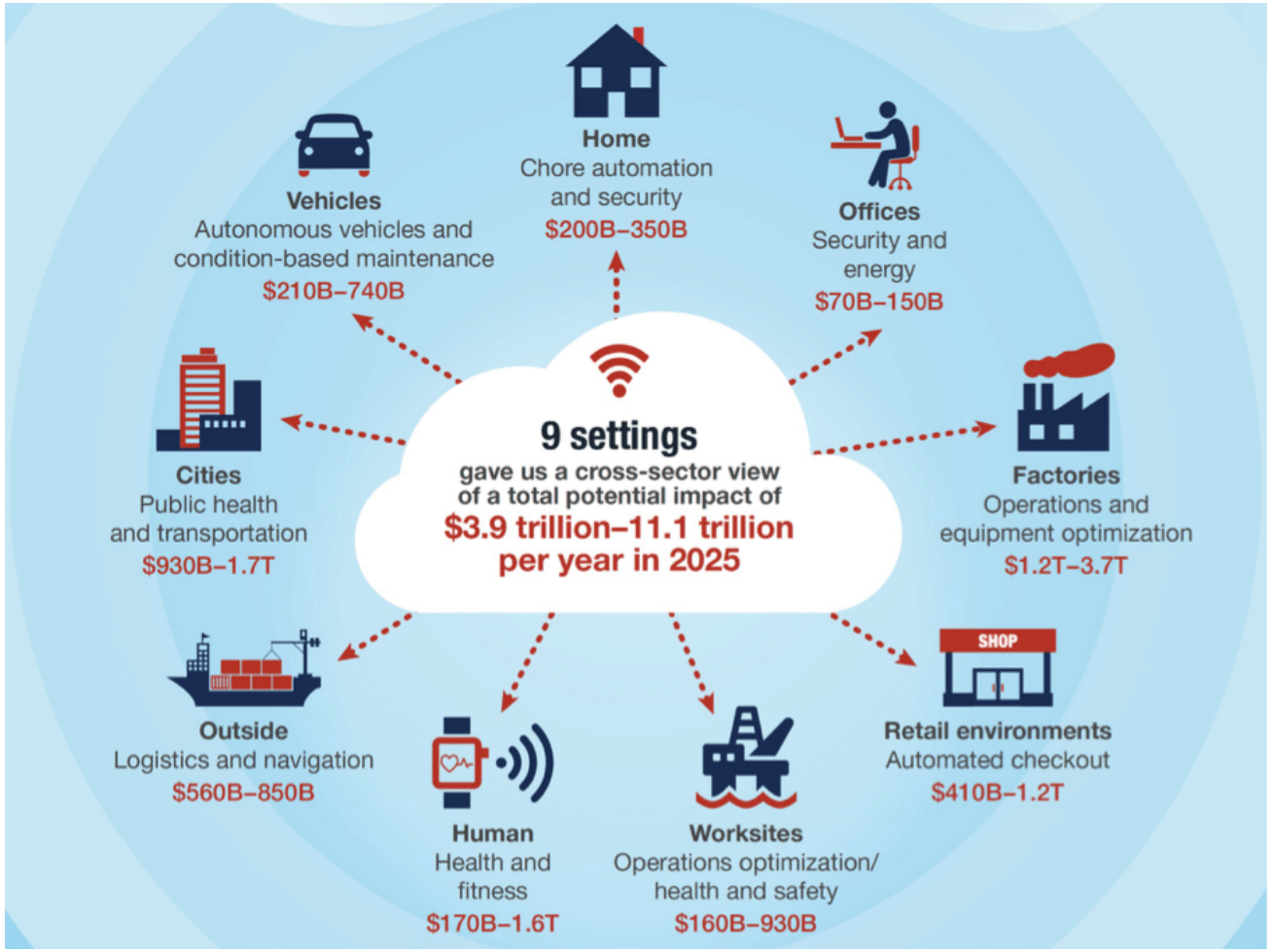
Components of Industry 4.0



IoT and Data Science

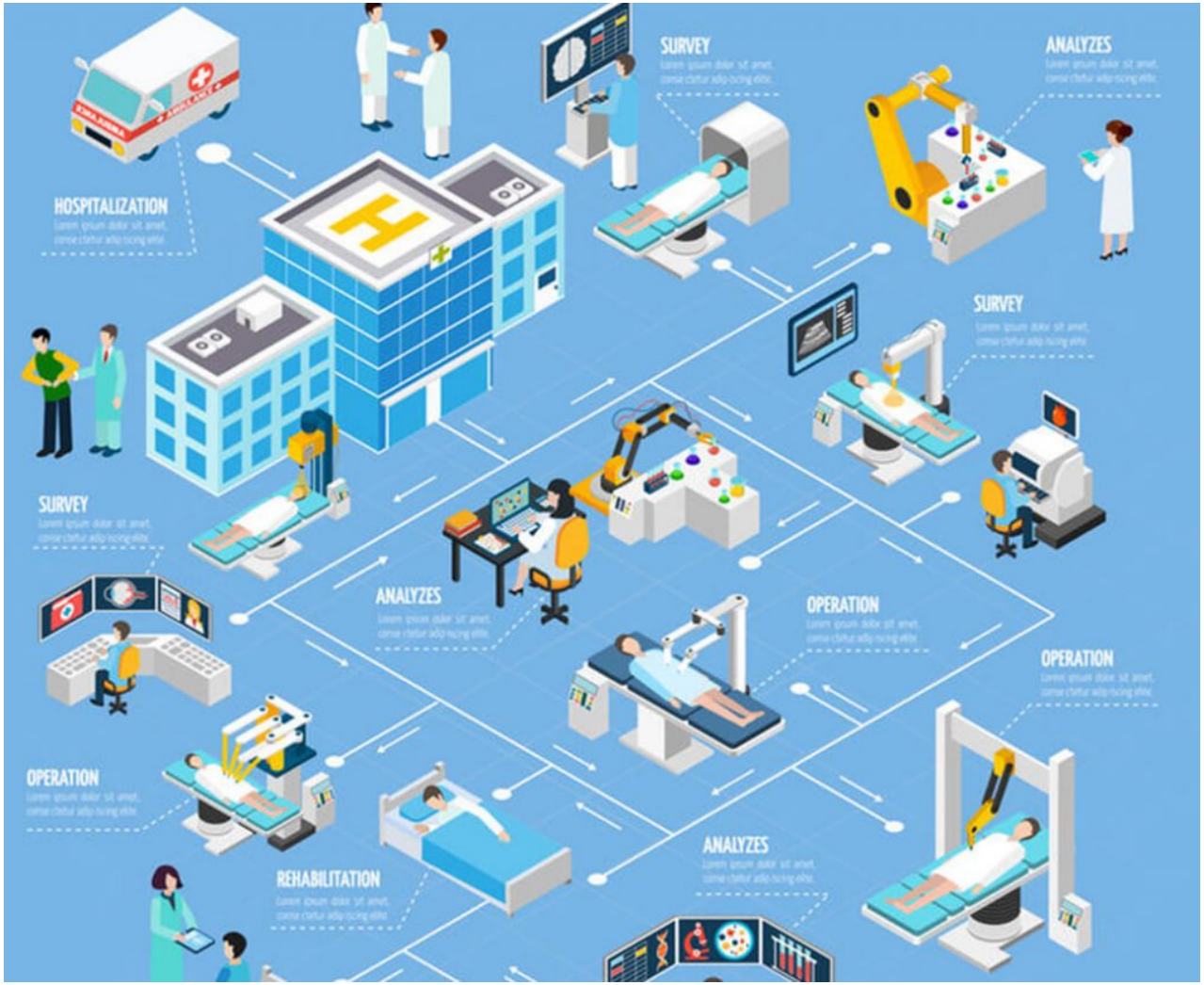


IoT 2025

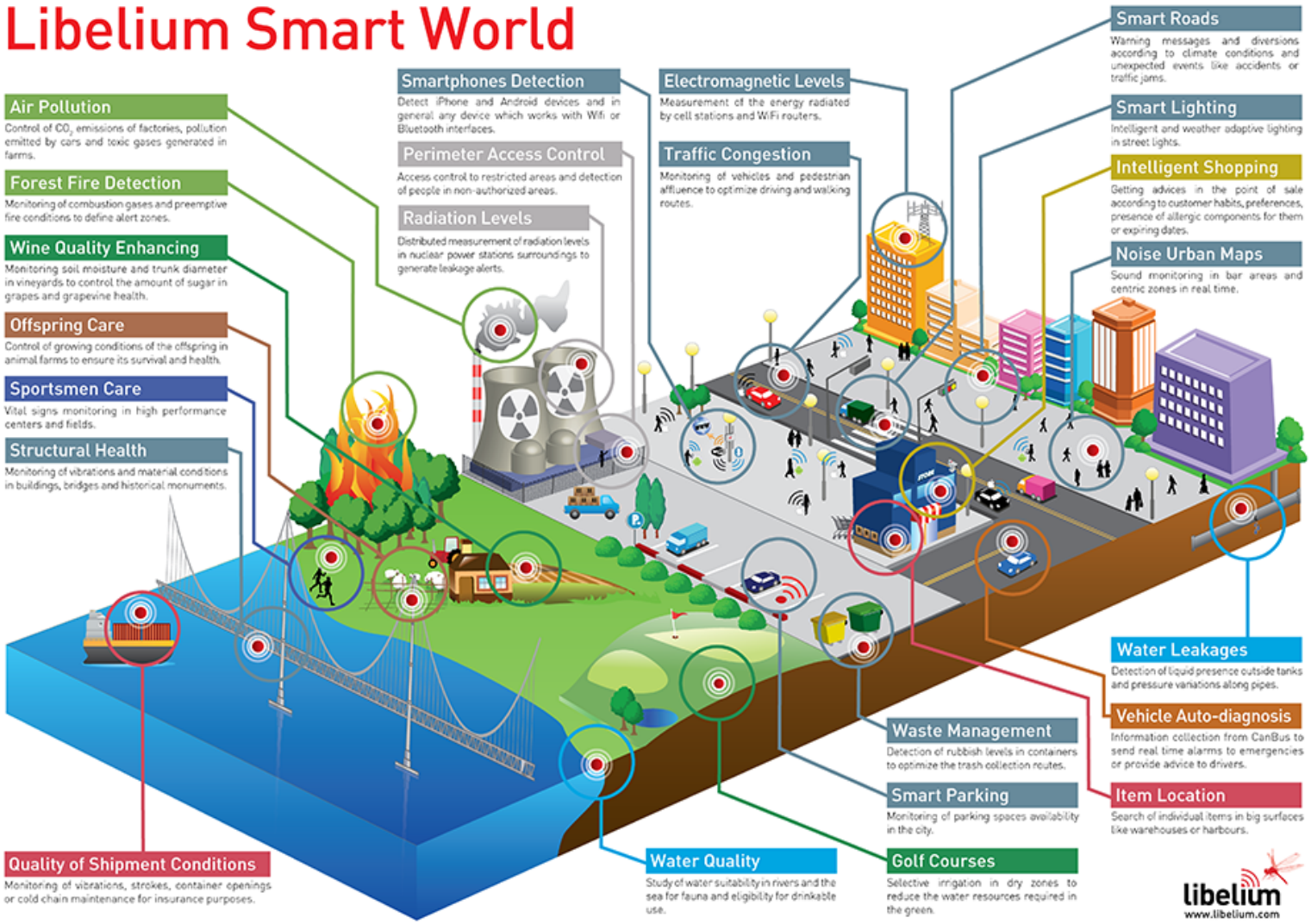

Home
Chore automation and security
$200B–350B
Vehicles
Autonomous vehicles and condition-based maintenance
$210B–740B
Offices
Security and energy
$70B–150B
9 settings
gave us a cross-sector view of a total potential impact of
$3.9 trillion–11.1 trillion per year in 2025
Cities
Public health and transportation
$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

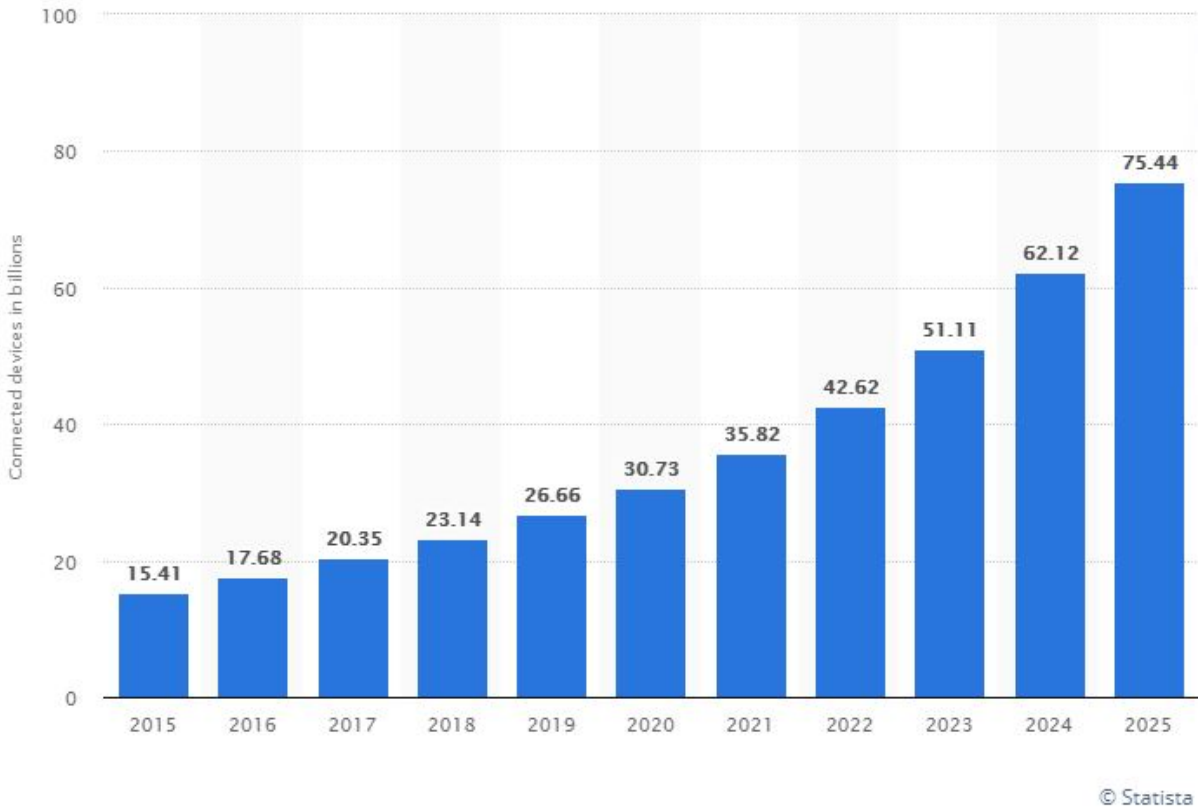


Smart World


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.
Waste Management
Detection of rubbish levels in containers to optimize the trash collection routes.
Smart Parking
Monitoring of parking spaces availability in the city.
Item Location
Search of individual items in big surfaces like warehouses or harbours.
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



How ubiquitous is the Internet of Things?

- There are approximately 31 billion IoT devices today.
- 127 new IoT devices are connected to the internet every SECOND.
- This morning, 1,828,800 IoT devices will be added to the internet.

Let's Begin Our Journey

Computer Languages

Mother Tongues

Tracing the roots of computer languages through the ages

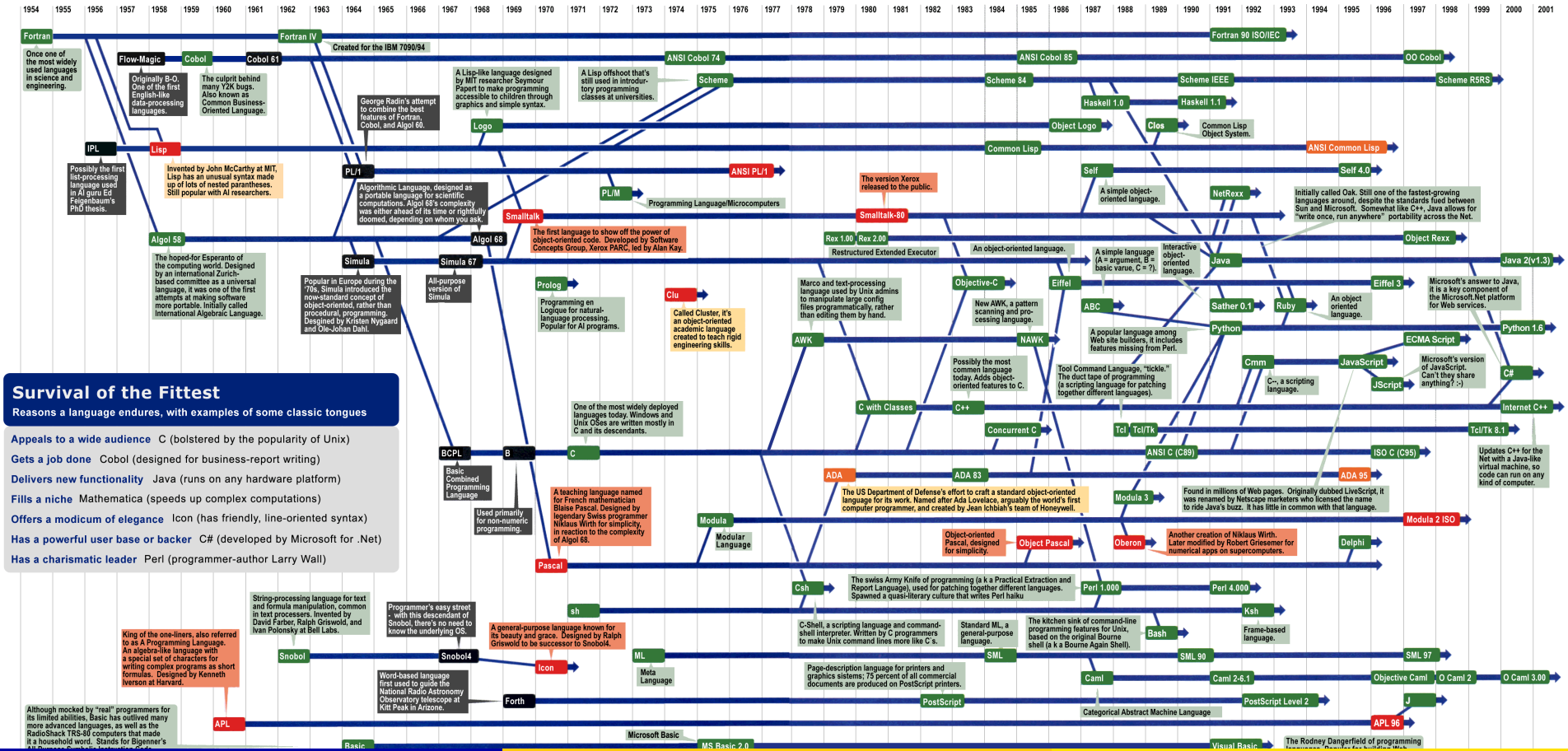
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

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**

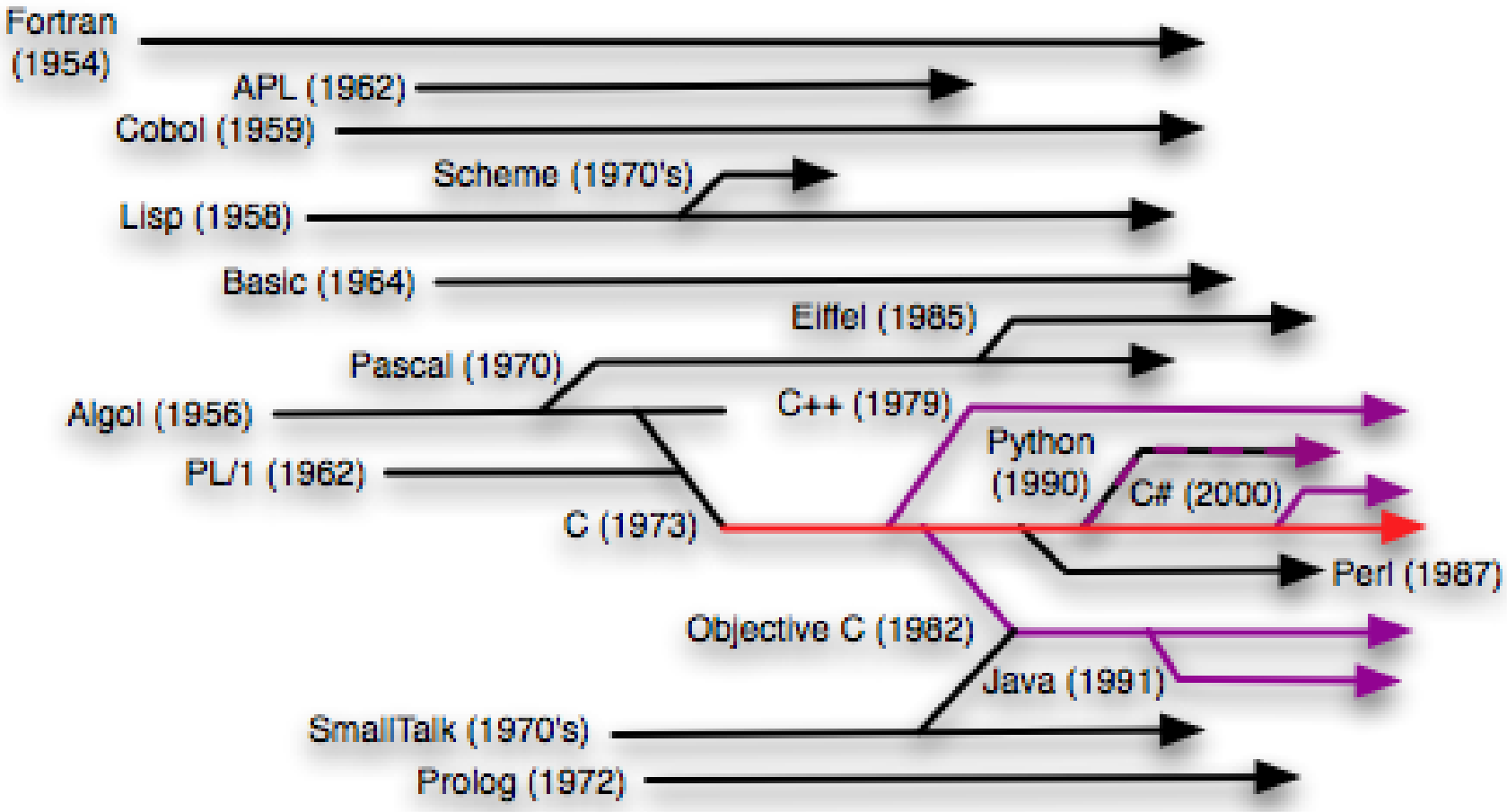


Survival of the Fittest

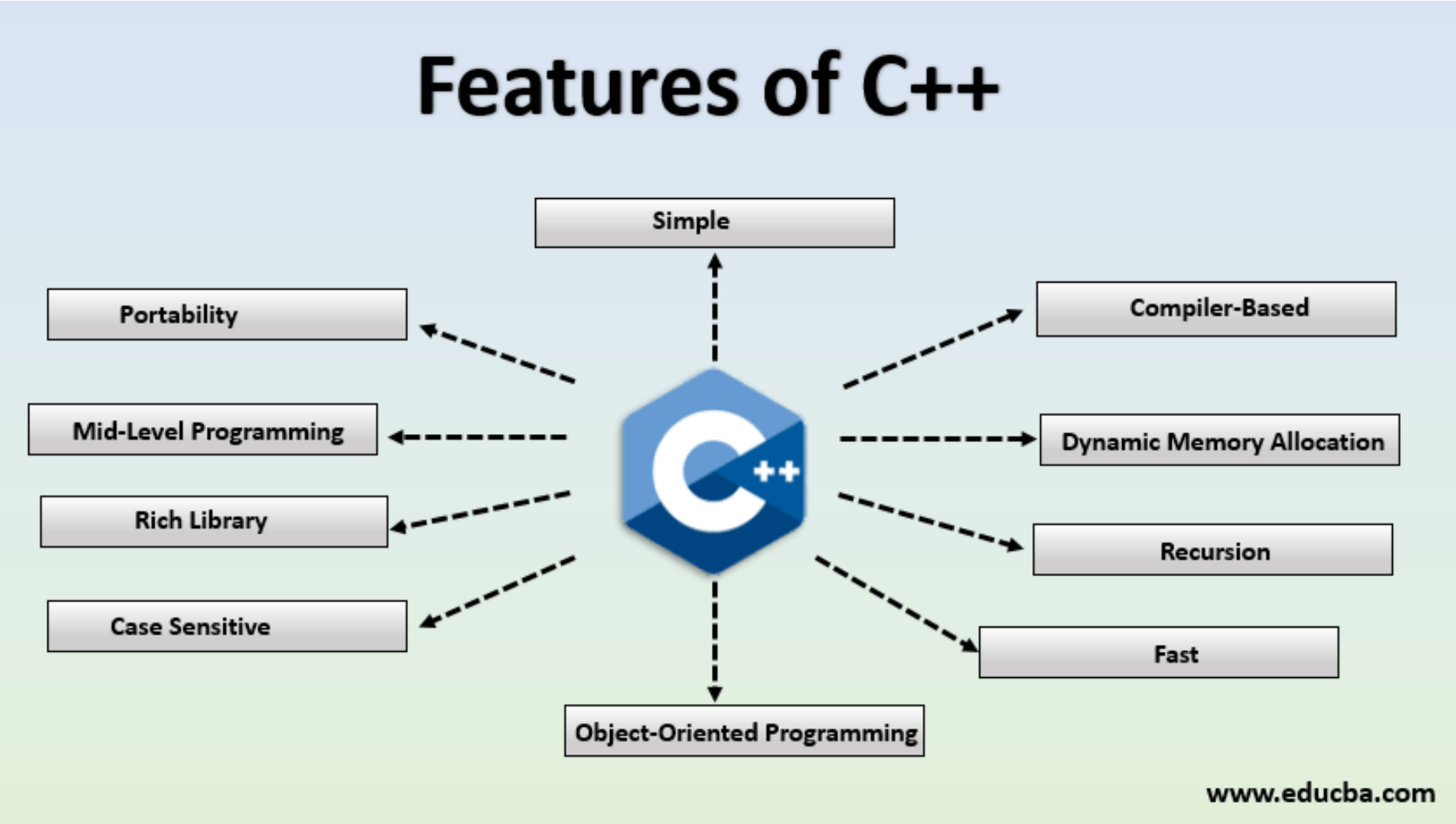
Reasons a language endures, with examples of some classic tongues

- **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)

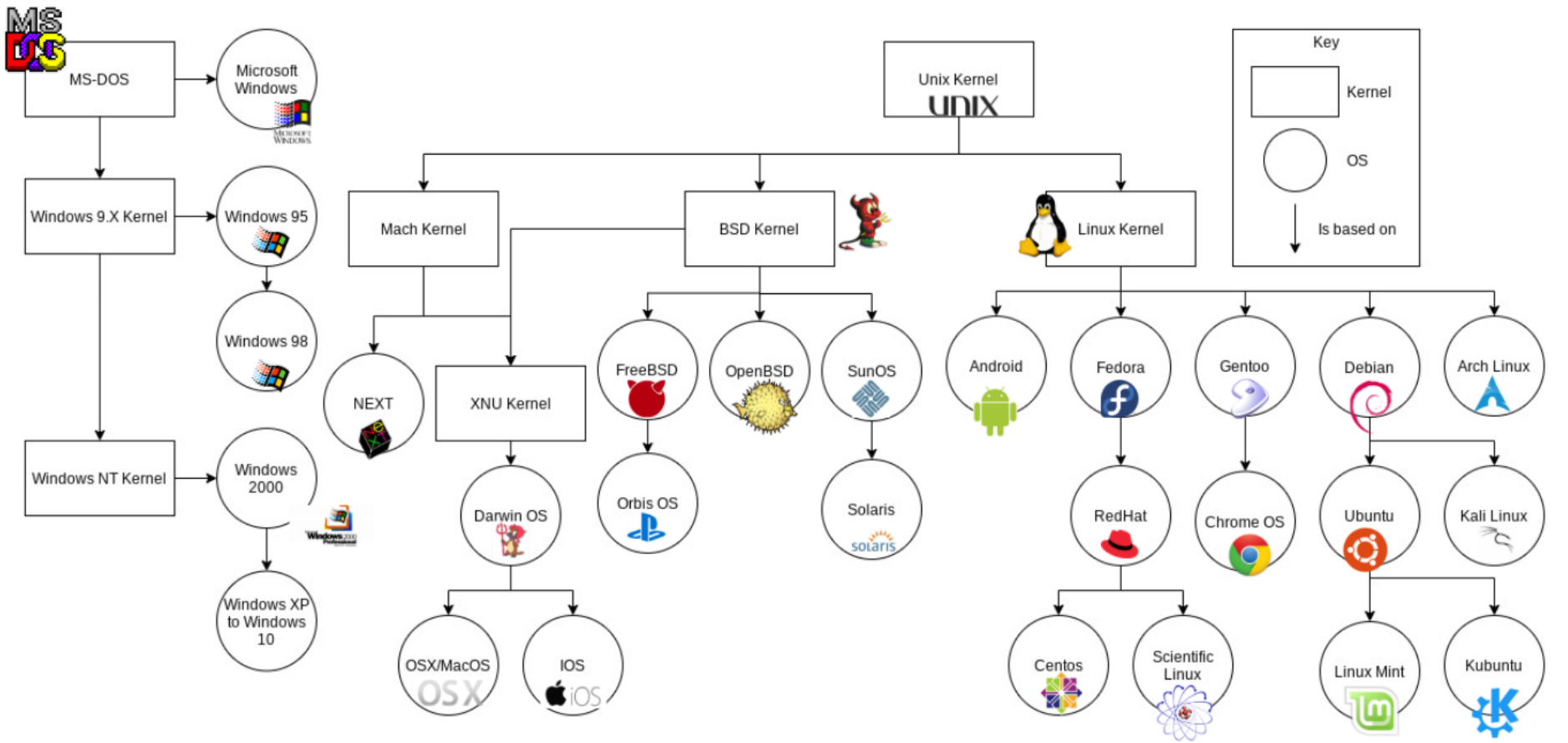
Computer Languages



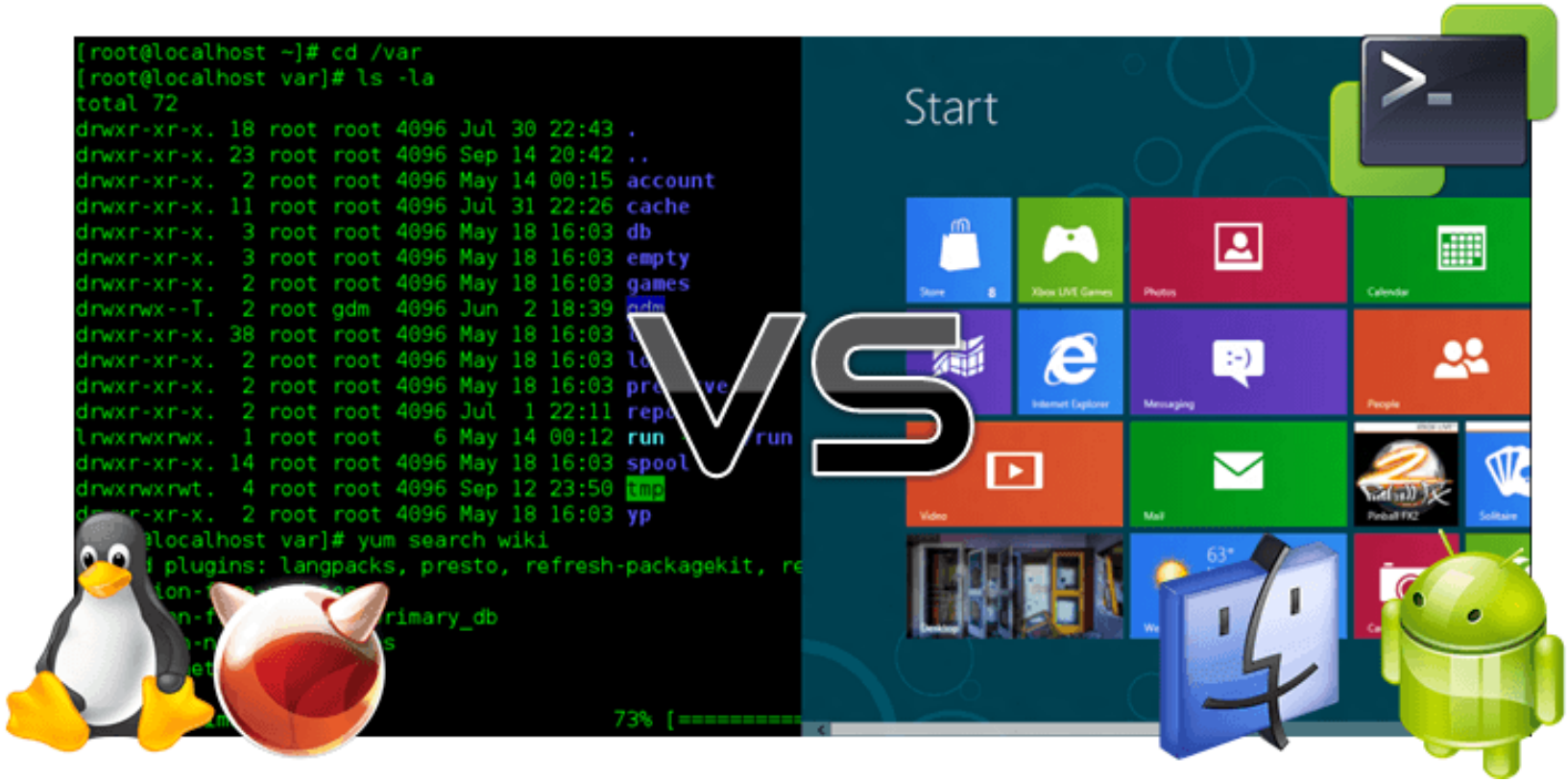
Why C++


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

Operating Systems



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

The Command Line Interface (CLI) will allow us to directly navigate the computers operating system. We will use:

- macOS or Linux: Terminal
- Windows: PowerShell

The following commands will work on all three systems, except where noted below. macOS and Linux are case-sensitive, Windows is not.

- 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. (not available in PowerShell).
- man: Used to show the manual of any command.

Command Line Interface - File and Directory Manipulation

- mkdir: Used to create a directory if it does not already exist. It accepts directory name as input parameter.
- rmdir: 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 a new name to the destination path. It accepts source file/directory and destination file/directory.
- mv: Used to move files or directories. This command is similar to the cp command but it deletes a copy of the file or directory from the source path.
- rm: Used to remove files or directories.
- touch: Used to create or update a file. (PowerShell New-Item).

Command Line Interface - Displaying the file contents

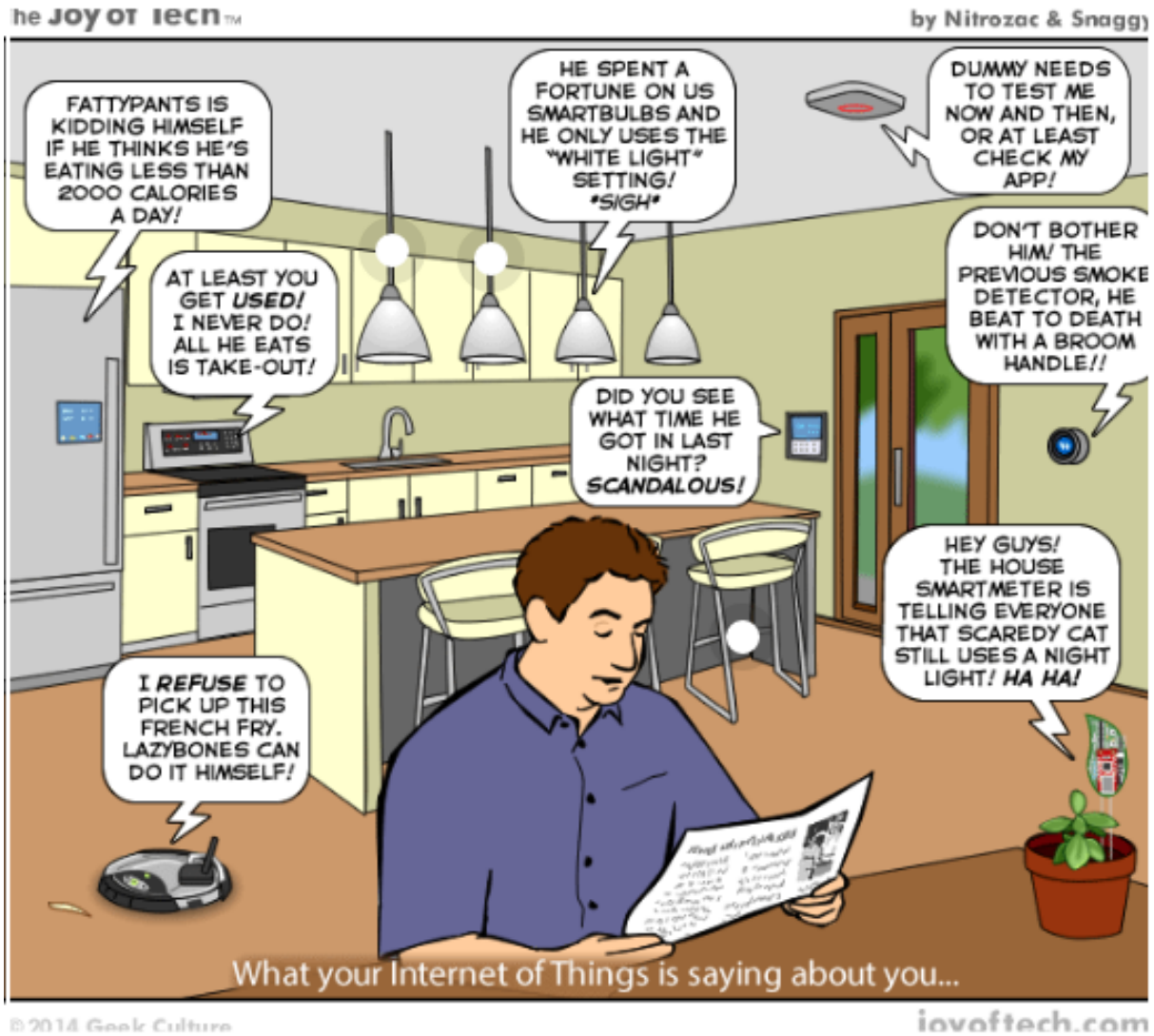
- cat: It is generally used to concatenate files. It gives the output on the standard output.
- more: It is a filter for paging through text one screenful at a time.

The below commands are not available in PowerShell:

- less: Used for viewing files instead of opening the file. Similar to the "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.

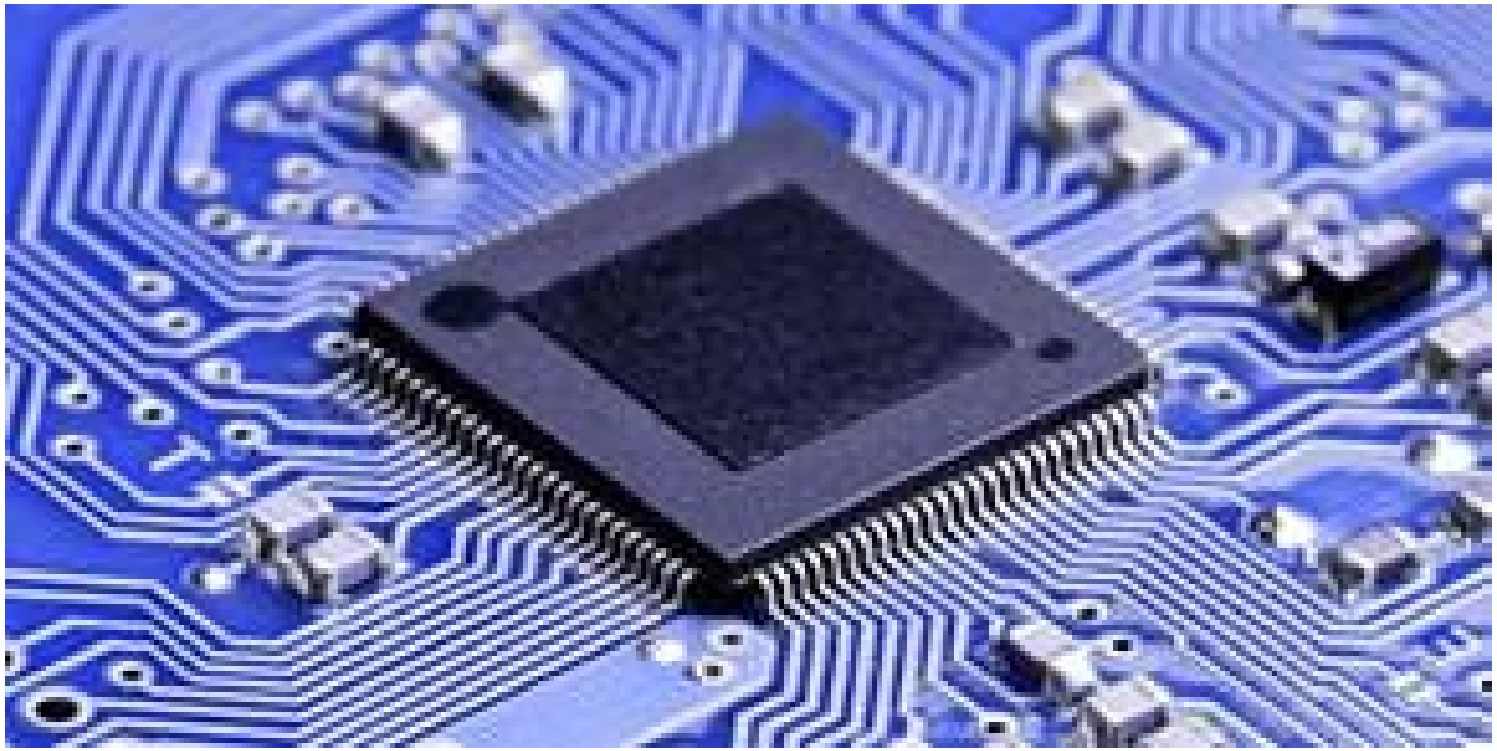
On all systems, commands can be "piped" together: ls | more < *file* >

IoT Fun

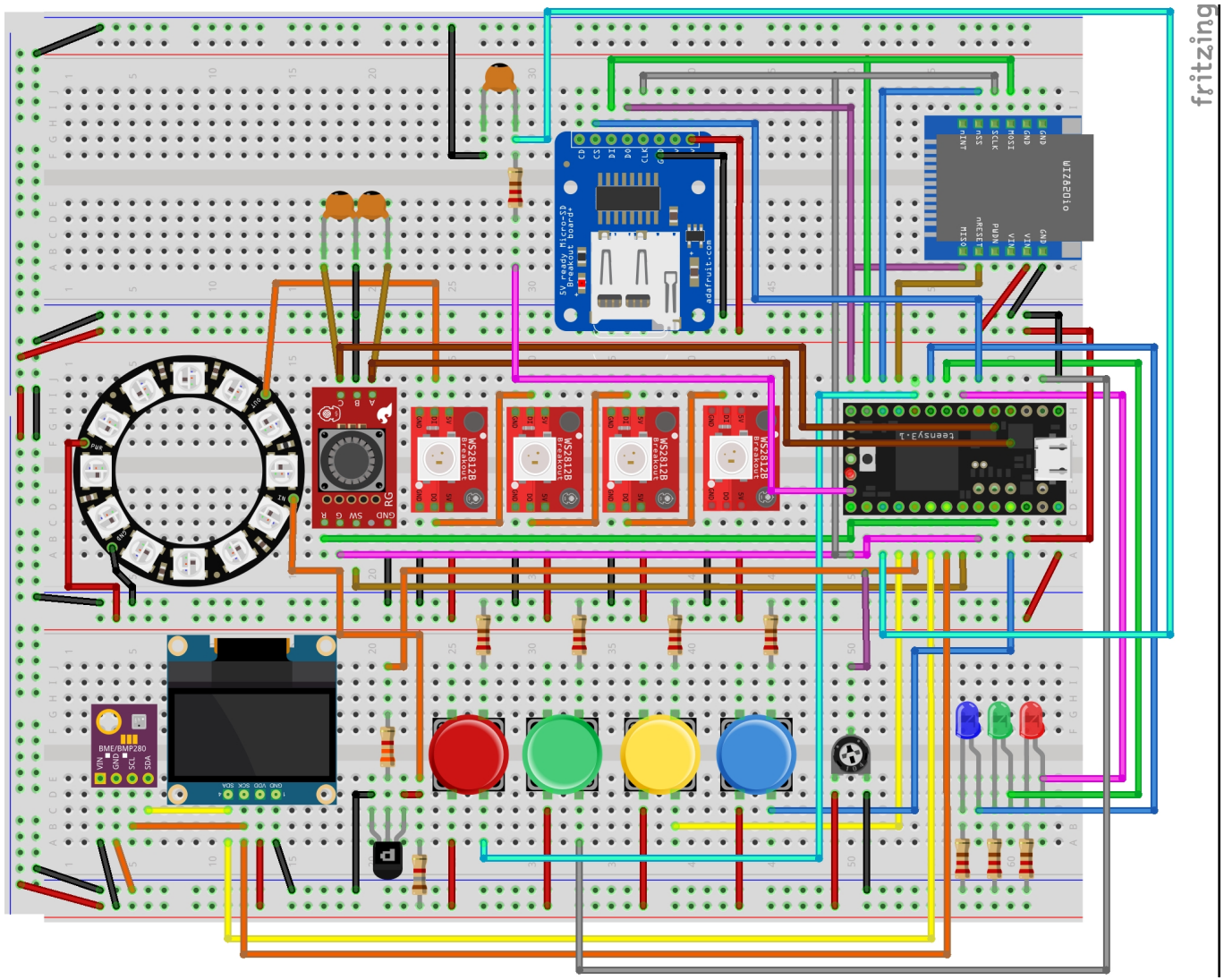

he Joy of Tech™
by Nitrozac & Snaggy
FATTYPANTS IS KIDDING HIMSELF IF HE THINKS HE'S EATING LESS THAN 2000 CALORIES A DAY!
HE SPENT A FORTUNE ON US SMARTBULBS AND HE ONLY USES THE "WHITE LIGHT" SETTING! *SIGH*
DUMMY NEEDS TO TEST ME NOW AND THEN, OR AT LEAST CHECK MY APP!
AT LEAST YOU GET USED! I NEVER DO! ALL HE EATS IS TAKE-OUT!
DON'T BOTHER HIM! THE PREVIOUS SMOKE DETECTOR, HE BEAT TO DEATH WITH A BROOM HANDLE!!
DID YOU SEE WHAT TIME HE GOT IN LAST NIGHT? SCANDALOUS!
HEY GUYS! THE HOUSE SMARTMETER IS TELLING EVERYONE THAT SCAREDY CAT STILL USES A NIGHT LIGHT! HA HA!
I REFUSE TO PICK UP THIS FRENCH FRY. LAZYBONES CAN DO IT HIMSELF!
What your Internet of Things is saying about you...
©2014 Geek Culture
joyoftech.com

Smart Room Controller

Our First Microcontroller

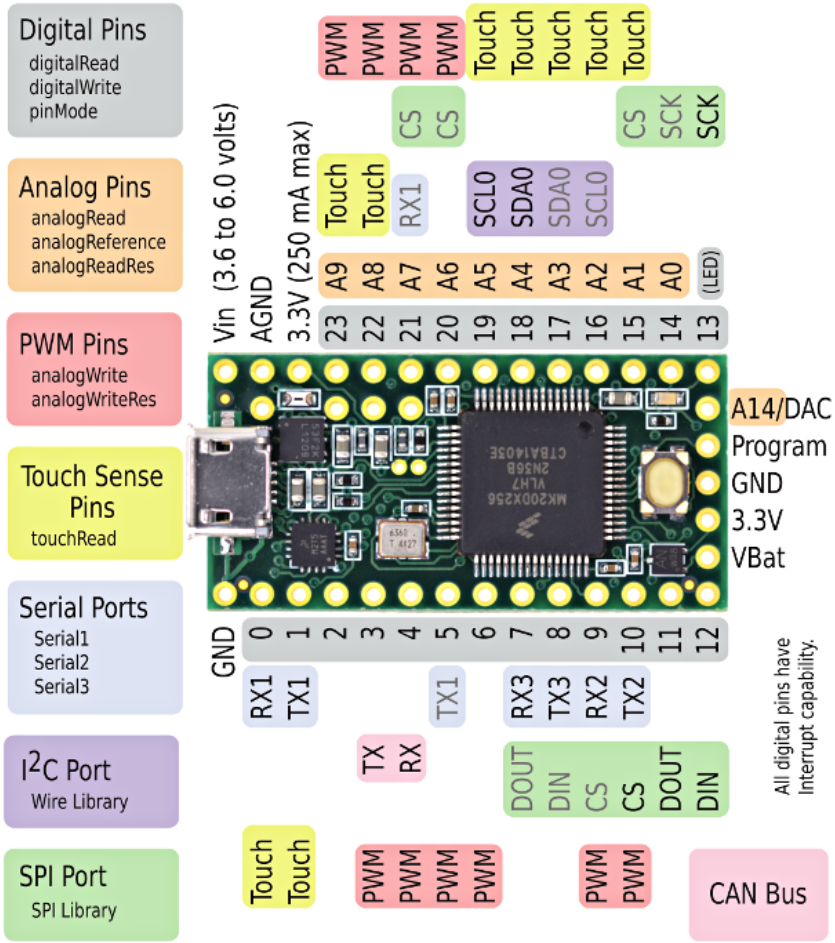


Smart Room Controller



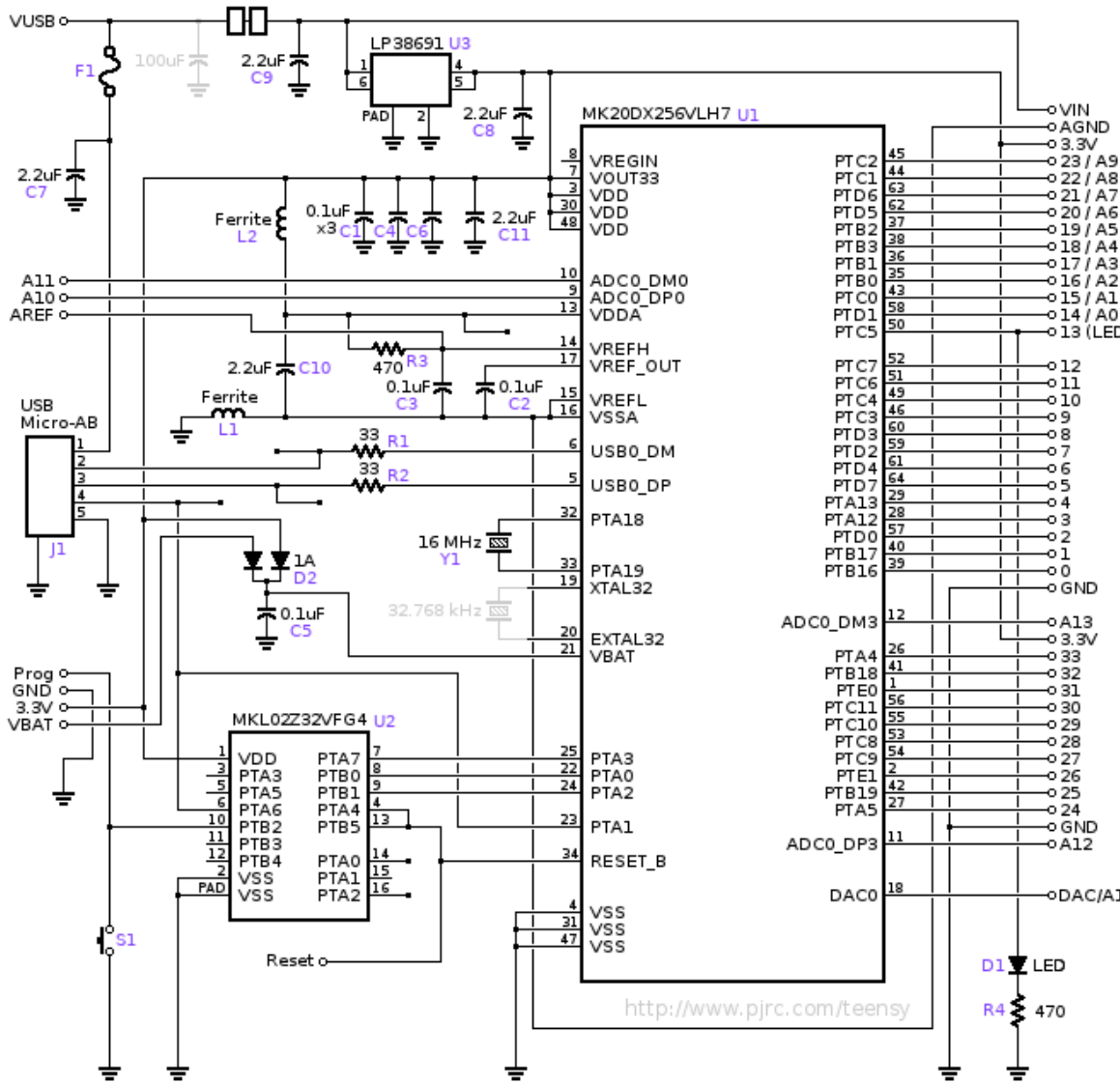
Teensy 3.2

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



Teensy 3.2 Schematic

- u1: Cortex-M4
- u2: Bootloader
- u3: Linear Regulator



Arduino IDE for Teensy

We are going to start off using the Arduino IDE[4]. The Arduino IDE is programmed essentailly using C++ code, but makes the compiling and loading onto the microcontroller simpler.

We begin by installing the Arduino IDE (Skip this step if on Mac):
`https://www.arduino.cc/en/main/software`

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

Finally, if OneDrive is enabled, create a folder in the local Documents called Ardunio and use File → Preferences → Sketchbook location to point to this new folder.

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

Other Software

1. Git
   - For Windows - `https://git-scm.com/download/win`
   - For Mac - `https://git-scm.com/download/mac`
   - For Linux - sudo apt-get install git-all
2. Fritzing
   - IoT Bootcamp Teams Site
   - Note: to extract faster, `https://www.7-zip.org/download.html`
3. Drawio
   - `https://app.diagrams.net/`
4. Adobe Illustrator
   - `https://www.adobe.com/creativecloud.html`
5. Formlab's Preform
   - `https://formlabs.com/software/`
6. Ultimaker's Cura
   - `https://ultimaker.com/software/ultimaker-cura`
7. Bookmark: `https://www.desmos.com/`

Solidworks

To install Solidworks (Windows only), go to
`http://www.SolidWorks.com/SEK`

- Enter your contact information.
- Check the radio button "Yes" under "I already have a Serial Number that starts with 9020".
- Select the version and click Request Download.
- On the next page, Accept the agreement and continue.
- On the final page, click the Download button to download the SolidWorks Installation Manager.
- Unzip the files to launch the Installation.
- Select the option for Individual/On this machine.
- Install using the following serial number provided by your instructor.

macOS and Linux users will use onShape:
`https://www.onshape.com/en/education/`

GitHub - Part 1

Git and GitHub: Your Version Control Friends

GitHub

What is a version control system?

Version Control Systems (VCS) record changes made to files so that you can

- compare and track changes over time
- revert single files to a previous state
- revert an entire project to an earlier version

Git is a VCS, or Version Control System.
It is similar to Backup on Windows or Time Machine on the Mac.

Installing Git

If you have not already done so, install Git on your computer.

- For Windows - `https://git-scm.com/download/win`
- For Mac - `https://git-scm.com/download/mac`
- For Linux - sudo apt-get install git-all

Note: For Mac, if you have XCode installed, then you will already have Git. To verify, you can type git –version in terminal.

Why use a version control system?

Before Version Control Systems.

- Save multiple versions of the file and try to remember which is which.
- Run a text comparison tool to see what changed between versions.
- Work with system engineer to get your changes merged with production version.

With Git.

- Commit as you code.
- Versioning and timestamping automatically happen.
- Easy to see differences between versions using git diff or visually if using GitHub.
- Easy to merge changes to production or rollback versions.
- Links with ticketing system for better task management and customer support.

Git vs. GitHub

Git is the version control system. This is on your **local system**.

GitHub is a GUI (Graphical User Interface) **cloud-based** product that allows you to save your work remotely and facilitates collaboration.

Commit vs. Push

To save files to your **LOCAL** repository. use the *commit* command. The file is given a timestamp and a unique commit number that is the file version.

→ git commit

To save a file to your **REMOTE** GitHub repository, use the *push* command.

→ git push

Remember to Push your files in order to save from data loss and to ensure your work is available for collaboration.

When should you commit and push your work?

When to commit? **OFTEN!**

- Any time you finish a task where you want to save or retain a version.

When to push? **OFTEN!**

- Any time you have finished a task, milestone, or significant project.
- At the end of each work session
  - Before you take a break
  - Before a meeting
  - Before lunch
  - Before you go offline for the day

Getting a GitHub account

If you do not already have a GitHub account, you will want to create one.

1. Go to `https://github.com`
2. Click Sign Up.
3. Type a unique username and password for your account.
   - *Note: you should consider a user name that is professional if you plan to share your GitHub account with prospective employers as part of your work portfolio.*
4. Complete the sign up process and Create Account.

GitHub Authentication: Using HTTPS and PAT

Effective August 2021, account passwords will no longer be allowed for command line GitHub access. Instead, you must create a Personal Access Token (PAT) to use in place of a less secure password.

To create a PAT:

1. Login to your GitHub account.
2. Click your account icon.
3. Click the Settings menu option.
4. Click Developer Settings.
5. Click Personal access tokens.
6. Generate a new token for GitHub Command Line Access.
7. Check the repo option.
8. Click Generate Token.

Now when you login to GitHub on the command line, use your PAT rather than your password to access your account.

GitHub Authentication: Switching from password to PAT

If you have been connecting to GitHub from the command line using your password, you will need to switch to using your Personal Access Token (PAT).

On Windows 10:

1. Open Credential Manager.
2. Click Windows Credentials.
3. Delete the git:https://github.com entry.

On Mac OSx:

1. Open Keychain Access application.
2. Search for github.
3. Delete github entries as desired.

The next time you attempt a git command on the command line, you will be prompted to enter your username and password or PAT. If you have trouble authenticating with PAT, check that your version is at least 2.30.

Using Git: Command line or GUI?

Why should I use the command line when I could use a GUI instead?

- It's generally better supported than GUI-based tools.
- It's independent from your IDE.
- It helps you come to a better understanding of the principles of version control.
- It's much more commonly used in the industry and thus is more likely to get you a job.
- It's much more likely to be useful if you find yourself in a sticky merge/rebase situation that you can't seem to fix.
- It's faster to type the commands than to go through the GUI. These time savings add up.

The command line

Windows

- PowerShell - This is what we are using in class.
- GitBash - Linux commands and editors available.

Mac or Linux

- Terminal - This is what our Mac users are using in class.

For basic commands, review IoT slides on *Command Line Interface*.

GitHub: Cloning and Pulling Code

To get an existing GitHub repository, you will clone it to your local system.

git clone <URL of repository>

To get updates from a GitHub repository after you have already cloned it to your local system, you will pull the code.

git pull

Note: make sure you are in the repository folder before doing a git pull.

GitHub: Getting Class Slides

You will need a copy of the class slides repository. To get class slides:

git clone `https://github.com/ddc-iot/class_slides`

Note: Every morning remember to pull a copy of the class slides so that you have the latest copy.

GitHub: Getting Assignments

ddc-iot-classroom-2

Accept the assignment —
L01_HelloWorld

Once you accept this assignment, you will be granted access to the `l01-helloworld-brashap` repository in the ddc-iot organization on GitHub.

Accept this assignment

You're ready to go!

You accepted the assignment, **L01_HelloWorld**.

Your assignment repository has been created:

https://github.com/ddc-iot/l01-helloworld-brashap

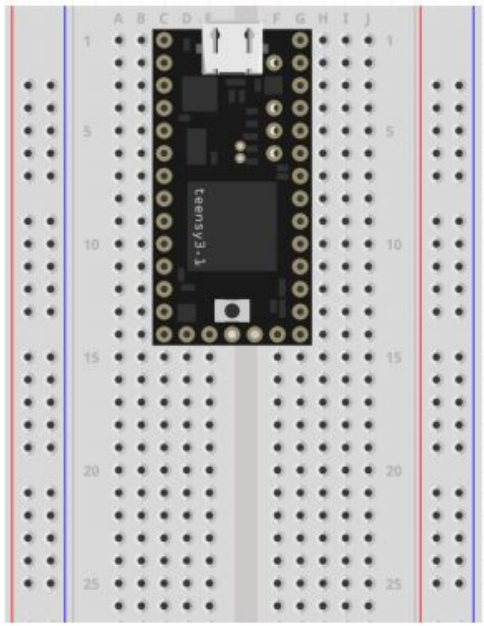
```
1  brian:~$ cd Documents/
2  brian:Documents$ mkdir IoT
3  brian:Documents$ cd IoT
4  brian:IoT$ git clone https://github.com/ddc-iot/L01_helloWorld-brashap
5  Cloning into 'L01_helloWorld'...
6  Username for 'https://github.com': brashap
7  Password for 'https://brashap@github.com':
8    remote: Enumerating objects: 4, done.
9    remote: Counting objects: 100% (4/4), done.
10   remote: Compressing objects: 100% (3/3), done.
11   remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
12   Unpacking objects: 100% (4/4), 321 bytes | 53.00 KiB/s, done.
```

GitHub: Cheatsheet. Memorize this!

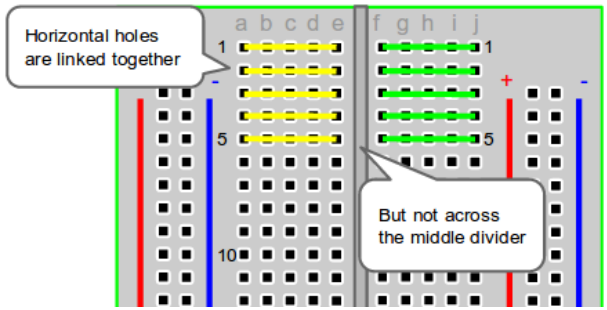
```
// In PowerShell go to ./Documents/IoT
// Get a repository that already exists and pull it into your local machine
  git clone <URL of repository>

// Send your changes up to the repository
  git add . //adds all changed files
  git commit -m "some comment"
  git push  //send your changes to the cloud

// The first time you use git, you may get asked to enter your GIT username
git config --global user.email "you@example.com"

// From the repository directory, get updates
  git pull
```

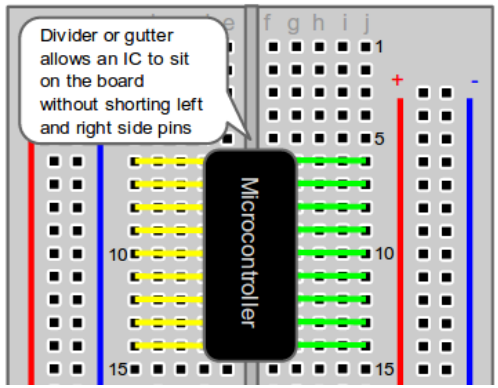
L01_HelloWorld

Teensy on Breadboard

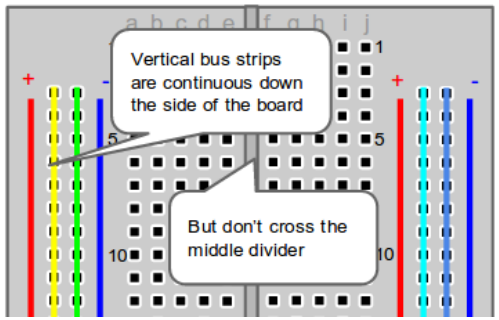


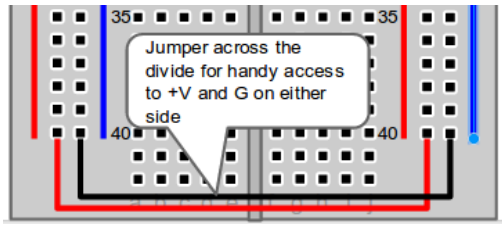
Horizontal Rows





Vertical Columns





Basic Structure of Arduino Sketch

```
// the "header" is used for GLOBALS

void setup() {
  // code in setup() runs once
  // it is used to initialize objects,
  // begin processes, and set variables
  pinMode(13, OUTPUT);  //set Pin 13 as an Output
}

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

Class Assignments

1. Lab Notebook - flow chart
2. Lab Notebook - 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
 */

// Single Line Comments
```

Hello World in some of the 603+ Coding Languages

Fortran

```
C     Hello World in Fortran

      PROGRAM HELLO
      WRITE (*,100)
      STOP
  100 FORMAT (' Hello World! ' /)
      END
```

C (K&R)

```
/* Hello World in C, K&R-style */

main()
{
  puts("Hello World!");
  return 0;
}
```

Python 2

```
# Hello world in Python 2

print "Hello World"
```

Assembler (Intel)

```
; Hello World for Intel Assembler (MSDOS)

mov ax,cs
mov ds,ax
mov ah,9
mov dx, offset Hello
int 21h
xor ax,ax
int 21h

Hello:
  db "Hello World!",13,10,"$"
```

Powershell

```
# Hello World in Microsoft Powershell

'Hello World!'
```

LabVIEW

LaTeX

```
% Hello World! in LaTeX
\documentclass{article}
\begin{document}
Hello World!
\end{document}
```

Unix Shell

```
# Hello world for the Unix shells (sh, ksh, csh, zsh, bash, fish, xonsh, ...)

echo Hello World
```

Lisp-Emacs

```
;;; Hello World in Emacs Lisp.

(defun hello-world()
  "Display the string hello world."
  (interactive)
  (message "hello world"))
```

BASIC

```
10 REM Hello World in BASIC
20 PRINT "Hello World!"
```

C++

```
// Hello World in C++ (pre-ISO)

#include <iostream.h>

main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

Perl

```
# Hello world in perl

print "Hello World!\n";
```

Python 3

```
# Hello world in Python 3

print("Hello World")
```

Pascal

```
{Hello world in Pascal}

program HelloWorld(output);
begin
  WriteLn('Hello World!');
end.
```

MATLAB

```
% Hello World in MATLAB.

disp('Hello World');
```

HTML

```
<HTML>
<!-- Hello World in HTML -->
<HEAD>
<TITLE>Hello World!</TITLE>
</HEAD>
<BODY>
Hello World!
</BODY>
</HTML>
```

Postscript

```
% Hello World in Postscript
%!PS
/Palatino-Roman findfont
100 scalefont
setfont
100 100 moveto
(Hello World!) show
showpage
```

Assignment L01_01_HelloWorld

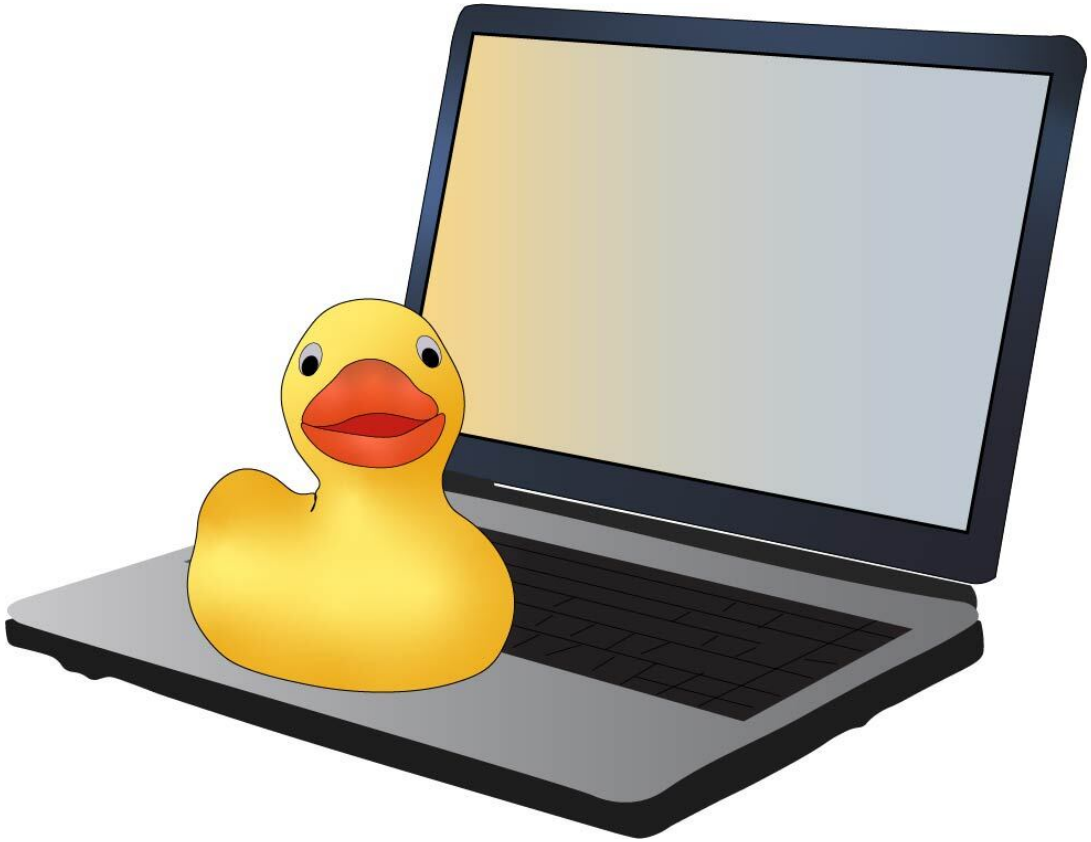
We will write our first program together as a class, using:

- pinMode(pin,mode)
- digitalWrite(pin,state)
- delay(delay_time)

How fast can you make it blink and still see it blinking?

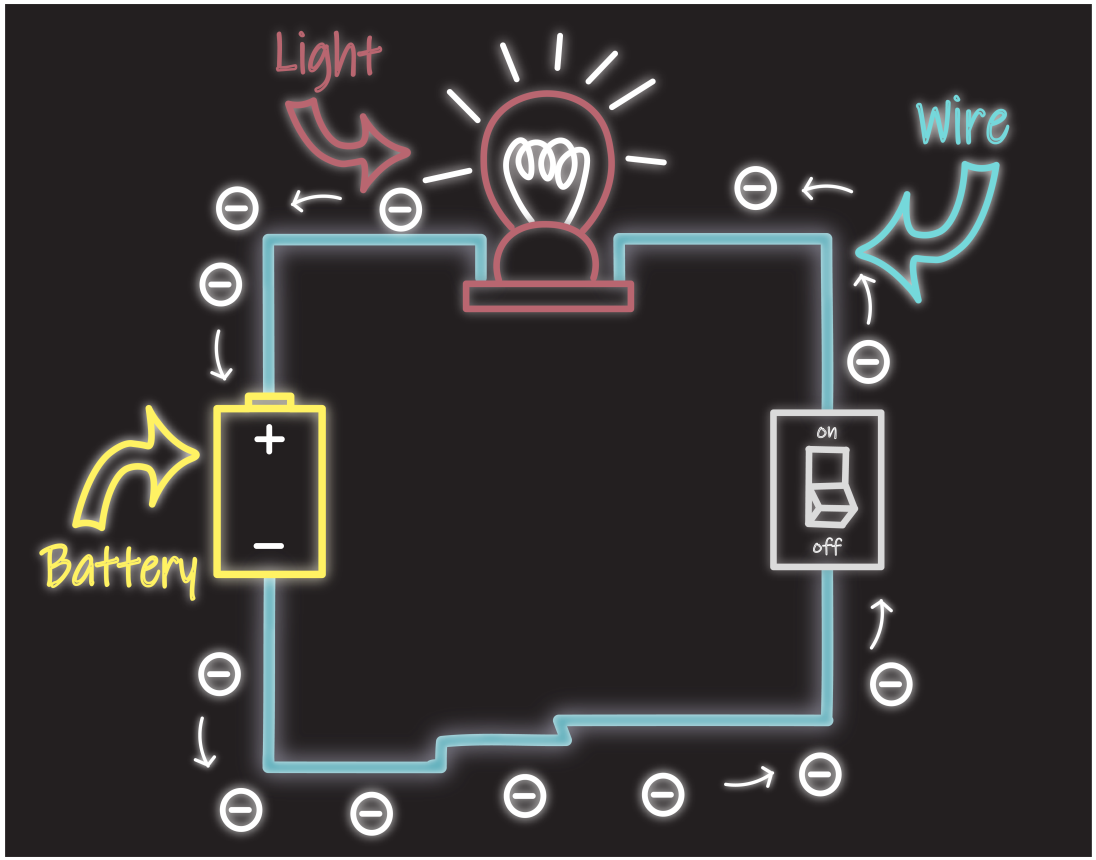
L02_HelloLED

Rubber Ducking - An Odd but Brilliant Tool

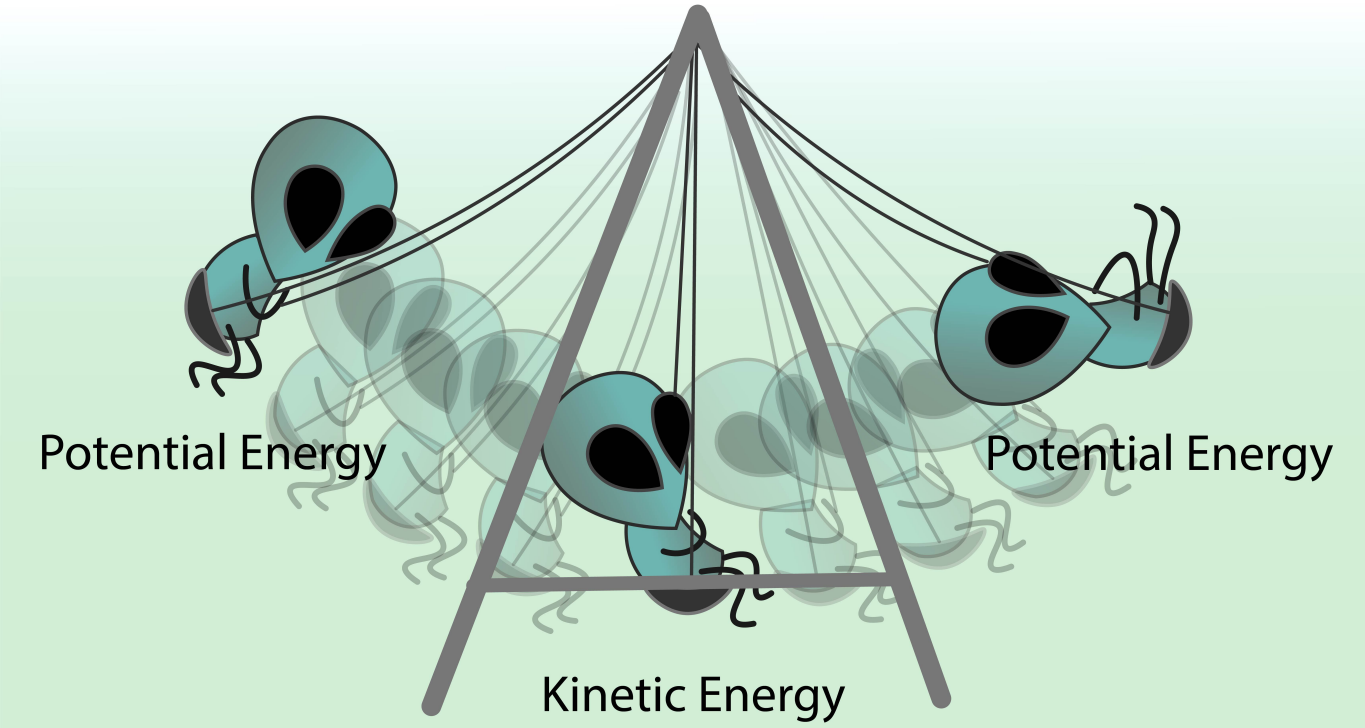


Rubber ducking is simply a method of debugging code. Programmers carry around a rubber duck with them, When they get stuck, they explain their code line-by-line to the rubber duck.

Introduction to Electrical Circuits



Energy

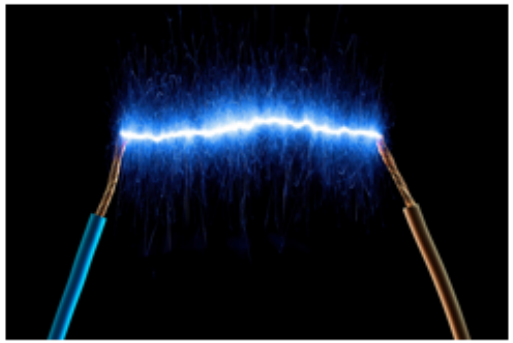


- Kinetic Energy - energy of motion
- Potential Energy - energy stored in an object

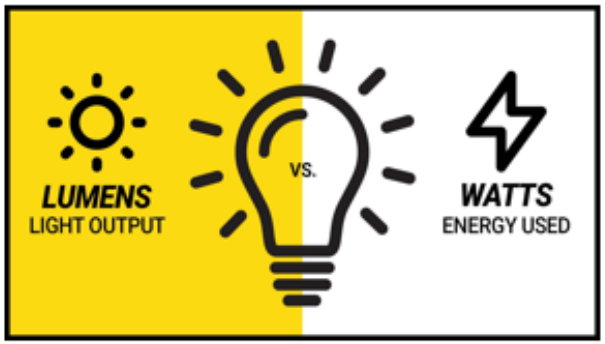
Electrical Circuit Terms

Voltage is **electric potential energy per unit charge** (V = J/C)

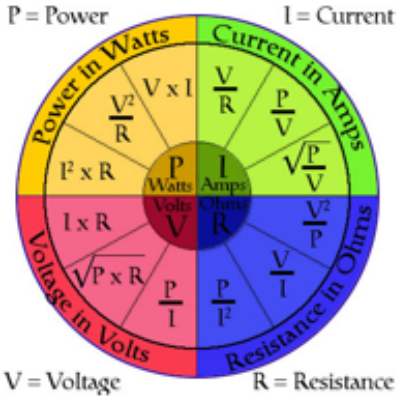
The Sub-atomic Particles			
Relative size	Name	Mass (Kg)	Charge (C)
	Proton	1.67×10^{-27}	$+1.602 \times 10^{-19}$
	Neutron	1.67×10^{-27}	0
* size exaggerated	Electron	9.11×10^{-31}	-1.602×10^{-19}



Electric current is the **rate of charge flow** (A = C/s)



Power is the rate of doing work or **the rate of using energy**. (W = J/S)

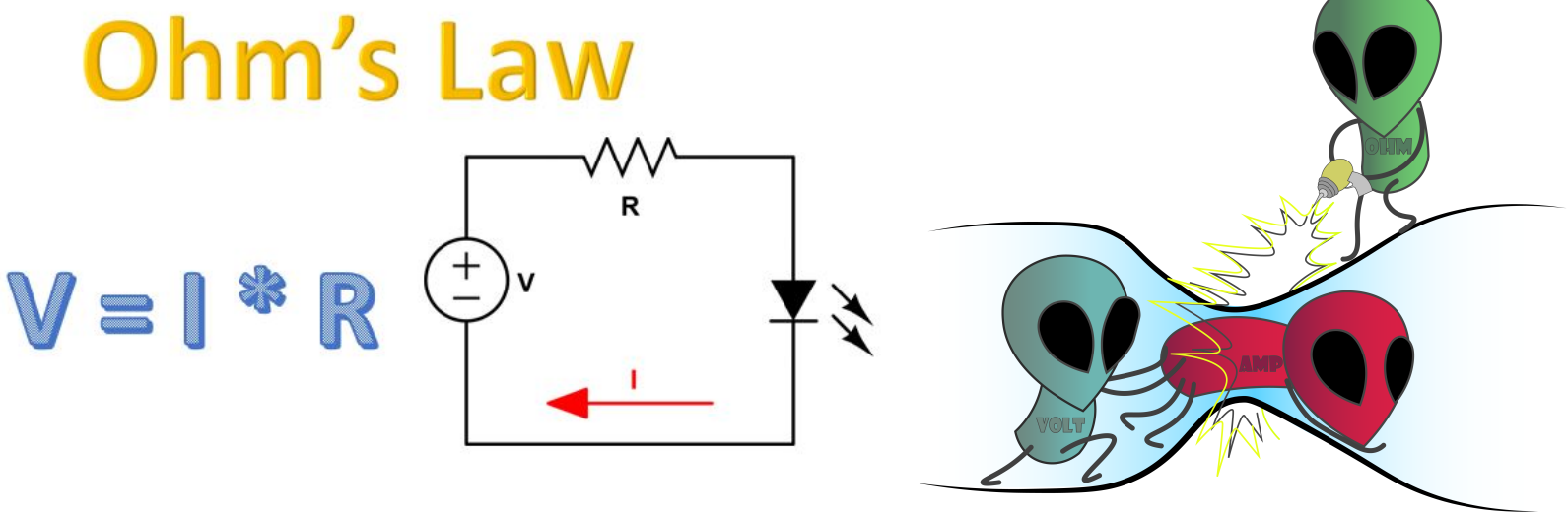


Power = Voltage x Current

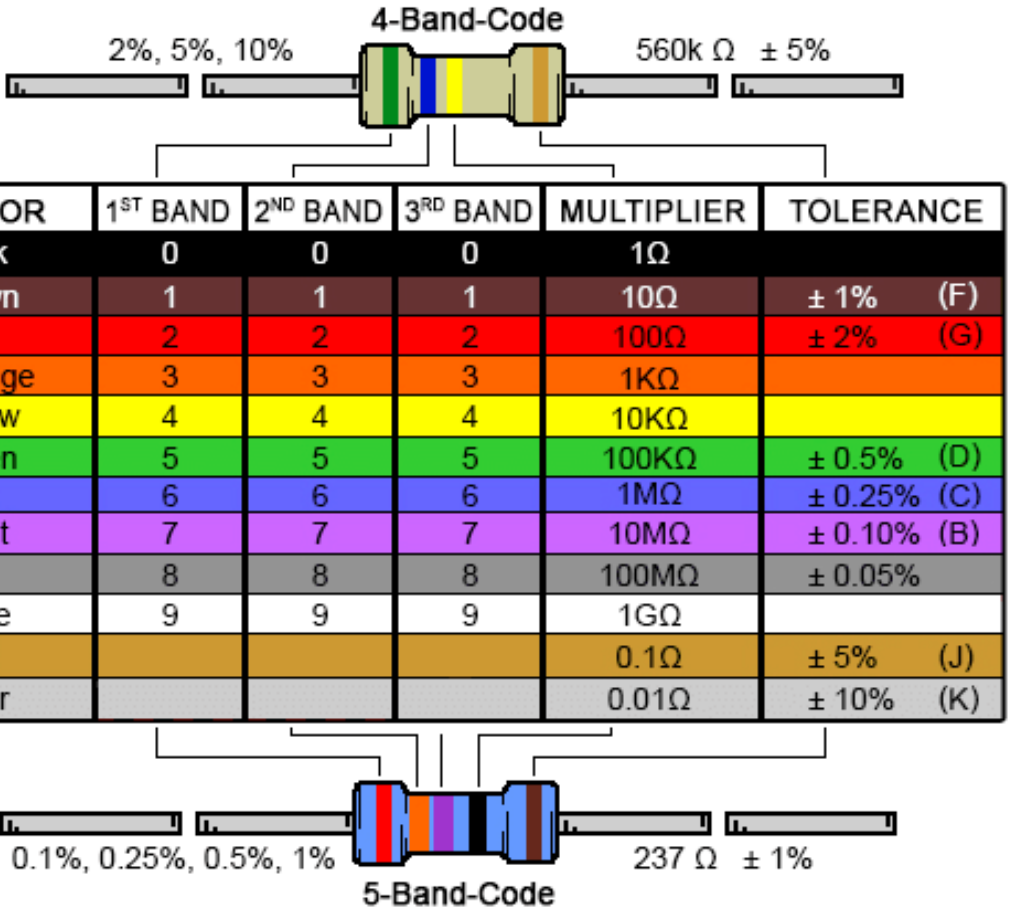
Energy is the **amount of power** produced or consumed **over a given time**. J = W x s

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:

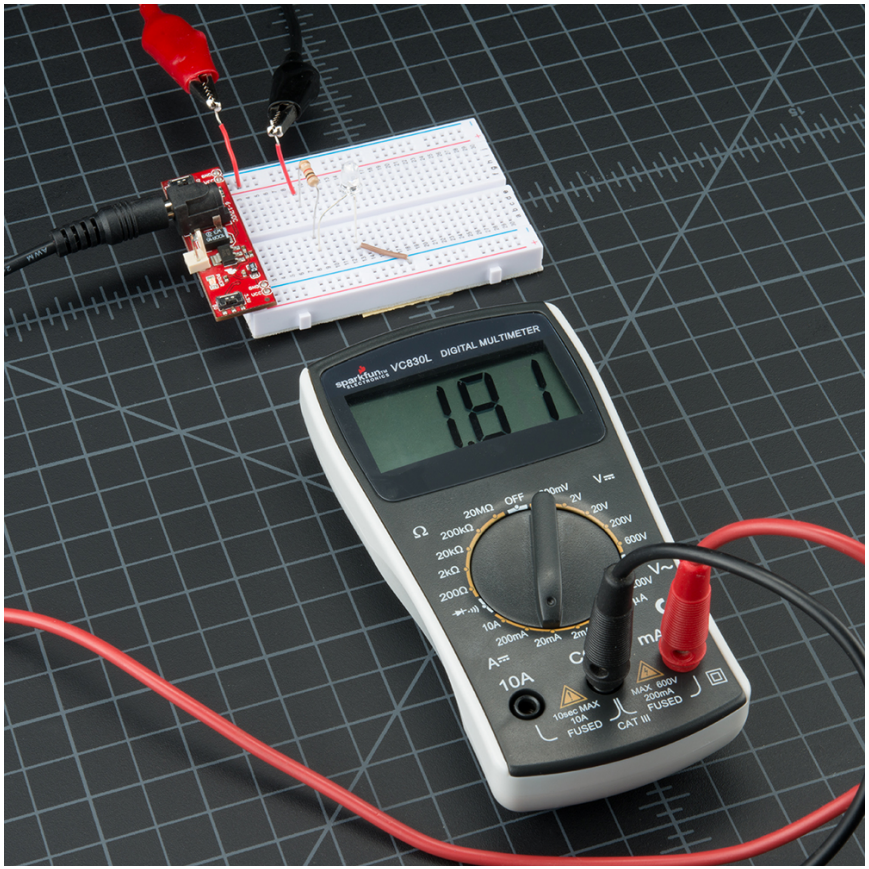


Resistor Color Bands

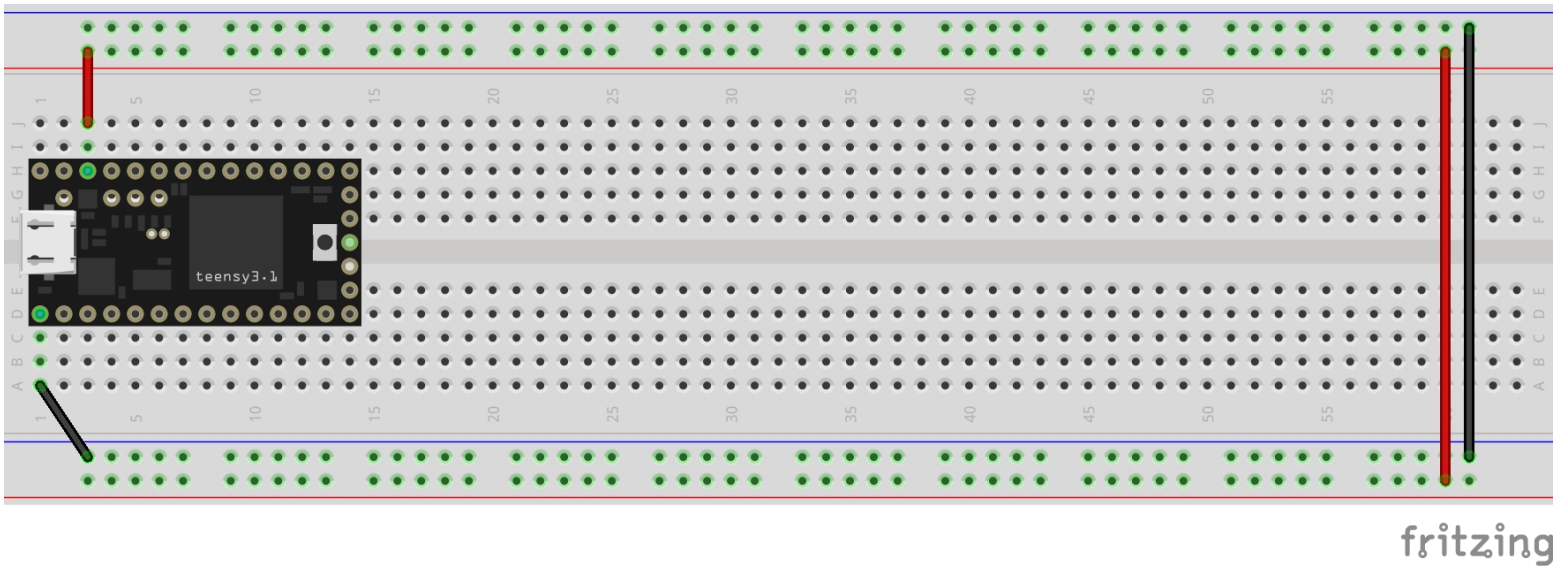


COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8	100MΩ	± 0.05%
White	9	9	9	1GΩ	
Gold				0.1Ω	± 5%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

Measuring Voltage, Current, and Resistance



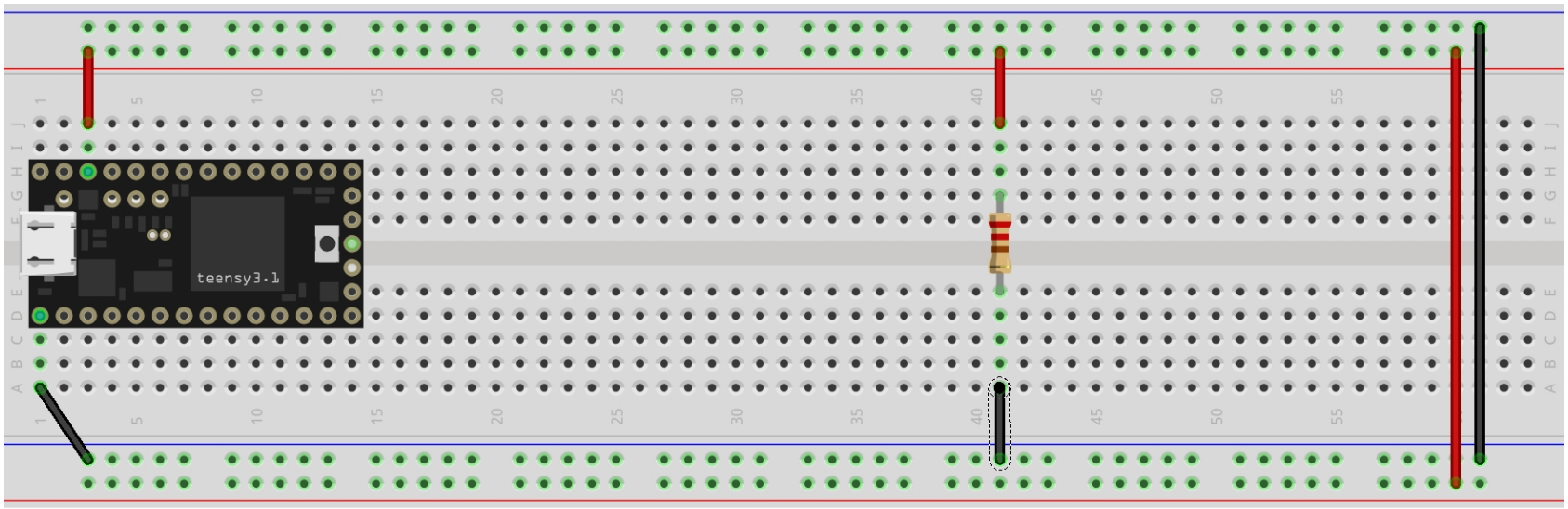
Power from the Teensy 3.2



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

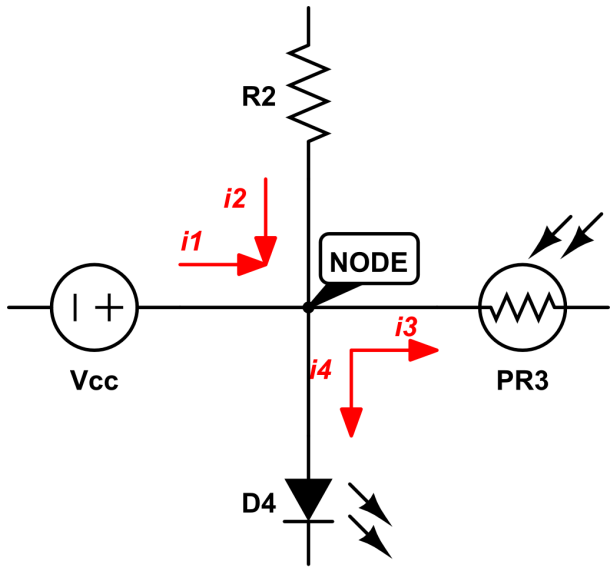


Using your multimeter, measure the voltage "across" and current "through" the resistor.

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:

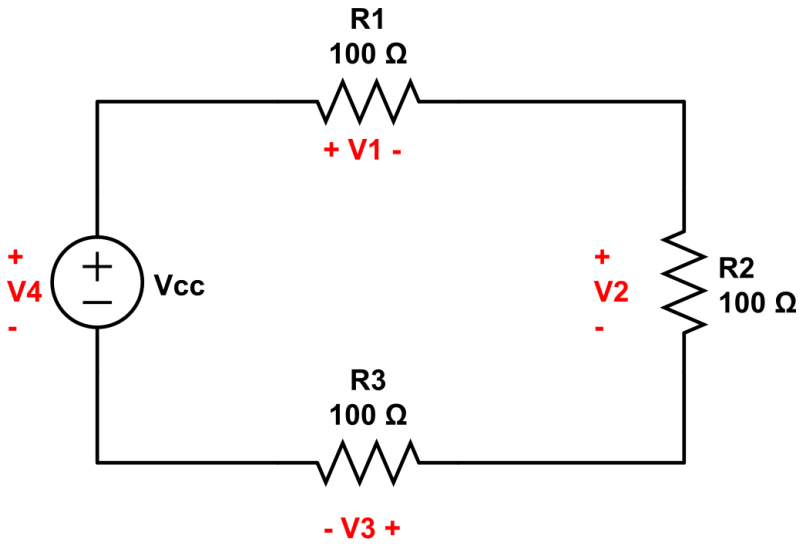
> In an electrical circuit, the sum of currents flowing into that node is equal to the sum of currents flowing out of that node



i1 + i2 = i3 + i4

Kirchhoff's Second Law

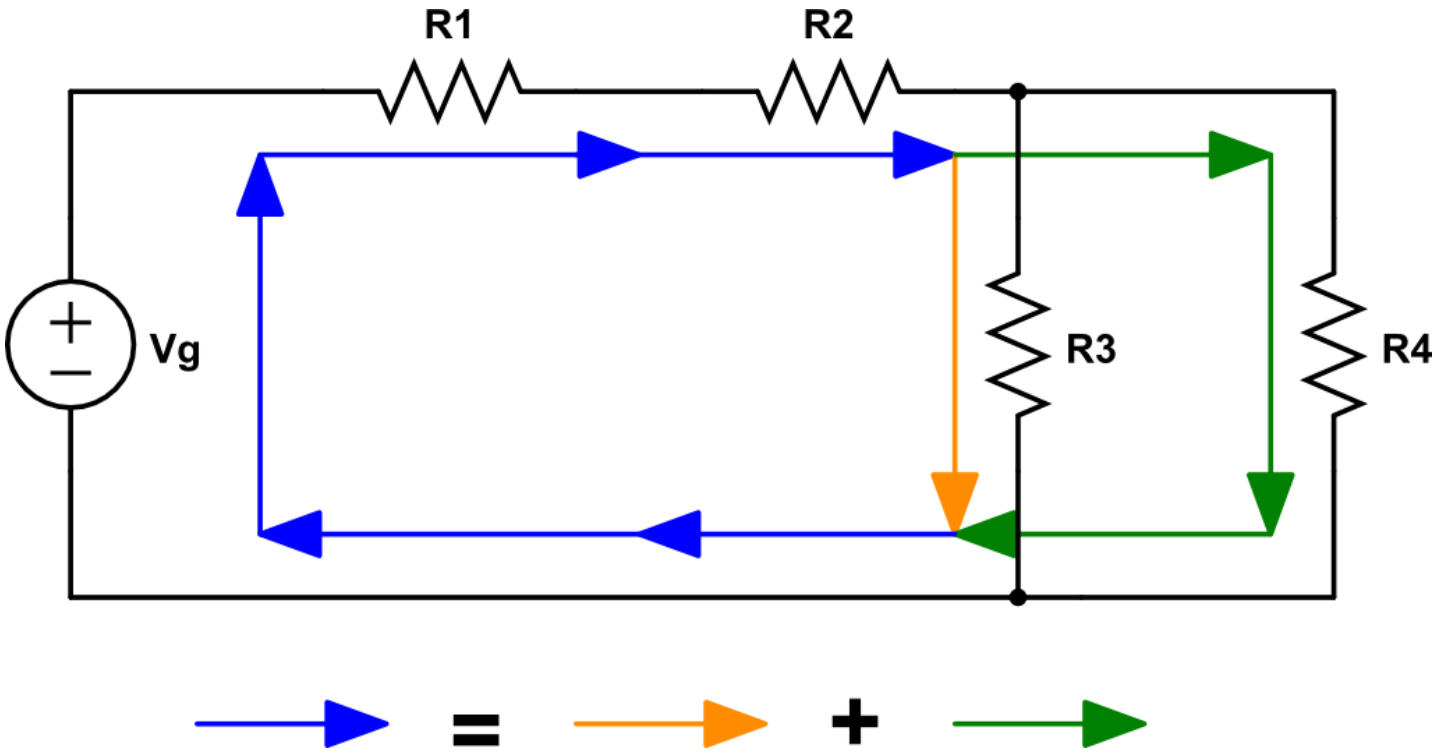
The directed sum of the potential differences (voltages) around any closed loop is zero.



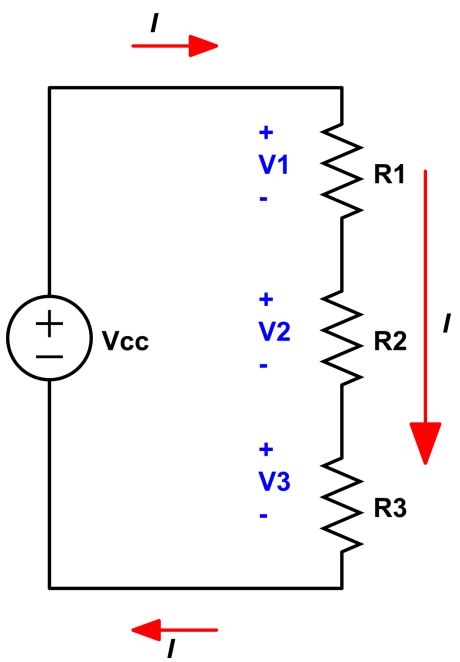
$$V4 - (V1 + V2 + V3) = 0$$

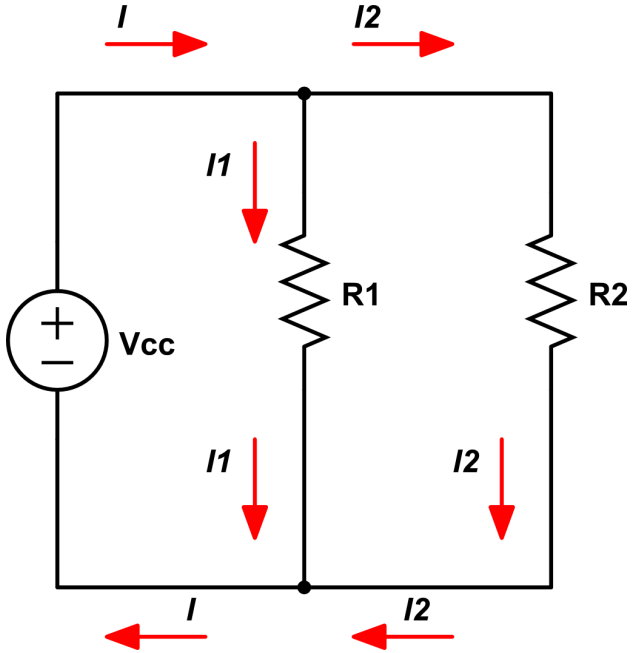
Kirchhoff's Second Law

Resistors in Series and Parallel



Resistors in Series and Parallel

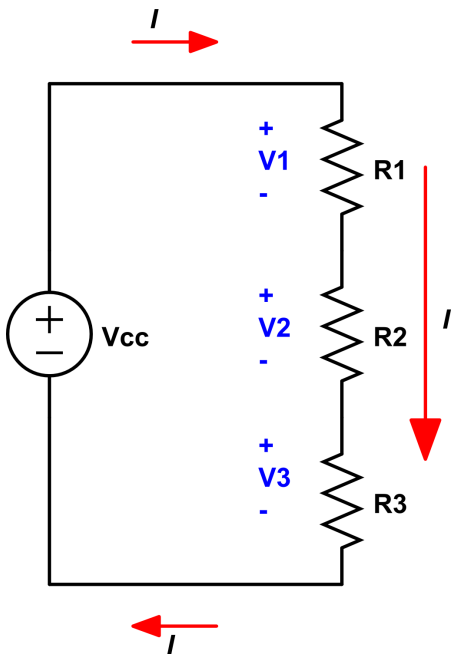




$$R_{eq} = R_1 + R_2 + R_3$$

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

Resistors in Series



$$V_{cc} = V_1 + V_2 + V_3 \quad (1)$$

$$V_{cc} = IR_1 + IR_2 + IR_3 \quad (2)$$

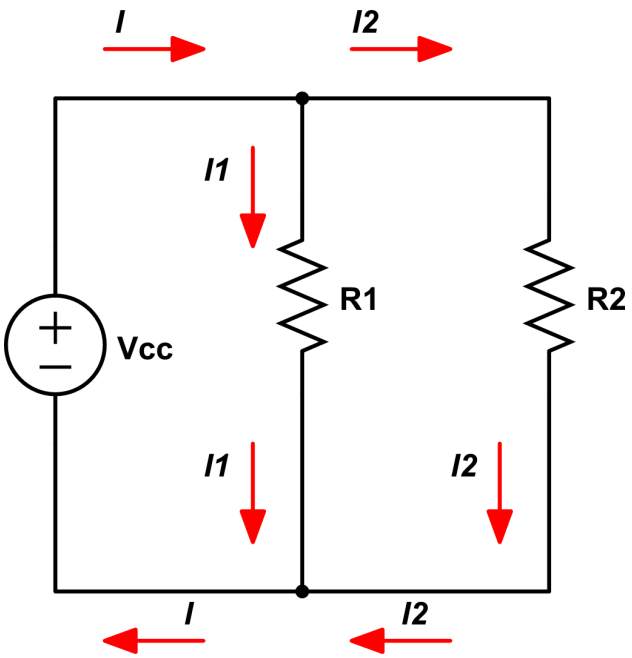
$$V_{cc} = I(R_1 + R_2 + R_3) \quad (3)$$

Node Law: $I = I_1 = I_2 = I_3$

Loop Law:
$V_{cc} - (V_1 + V_2 + V_3) = 0$

$$\boxed{R_{eq} = R_1 + R_2 + R_3} \quad (4)$$

Resistors in Parallel



Node Law: $I = I_1 + I_2$

Loop Law: $V_{cc} = V_1 = V_2$

$$I = I_1 + I_2 \tag{5}$$

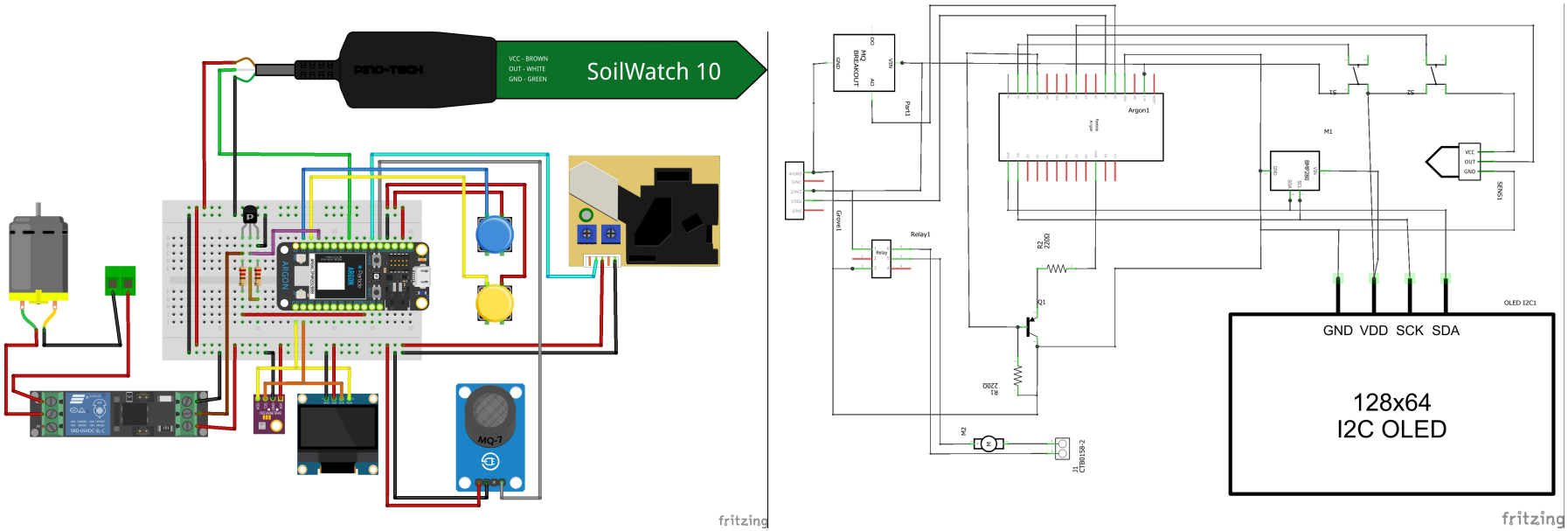
$$I = \frac{V_1}{R_1} + \frac{V_2}{R_2} \tag{6}$$

$$I = \frac{V_{cc}}{R_1} + \frac{V_{cc}}{R_2} \tag{7}$$

$$\frac{V_{cc}}{R_{eq}} = V_{cc}(\frac{1}{R_1} + \frac{1}{R_2}) \tag{8}$$

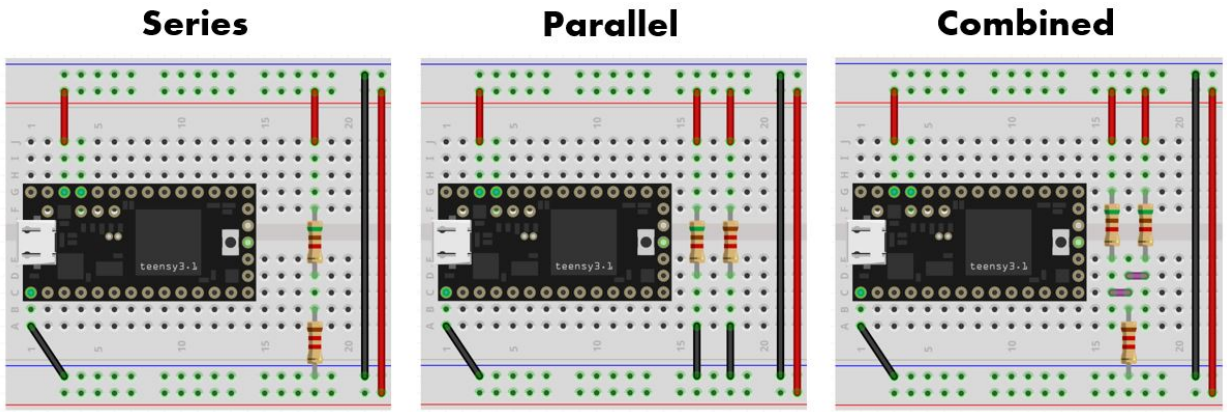
$$\boxed{\frac{1}{R_{eq}} = (\frac{1}{R_1} + \frac{1}{R_2})} \tag{9}$$

Install Fritzing - download from Teams



Smart House Plant Watering System

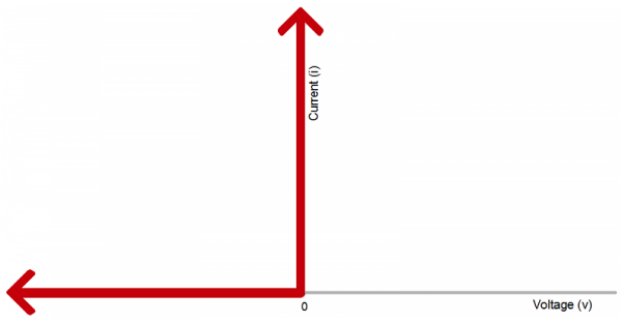
Assignment: L02_00_Resistors

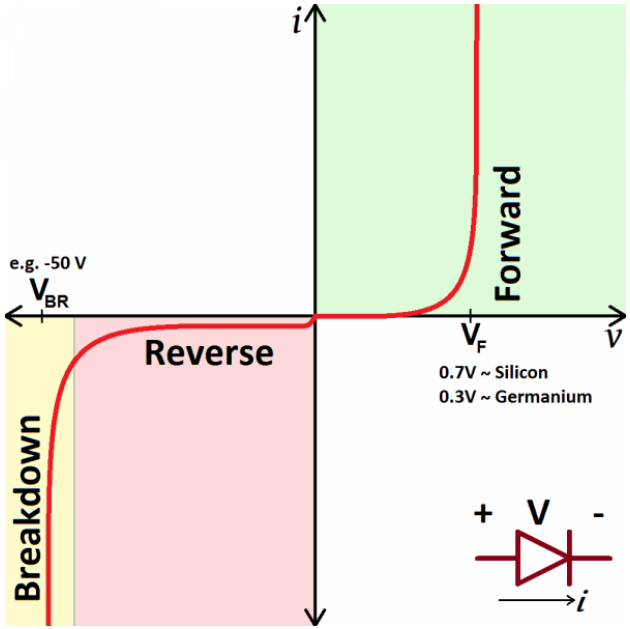


- In your lab notebook, draw the circuit diagrams
  1. Series: $5.1k\Omega$ and $1.2k\Omega$
  2. Parallel: $5.1k\Omega$ and $1.2k\Omega$
  3. Combined: Two parallel $5.1k\Omega$ in series with $1.2k\Omega$
- Calculate the combined resistance, the voltage at each node, and the current through each component.
- Create Fritzing diagram.
- Build **(one at a time)** on your breadboard and test your calculations with a multimeter.

Diodes

The key function of a 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.

diode

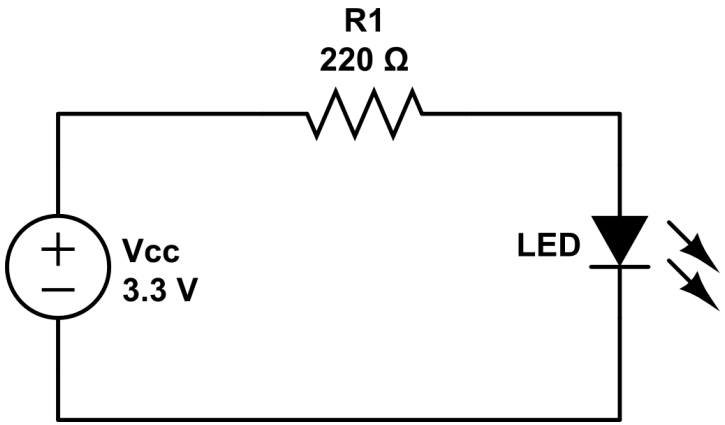
light emitting diode

Current Limiting Resistors

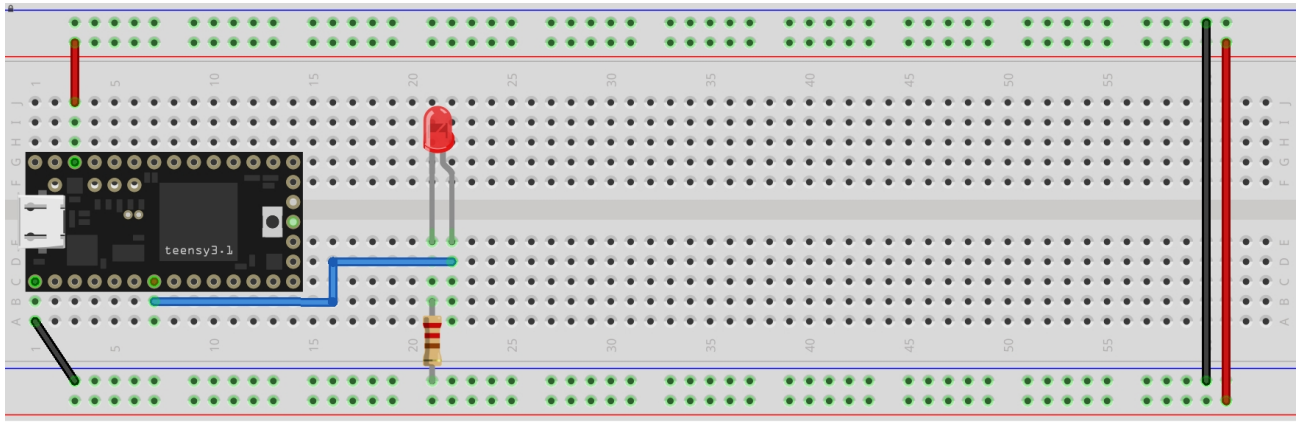
As an LED has very little resistance, when it is connected directly to a power supply, the current draw will exceed its specifications and it will burn out.

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

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.
- Measure the voltage at both leads of the LED and record the voltage in your notebook.
- Change the resistor to $1k\Omega$ and then $10k\Omega$. What happens to the brightness? Measure the voltage and current in each case. Record in your notebook.

REMEMBER: Lab notebook, Fritzing, breadboard, then code

Constants and Variables

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

- A **Constant** is a declaration that does not change throughout the code. For example, the pin that an LED is attached to.
- A **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 editing a single declaration.
- It allows the code to do math.

The first two Data Types that we will be using:

- **int**: an Integer between $\pm 2,147,483,648$.
- **float**: a Floating point number with 7-digits precision.

Operators

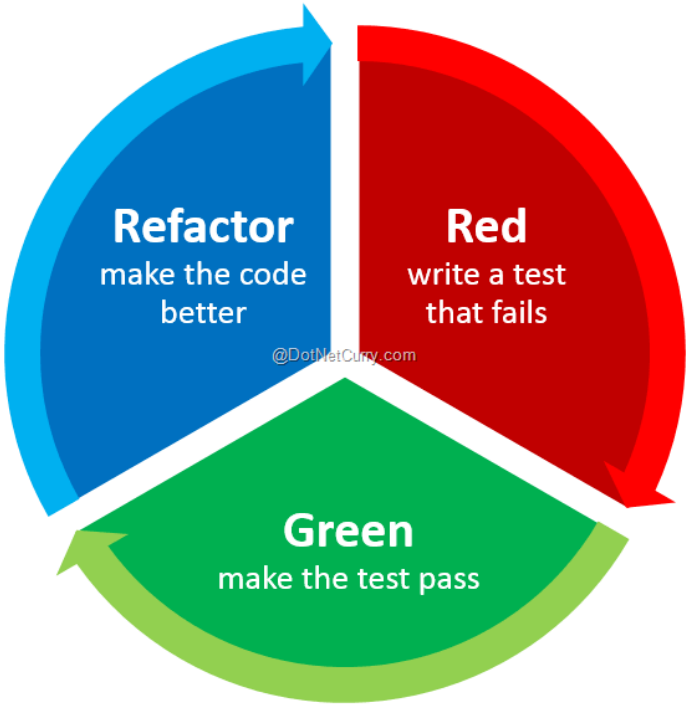
There are a number of operators that act on variables:

```
// Assignment
x = y;       // assign x to be equal to y

// Math Operators
sum = x + y;
difference = x - y;
product = x * y;
quotient = x / y;
remainder = x % y;

// Incrementing
i = i + 1;
i += 1;
i++;

// Decrementing
i = i - 1;
i -= 1;
i--;

// Comparison
(x == y); // true if x is equal to y
(x != y);    // true if x is not equal to y
(x > y);     // true if x is greater than y
(x >= y);    // true if x is greater than or equal to y
(x < y);     // true if x is less than y
(x <= y);    // true if x is less than or equal to y
```

Constants and Variables Example

```
const int LEDPIN = 5;
const int LEDDELAY = 1000;
int i;

void setup() {
  pinMode(LEDPIN, OUTPUT);//set LEDPIN as Output
  i = 100;
}
void loop() {
  digitalWrite(LEDPIN,HIGH);
  delay(LEDDELAY);
  digitalWrite(LEDPIN,LOW);
  delay(LEDDELAY+i);
  i = i + 100;
}
```

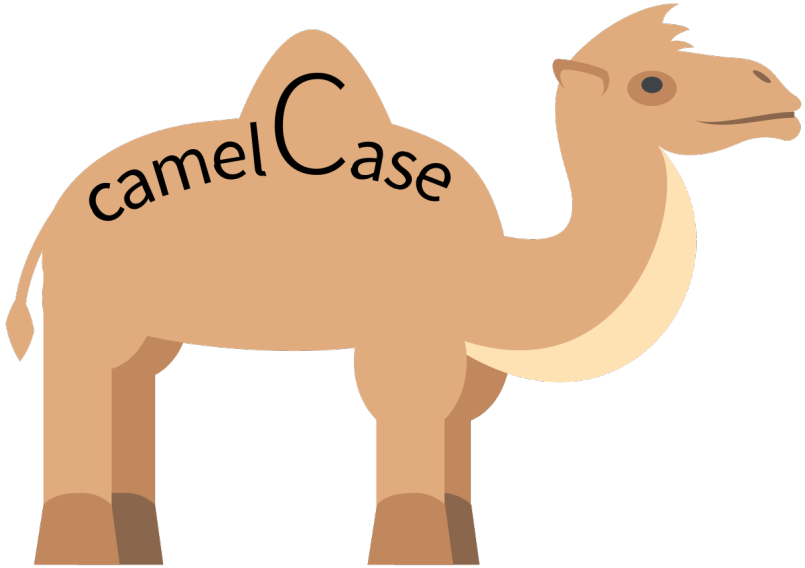
Assignment L02_02_helloLEDvar



- Refactor your L02_01_helloLED code to use constants and/or variables.

Note: refactoring code is when you rewrite portions of your code to make it more readable, or to make it run more efficiently.

IoT Style Guide - camelCase

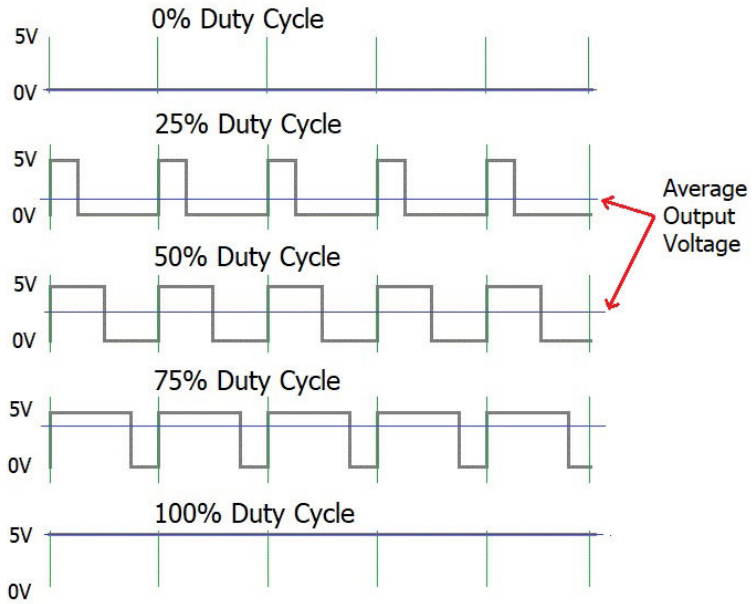


Camel Case is the practice of writing phrases without spaces or punctuation, indicating the separation of words with a single capitalized letter, and the first word starting with either case. In IoT, we will start the first word as lowercase and subsequent words with upper case to delineate words.

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:

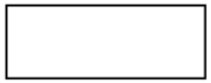
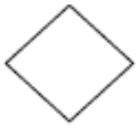
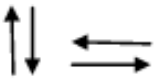
- 255
- 63
- 171
- 16

Measure the voltage with your multimeter at each value.

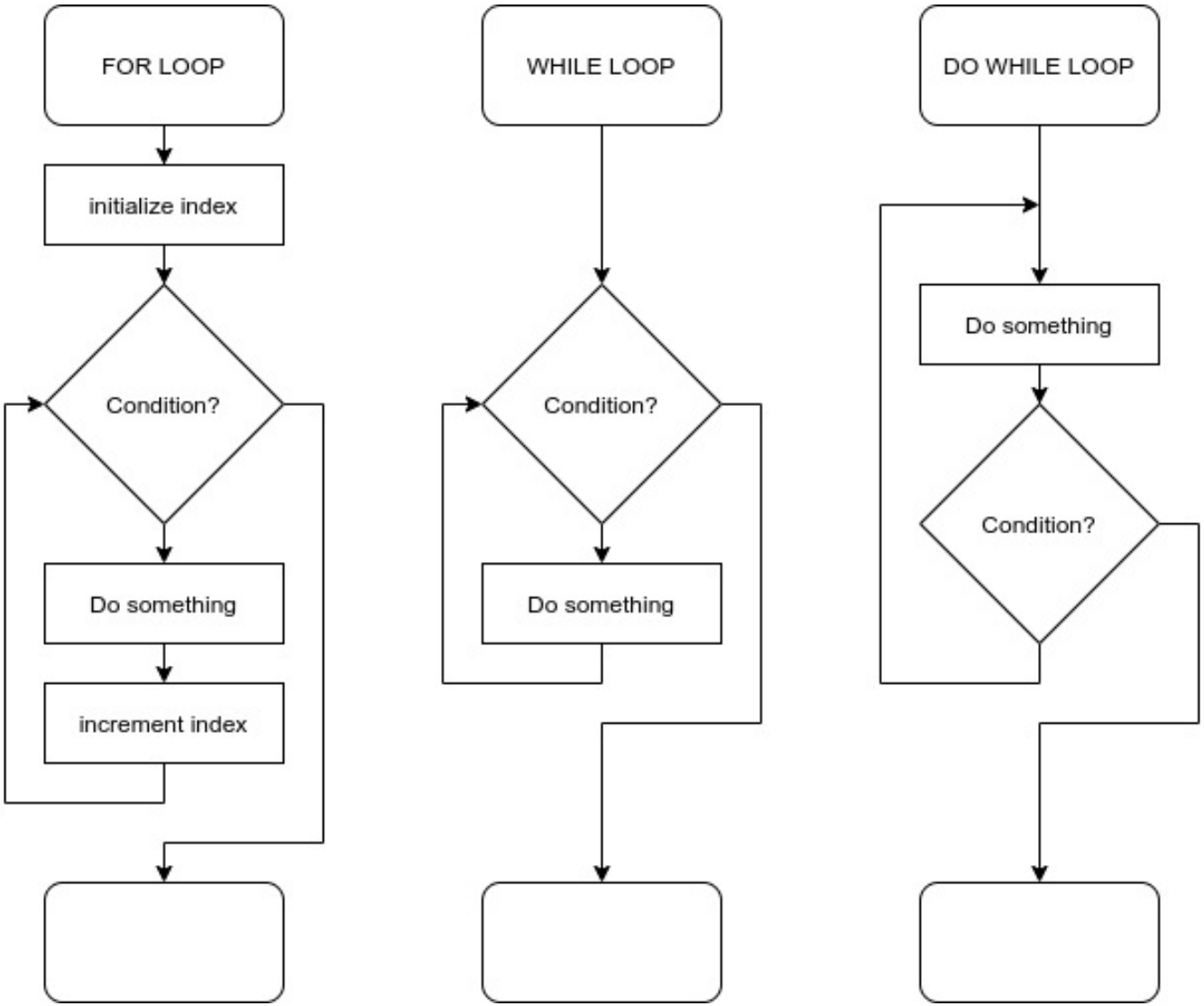
Syntax: analogWrite(pin,value);

- pin: the pin to write to
- value: the duty cycle of the pulses, an int between 0 (always off) and 255 (always on)

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 }
```

Note: the third parameter of a FOR loop can also decrement; i.e. $j--$ or can go up or down by a specified amount; i.e. $j = j + 2$

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

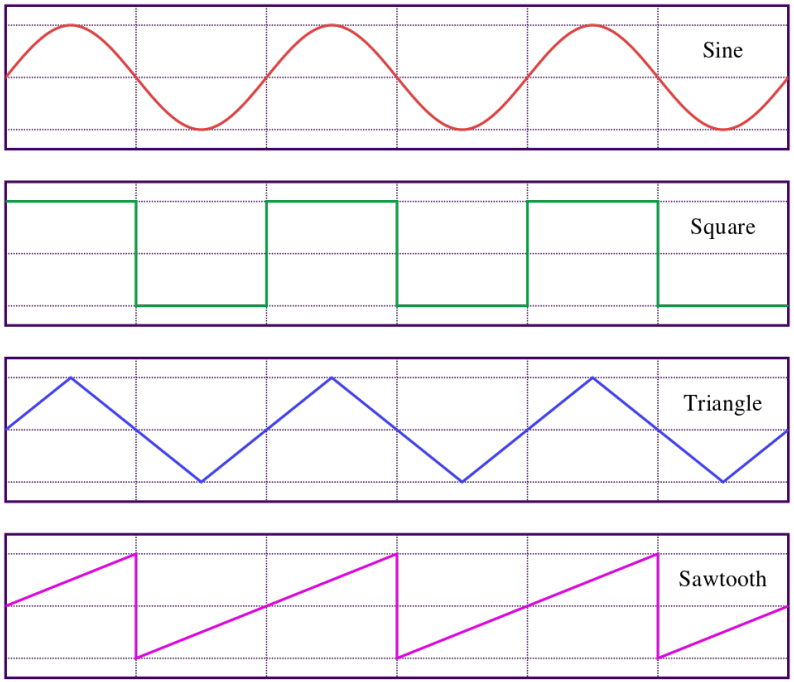
For vs While Loop

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.

Before you write code, create a flow chart of the logic to create a triangle wave.

While or Do While

Number Systems

Decimal

92_{10}

Digits: 0,1,2,3,4,5,6,7,8,9

Binary

01011100_{2}

Digits: 0,1

Hexadecimal

$5C_{16}$

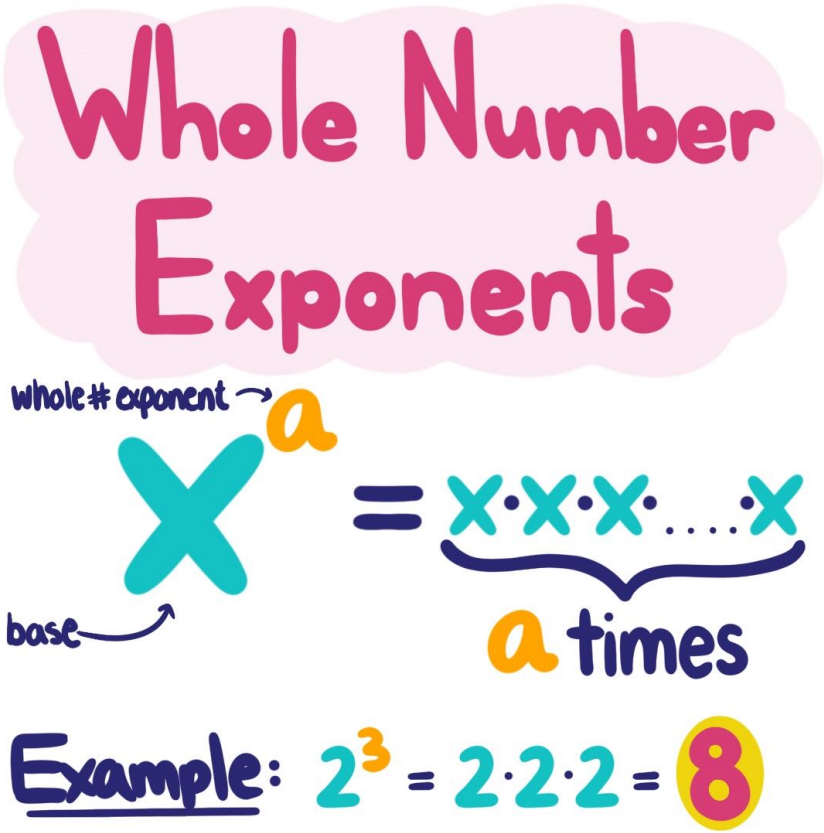
Digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Octal

134_{8}

Digits: 0,1,2,3,4,5,6,7

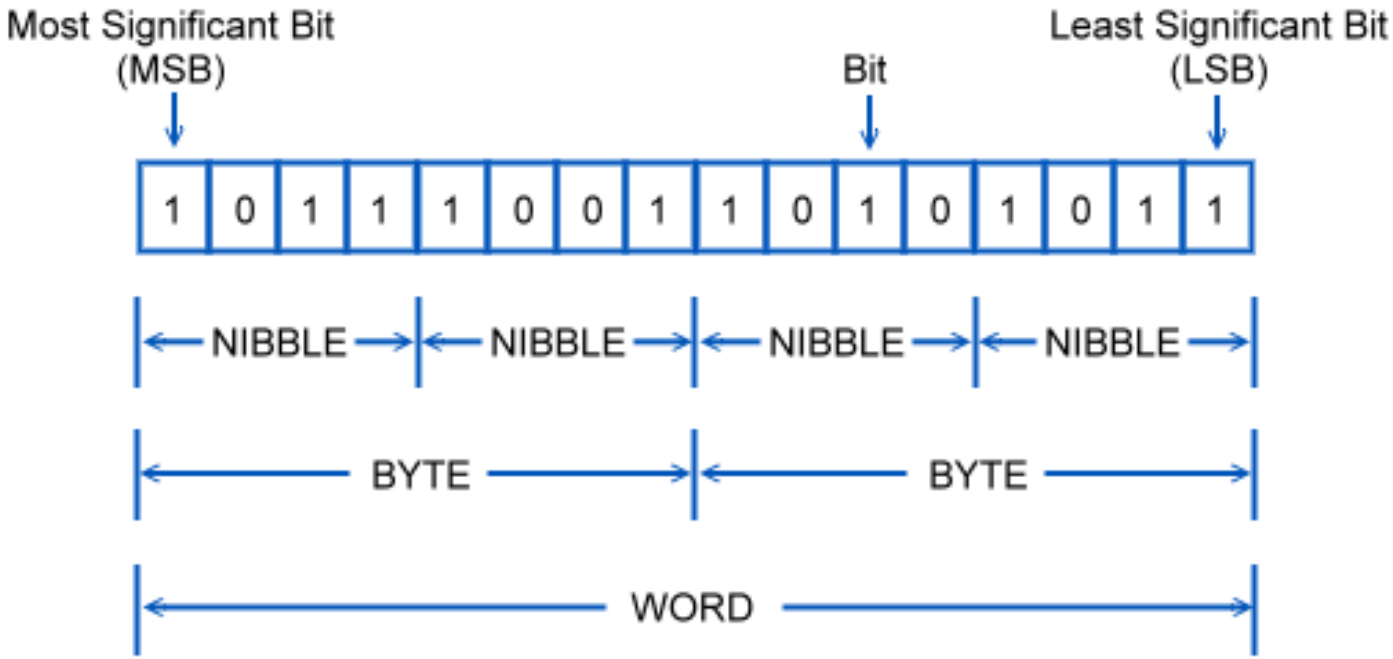
Exponents



The rule for zero as an exponent:

Any nonzero real number raised to the power of zero is one, this means anything that looks like x^0 will always equal 1 if x is not equal to zero.

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						
uint8_t	1	n/a	0 to 255	1	n/a	0 to 255
int8_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.

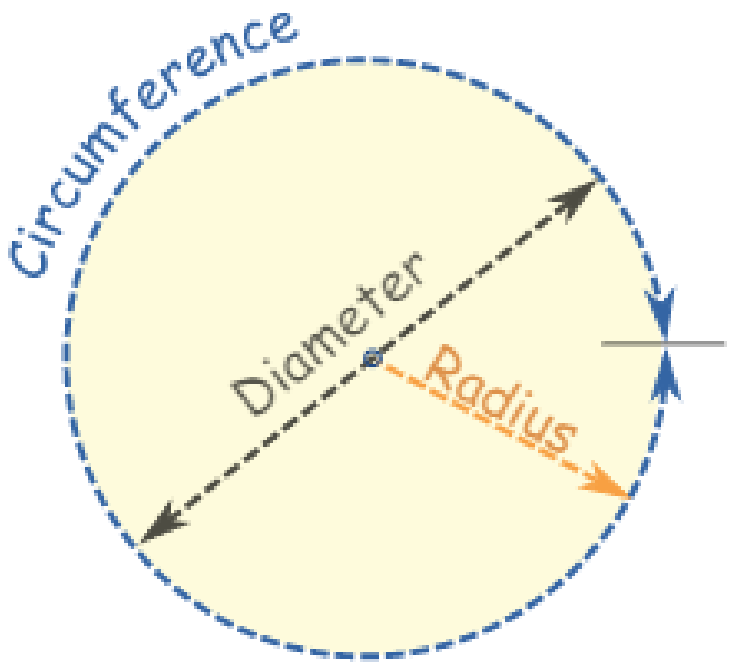
Math Warning and Type Casting

Be cognizant of the data type when performing math operations.

```
int x = 3;
int y = 2;
float yf = 2.0;
float z;

//int divided by an int returns an int
z = x/y;                        // z = 1.0
z = x/yf;                       // z = 1.5
z = x / 2;                      // z = 1.0
z = x / 2.0;                    // z = 1.5

//type casting used to change datatype
z = (float) x / (float) y;   // z = 1.5
z = x / (float) y;           // z = 1.5

z = (int) (x / yf);          // z = 1.0
y = x / yf;                  // y = 1
```

Type Casting is a way to ensure that you are correctly moving between datatypes.

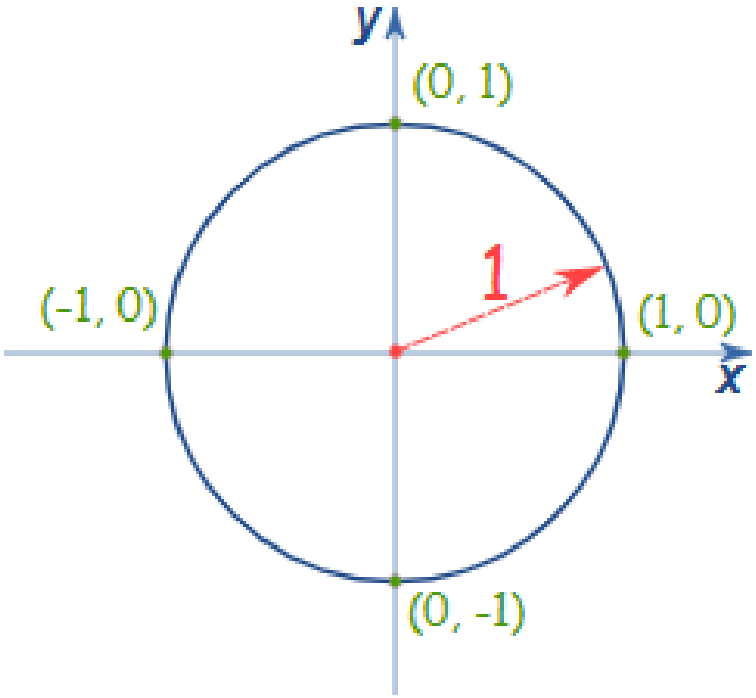
Pi (π)

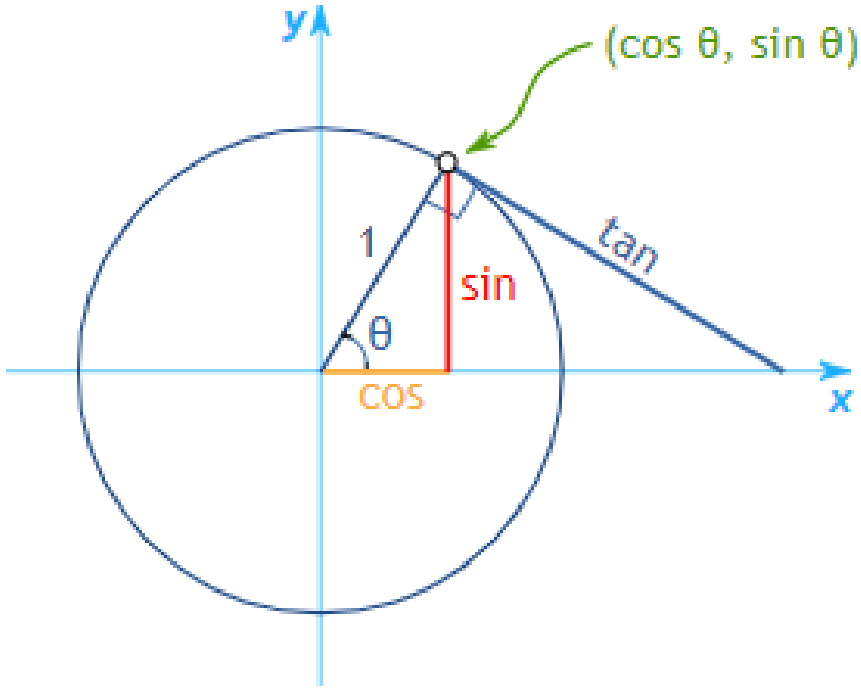


$$\frac{\text{Circumference}}{\text{Diameter}} = \pi = 3.14159\ldots$$

Unit Circle and Trigonometric Functions

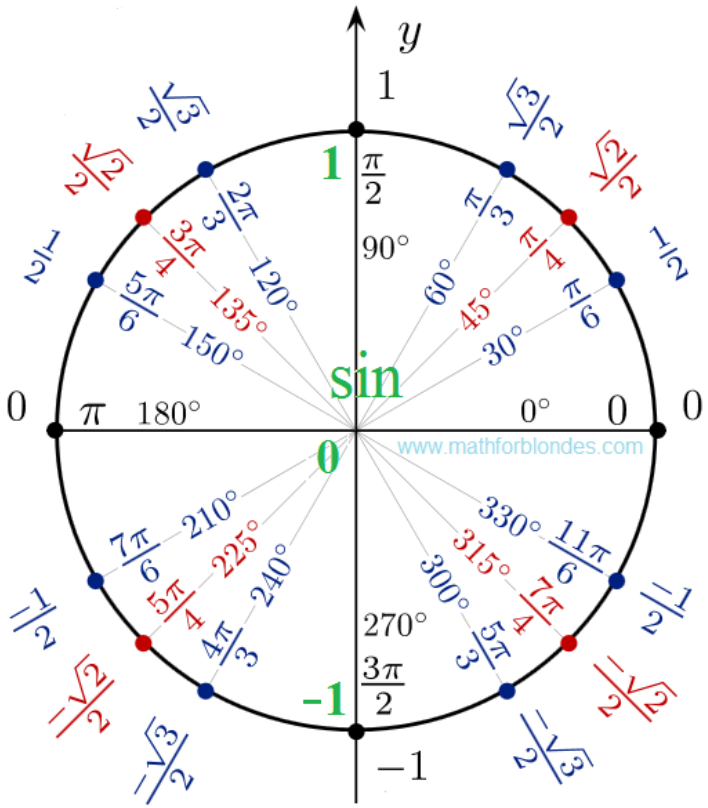
The Unit Circle is a circle with a radius of 1.

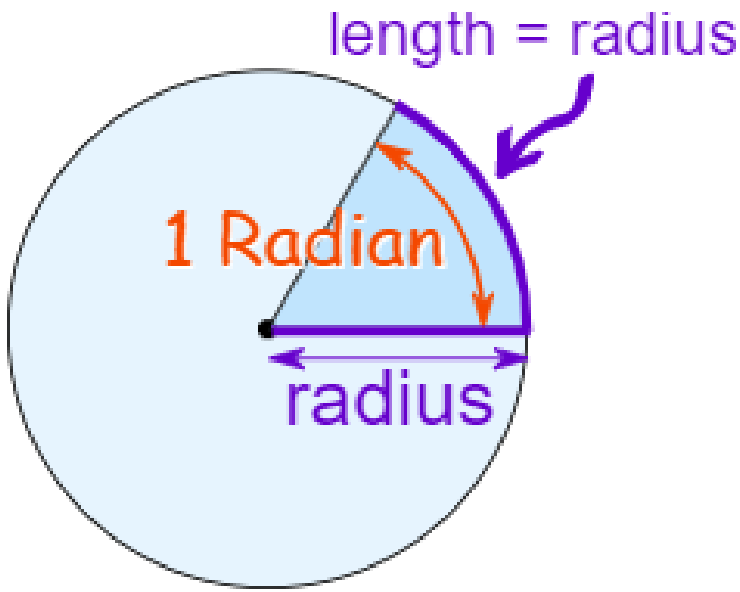




The Unit Circle can be used to map out the trigonometric values of sine, cosine, and tangent.

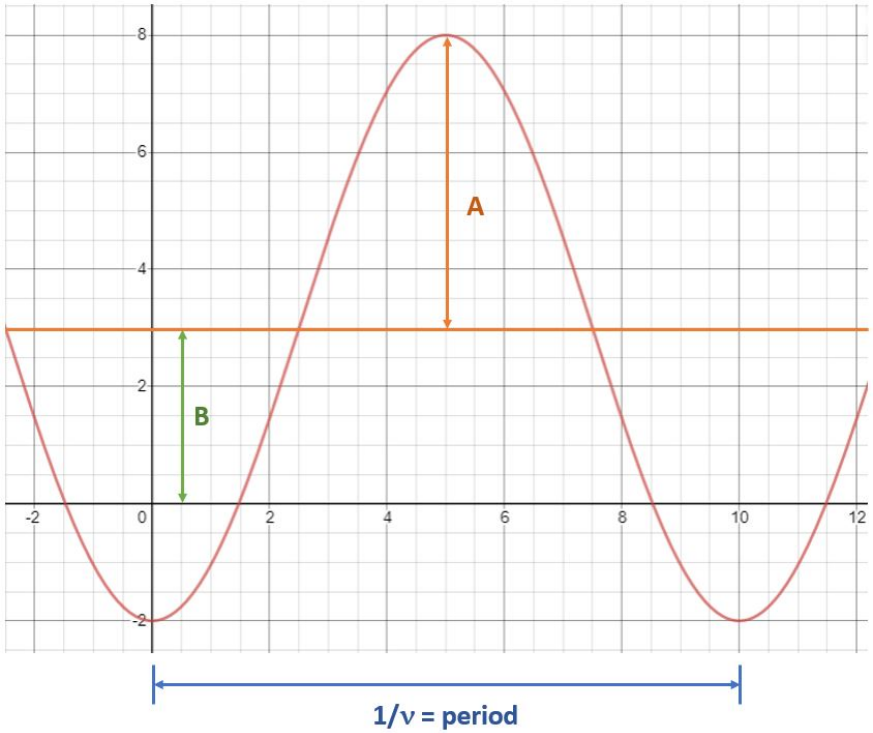
Unit Circle and the Value of $sin(\theta)$





- $sin(\theta)$ is the y-value of the point on the Unit Circle at angle θ.
- In our trig functions, θ is measured in radians (rad), not degrees.
- 360 degrees $= 2\pi$ radians.

Sine Waves



$$y = A * sin(2 * \pi * \nu * t) + B$$

where A = amplitude, B = offset, ν = frequency = $\frac{1}{period}$, and t = time in 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:

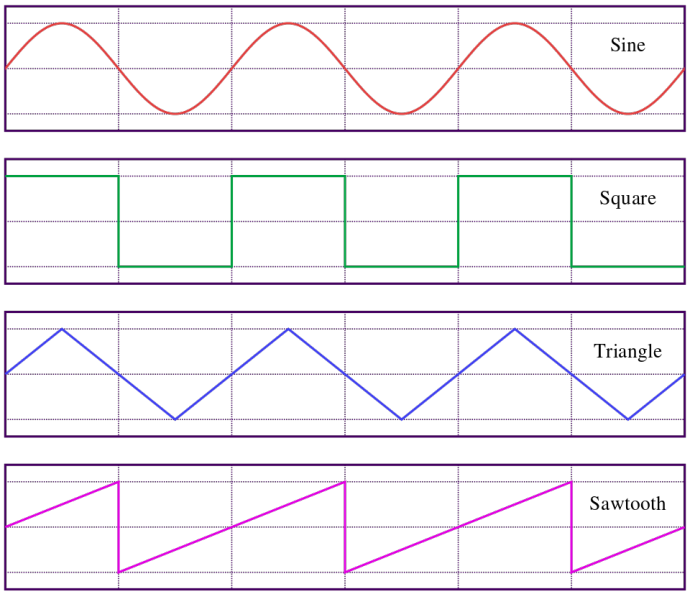
- #include $<$ *file.h* $>$ for system header files.
- #include "*file.h*" for user-created header files in the directory that contains the current code.

An example of a system header file is the math.h header that defines various mathematical functions.

Basic Structure of Arduino Sketch Revisited

```
#include <math.h>           // include header files
const int LEDPIN = 5;       // declare constants
float value,n;              // declare variables

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

void loop() {                // loops indefinitely
  value = sin(2*PI*n);
  n = n+0.25;           // move a quarter around
                        // the unit circle
}
```

Assignment L02_05_helloLEDsin



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

- Use math.h.
- Function sin() takes a double as an input and returns a double.
- Set the period to 5 seconds.

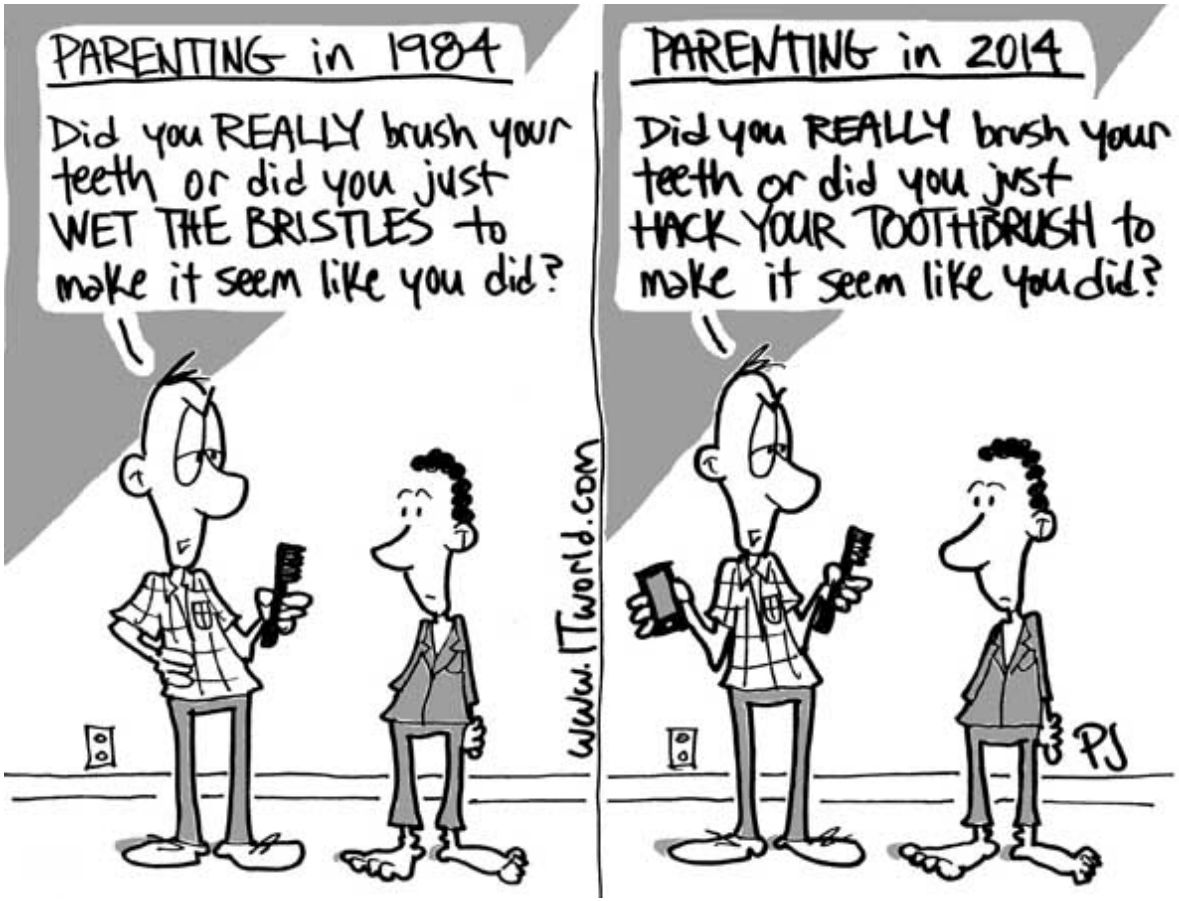
Recall: for a sin wave: $y = A * sin(2 * \pi * \nu * t) + B$

The function millis() returns milliseconds since the Teensy has been powered on. For the sine equation, use t = millis() / 1000.0.

Why is adding the decimal after 1000 important?

L03_Buttons

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

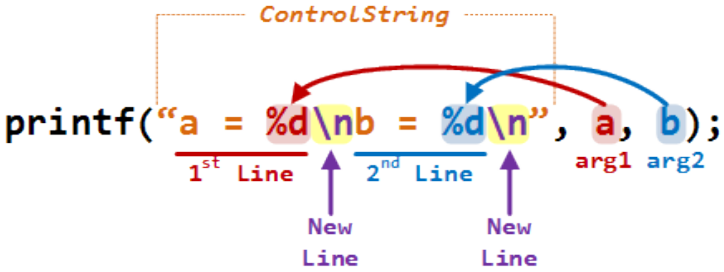
Displaying to the Screen: The Serial Monitor

```
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



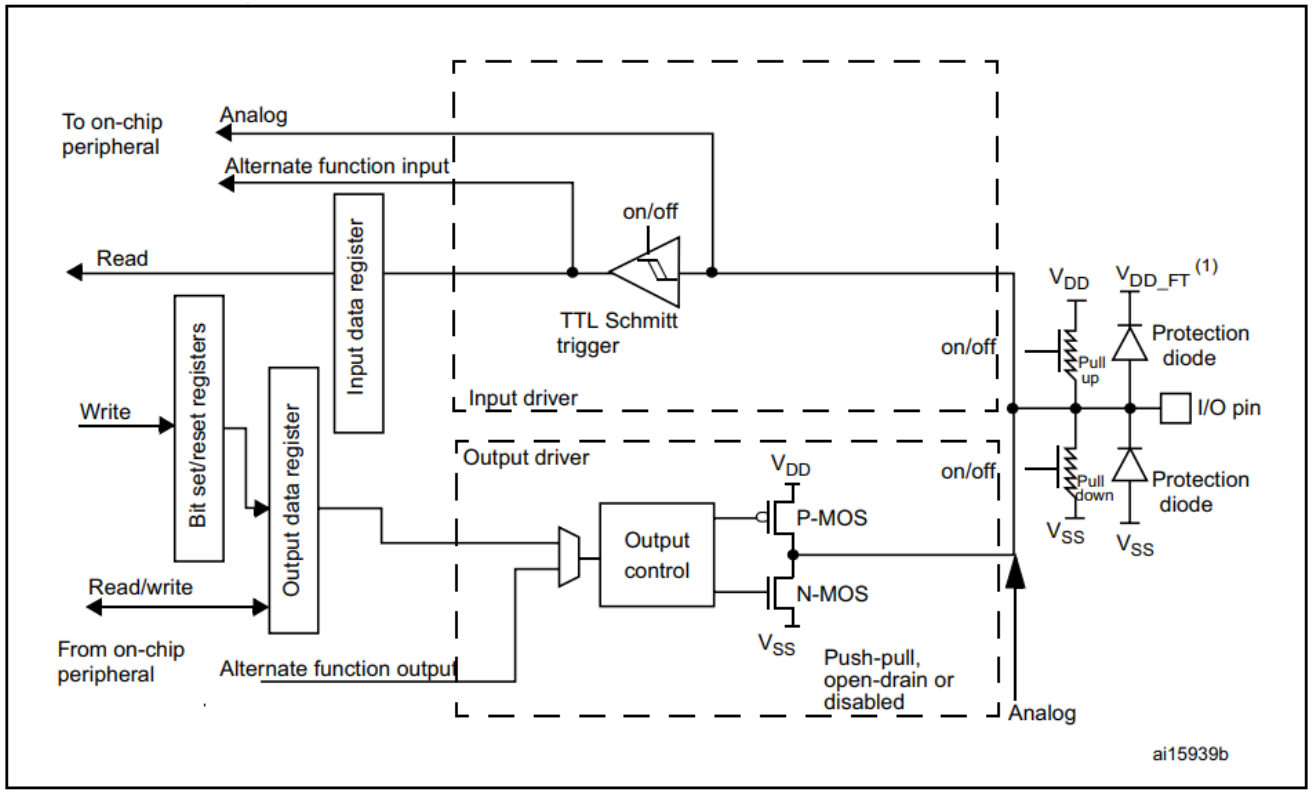
specifier	Output	Example
d or i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (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.	%

```
1 int count = 42;
2 float value = 3.14159;
3 Serial.printf("Print an integer %i and a float %0.4f\n",count,value);
4
5 //Output: Print an integer 42 and a float 3.1415
```

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, three times by using:
   - Serial.print();
   - Serial.println();
   - Serial.printf();

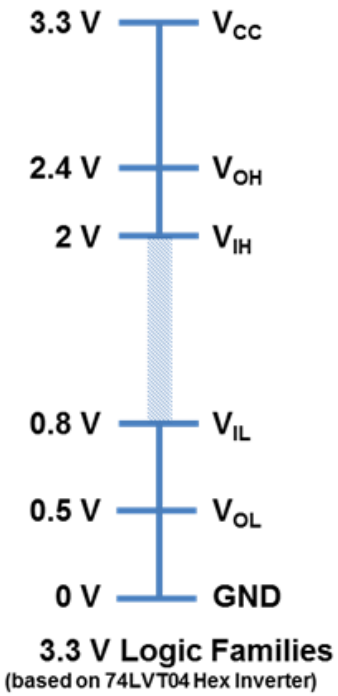
One Pin - Many Functions



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.

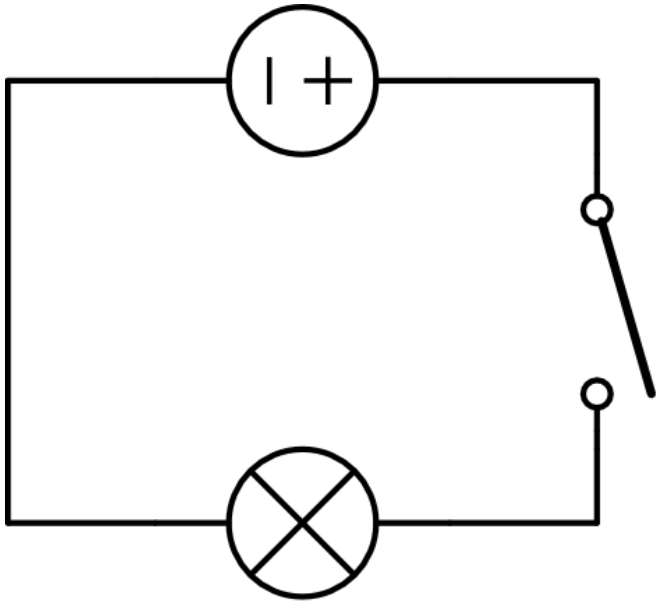


3.3 V Logic Families
(based on 74LVT04 Hex Inverter)

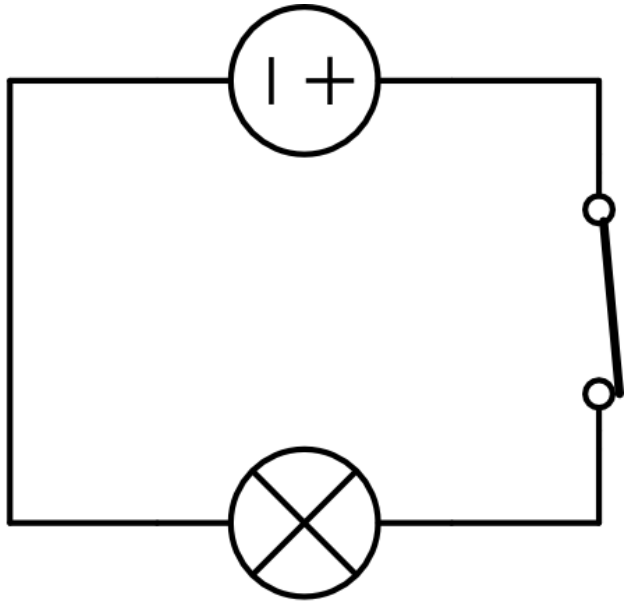
- digitalWrite(pin,value);
- inputValue = digitalRead(pin);

where, value equals HIGH or LOW.

Switches

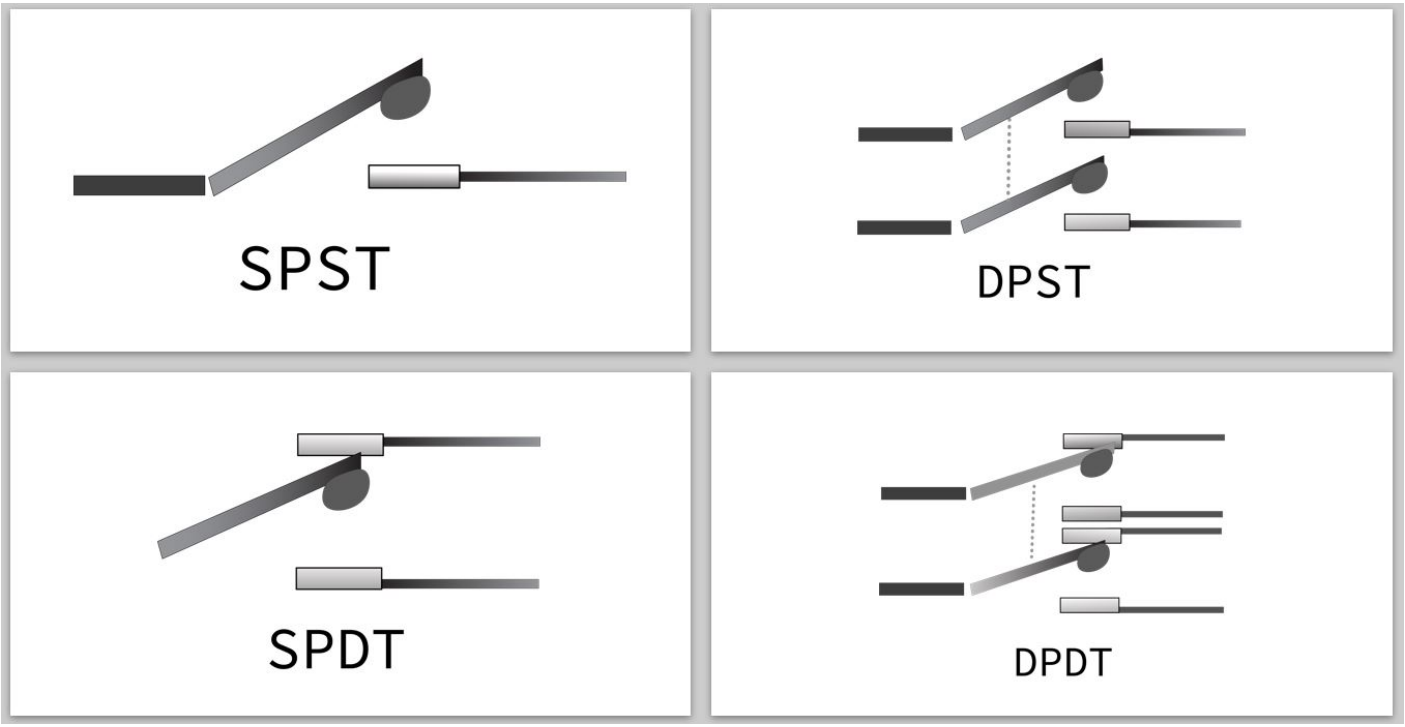


Lamp Off



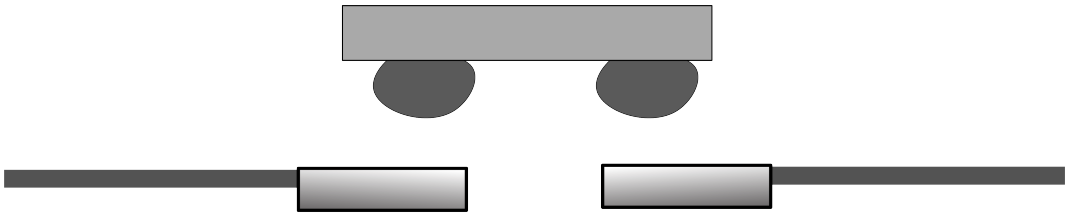
Lamp On

Poles and Throws



- Poles indicates the number of circuits that one switch can control for one operation of the switch.
- Throws indicates the number of contact points.

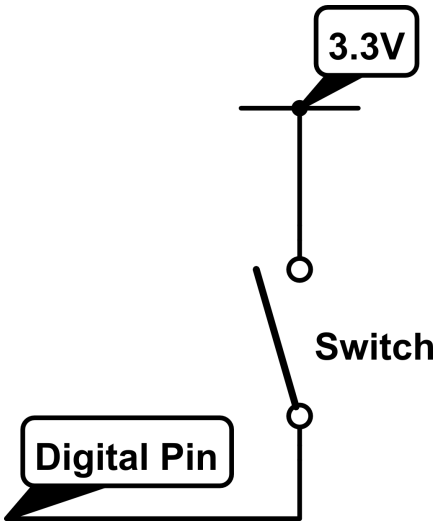
A Button - SPST

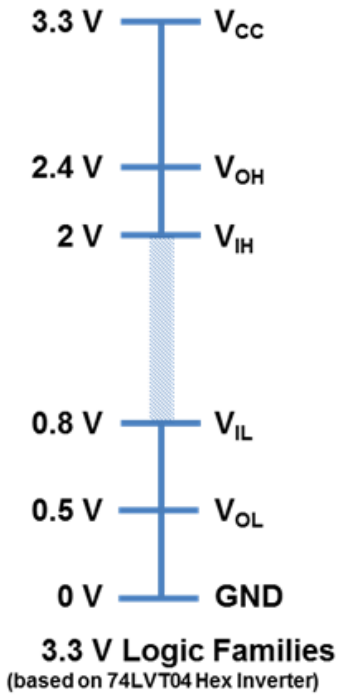


SPST

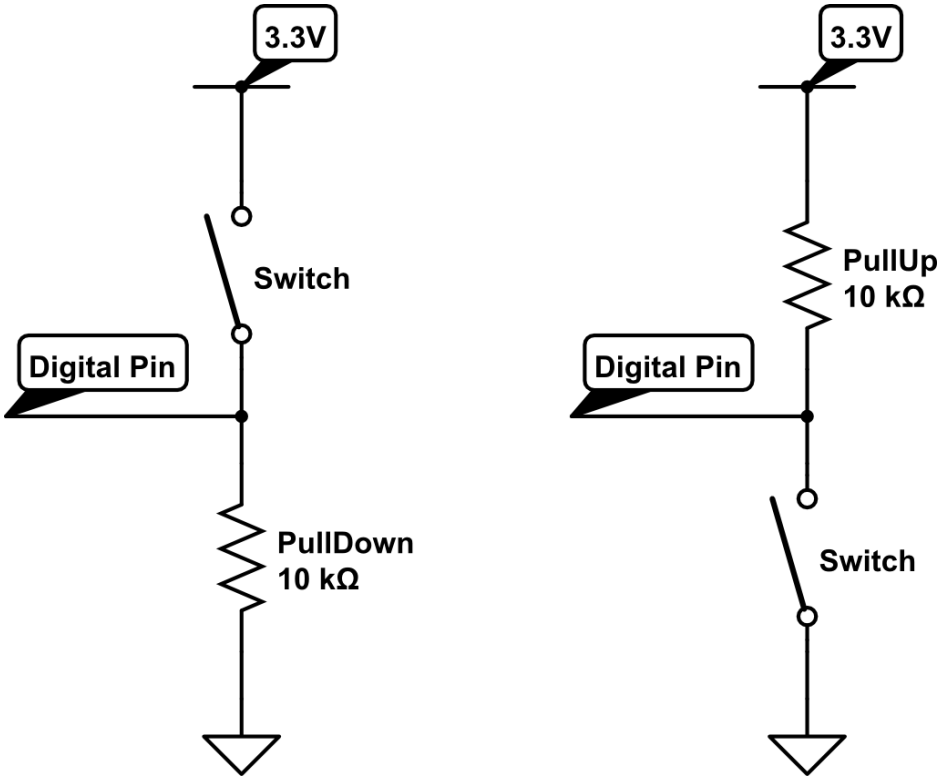
double break

Floating Inputs



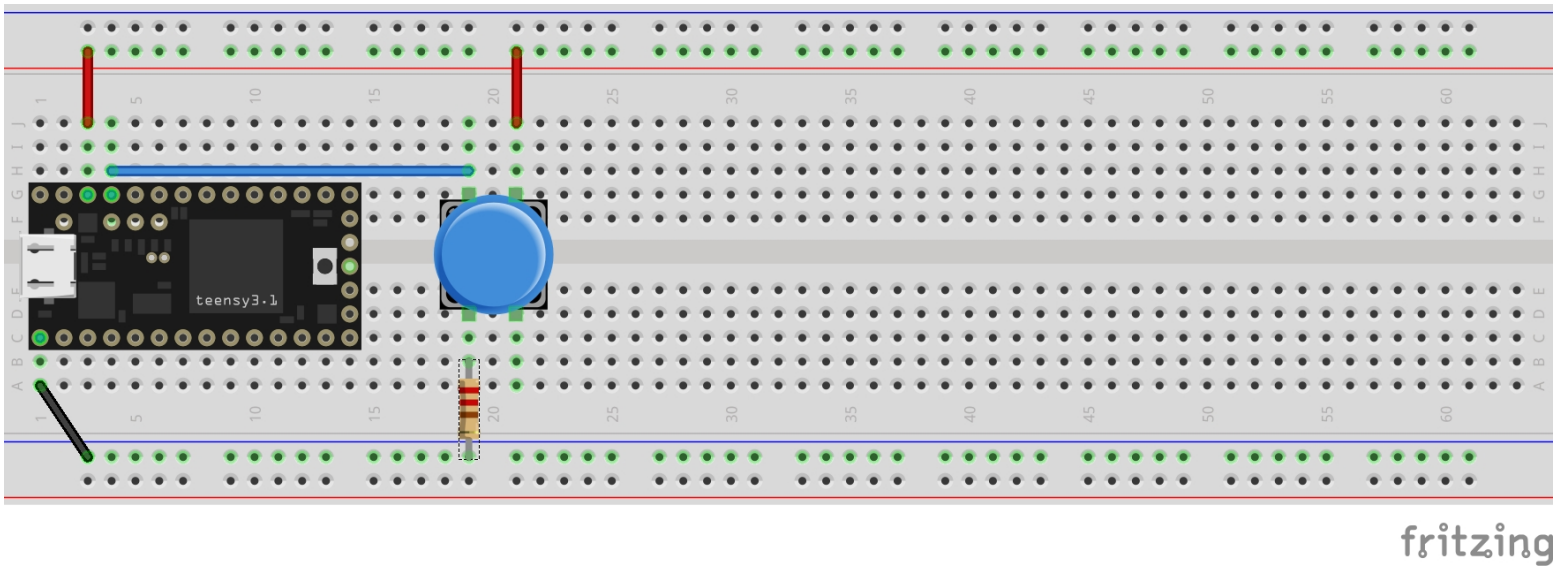


Pull Down and Pull Up Resistors



- What happens if the digital pin is left floating when the switch is open?
- What happens if the pin is connected directly to GND (or V_{cc}) without a resistor?

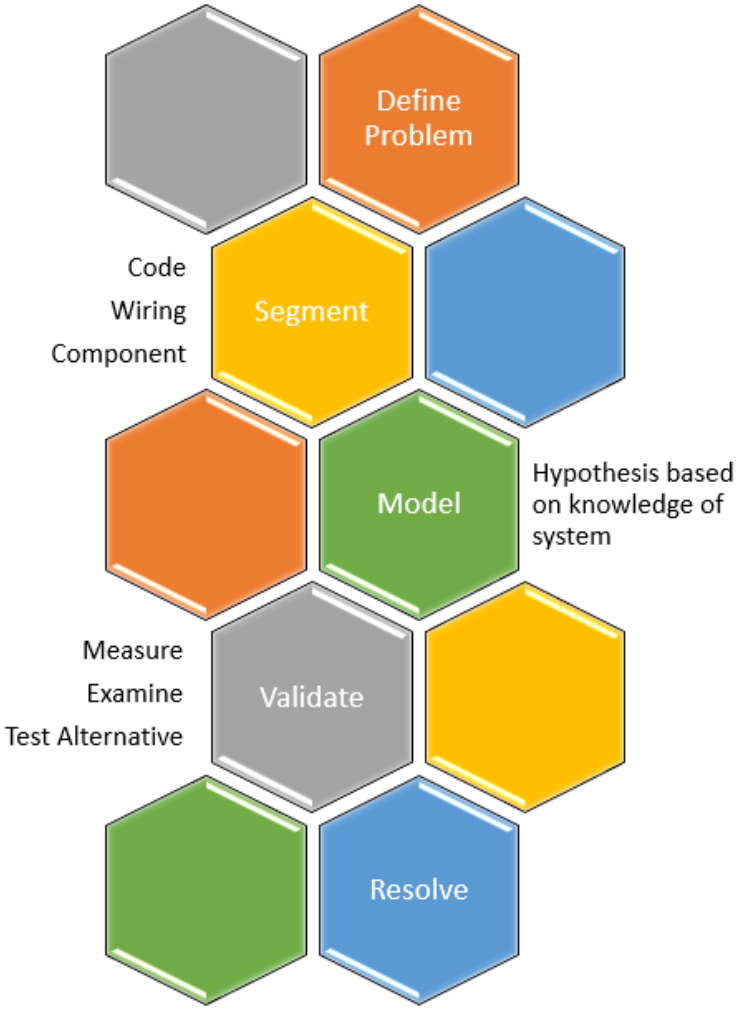
Our First Button and Pull Down Resistors



Reading from a digital pin:

- inputValue = digitalRead(pin);
  where
  - pin is the digital pin that the button is connected to
  - inputValue is an int (declared in the header)

Model Based Troubleshooting



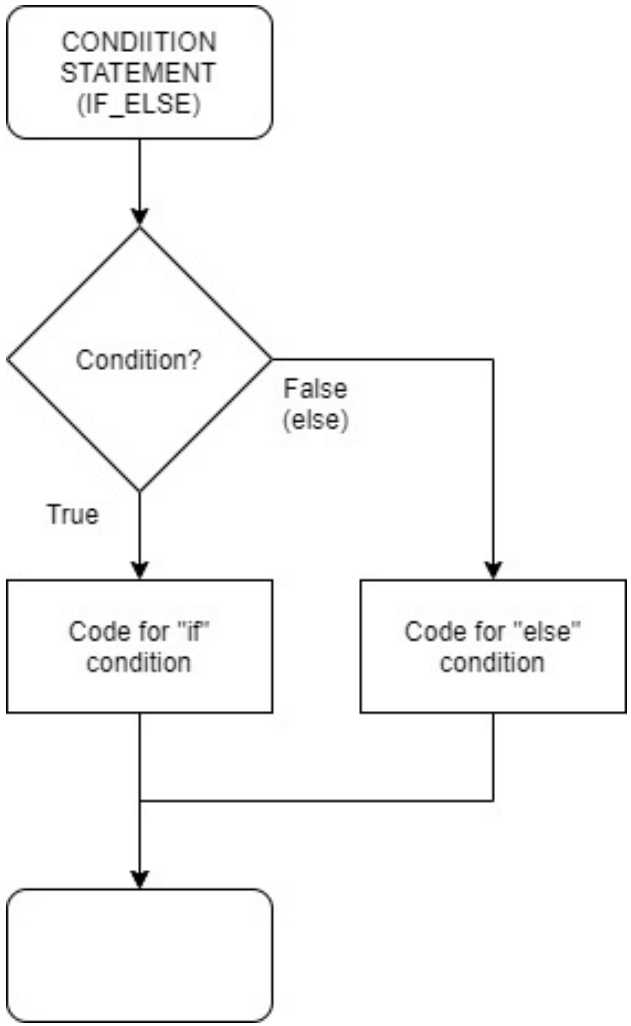
Assignment: Buttons

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

Connect a button (and a multimeter to measure the voltage) to Pin 23.

1. L03_01_button
   - Use digitalRead() to input the button state.
   - Print button state to the screen.
   - Remove the resistor. How does this affect the button state and the voltage?
   - Replace the pull-down resistor with a pull-up. How does the logic change?
     - Not pressed: 3.3V
     - Pressed: GND
2. L03_02_button_input_pullup
   - Remove the pull-up resistor.
   - Implement: pinMode(pin,INPUT_PULLUP);

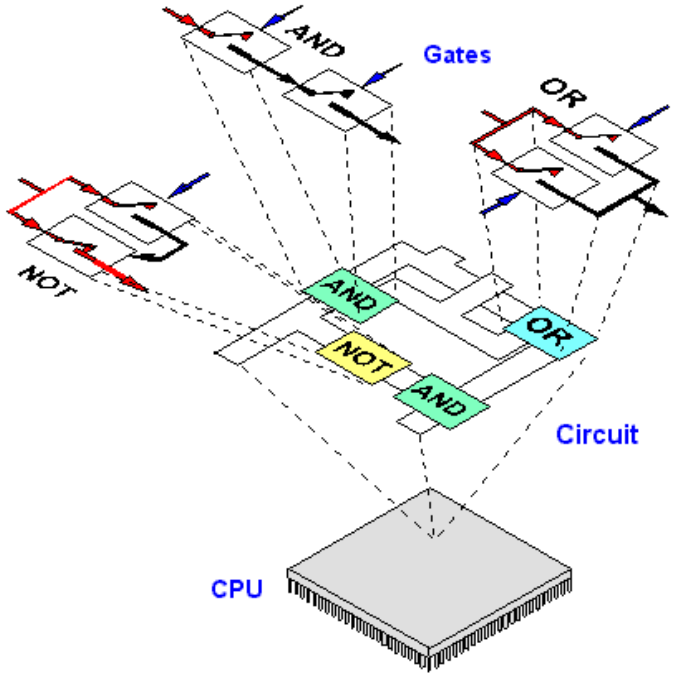
IF-ELSE Statements



IF-ELSE Statements

```
1  // IF statement SYNTAX
2  if (condition) {
3    //statement(s)
4  }
5  else {
6    // else statement(s)
7  }
8
9  // EXAMPLE
10 if (buttonState) {
11   Serial.printf("Button is pressed \n");
12 }
13 else {
14   Serial.printf("Button is not pressed \n");
15 }
```

Data Types: Boolean and Boolean Logic



Boolean datatype (bool) holds either a TRUE or FALSE

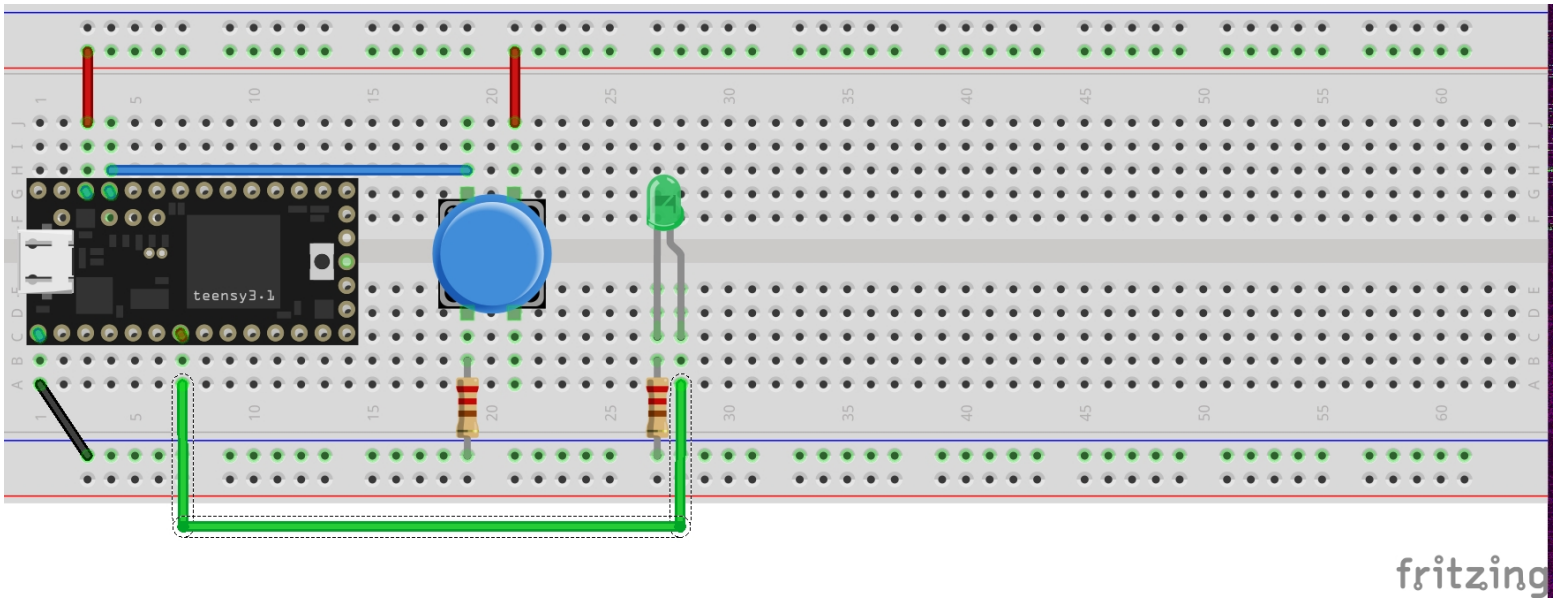
Boolean Logic Operations (condition statements)

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 Logic In Action

```
1  bool x;
2  bool buttonState1, buttonState2;
3  int y, z;
4
5  // ! (NOT) assignment function
6  x = !x;        // toggle x (if x = 1, then set x = 0, and visa-versa)
7
8  // ! (NOT) comparison
9  if (!x) {      // if x is false
10   // do something
11 }
12
13 // LOGICAL AND: if both pins are pressed
14 buttonState1 = digitalRead(PIN1);
15 buttonState2 = digitalRead(PIN2);
16 if ((buttonState1) && (buttonState2) {
17   // do something
18 }
19
20 // LOGICAL OR: if either value is greater than zero
21 if (y > 0 || z > 0) {
22   // do something
23 }
```

Button and LED



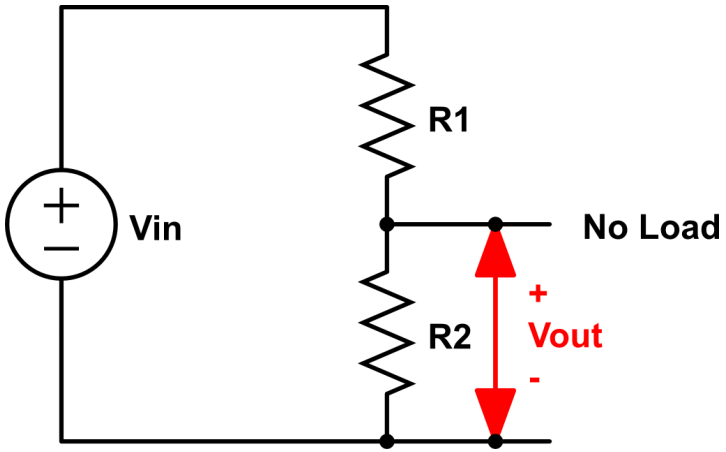
Assignment: Buttons and LEDs

Update your schematic and Fritzing

1. L03_03_buttonLED
   - Add a Green LED to Pin 5 and use the button to turn the LED on/off.
   - Also, print button state to the screen
2. L03_04_twoButtonLED
   - Add a second button (Pin 16) and Yellow LED (Pin 6).
   - Have each button control one LED.
   - Also, print button states to the screen.
3. Extra Credit: Modify twoButton
   - The Green LED lights up if both buttons are pressed
   - The Yellow LED lights up if either button is pressed

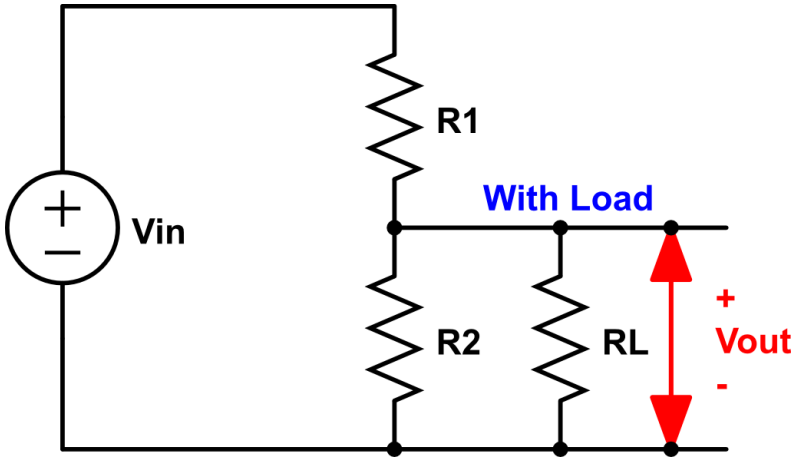
Resistors in Series and Parallel

Open Circuit Behavior



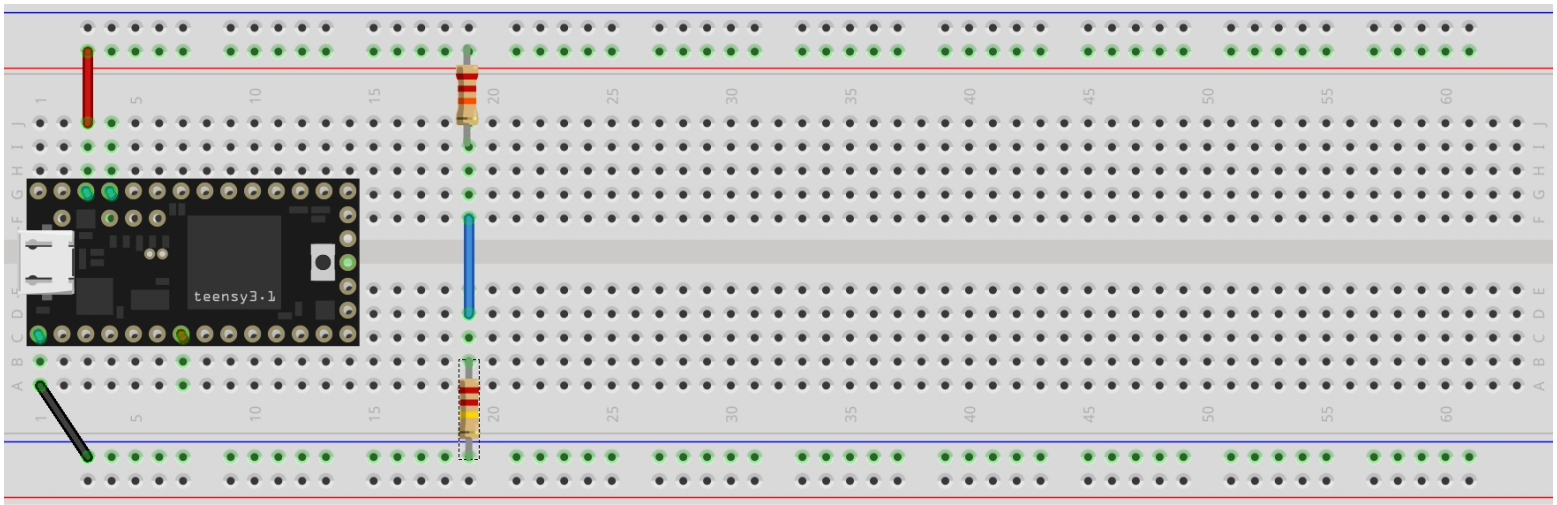
$$V_{out} = V_{in}\frac{IR_2}{I(R_1+R_2)} = \frac{R_2}{R_1+R_2}V_{in}$$

Behavior Under Load



$$V_{out} = \frac{R_2 \| R_L}{R_1+R_2\|R_L}V_{in}$$

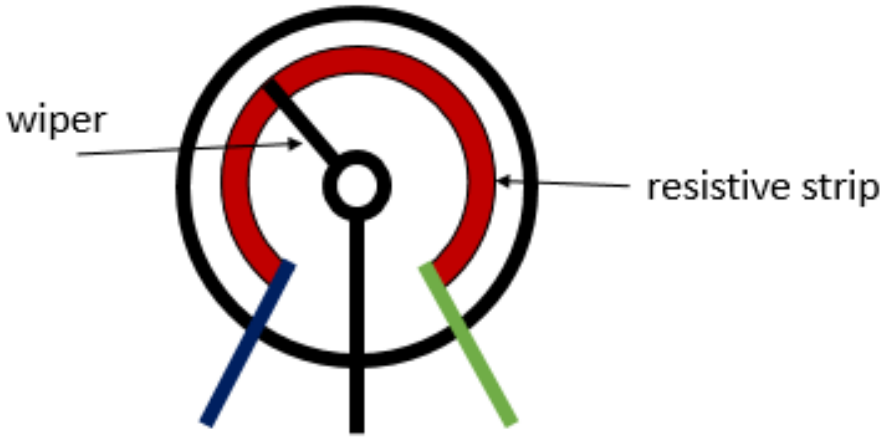
Voltage Dividing



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

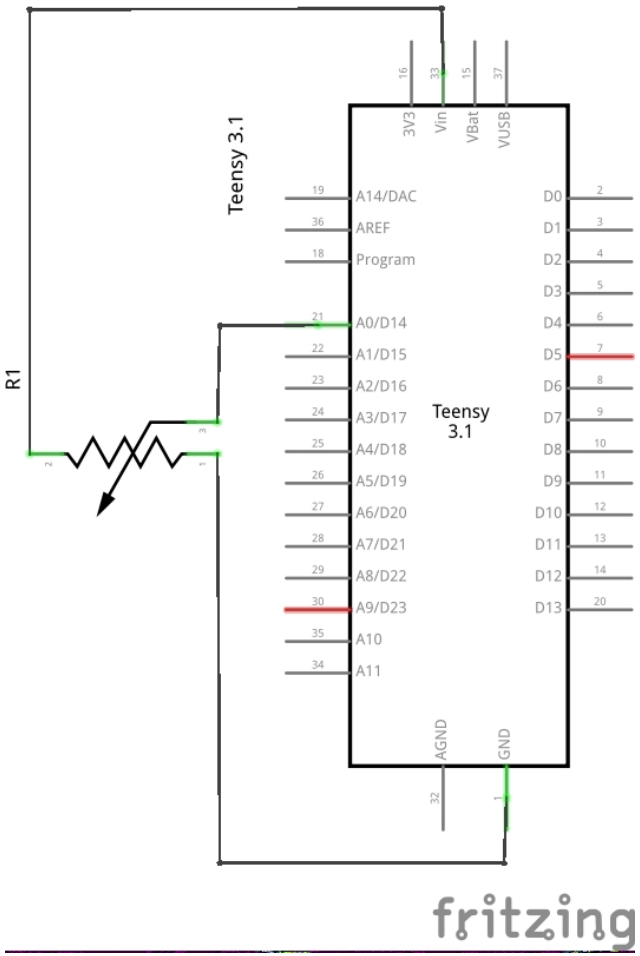
- Use various combinations of resistors between $1k\Omega$ and $22k\Omega$.
- Calculate the Series resistance and the voltage between the two resistors in your Lab Notebook.
- Measure with your multimeter and compare.

Potentiometer - Variable Resistor



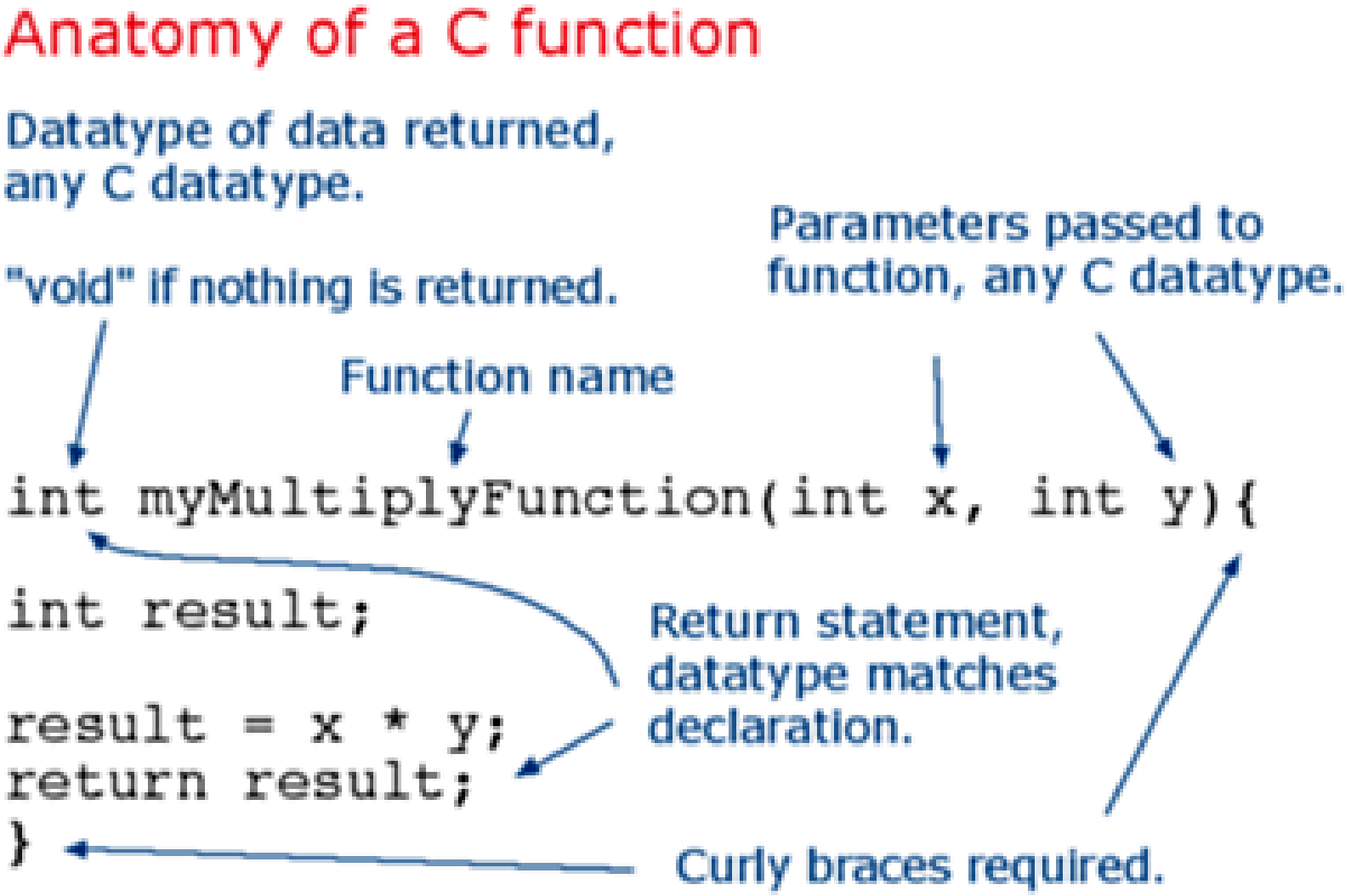
A potentiometer has 3 pins. Two terminals (the blue and green) are connected to a resistive element and the third terminal (the black one) is connected to an adjustable wiper.

Assignment L03_05_AnalogInput



1. Look up the syntax of analogRead() in the Ardunio Reference.
2. Utilize analogRead() to measure analog input across a potentiometer (voltage divider) using Pin 14.
3. Determine the range of the analogRead() across the entire range of the potentiometer.

Anatomy of a Function



Types of Variables

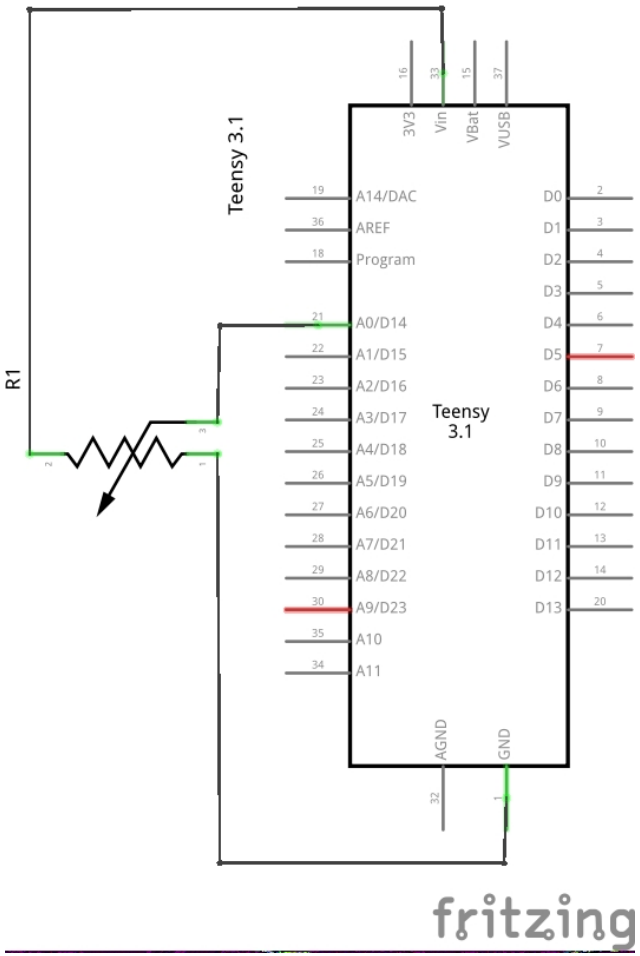
```
1  int x;            // x is a global variable available in the entire program
2  void setup() {
3    x = 1;
4  }
5  void loop() {
6    x = addx();
7  }
8  int addx() {
9     int y = 0;          // y is a local variable, resets every function call
10    static int z = 0;   // z is a static local variable, maintains its value
11    y = y + x;
12    z = z + x;
13    Serial.printf("x = %i, y = %i, and z = %i \n",x,y,z);
14    return z;
15 }
```

1. Global Variables
   - Accessible throughout the program and all functions.
2. Local Variables
   - Accessible only in the function.
   - Created when function is called. Destroyed when function is returned.
3. Static Local Variables
   - Accessible only in the function.
   - Maintains value across multiple calls of a function.

Basic Structure of Arduino Sketch with Functions

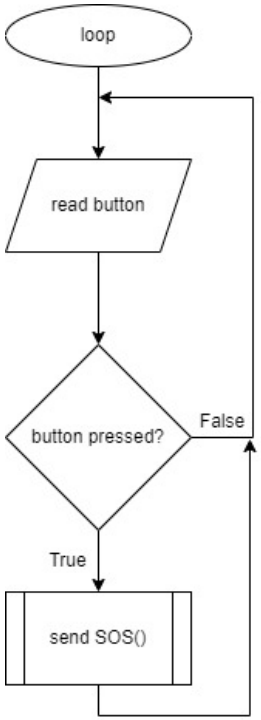
```
/*
 * This is an example of how to use functions
 * The code below converts inches to feet
 */

const int INCHPIN=14;
int inches;
float feet;

void setup() {
  Serial.begin(9600);  // Turn on Serial Monitor
  while(!Serial);      // Wait for Serial Monitor to be running
  pinMode(INCHPIN,INPUT);
}

void loop() {
  inches = analogRead(INCHPIN);
  feet = inchestoFeet(inches);
  Serial.printf("%i inches equal %0.2f feet \n",inches, feet);
  delay(1000);
}

float inchestoFeet(int measurement) {
  float answer;            // declare answer as a local variable

  answer = measurement / 12.0;
  return answer;
}
```

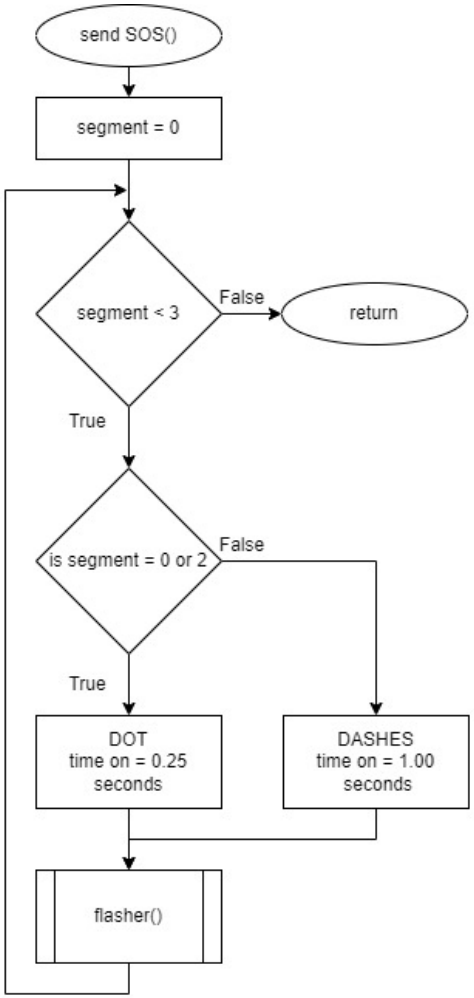
Assignment L03_05_AnalogInput Revisited

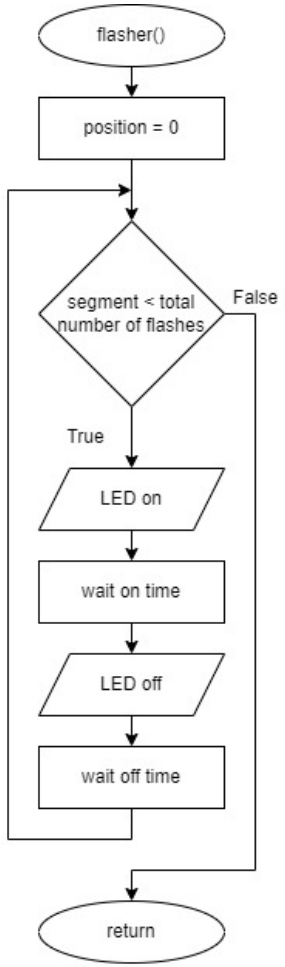


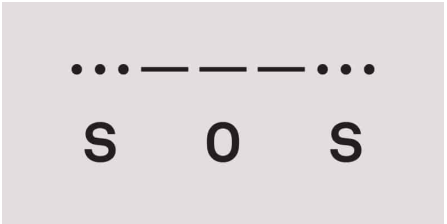
1. Modify your code by adding a function, intoVolts(), that converts the analog input value to voltage.
2. Print both the raw analogInput and the associated voltage to your screen.

Optional Week 1 Review: L03_00_SOS









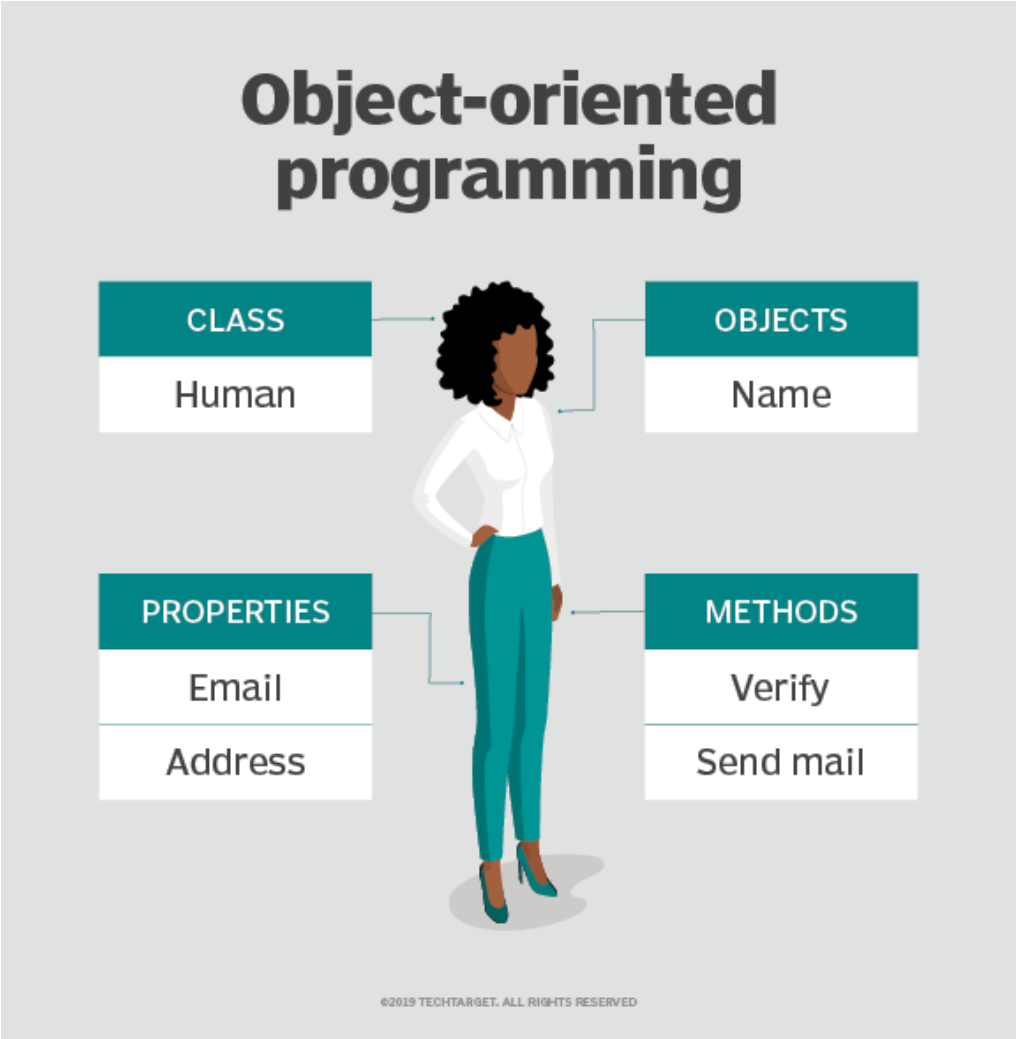
- Flowchart is in english, not code syntax
- Wire one button and one LED to your Teensy
- Declare the appropriate global variables and constants
- First function: void sos
- Second function: void flasher with parameters number of flashes, on time, off time between flashes
- Variables within the functions should be local, not global

L04_oneButton

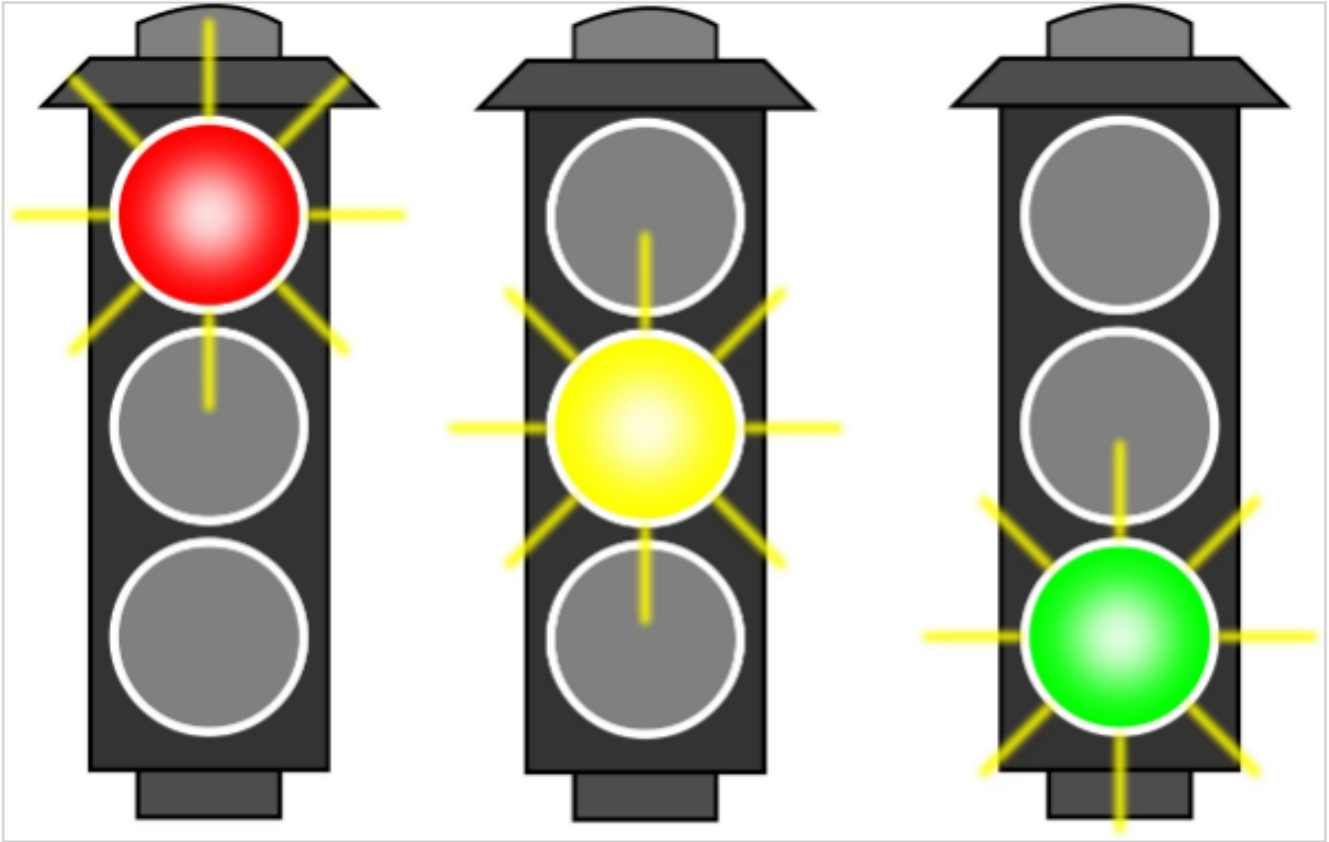
IoT Humor

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

Objects

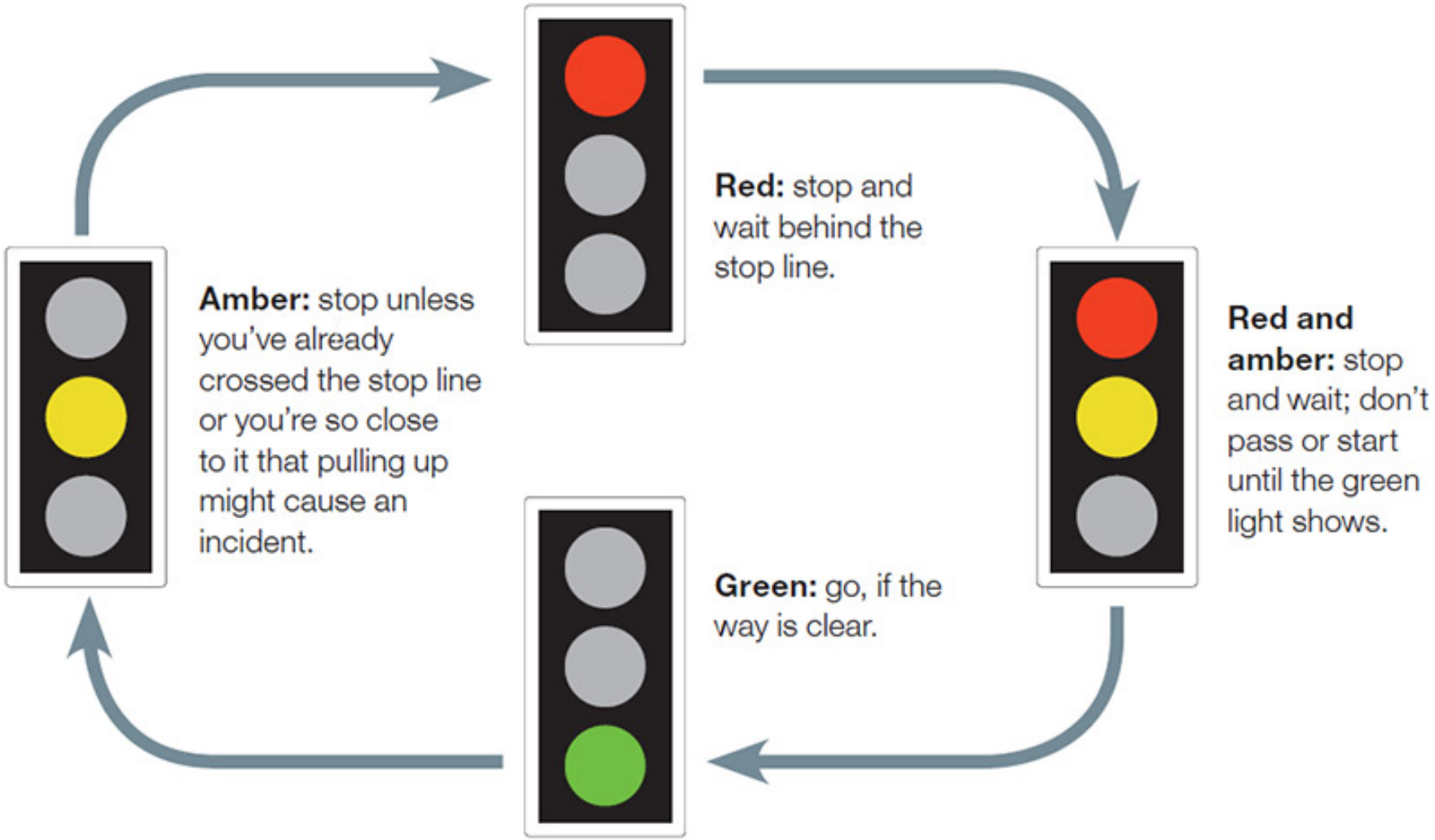

Object-oriented programming
CLASS
Human
OBJECTS
Name
PROPERTIES
Email
Address
METHODS
Verify
Send mail
©2019 TECHTARGET. ALL RIGHTS RESERVED

Traffic Light



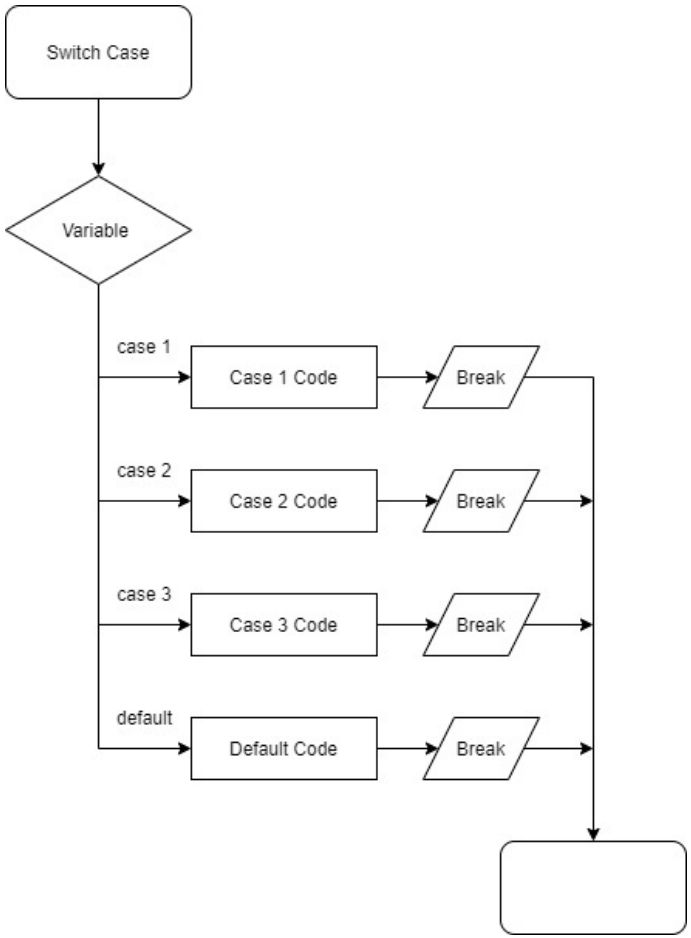
Let's use the traffic light to build our own Objects.

State Machine - Traffic Lights, British Style



SWITCH...CASE syntax - multiple IFs

```
// SWITCH...CASE syntax
switch (variable) {
  case constant1:
    // statements
    break;
  case constant2:
    // statements
    break;
  case constant3:
    // statements
    break;
  default:
    // statements
    break;
}
```



The variable is compared to the constants and the applicable code is called. If variable doesn't match any of the constants, then the default case is called.

Enumeration (enum)

The C-language has a user-defined datatype, enum, which allows the user to create a variable with various names for each of its states.

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

```
1 // Datatype State: the four traffic light states
2 enum State{
3   GREEN,
4   YELLOW,
5   RED,
6   RED_YELLOW
7 };  //note the ; after the }
```

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

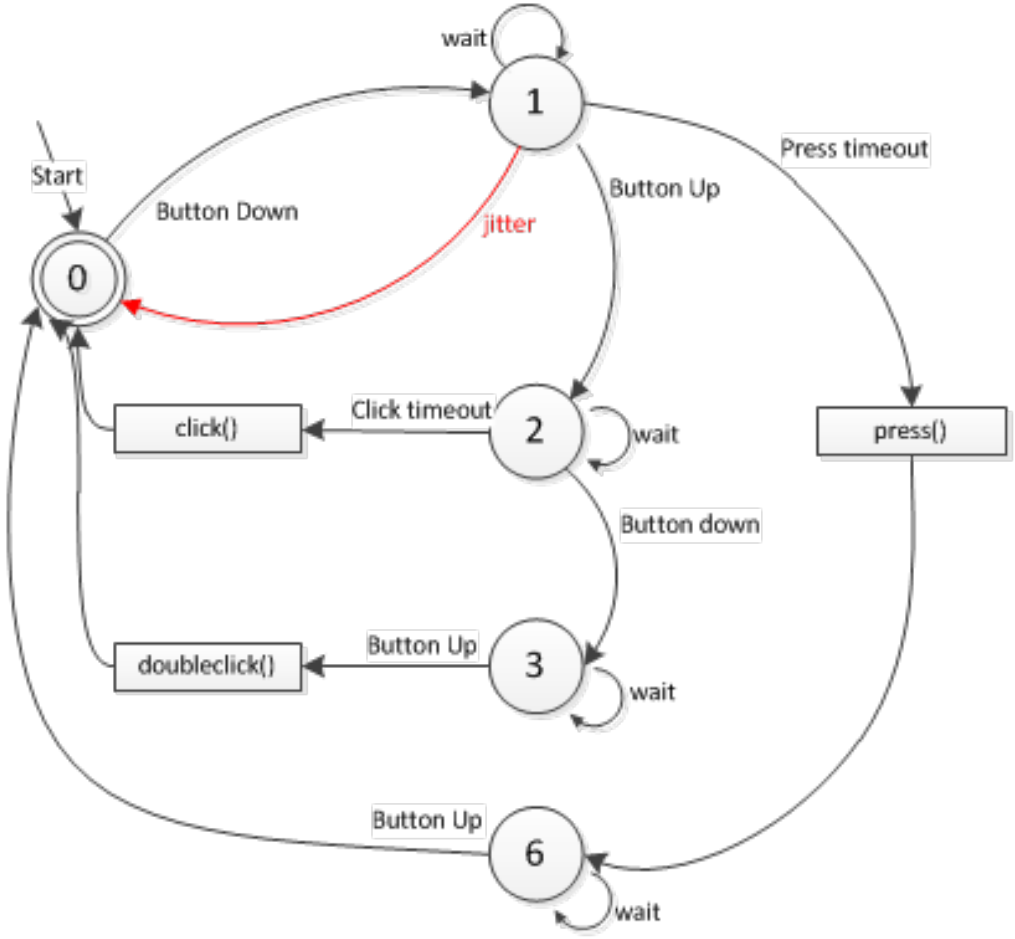
Preprocessor commands

The C-compiler preprocesor executes the # commands before converting the program into code the Teensy can understand.
We have previously used the #include preprocessor command. Now, we will learn a few more:

```
#ifndef _BUTTON_H_   //if not defined, then execute the rest of code

  #define _BUTTON_H_ //define the label

  // Place your Header.h code here

#endif // _BUTTON_H_
```

- #ifndef - if the label is not defined, then execute code until #endif
- #define - define the label
  - the #define label can be set to a value
  - the compiler then replaces the label with the value where ever it sees it meaning it is the similar to const <datatype> = <value>.
  - NOTE: we will use the CONSTANT and not the #define

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

```
1  // the "header" is used for GLOBALS
2  #include <OneButton.h>              // system library files
3  #include "TrafficLight.h"           // local library files
4
5  OneButton button1(23, false);     // define objects
6
7  const int GREENLED = 5;           // declare global constants
8  bool greenState;                   // declare global variables
9
10 // setup() is used for code that runs once at the beginning
11 void setup() {
12   Serial.begin(9600);              // begin processes
13   pinMode(GREENLED,OUTPUT);        // define input/output modes
14   button1.attachClick(click);      // initialize objects
15   greenState = false;              // set variables
16 }
17
18 // loop() is used for code that runs continuously
19 void loop() {
20   button1.tick();
21   digitalWrite(GREENLED,greenState);
22 }
23
24 // user defined functions
25 void click() {
26   greenState = !greenState;
27 }
```

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);
}
```

OneButton parameters (not variables):

- pin (int): the pin the button is connected to.
- activeLOW (bool): Button wire: false = pull down, true = pull up
- pullUP (bool): pinMode: false = INPUT, true = INPUT_PULLUP

Using OneButton

```
1
2  void loop() {
3    button1.tick();// check the state of the button
4  }
5
6  void click1() {
7    Serial.printf("Hi, my name is Brian.\n");
8  }
9
10 void doubleClick1() {
11   Serial.printf("Hope your day is well.\n");
12 }
```

Assignment: OneButton

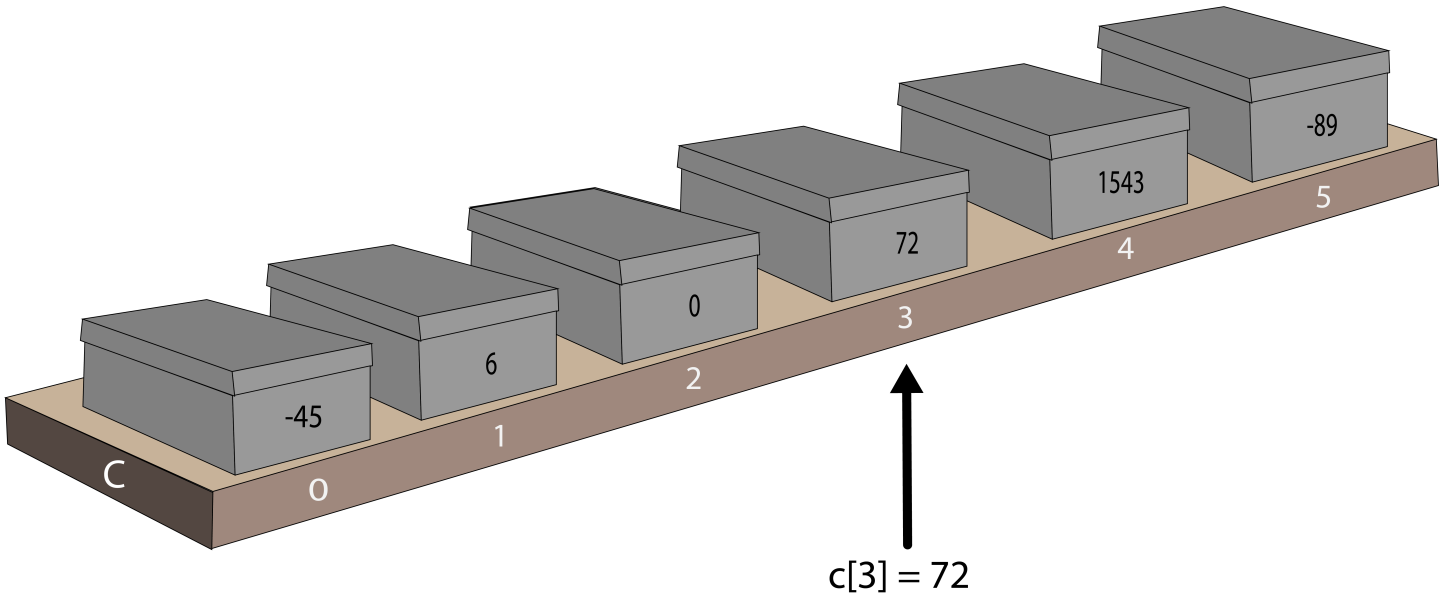
- Notebook: flowchart
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

- L04_01_oneButton
  1. Use OneButton libary and button on Pin 23.
  2. Click() - toggle bool variable buttonState.
  3. doubleClick() - toggle bool variable blinker.
  4. Print both variables to your Serial.Monitor.
- L04_02_oneButtonLED - Without changing the Click() and doubleClick() functions:
  1. Single Click: toggle LED on/off using buttonState variable.
  2. Double Click: when LED is on, toggle between solid and rapidly blinking (50ms) using blinker variable and a delay() like L02_01_HelloLED.
  3. What happens if blinking is made slower (try $>$ 1 second)?
  4. Modify the second bullet to work without using delay().

Avoiding Delays

```
1  void loop() {
2    //run constantly
3    currentTime = millis();
4
5    //run once per second
6    if((currentTime-lastSecond)>1000) {
7      Serial.printf(".");
8      lastSecond = millis();
9    }
10
11   //run once per minute
12   if((currentTime-lastMinute)>60000) {
13     Serial.printf("\nMinute\n");
14     lastMinute = millis();
15   }
```

Arrays



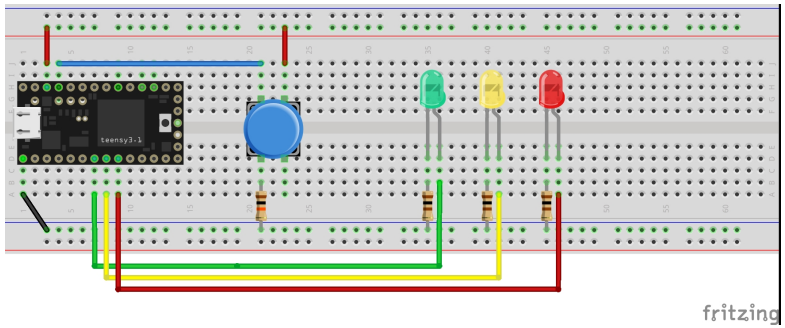
- Syntax: datatype var[ ] = {element 1, element 2, element 3};
- Example: int c[ ] = {-45, 6, 0, 72, 1543, -89}

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.printf("Pin %i selected \n",myPins[i])
   ;
  }
}
```

Note: An array index starts at 0 (not 1).

Assignment: OneButton



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

1. L04_03_oneButtonArray
   - Use 3 LEDs and one Button.
   - Single Click - toggle current LED on/off.
   - Double Click - using an array, select the next LED.
   - Long Press (start) - light up the three LEDs in sequence with a one second delay between.
   - Long Press (end) - turn off the three LEDs in reverse order with a delay between.

String datatype and Serial Read

- We can enter input via the Serial Monitor using the String class.
- The String class acts like a datatype.
- And, it also allows methods, such as toInt().

```
1  String inputString;
2  int value;
3
4  void setup() {
5    Serial.begin(9600);
6    while(!Serial);
7  }
8
9  void loop() {
10   inputString = "";                                  // clear the variable inputString
11   while(inputString=="") {                           // wait for input from Serial Monitor
12     inputString = Serial.readStringUntil('\n'); // get input when enter is pushed
13   }
14   value = inputString.toInt();                       // convert String to integer
15   Serial.printf("The number you entered is %i \n",value);
16 }
```

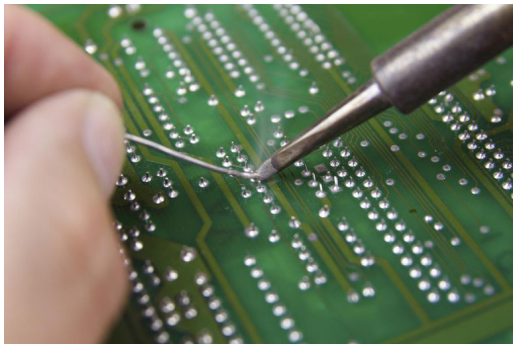
Assignment: Serial Read

L04_04_timer - Create a Stop Watch and Countdown Timer

1. Use OneButton:
   - Single click - start and stop
   - Double click - switch between stop watch and timer
2. Functionality:
   - In Stop Watch mode, start counting up from zero.
   - In Timer mode, prompt the user on the Serial Monitor to enter the time to countdown from.
   - Display time in seconds to the serial monitor each second.
   - For Stop Watch, display time to 3 decimal points when stop is pressed.
   - For Timer, display DONE when zero is reached.

L05_NeoPixels

Soldering



Soldering is one of the most fundamental skills needed to construct IoT devices.

- Solder the Noun: the alloy (a substance composed of two or more metals) that typically comes as a long, thin wire in spools or tubes
- Solder the Verb: to join together two pieces of metal in what is called a solder joint.

So, we solder with solder!

Soldering

- Lead vs Lead-free: Traditionally, solder was composed of mostly lead (Pb), tin (Sn), and a few other trace metals. However, lead is hazardous in large quantities.
  - Lead solder has superior properties - lower melting point, flows well, less internal flaws after cooling.
  - Lead-free solder has a higher melting point and thus needs assistance to flow. Many have flux core, a chemical that aids in the flowing.
- Solder flux: Flux is a chemical cleaning agent used before and during the soldering process of electronic components. The flux also protects the metal surfaces from re-oxidation during soldering and helps the soldering process by altering the surface tension of the molten solder.

Solder Irons

- Solder tips - the tip transfers heat, raising the temperature of the metal components to the melting point of the solder. They come in a variety of shapes and sizes.
- Wand - the wand holds the tip and may have a separate base. The wand controls the temperature of the tip. The temperature range depends on the type of solder.
  - Lead solder - 600°-650°F (316°-343°C)
  - Lead-free solder - 650°-700°F (343°-371°C)
- Brass Sponge - During soldering the tip will start to oxidize, it will change color and less readily accept solder. The brass sponge will reduce buildup. Alternatively, a wet sponge can be used.

Solder Tip Care

CLEANING YOUR TIPS

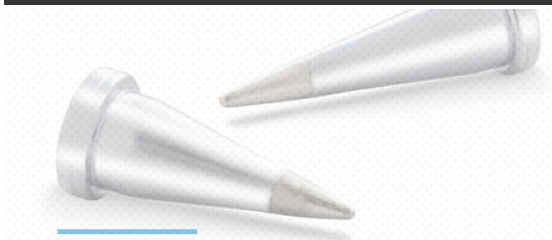
To clean your tips, use either **brass or stainless steel wool.** Brass wool is softer and less abrasive, while the harder stainless steel wool has a longer life. After cleaning, immediately wet the tip with fresh solder to **prevent oxidation.**

TINNING YOUR TIPS

Tinning stops your tips from oxidizing by creating a **protective layer** between the air and the iron. Preventing oxidation through tinning **extends the life** of your tips.

OXIDIZED TIP

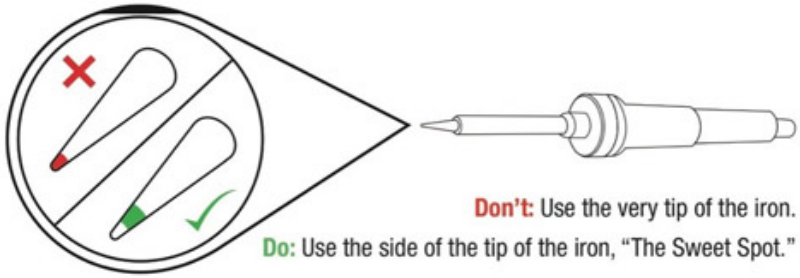
TINNED TIP



Properly Storing Tips

If storing your tips for an extended period, you should **clean and tin them before putting them away,** which will help prevent them from oxidation. After letting them cool, you may also want to **store them in a sealed container** to further **protect them** from oxidation, humidity and contamination.

Good vs Bad Solder Joints



Don't: Use the very tip of the iron.

Do: Use the side of the tip of the iron, "The Sweet Spot."

Do: Touch the iron to the component leg and metal ring at the same time.

Do: While continuing to hold the iron in contact with the leg and metal ring, feed solder into the joint.

Don't: Glob the solder straight onto the iron and try to apply the solder with the iron.

Do: Use a sponge to clean your iron whenever black oxidization builds up on the tip.

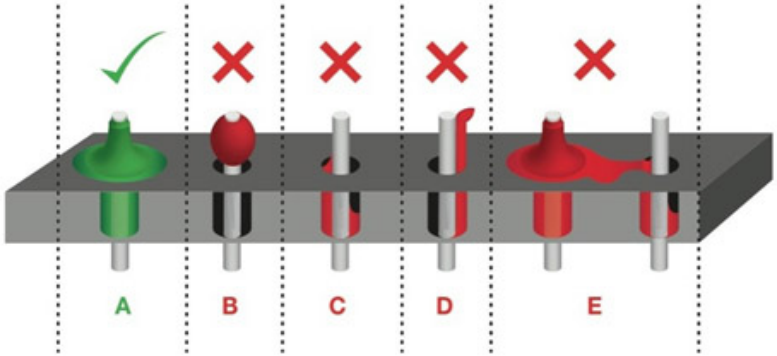
A Solder flows around the leg and fills the hole - forming a volcano-shaped mound of solder.

B Error: Solder balls up on the leg, not connecting the leg to the metal ring.
Solution: Add flux, then touch up with iron.

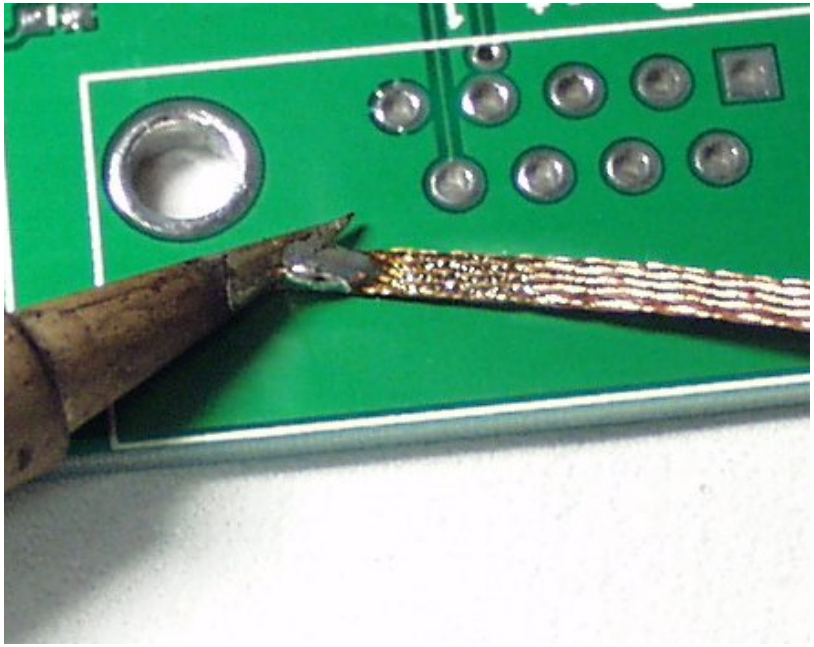
C Error: Bad Connection (i.e. it doesn't look like a volcano)
Solution: Flux then add solder.

D Error: Bad Connection...and ugly...oh so ugly.
Solution: Flux then add solder.

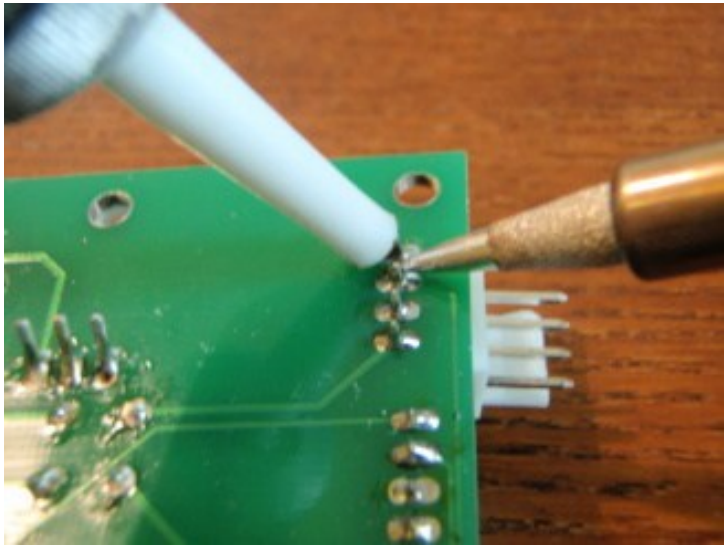
E Error: Too much solder connecting adjacent legs (aka a solder jumper).
Solution: Wick off excess solder.



Desoldering



Solder Wick



Desoldering Pump

Generating Random Numbers

```
/*
 * The random() function generates pseudo-random
   numbers.
 *     random(min,max)
 *     random(max)     //assumes min = 0
 * returns a number between min and max-1
 */

  // print a random number from 0 to 299
  randNumber = random(300);
  Serial.printf("The number is = %i \n",randNumber);

  // print a random number from 10 to 19
  randNumber = random(10, 20);
  Serial.printf("The number is = %i \n",randNumber);
```

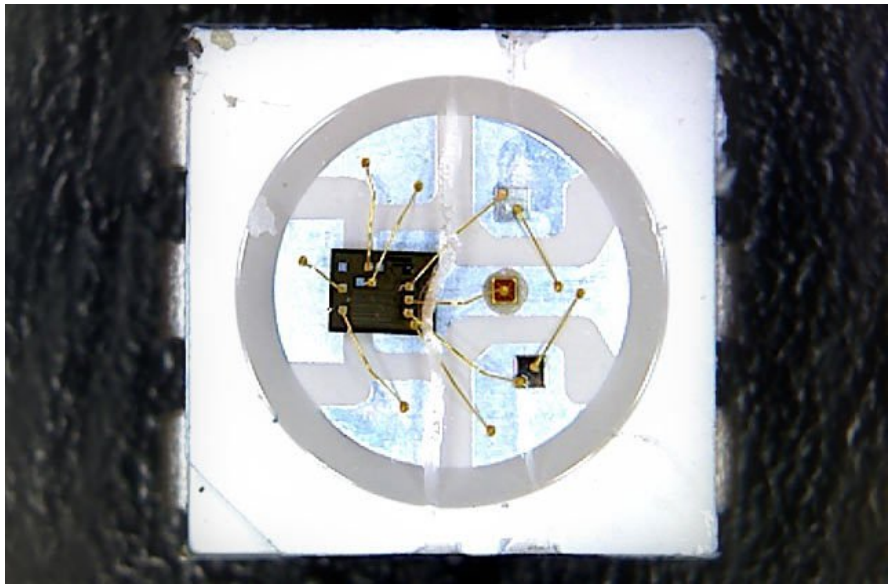
Nested Statements

- Loops and conditions can be nested within each other
- A nested loop is a loop within a loop, an inner loop within the body of an outer one. How this works is that the first pass of the outer loop triggers the inner loop, which executes to completion. Then the second pass of the outer loop triggers the inner loop again. This repeats until the outer loop finishes.
- The break command can be used to exit a loop before completion

Nested For Loops

```
int tens;
int ones;

// Nested FOR Loops
for(tens=0;tens<10;tens++) {
  for(ones=0;ones<10;ones++) {
    Serial.printf("Combining %i(tens) and %i(ones) gives %i%i\n",tens,ones,tens,ones);
  }
}

// An IF nested in Nested FOR Loops
for(tens=0;tens<10;tens++) {
  for(ones=0;ones<10;ones++) {
    if(tens != ones) {
      Serial.printf("Combining %i(tens) and %i(ones) gives %i%i\n",tens,ones,tens,ones);
    }
  }
}

// Using break to break out of a loop
for(tens=0;tens<10;tens++) {
  for(ones=0;ones<10;ones++) {
    Serial.printf("Combining %i(tens) and %i(ones) gives %i%i\n",tens,ones,tens,ones);
    if((tens+ones) = 10) {
      break;
    }
  }
}
```

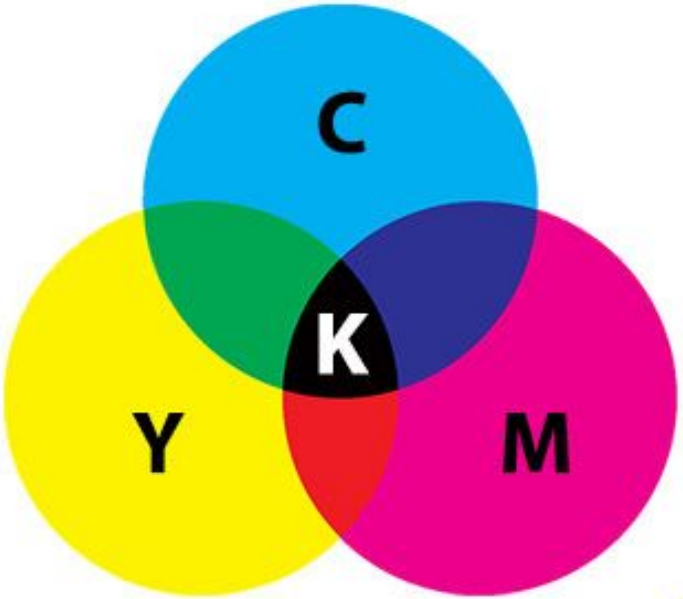
NeoPixels



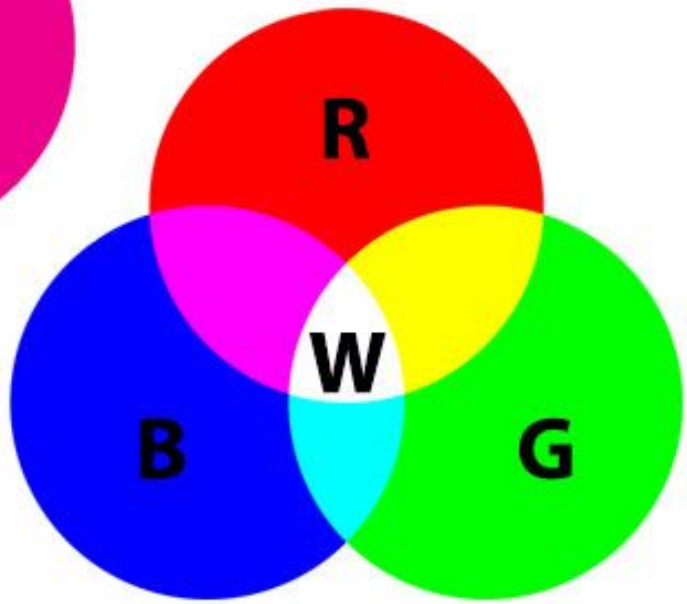
NeoPixels are:

- 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.

CYMK vs RGB Colors

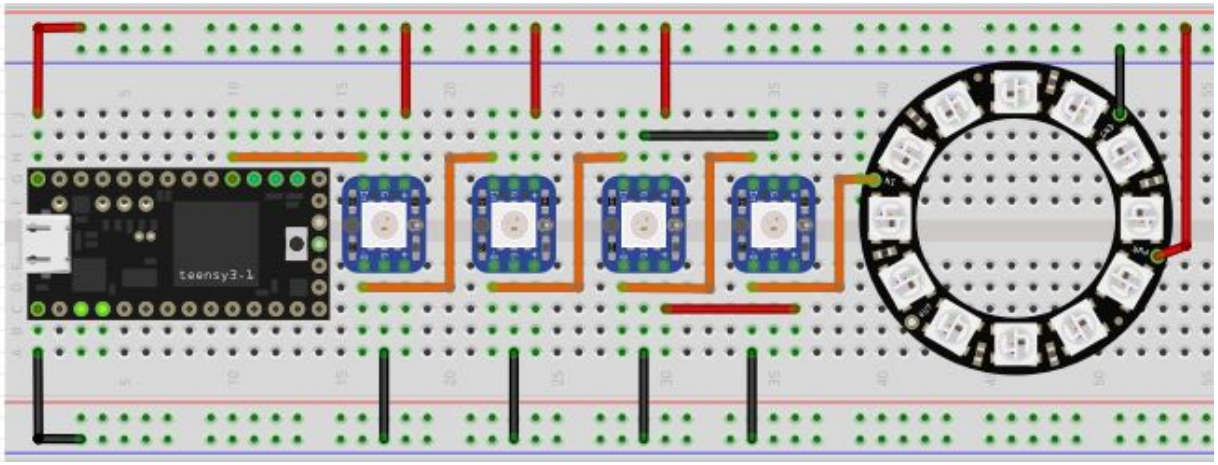


CMYK / Print
(Subtractive Colors)

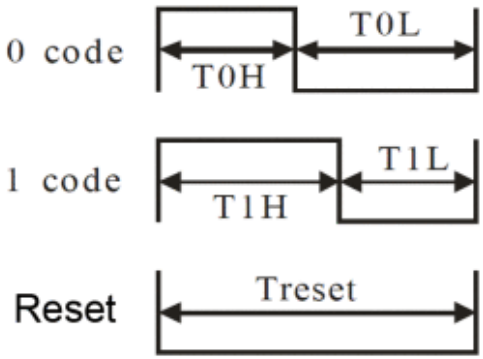


RGB / Screen
(Additive Colors)

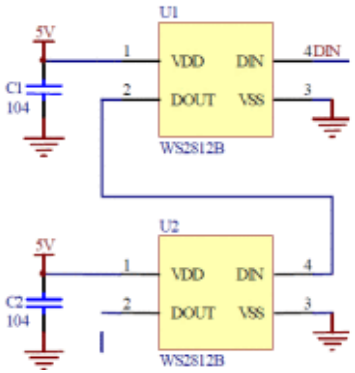
NeoPixel Programming



WS2812 Protocol



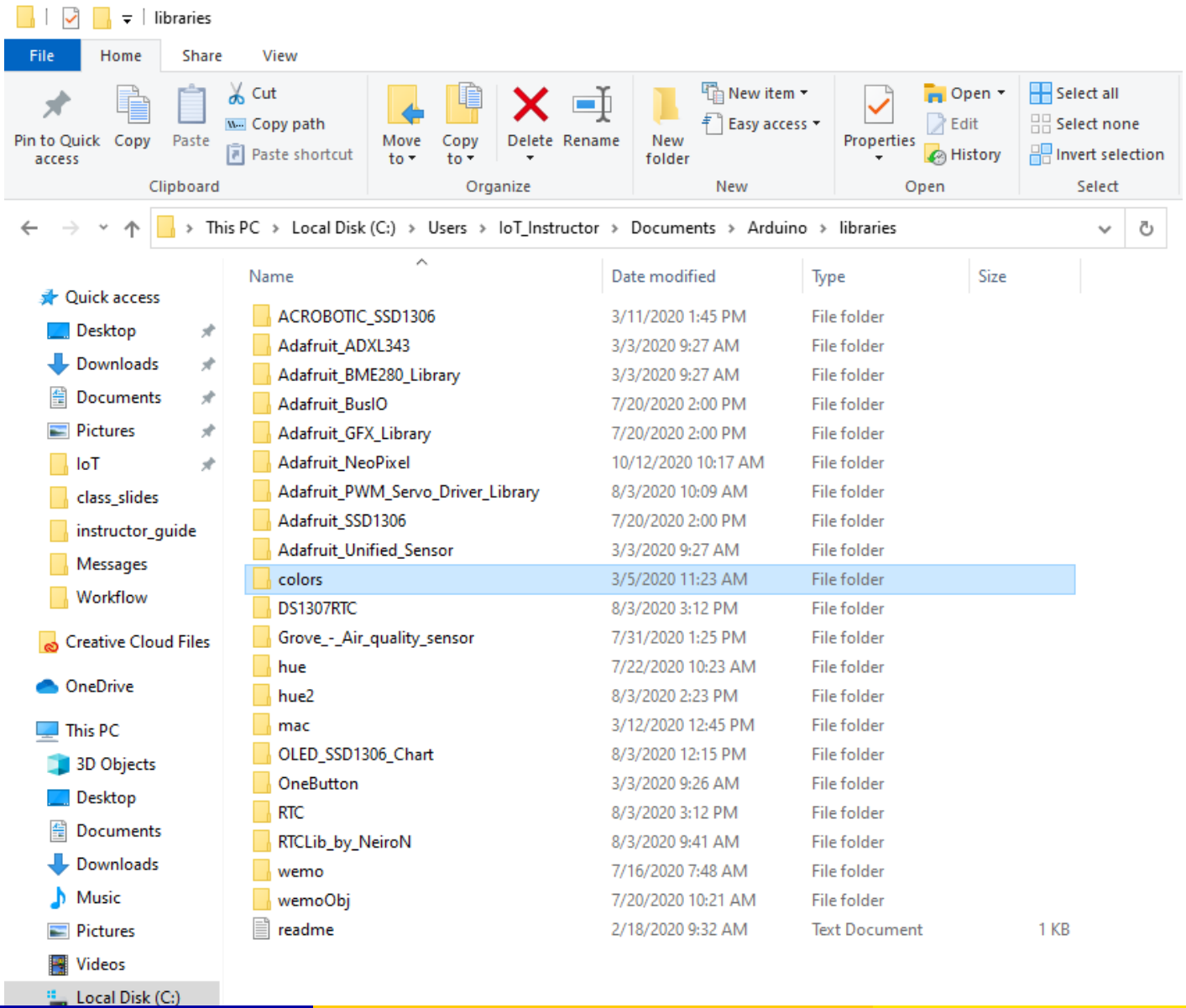
LED-Chain



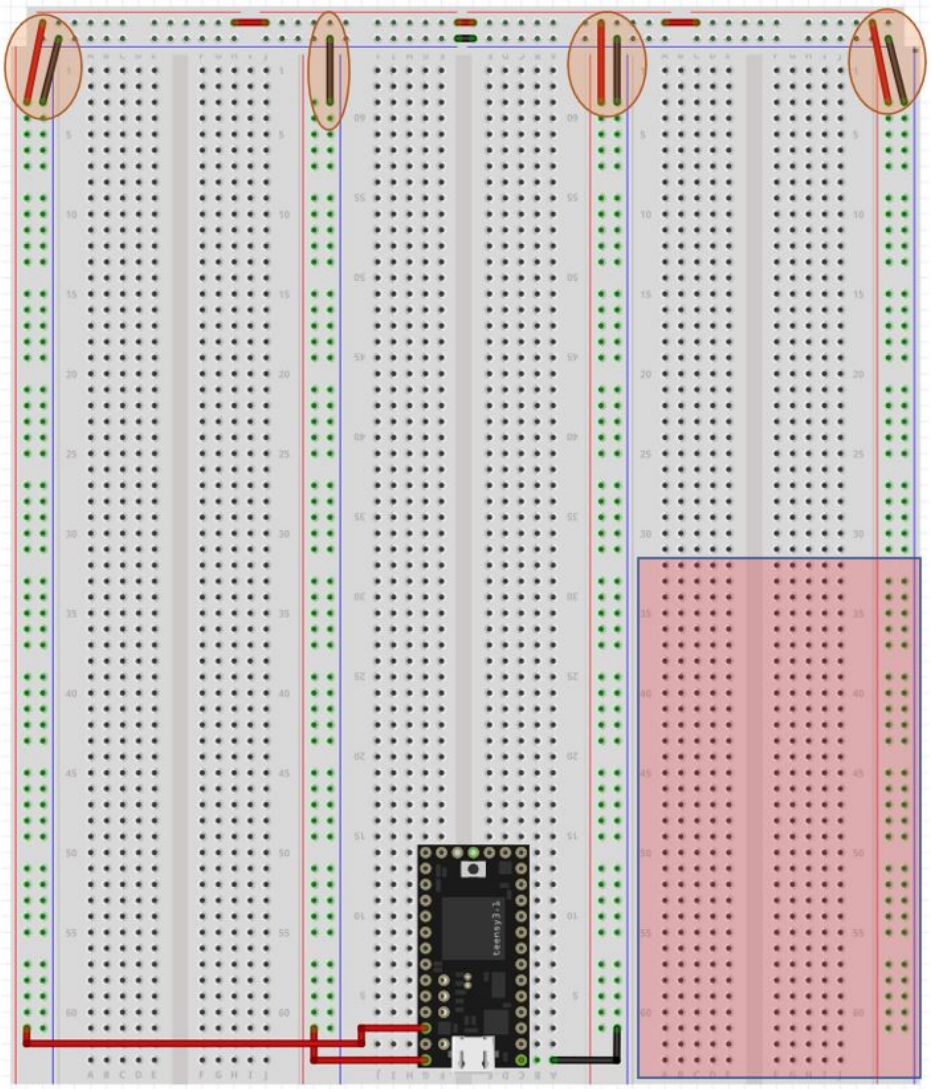
Using NeoPixel Class and Methods

```
#include <Adafruit_NeoPixel.h>

const int PIXELPIN = 17;    // Pin the NeoPixels are connected to
const int PIXELCOUNT = 16;  // Total number of NeoPixels

Adafruit_NeoPixel pixel(PIXELCOUNT, PIXELPIN, NEO_GRB + NEO_KHZ800); //declare object
/* Argument 1 = Number of pixels
 * Argument 2 = GPIO pin number
 * Argument 3 = Pixel type flags, add together:
 * Use:
 *    NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
 *    NEO_KHZ800  800 KHz bitstream (WS2812)
 *    Other options for Argument 3:
 *    NEO_KHZ400  400 KHz (WS2811)
 *    NEO_RGB     Pixels are wired for RGB bitstream (v1)
 *    NEO_RGBW    Pixels are wired for RGBW bitstream
 */
void setup() {
  pixel.begin();
  pixel.show(); //initialize all off
}

void loop() {                                  //n is the pixel number being set
  pixel.setPixelColor(n, red, green, blue); //red,green,blue = 0 - 255
  pixel.setPixelColor(n, color);            //hex code 0x000000 - 0xFFFFFF
  pixel.fill(color, first, count);
  pixel.setBrightness(bri);                 // 0 - 255
  pixel.show();                             //nothing changes until show()
  pixel.clear();
  pixel.show();                             //even clear() needs a show()
}
```

Where do global header files go?



Move to Big Breadboard

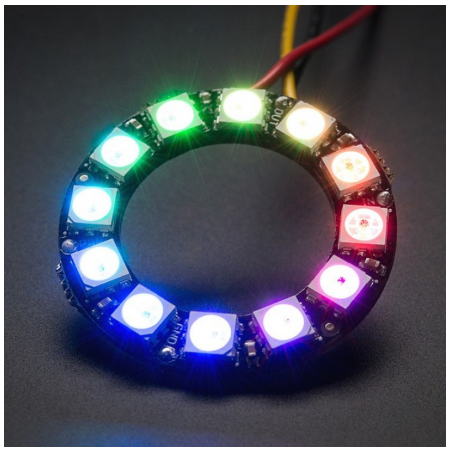


Connect the 3.3V and ground rails together

Leave the 5V rail for NeoPixels

Reserve this space for SPI components

Assignment: NeoPixels



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

1. L05_01_neoPixel
   - Using FOR loop and individual R/G/B, light up 16 pixels; small delay between.
2. L05_02_colorHeader
   - Implement a header file that contains the pixel colors.
3. L05_03_neoStrip, use setPixelColor() to implement functions:
   - 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 7 segments of different colors using the fill() method.

RandomSeed()

- The "pseudo" in pseudo-random indicates it is not truly random.
- randomSeed() initializes the pseudo-random number generator causing it to start at an arbitrary point in its random sequence. This sequence, while very long, and while appearing random, is always the same.
- If it is important for a sequence of values generated by random() to differ, on subsequent executions, use randomSeed() to initialize the random number generator with a fairly random input, such as an analogRead() on an unconnected pin.

```
1  // Leave an Analog Input (A0) floating
2  pinMode(A0,INPUT);
3  randomSeed(analogRead(A0));
4
5  // print a random number hex color value
6  randNumber = random(0x0000,0xFFFFFF);
7  Serial.printf("My color is 0x%06X \n",randNumber);
```

L06_Encoders

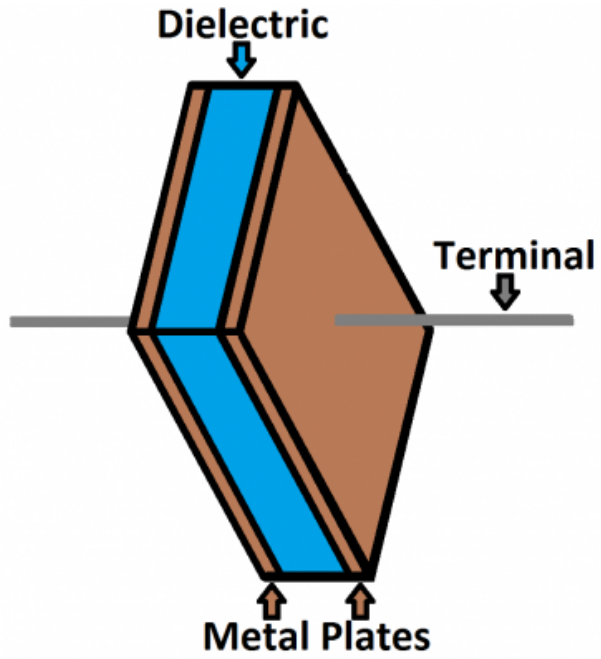
IoT Humor


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

Capacitors

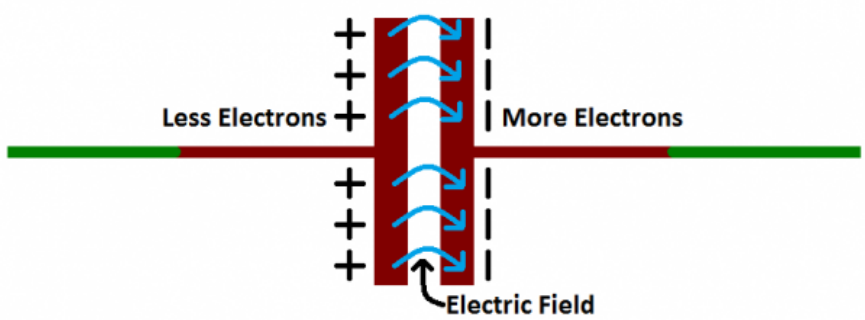
A capacitor is created out of two metal plates and an insulating material called a dielectric. The metal plates are placed very close to each other, in parallel, but the dielectric sits between them to make sure they don't touch.

- 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.



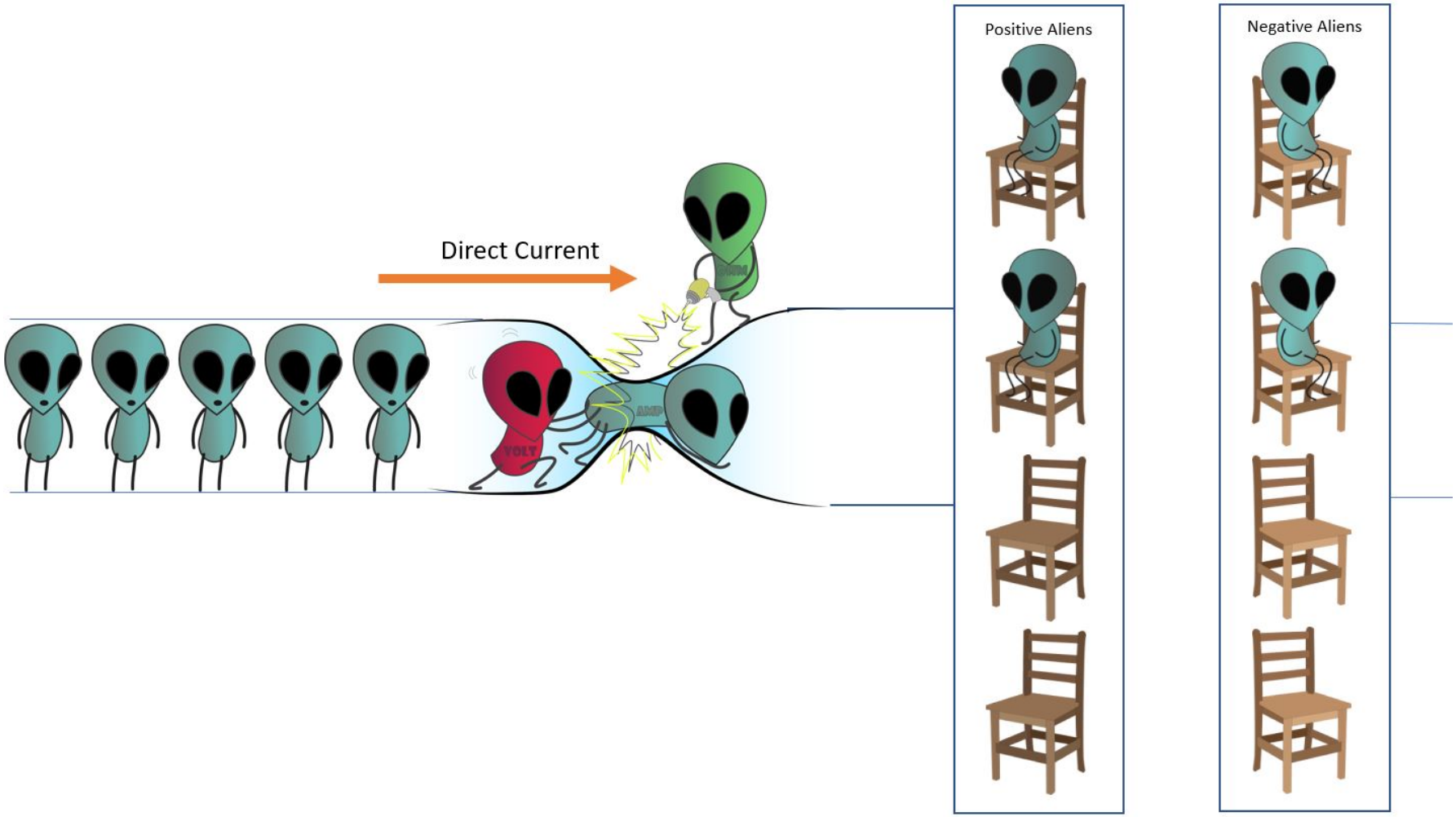
Capacitors

When current flows into a capacitor, the charges get "stuck" on the plates because they cannot 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.

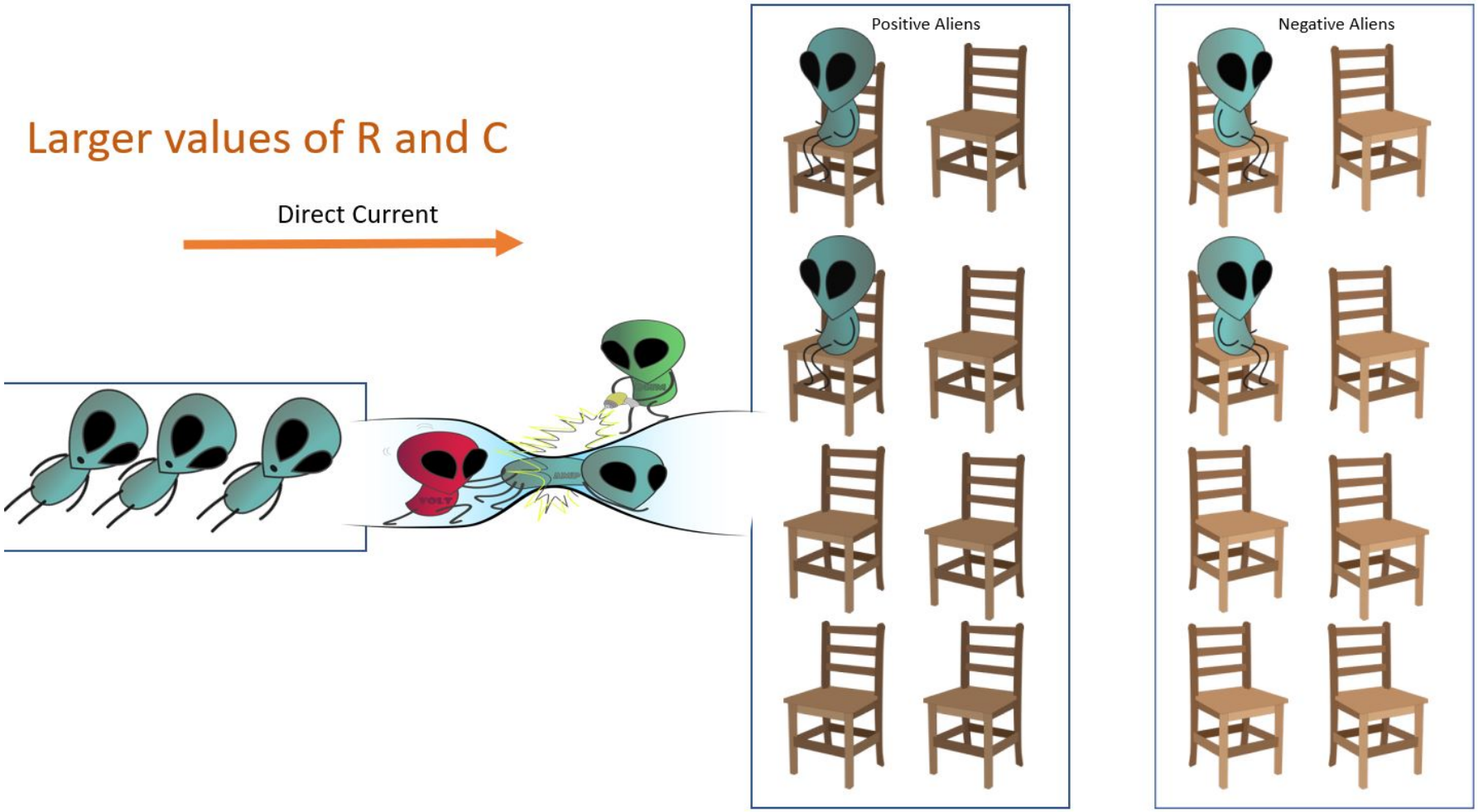


The stationary charges on these plates create an electric field, which influences electric potential energy and voltage. When charges group together on a capacitor like this, the capacitor is storing electric energy just as a battery might store chemical energy.

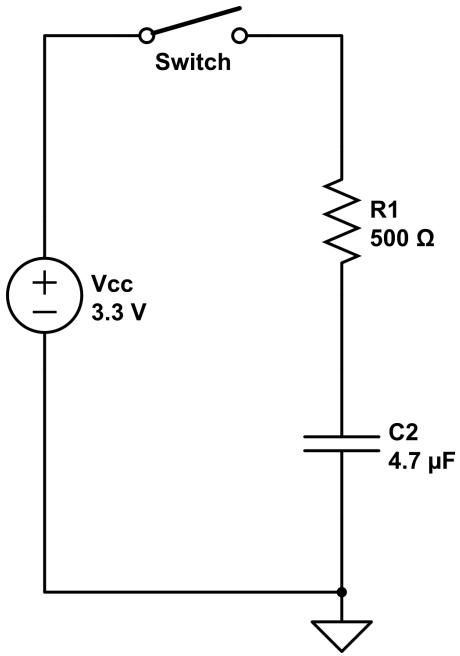
Capacitors in Circuits



Capacitors in Circuits



RC Time Constant



$$V_c(t) = V_c(0) * e^{\frac{t}{-\tau}}$$

Capacitance is defined as:

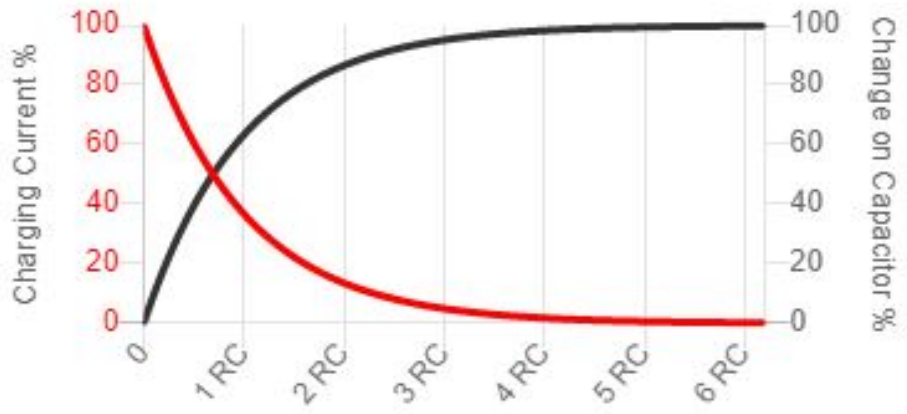
$$C = \frac{Q}{V}(\frac{Coulombs}{Volt})$$

The current through a capacitor is:

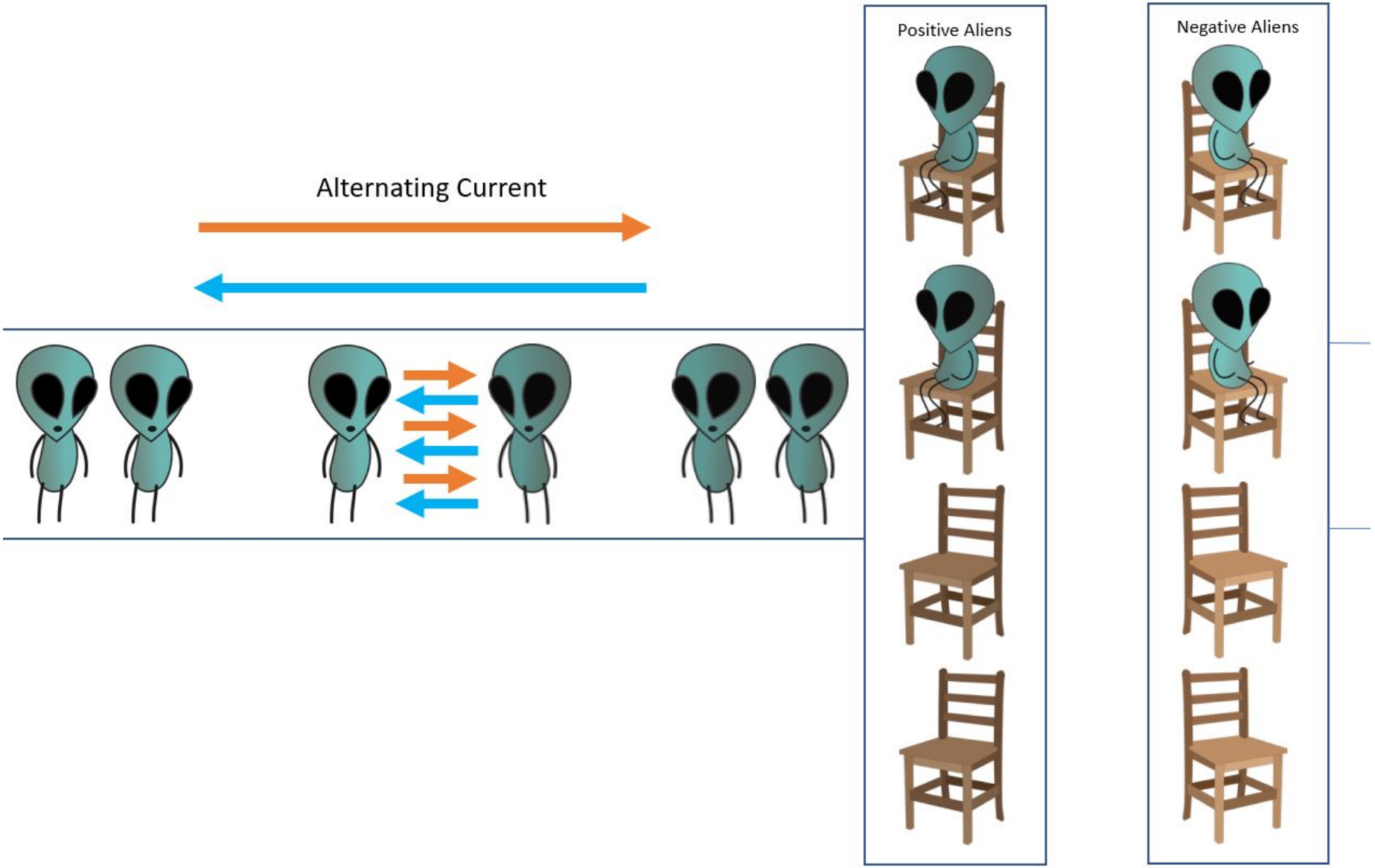
$$I = C\frac{\Delta V}{\Delta t}$$

And, therefore, the capacitor charges with a time constant (τ):

$$\tau = RC$$

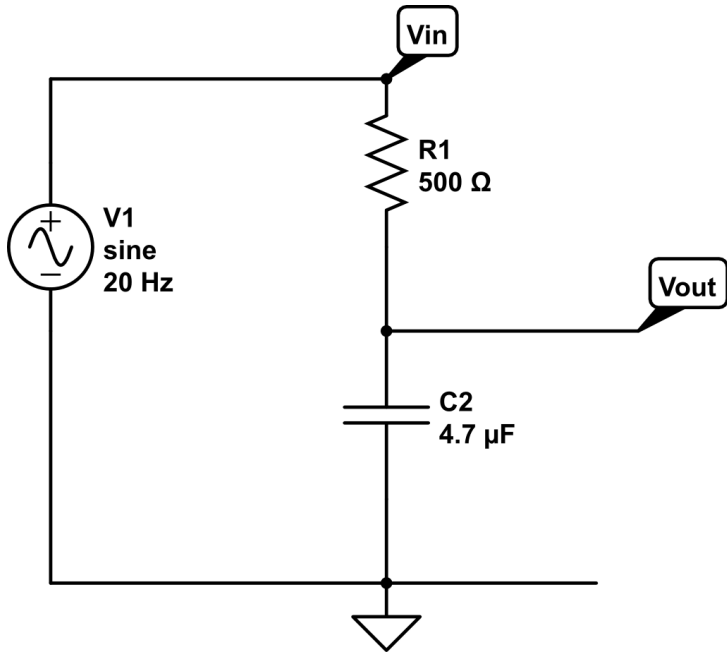


Alternating Current



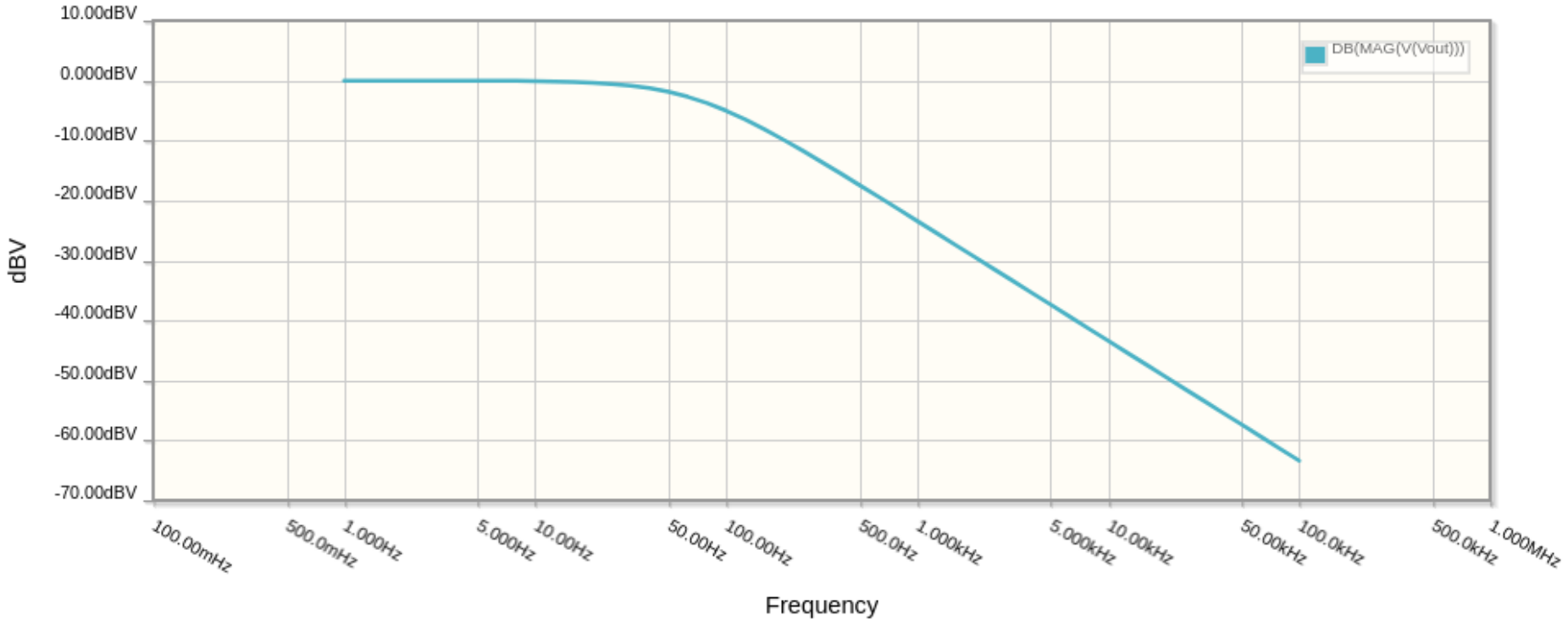
Low Pass Filter - cutoff frequency f_c

- 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.



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

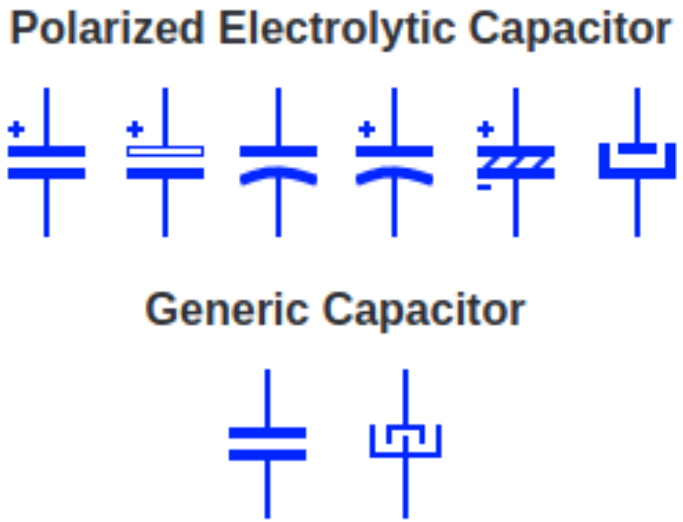
Low Pass Filter Response



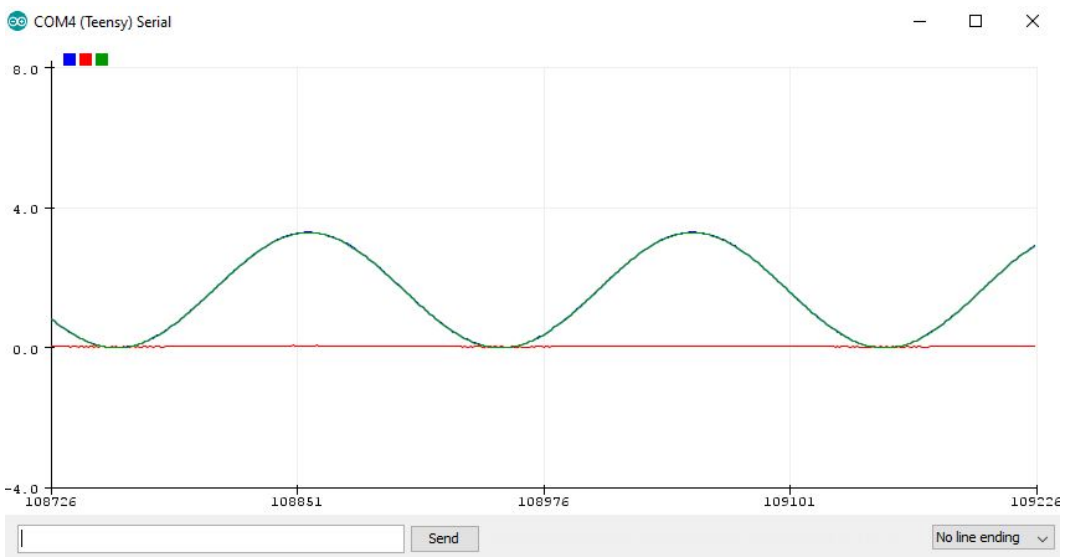
$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi(500)(4.7x10^{-6})} = 67.5678Hz$$

Capacitors - does it matter how they are placed

- Some types of capacitors (electrolytic and tantalum) are polarized (they 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.



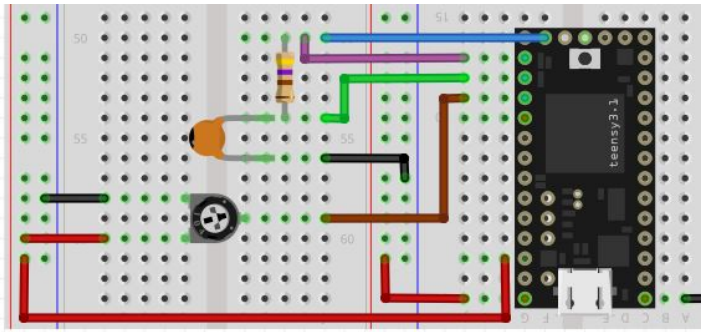
Serial Plotter



The Serial Plotter can give you visualizations of variables in real-time. This is very useful for visualizing data, troubleshooting your code, and visualizing your variables as waveforms.
Data is plotted using Serial.printf() with the data separated by commas. For example:

- Serial.printf("%i , %i, %0.3f \n",data1,data2,data3);

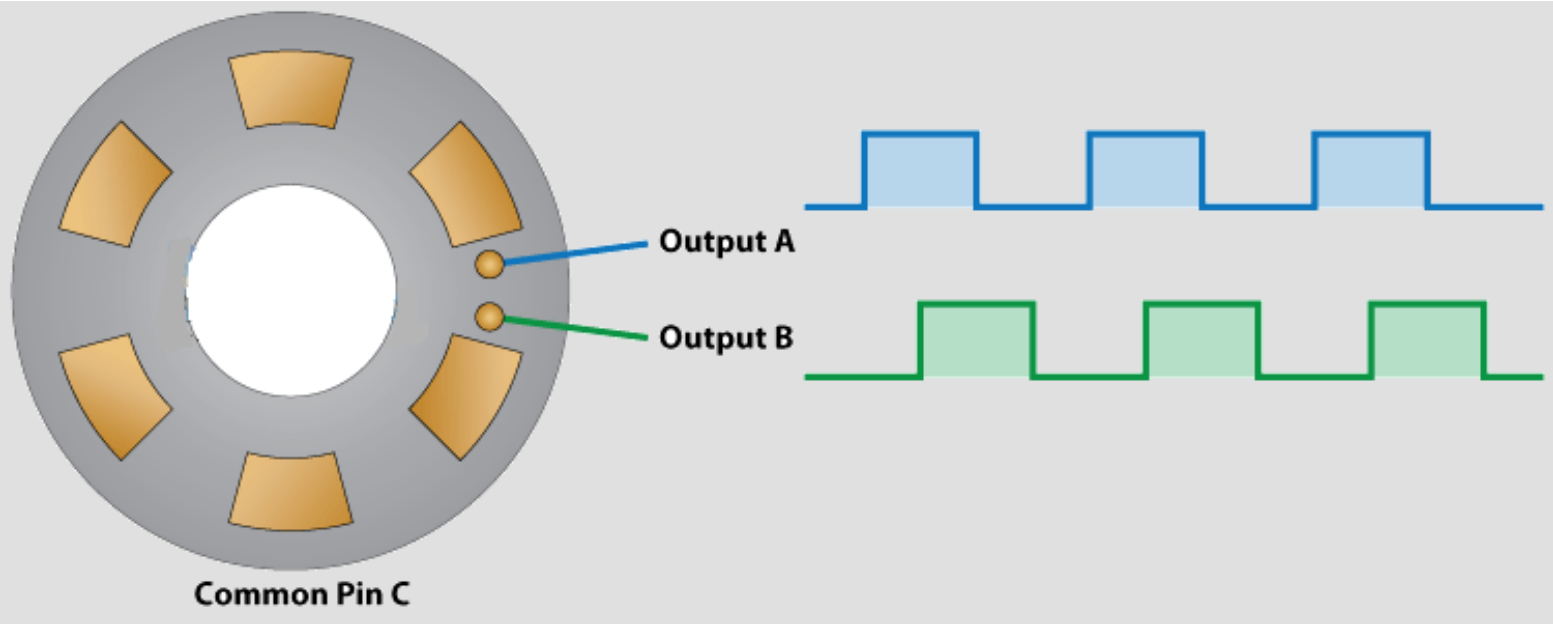
Assignment: Low Pass Filters



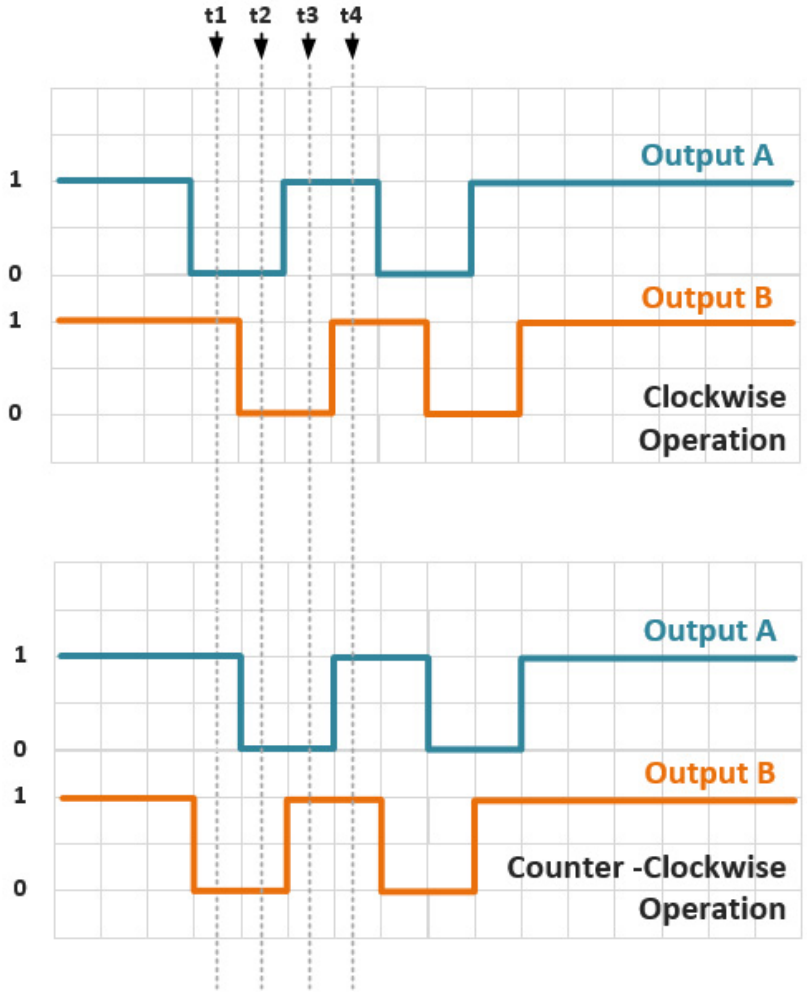
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L06_00_lowPass
   - Connect a potentiometer to an Analog Input (brown wire). Use it to vary a frequency (ν) from 0 to 500 Hz.
   - Create code that generates a sine wave of frequency ν: $sin(2\pi\nu t)$
   - Create a low pass filter with $f_c \approx 67Hz$
   - Connect the DAC (digital to analog) output A14 (blue wire) to the input to the filter and to an Analog Input (purple wire).
   - Connect the output of the filter to another Analog Input (green wire).
   - Use the Serial Plotter to view the frequency for both inputs.
   - Record in your notebook how the signals change when ν is varied.

Encoders



Encoders



Clockwise Sequence

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

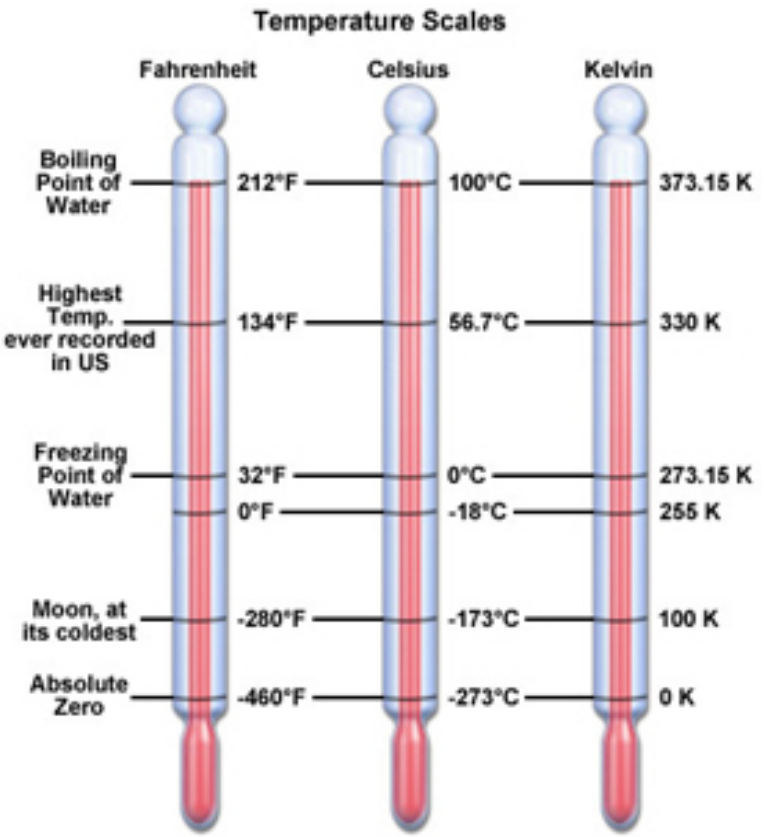
Counter-Clockwise Sequence

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

The Encoder Class

```
#include <Encoder.h>
Encoder myEnc(PINA,PINB);
// The "c" pin on the encoder is connected to GND

void setup() {
}

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

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

Mapping (or Converting)



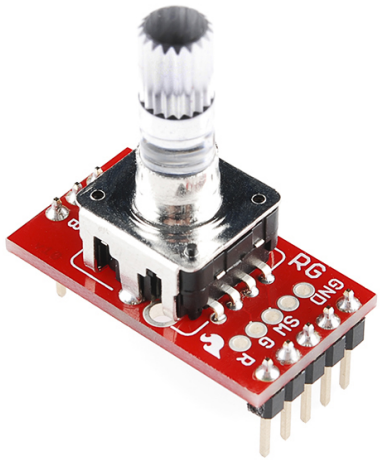
Mapping is the conversion from one set of units to another. For example converting from Celsius to Fahrenheit:

$$Temp(^\circ F) = \frac{9}{5} * Temp(^\circ C) + 32$$

C++ provides us with a function to do this mapping:
newVal = map(value, fromLow, fromHigh, toLow, toHigh);

For example:
tempF = map(tempC,0,100,32,212);

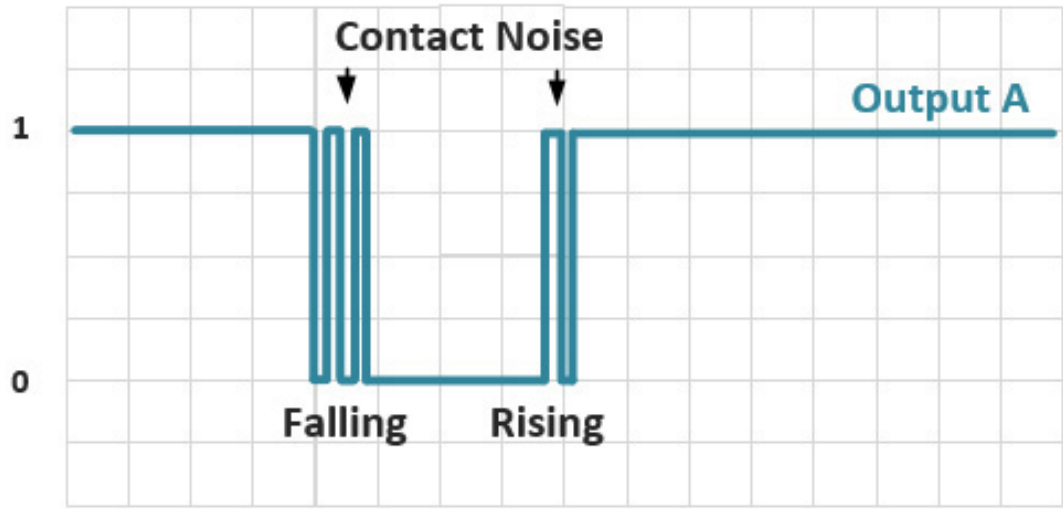
Assignment: Encoders



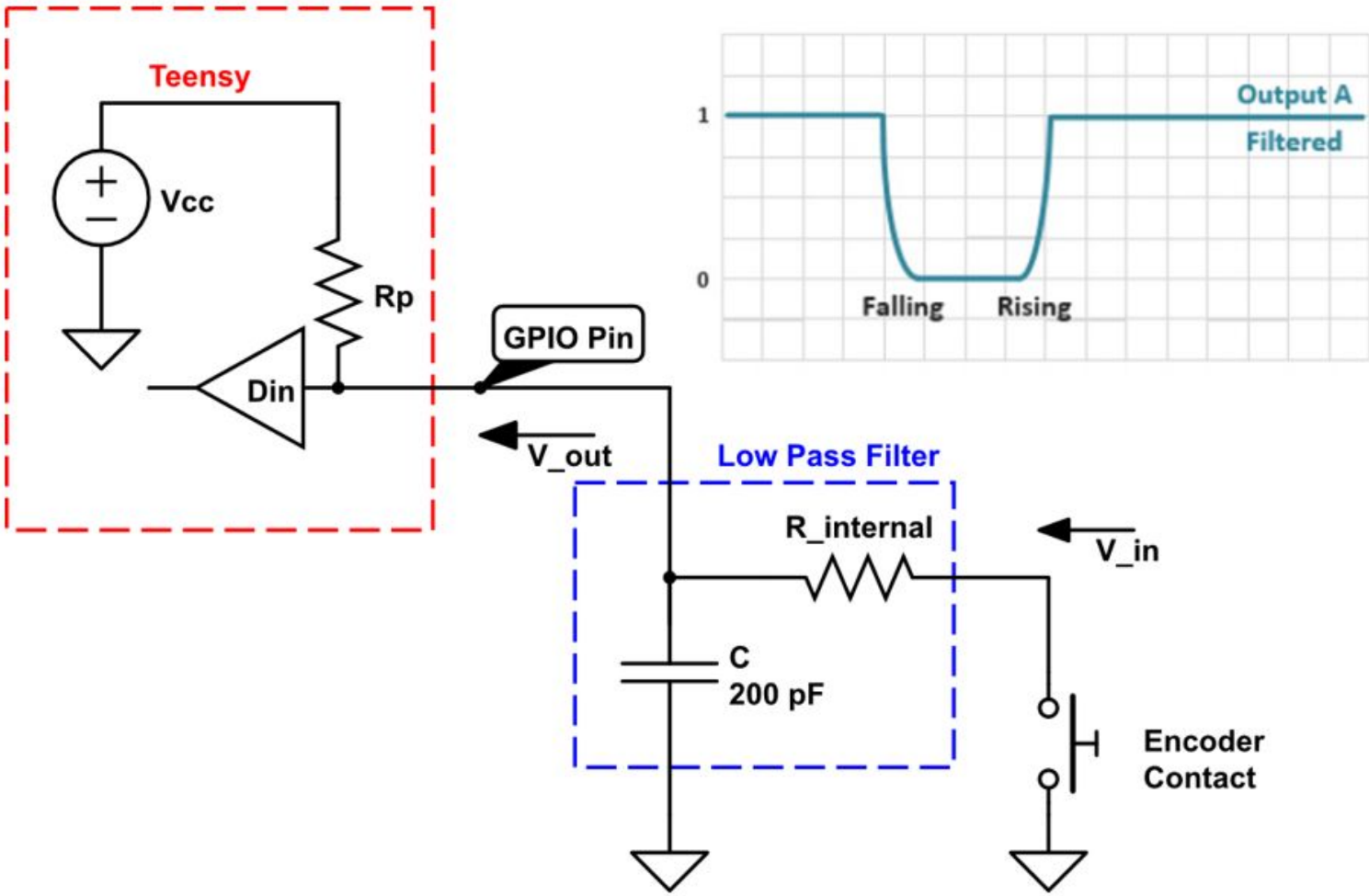
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L06_01_encoder
   - Display the encoder position to the screen, but only when encoder is moved.
   - What do you notice about the encoder position as you turn it slowly?
2. L06_02_NeoPixel
   - The encoder has 96 positions. Map the encoder input to 16 NeoPixels.
   - Bound the input, so the mapped range is from 0 to 15.
   - What is the difference in performance if you bind the input vs the mapped range?
   - Use the encoder to light up the four NeoPixels and the ring.

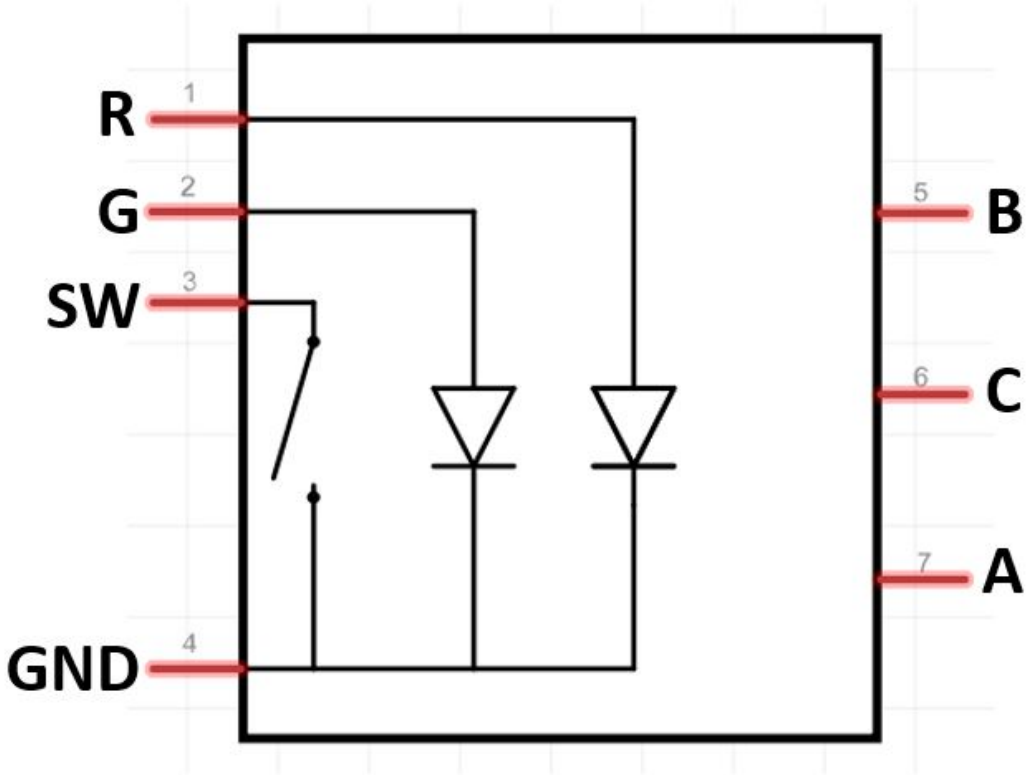
Encoder Jitter



Encoder - Low Pass Filter



Encoder - LEDs and Button



The encoder has a Red and Green LED, as well as a Switch (technically a Button), that act like discrete components and are not associated with the Encoder.h library.

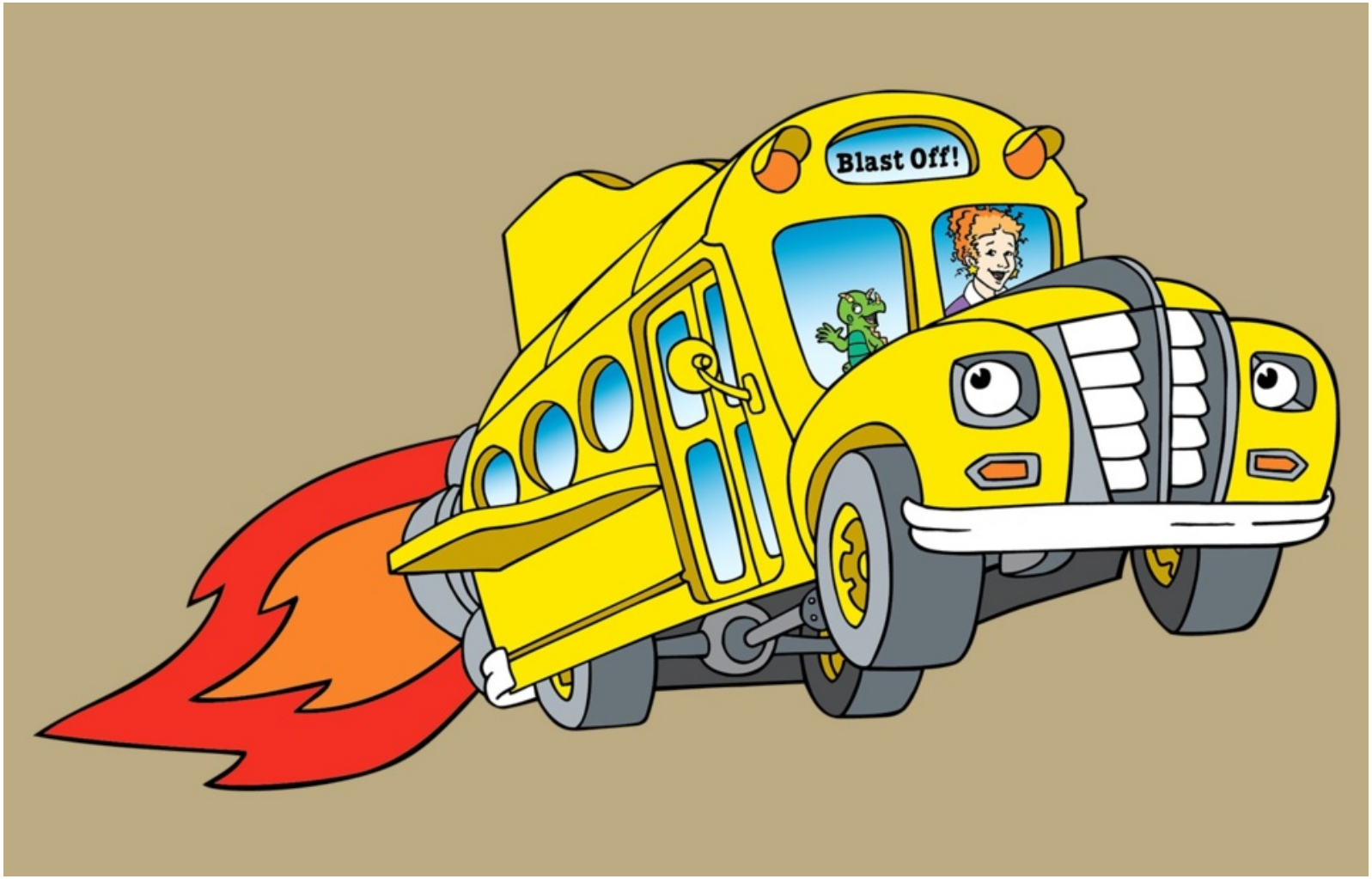
Assignment: Encoders

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

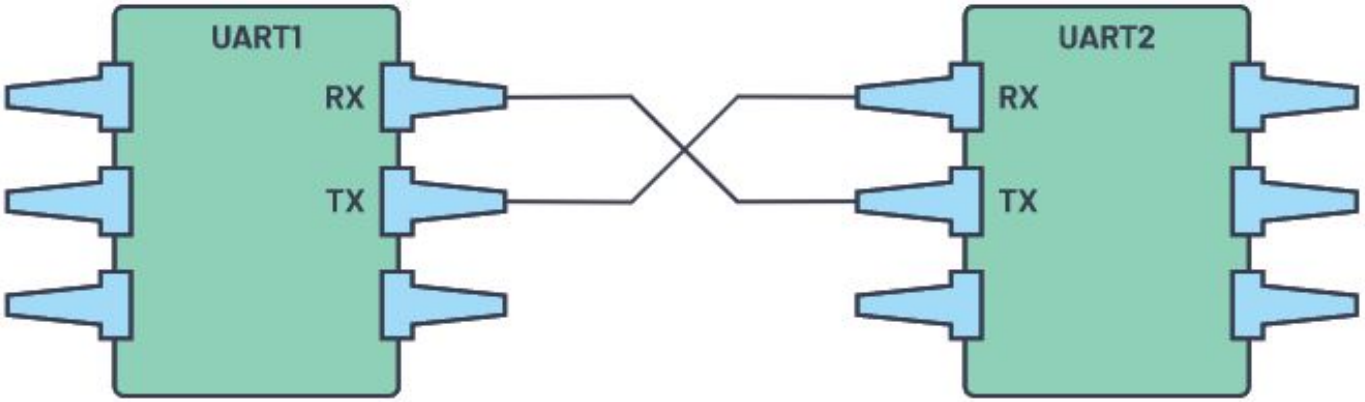
1. L06_03_encoder_switch
   - Connect the encoder switch and LEDs.
   - Use the switch to turn on/off the NeoPixels.
   - Set encoder LED to red for off and green for on.
   - Disable turning the encoder when off.
2. L06_04_rainbow1
   - Without OneButton: Use a button to cycle the NeoPixel ring colors through the colors of the rainbow; one color change each time button is pressed.
3. L06_05_rainbow2
   - With OneButton: Have it continuously cycle (i.e, while pressed colors change every one second).

L07_SPI

Buses and Interfaces



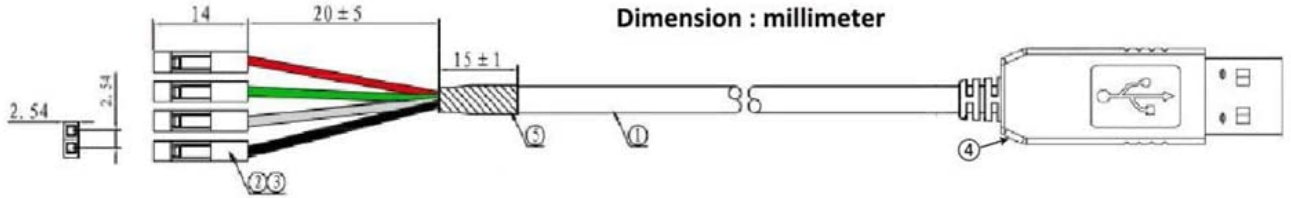
UART



Universal Asynchronous Receiver/Transmitter

- Each device have a transmit (Tx) and receive (Rx) pin.
- UART1 Tx is connected to UART2 Rx (and visa versa)

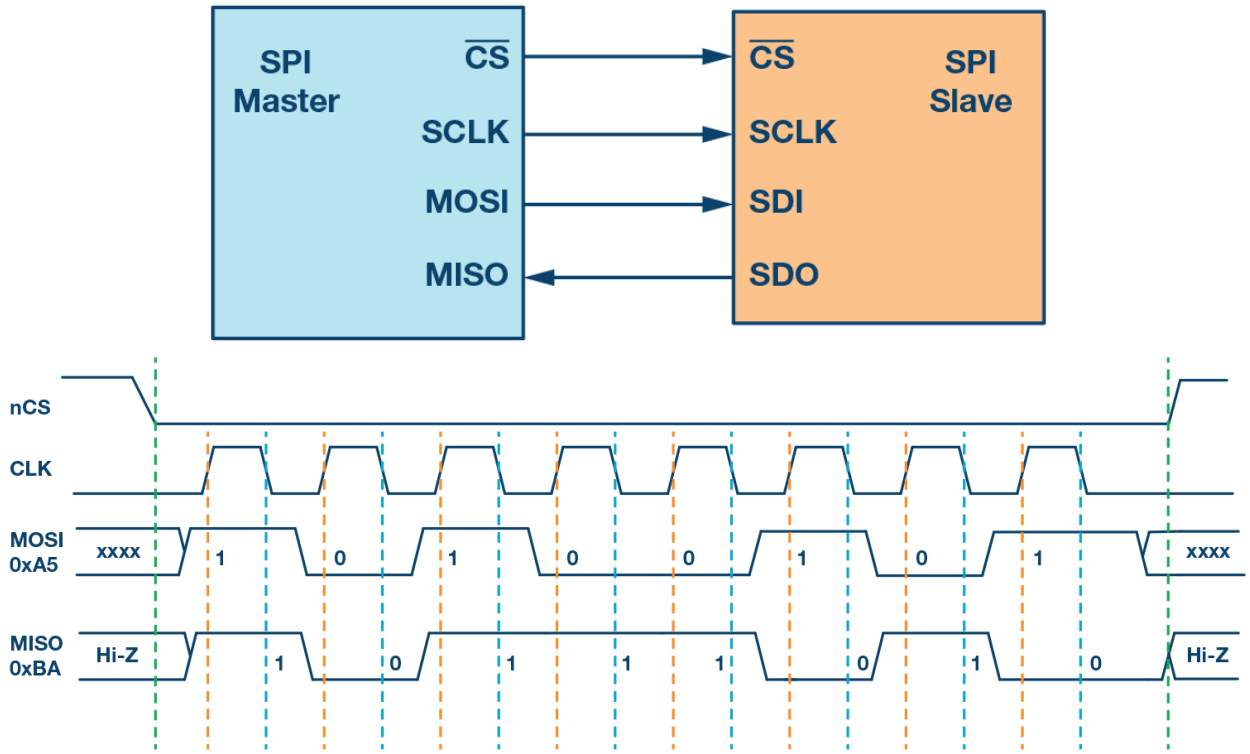
The USB Cable is a UART connection



① PVC Jacket
② 2.54mm pitch socket
③ 2.54mm 1*4p terminal ,female
④ USB A male, PL2303TA TTL 3.3V PCBA inside
⑤ Heat shrink tube, black, OD=5.0

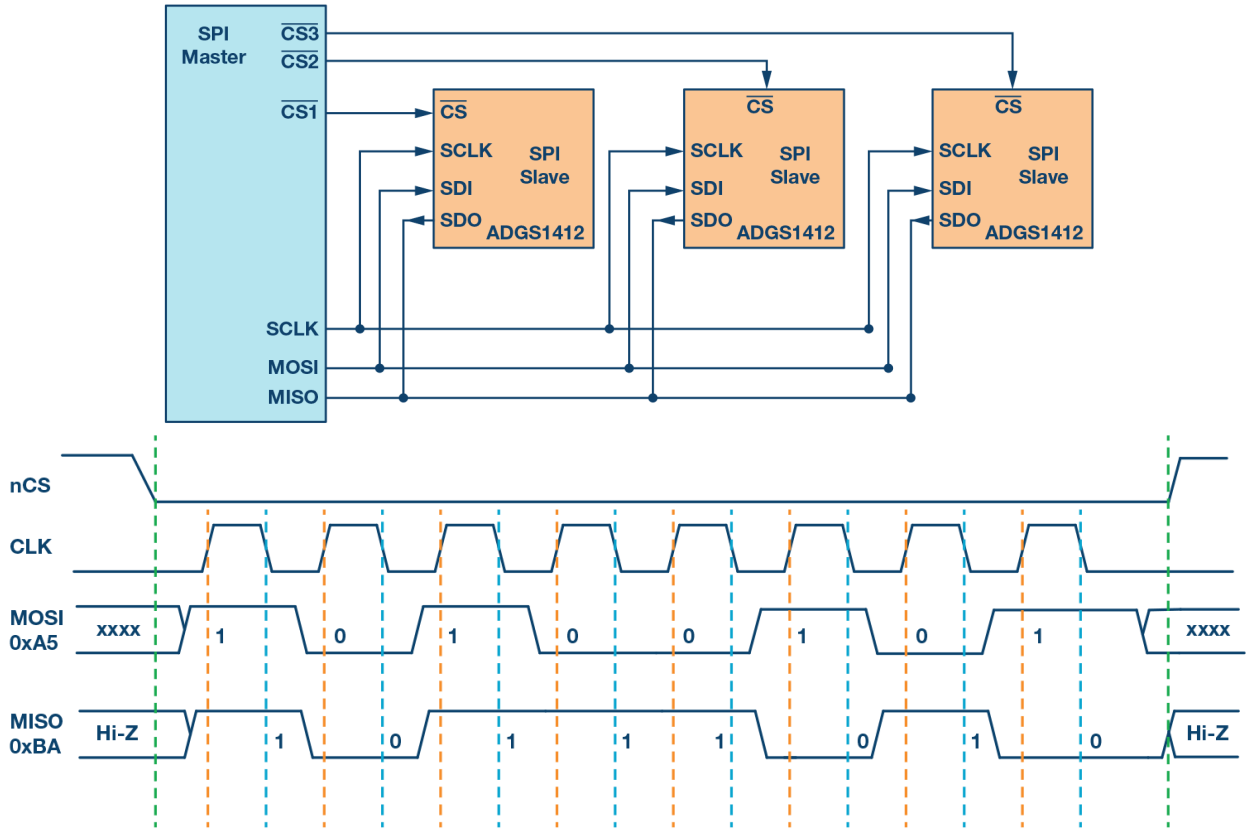
1*4P Female Socket	Name	Colour	Description
Pin 1	TXD	White	Transmit Asynchronous Data
Pin 2	RXD	Green	Receive Asynchronous Data
Pin 3	GND	Black	Device ground supply
Pin 4	VCC	Red	+5V

Serial Peripheral Interface



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

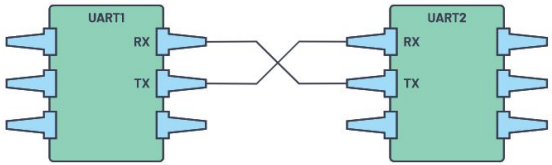
SPI - Chip Select



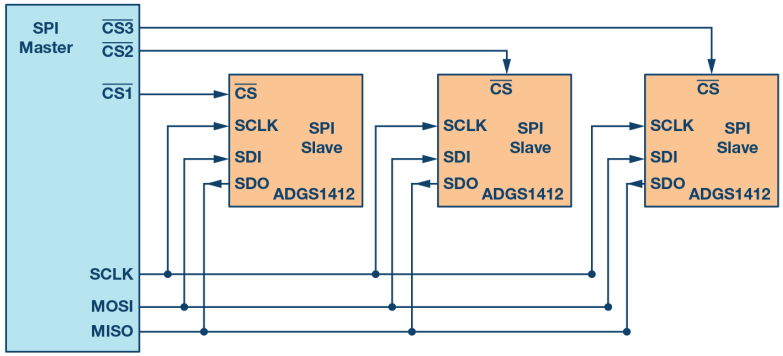
- SPI uses the $\overline{CS}$ lines to select which peripheral is active.
- Having two SPI devices selected at the same time causes interference.
- In void setup(), always initalize all SPI devices as "off"
  - Note: $\overline{CS}$ is active LOW ("off" is HIGH)

Serial UART vs SPI

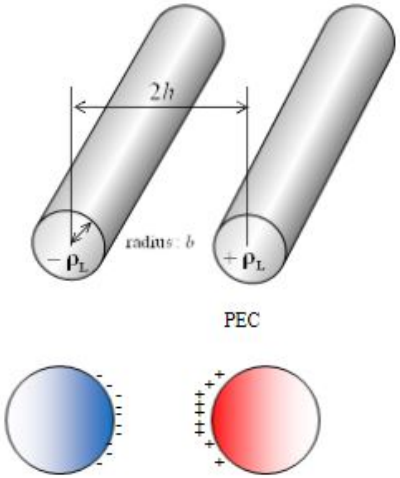
UART

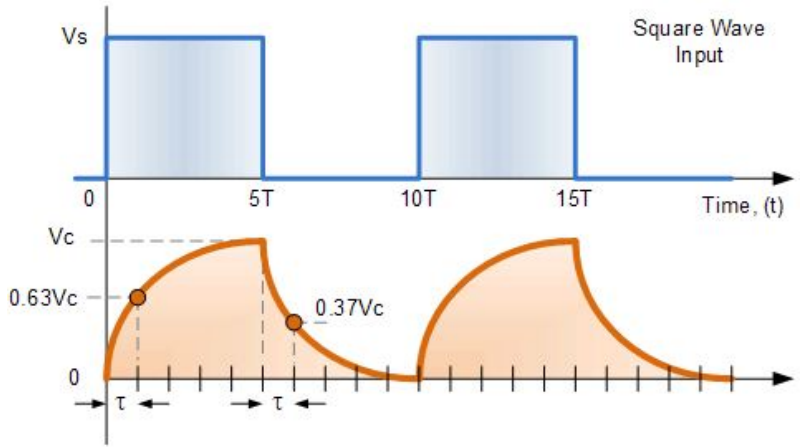


SPI



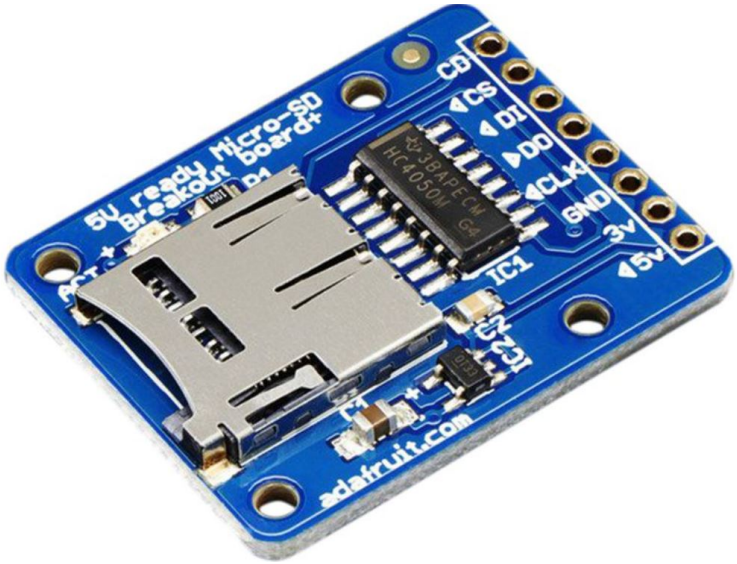
RC Time Constant





- Up until now, we have considered wires as ideal conductors; however:
  - Wires have non-zero resistance, longer and thinner wires have more resistance.
  - When two wires are close to each other, there is a parasitic capacitance between them.
- The time constant ($\tau = \frac{1}{RC}$) of an RC circuit determines the amount of time it takes a square wave input to reach 63% of it's final value.
- Some communication protocols expect sharp transitions between 0 and 3.3V for clock and data signals.

μSD Card Module



1. SD cards are sensitive to interface to the pins
   - Keep wires short
   - Data lines need to be 3.3V, use level shifter is needed
2. FAT16 or FAT32 format
   - File naming - 8.3 (e.g., myfile12.csv)
3. Pinout
   - ◀ 5V - Power input(3.3V or 5V)
   - 3V output to power other devices
   - GND - Ground
   - ◀ CLK - Clock
   - ▶ DO - MISO
   - ◀ DI - MOSI
   - ◀ CS - Chip Select
   - CD - Card Detect

Assignment: L07_01_myLogger

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

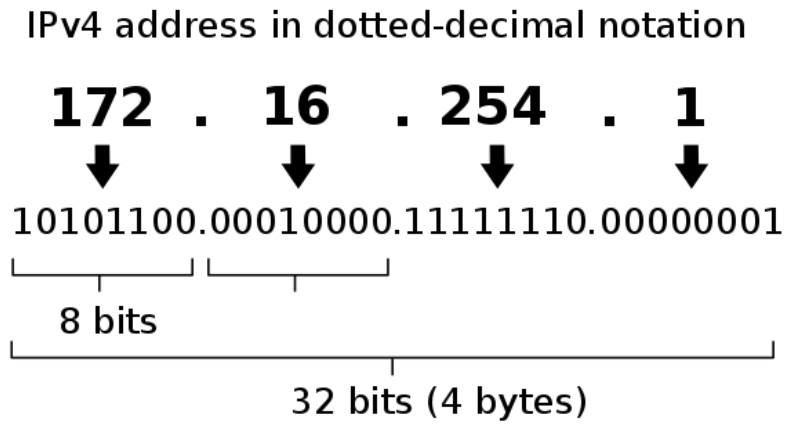
1. L07_00_dataLogger
   - The starter code details the pin assignments for SPI. Use these for your schematic and Fritzing.
   - After drawing schematic, fritzing, and wiring your μSD module, run the sample code to validate your configuration.
2. Save the starter code as L07_01_myLogger and modify it:
   - Read two inputs:
     1. The value of the encoder (unbounded)
     2. The value of a potentiometer setup as a voltage divider.
   - Write to μSD Card a timestamp (in millis()) and the two input values every 5 seconds without using delay().

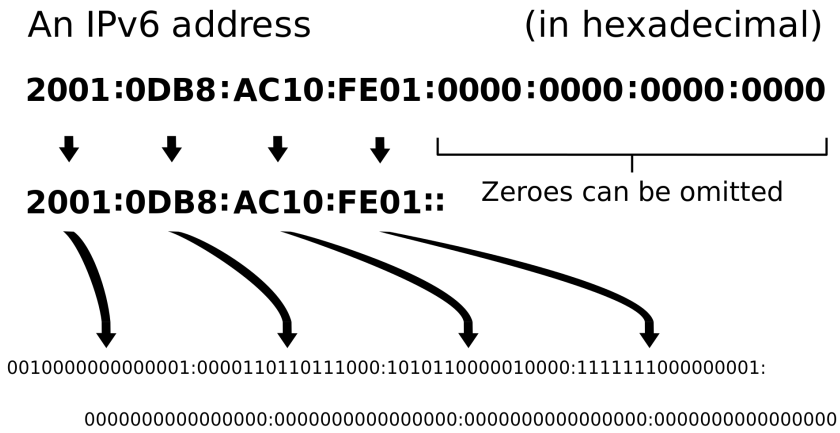
L08_Ethernet

The Internet

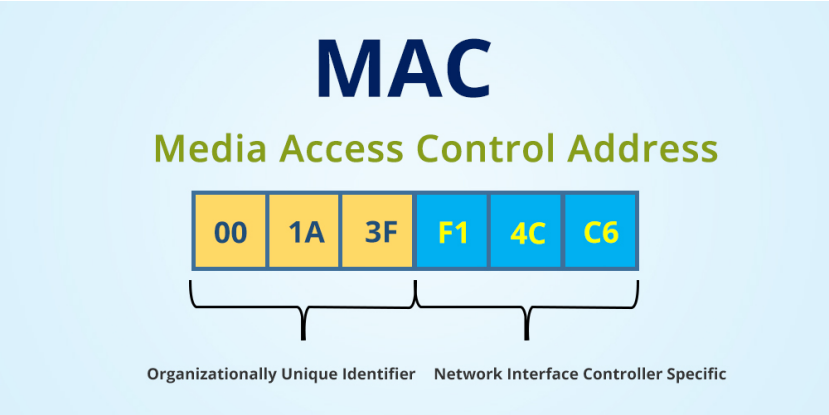
IP Addresses

- When a device joins the network, it is given an internet address.
  - static or dynamic
- IPv4 (32-bit) - 4.2 billion
- IPv6 (128-bit) - $340 * 10^{27}$ (quadrilliard)
- In Powershell: ipconfig /all
- In Terminal (MAC/Linux): ifconfig





MAC Address

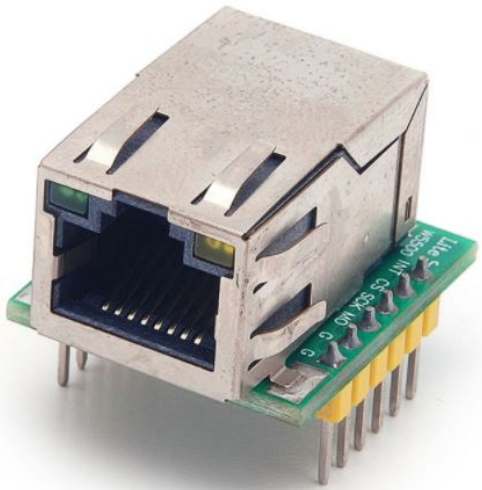


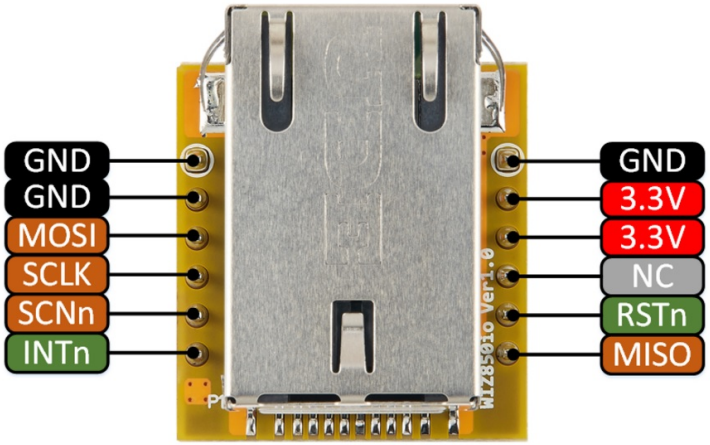
A MAC Address is a unique 6-byte (48-bit) address that is usually permanently burned into a network interface card (NIC) and uniquely identifies the device on an Ethernet-based network. The uniqueness of MAC addresses is ensured by IEEE.

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.

Ethernet Port





Assignment: Wemo

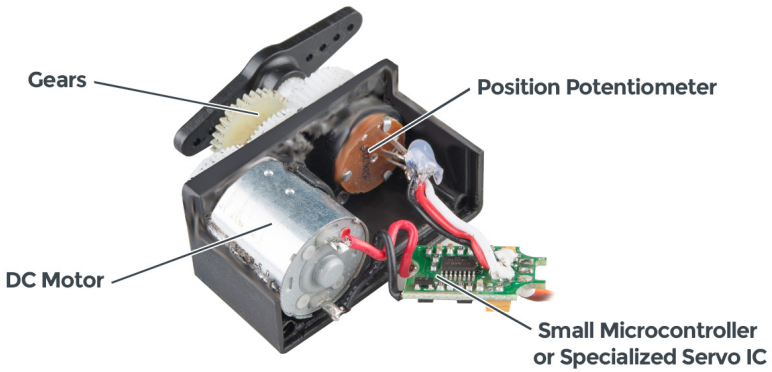
- Schematic
- Fritzing diagram (Part: wiz280io)
- Wire your circuit
- Write the code

Add the Ethernet (CS=10) to your breadboard along with a button in Pin 23.

1. L08_00_EthernetTest
   - Create your own mac.h MAC Address.
2. L08_01_Wemo
   - Using wemo.h, use the button to toggle Wemo on/off (one toggle per press)
   - Implement a method to select different Wemo (encoder, doubleClick, etc.)
3. L08_02_Wemo_Timer
   - Create timer that turns off a Wemo 10 secs after you push "off" button without using delay().
4. L08_03_Wemo_Object
   - Copy wemo.h to /wemoObj/wemoObj.h
   - Modify it to be use Class and Methods.
   - Modify your code to use a wemoObj object.

L09_Servo

Servo Motors



- A servo is any motor-driven system with a feedback element built in.
- A servo motor basically has three core components:
  1. a DC motor,
  2. a potentiometer that measures its position,
  3. a feedback controller circuit
- The servo is controlled by a PWM signal from a digital pin. The width of the pulse determines the position that the servo moves to.

Servo library

For the Teensy 3.2, we will be using the PWMServo.h library

1. Header
   - PWMServo myServo; - create object myServo of class PWMServo
2. void setup()
   - myServo.attach(pin) - attach the Servo object to a pin (this must be a PWM pin)
3. void loop()
   - myServo.write(angle) - move servo to angle (in degrees)
   - myServo.read() - returns the current angle of the servo

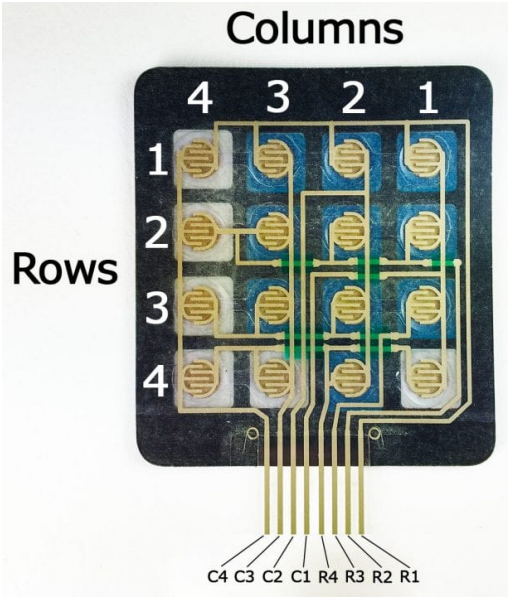
Assignment: L09_Servo

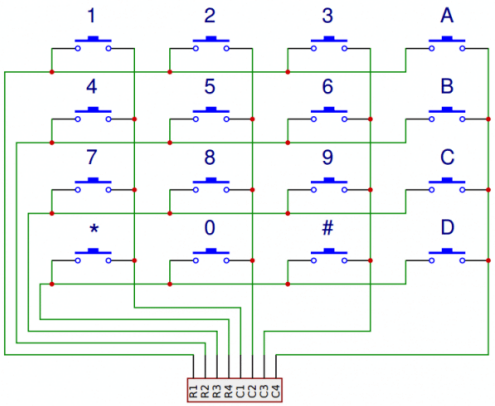
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L09_01_Servo
   - Connect the servo and a button to the Teensy
   - Create a HelloServo-type code that moves the servo to 180 degrees, waits, and then moves it to zero.
   - Modify the code to have the servo oscillate between 0 and 180 degrees using a sine wave pattern.
   - Without using OneButton, have the button to start and stop the motion.
   - Extra: Have the motion begin again at the point in the cycle where it stops.

On the Teensy, we can use PI for π, note however, in C++ (math.h) the math constants are M_<name>. For example, M_PI $= \pi$. Use M_PI for this exercise.

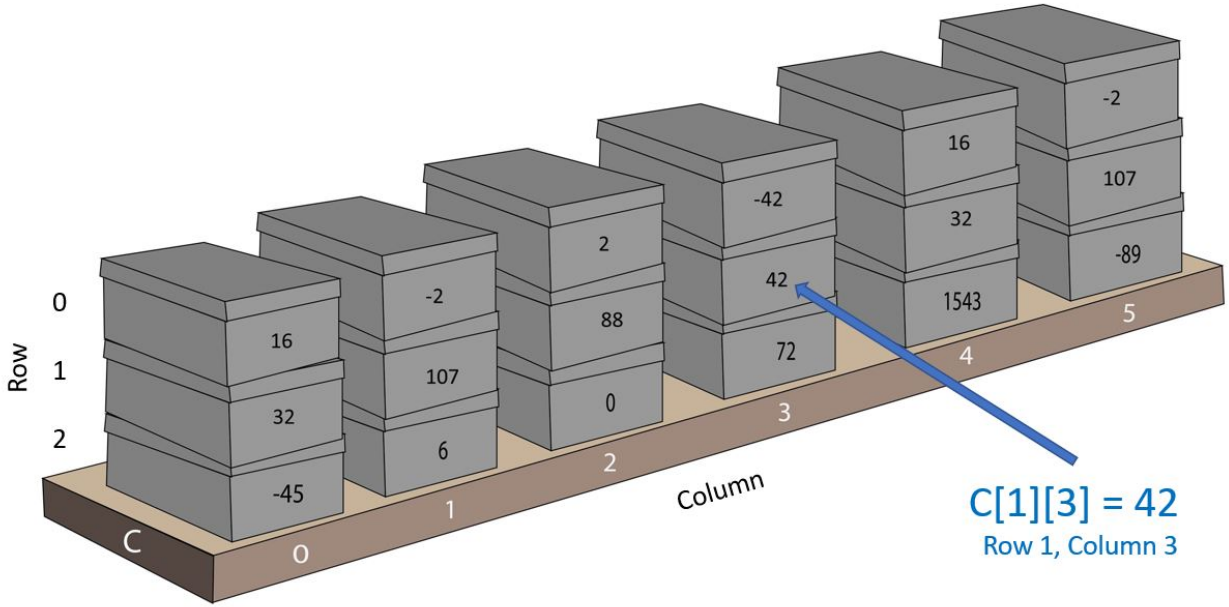
KeyPad





- The Keypad is a two-dimensional array of buttons.
- Pushing a button connects one row pin with one column pin.
- We will use the Keypad.h library to access the Keypad.
- NOTE: due to the onboard LED, you can not use Pin 13 for the keypad

2-dimensional arrays



- Declare Array: int c[3][6] = {{16,-2,2,-42,16,-2},{32,107,88,42,32,107},{-45,6,0,72,1543,-89}};
- Set a Cell: c[1][3] = 42;
- Access a Cell: x = c[1][3]; → x = 42

Char and Byte Datatype

- The char data type is a single byte in size and can be used to represent text characters (ASCII).
- Alternatively, the datatype byte can also be used for a single byte (8-bit number)

ASCII TABLE

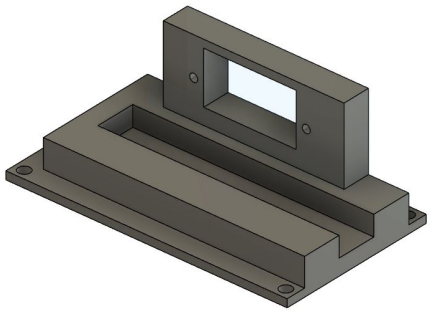
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]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

ASCII: American Standard Code For Information Interchange

Keypad.h

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;
char customKey;

char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {9, 8, 7, 6};     \\keypad leads 8,7,6,5
byte colPins[COLS] = {5, 4, 3, 2};     \\keypad leads 4,3,2,1

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){
  Serial.begin(9600);
}

void loop(){
  customKey = customKeypad.getKey();

  if (customKey){
    Serial.printf("Key Pressed: %c\n",customKey);
    Serial.printf("Key Pressed (Hex Code) 0x%02X\n",customKey);
  }
}
```

Assignment: L09_02_Lock



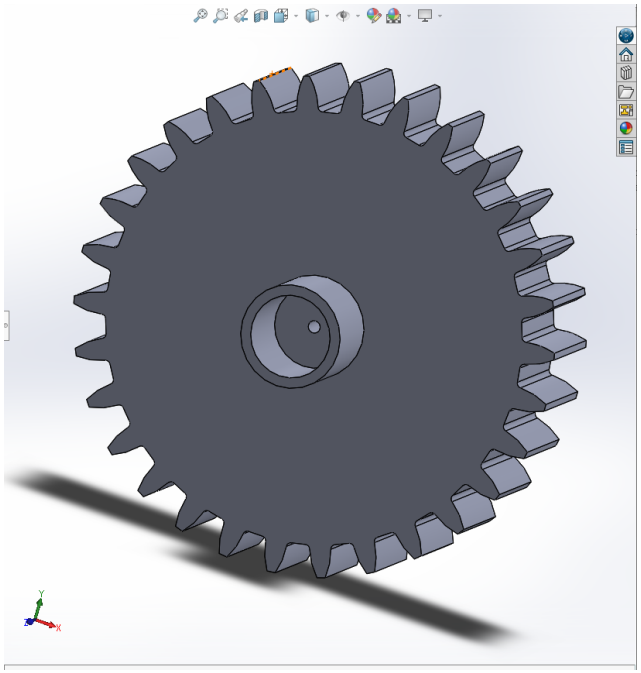
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. Design and print a lockholder based on the class example.
   - Optional: design and print your own gear and lock slider
2. Using the keypad, create a digital lock
   - Implement 4-digit digital "key" in an array.
   - Use the keypad to enter a code and store the entered code in an array
   - Create a bool function[a] that compares entered code array to the key array
   - Use the servo to lock and unlock based on a correctly entered code.
   - Light green LED and disengage lock when unlocked.
   - Light red LED and engage lock when locked.

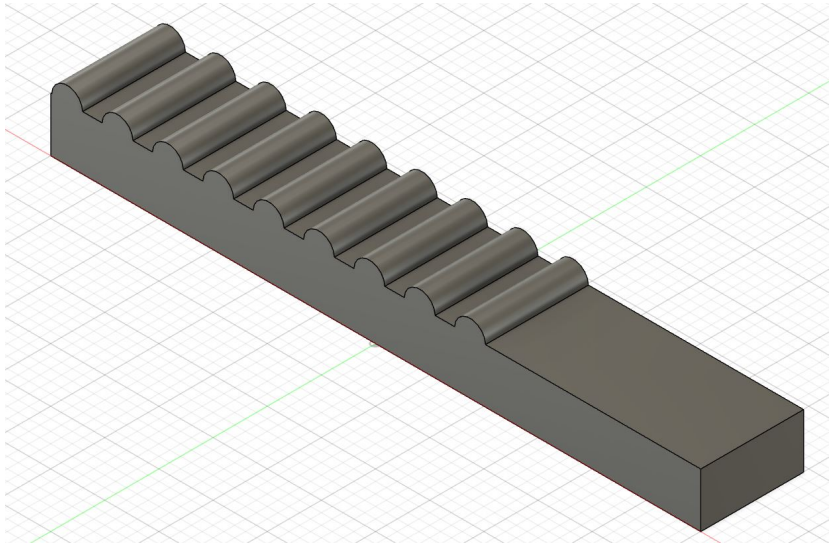
[a] Remember to use local variables in the function

Optional Designs

GEAR



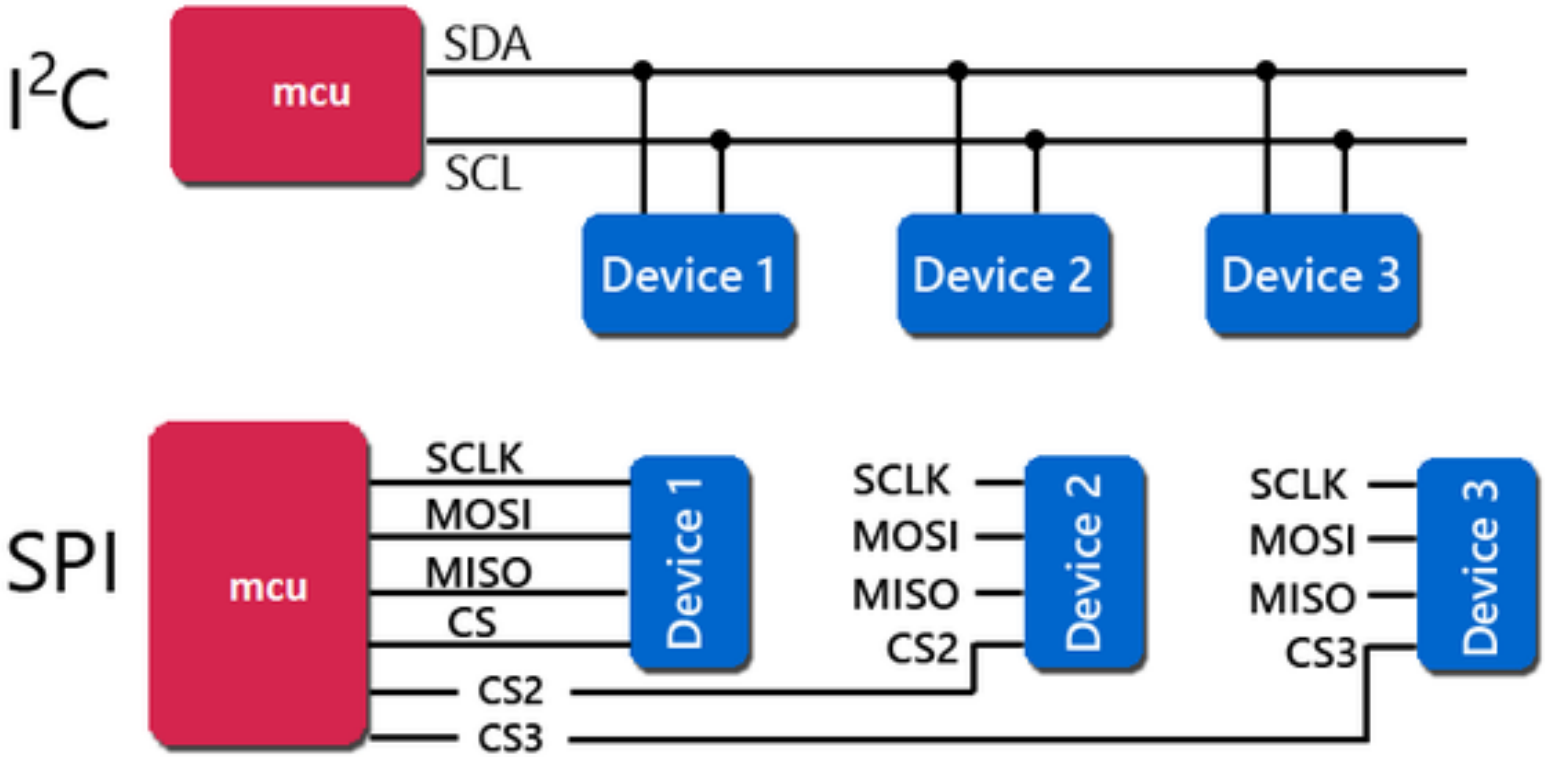
LOCK SLIDER



L10_I^2C

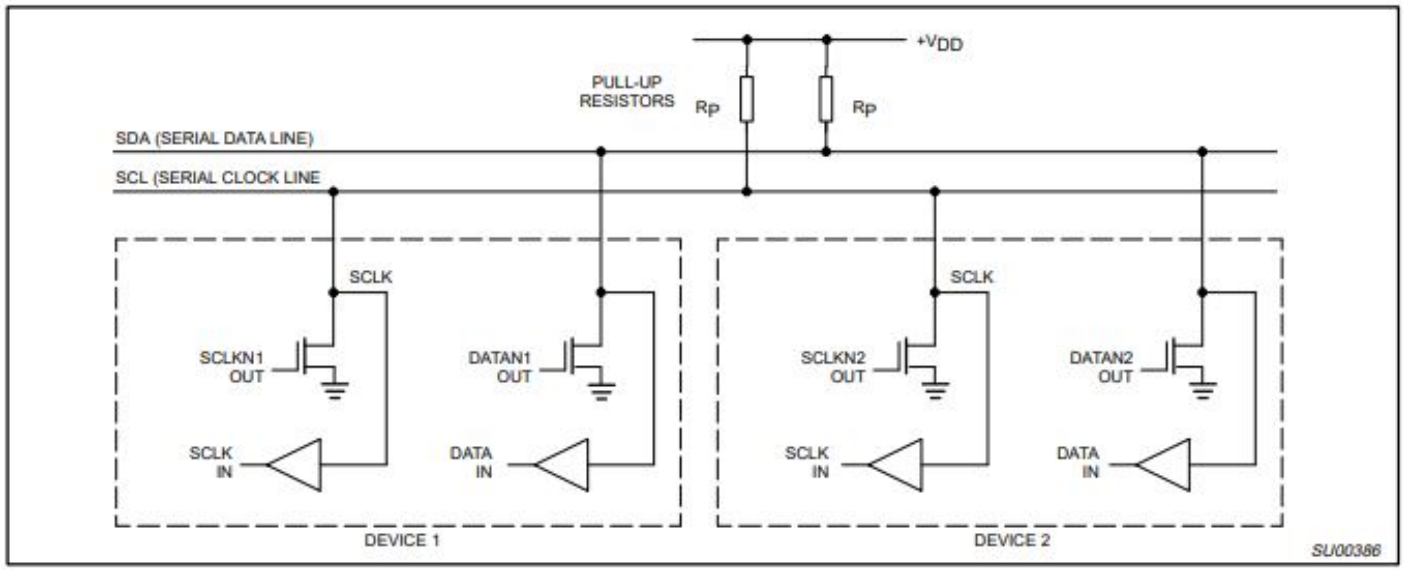
IoT Humor

Inter-integrated Circuit (I^2C)



I^2C Pullup Resistors

The I^2C drivers are "open drain". They can pull the corresponding signal line low, but cannot drive it high. This is done to prevent a short when one device is driving the bus low, while another is driving it high.



Each I^2C line needs a pull-up resistor on it to restore the signal to high when no device is asserting it low. The Teensy (and Argon) have internal pull-up resistors that are usually sufficient to restore the high signal. For I^2C greater than $1m$, an external 4.7Ω should be added to each line.

I^2C vs SPI

I2C v/s SPI

I2C	SPI
Speed limit varies from 100kbps, 400kbps, 1mbps, 3.4mbps depending on i2c version.	More than 1mbps, 10mbps till 100mbps can be achieved.
Half duplex synchronous protocol	Full Duplex synchronous protocol
Support Multi master configuration	Multi master configuration is not possible
Acknowledgement at each transfer	No Acknowledgement
Require Two Pins only SDA, SCL	Require separate MISO, MOSI, CLK & CS signal for each slave.
Addition of new device on the bus is easy	Addition of new device on the bus is not much easy a I2C
More Overhead (due to acknowledgement, start, stop)	Less Overhead
Noise sensitivity is high	Less noise sensitivity

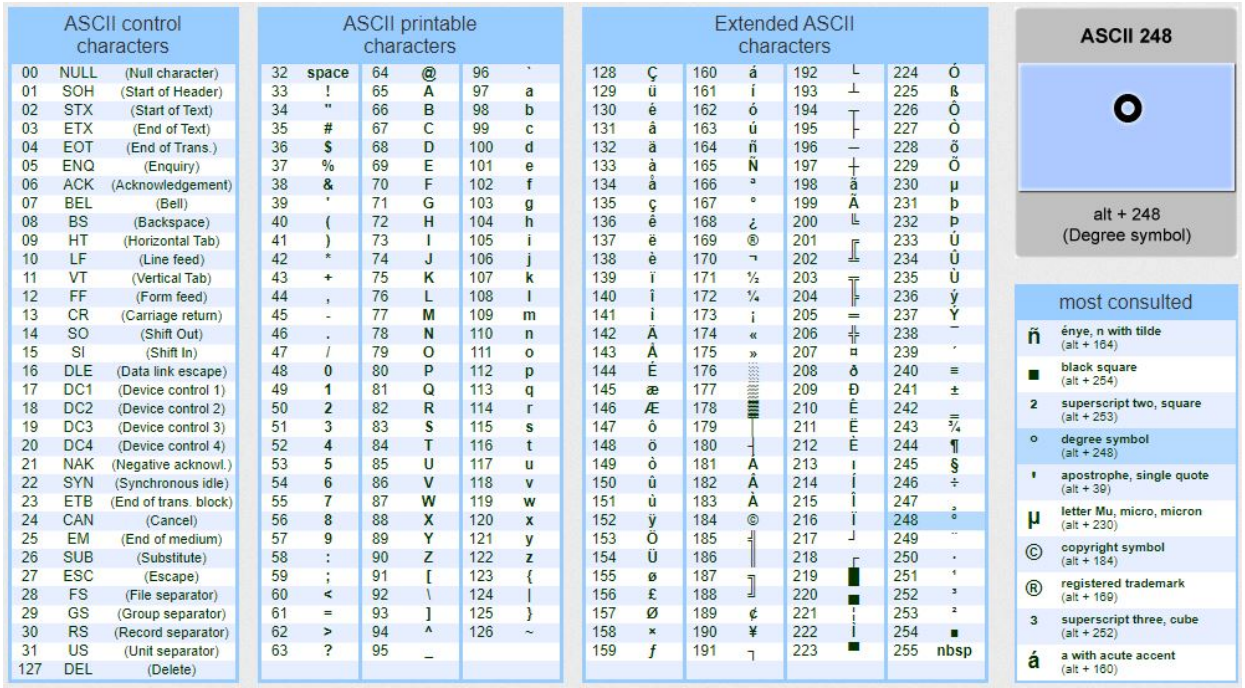
L10_00_I2CScanner

Let's create an I2C scanner

1. On your large breadboard, add the BME280 and OLED.
2. Follow along to create the I2C code.
   - Use library wire.h
   - Wire.begin();
   - Wire.beginTransmission(i);
   - Wire.endTransmission();
     - 0: Transmission Successful
     - 1: Data too long to fit in transmit buffer
     - 2: Received NACK (Negative Acknowledgment) on transmit of address
     - 3: Received NACK on transmit of data
     - 4: Other error
3. Determine the I2C addresses of each device, document in your lab notebook.

Char Datatype - Revisited

Reminder - the char data type is a single byte in size and can be used to represent text characters (ASCII).



ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	▒	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▒	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	▉	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

ASCII 248

alt + 248
(Degree symbol)

most consulted

ñ	ényе, n with tilde	(alt + 164)
■	black square	(alt + 254)
2	superscript two, square	(alt + 253)
°	degree symbol	(alt + 248)
'	apostrophe, single quote	(alt + 39)
µ	letter Mu, micro, micron	(alt + 230)
©	copyright symbol	(alt + 184)
®	registered trademark	(alt + 169)
3	superscript three, cube	(alt + 252)
á	a with acute accent	(alt + 160)

```
1 const char degree = 0xF8;  // Decimal 248 = 0xF8
2 float temp = 98.6;
3 void setup() {
4   Serial.begin(9600);
5   //NOTE: extended ASCII characters don't always print correct to Serial Monitor
6   Serial.printf("My temperature is %0.1f %c",temp,degree);
7 }
```

A word about example code

Example code is sometime misleading as each author has their own style

1. .print() and .println() vs. .printf()
   - Some arduino-type embedded controllers don't have .printf() available as a command, so .print() and .println() are used instead.
   - Per the IoT Style Guide, we use .printf()
2. Serial.printf(F("Hello World"))
   - AMR Cortex is segmented into program memory and data memory.
   - The compiler usually stores Strings as constants in data memory.
   - The F() forces the String to be stored in program memory. This is useful when the data memory is "small"
   - The ARM Cortex compilers automatically store Strings in program memory, so F() is redundant and not needed.
3. #define pre-compiler directive
   - #define can be used to create a label that represents a value. Before compiling, the anywhere the label exists in the code, it is replaced by the value.
   - Our IoT Style guide is to use "const datatype NAME=value" instead of "#define NAME value"

Assignment: I^2C

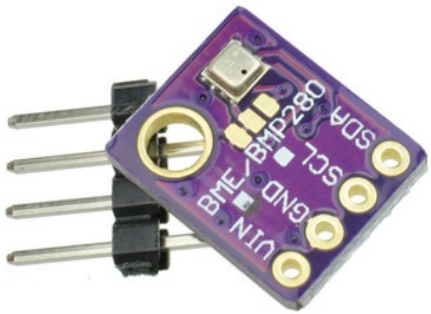
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. Install Adafruit_SSD1306 library
   - Run SSD_1306_128x64_i2c example, changing the I2C address to match your OLED.
   - In your notebook document the commands to create an object, initialize the object, and the various commands to display text.
2. L10_01_OLEDWrite
   - Without cut/paste, using your notes and printf() display:
     - Hello World
     - Your Name using spanish honorific (señor, señora, señorita).
     - Your Birthday (e.g., 04/03/1968) using separate variables for month, day, and year.
   - Rotate screen using setRotation(rot) method.
     - rot is an int from 0 to 3
   - OPTIONAL: Make your own bitmap: `https://diyusthad.com/image2cpp` and `https://www.reduceimages.com`

Using the Adafruit_BME280 class

```
// Define BME280 object
Adafruit_BME280 bme; //this is for I2C device

// Initialize the BME280 in void setup()
status = bme.begin(hexAddress);
if (status == false) {
  Serial.printf("BME280 at address 0x%02X failed to
    start", hexAddress);
}

// Getting data from BME280
tempC = bme.readTemperature(); //deg C
pressPA = bme.readPressure();  //pascals
humidRH = bme.readHumidity();  // %RH
```

Assignment: I^2C

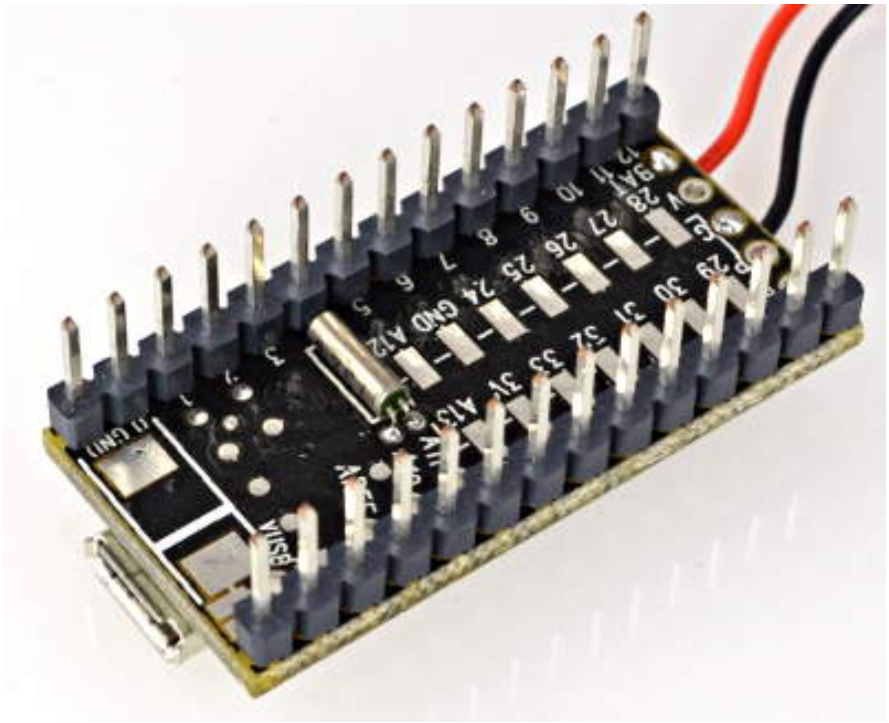


- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L10_02_BME280
   - Read BME280 data.
   - Convert to tempF and inHg.
   - Print to Serial.Monitor and the OLED display.
2. L10_03_BME280_SDMicro
   - Add in saving data to the μSD card once per minute using millis() as the timestamp.
   - Use your NeoPixels to give a visiual indication of room conditions.
3. L10_04_RTC (Optional)
   - Add the 12.5 pF crystal to the Teensy.
   - Display the time on the OLED.
   - Add a true timestamp (actual time, not millis()) to the saved data.

Real Time Clock

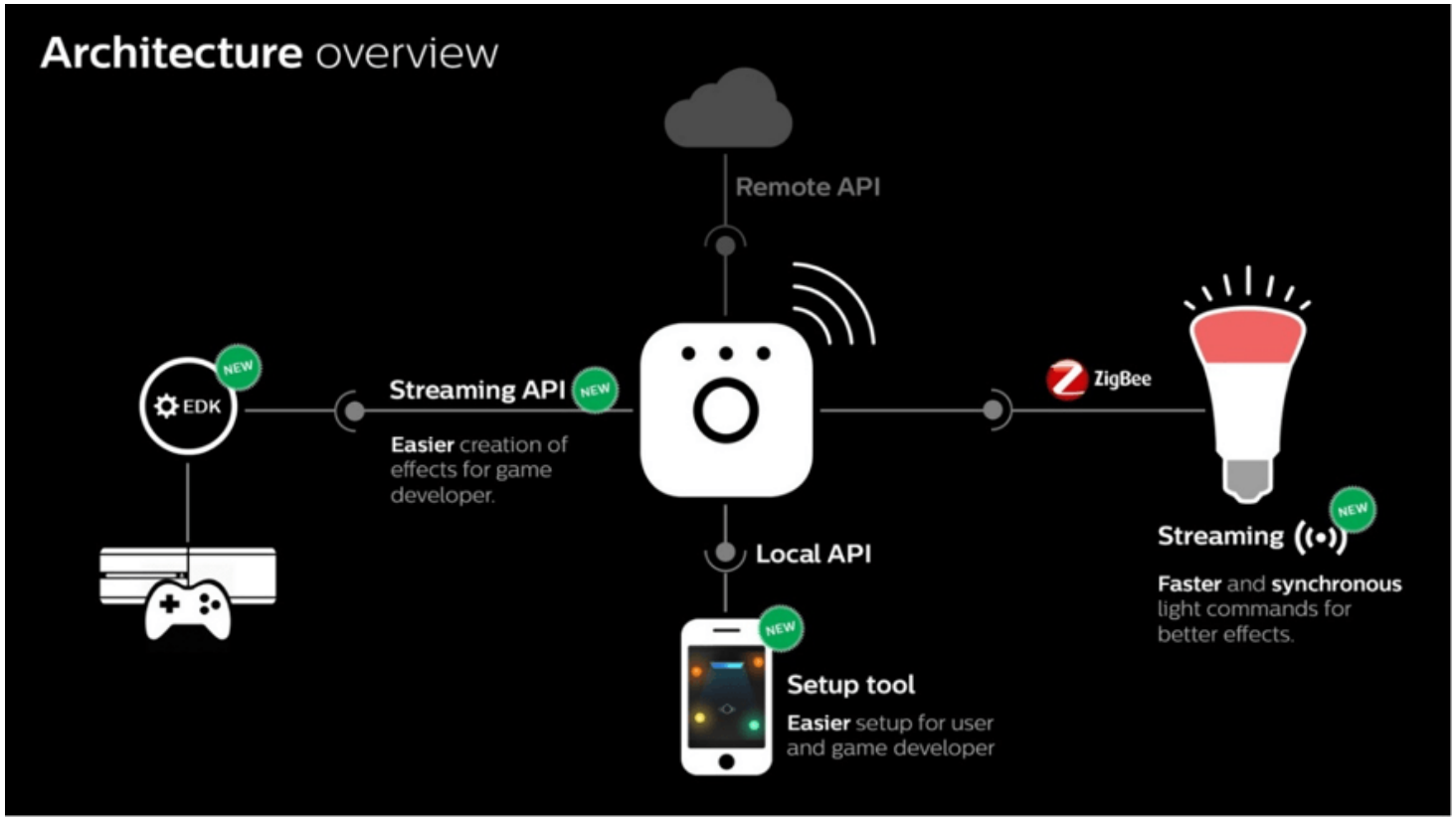
To use the Teensy 3.2 RTC, you need to add a 32.768 kHz, 12.5 pF crystal to the bottom side of the board.



Example can be found at FILE → Examples → Time → TimeTeensy3.

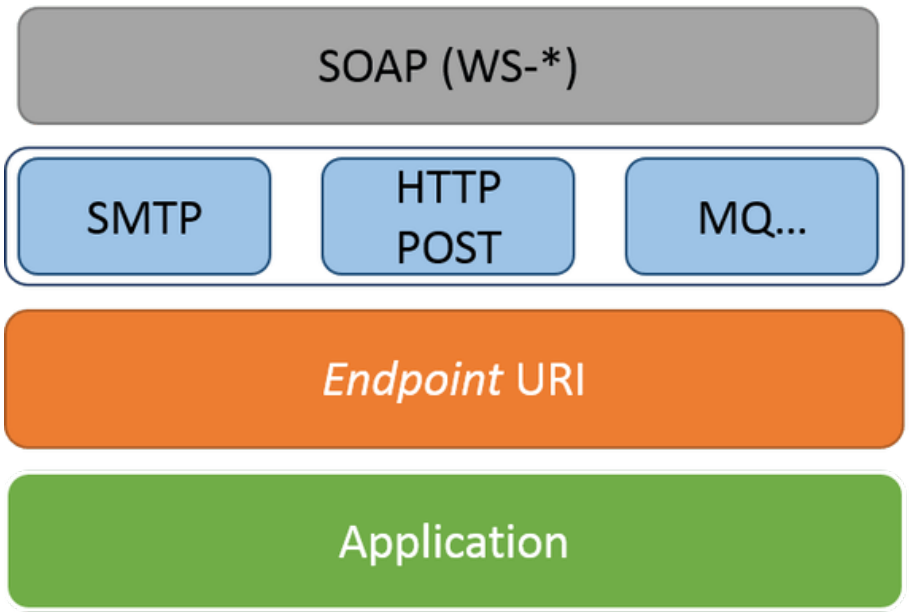
L11_Hue

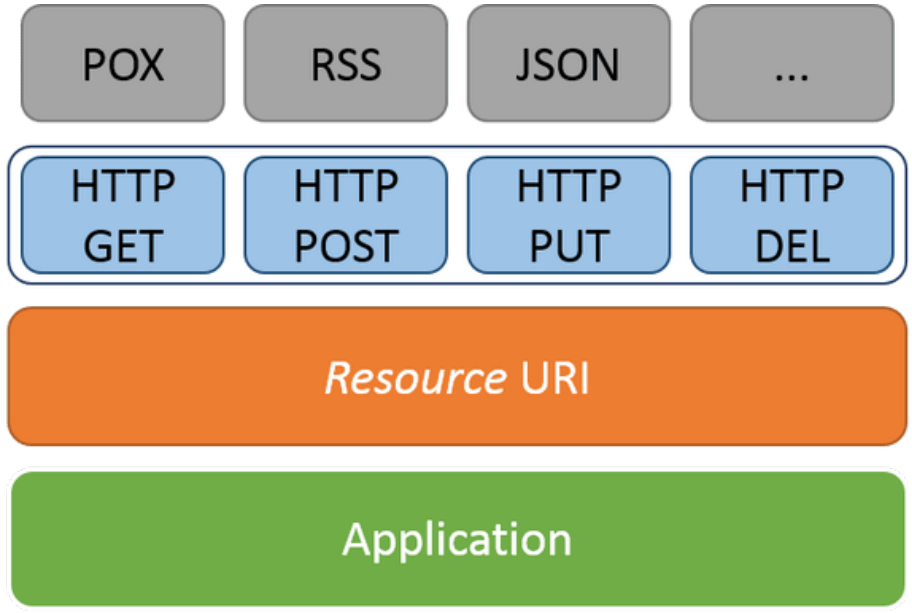
Phillips Hue API



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

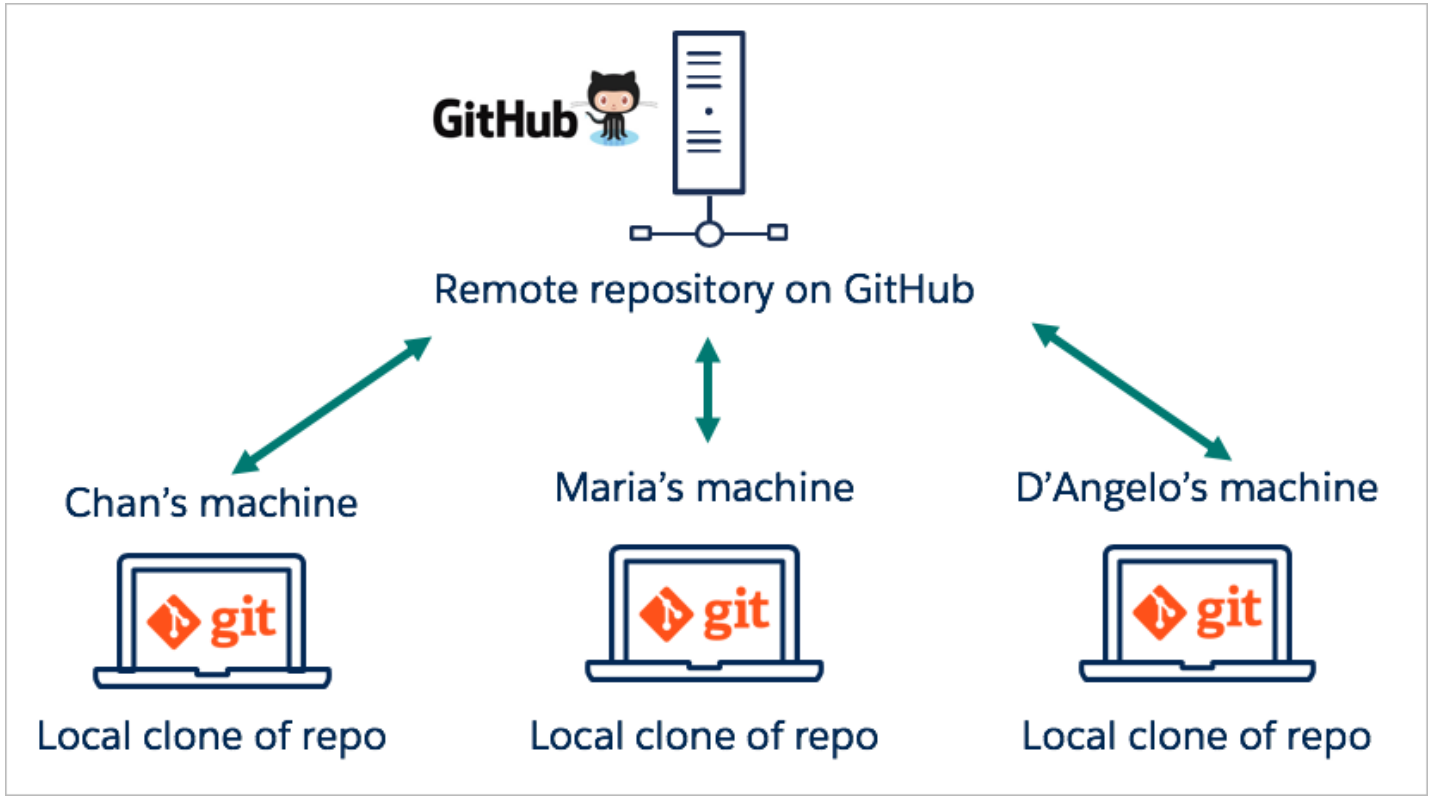
1. Using the hue.h library and L11_00_HueHeader as a template, create code that:
   - has a button that turns on and off the Hue light at your pod,
   - uses the encoder to change the brightness of the Hue bulb,
   - has a method of cycling the Hue light through the colors of the rainbow.

GitHub - Part 2

The repository - aka the repo

- The git repository is the location where files are saved/staged.
- The git repository is also called a "repo".
- You may create a new repository from scratch.
  - Your smart room controller, flower pot, and capstone projects
- You may clone a local repository from an existing remote repository.
  - Your class exercises, a 3rd party library
- The git repository keeps your version history in a hidden ".git" folder.
  - To view on Windows, in your project folder in File Explorer, click View, then check "Hidden Items"
  - To view in PowerShell, in your project folder, type dir -Force.
  - To view on Mac, in your project folder in Finder, press Shift + Control + dot

Personal Repositories



The readme, license, .gitignore files

1. README.md - Whether public or private, this file describes the purpose of your repo. If your repo is public, you can use the readme.md file to advertise your application. Read `https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/about-readmes` for more information.
2. LICENSE - The license file should be added for all public repos. Read `https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/licensing-a-repository` for more information.
3. .gitignore - Allows you to exclude files from your repo. Read `https://docs.github.com/en/github/using-git/ignoring-files` for more information.

Practice: Create Your Midterm Repo In GitHub

1. Login to your GitHub account. If you do not have a GitHub account, then create one at `https://Github.com`
2. On your GitHub home page, click "New" to create a new repository.
3. Type the following name for your new repo.
   - smart room controller
4. Indicate if the repo is public or private.
5. Select the readme and license MIT license options.
6. Click "Create Repository"
7. Your new repository is created.

Practice: Clone the repo to your local machine

1. Open your new repository in GitHub if it is not already open.
2. Click the "Code" button.
3. Copy the https address for your github repo.
   - Note: there is also an SSH option if you choose to link your public SSH key to your github repository.
     `https://docs.github.com/en/github/authenticating-to-github/about-authentication-to-github`
4. cd into your IoT directory.
5. git clone the https GitHub repo to initialize the git repo on your local machine. This creates your project folder and the hidden .git folder.

README.md: some common markdowns

A readme file can more effectively communicate your project purpose if you use markdowns. Markdowns are symbols that, when placed in text, allow you to format your text.

- # preceding text will show the text as a title (i.e. bold, larger)
- $*sometext*$ single asterisks surrounding text will present the text in italics.
- $**sometext**$ double asterisks surrounding text will present the text in bold.
- $*$ placed in front of text will create a bullet for a bulleted list.
- 1. placed in front of text will create a numbered list.
- [*GitHub*](*https* : //*github.com*) will create a hyperlink labeled "GitHub" that links to https://github.com.
- [*githublogo*](*github.png*) will display the image called github.png located at the root of your project.

For more markdown information, read `https://guides.github.com/features/mastering-markdown/`.

Practice: customizing your README.md file

1. Create the following headers in your file:
   - Overview
   - Details
   - Summary
2. Draft a brief overview of your project in the Overview section of your file. Emphasize some words with bold or italics.
3. Add a bulleted list showing project features to the Details section of your file.
4. Add a numbered list showing project updates or features to one of the sections of your file.
5. Add a hyperlink in the Summary section of your file.
6. BONUS: Add an image to one of the sections of your file.

Commit and push your changes to GitHub. View your file on GitHub to see how your markdowns are displayed.

Practice: Committing and pushing changes

1. Go back to your repo directory on your computer.
2. Edit the README.md file.
   - *You can use Notepad, TextEdit or any text editor.*
3. Make some changes to the file.
4. Commit and push the file.
5. Make some more changes to the file.
6. git diff [filename] to see the changes that will be committed.
7. Commit and push the file.
8. Go to GitHub and view "commits"
9. See how GitHub displays your changes.

Practice: View changes

1. Go to your project directory
2. git log - to view version history
3. Copy one of the commit numbers to your clipboard
4. git show [commit] - to view content changes for that commit

The .gitignore file

There may be times you want to omit directories and/or files from git.

Examples:

- Files with sensitive information such as password/credential files.
- Directories with 3rd party libraries. These will likely already be installed on the deployment server so do not need to go to GitHub.

Such files and directories are listed in a .gitignore file that usually resides at the root of your project directory.

Practice: Creating and using a .gitignore file

1. Open your test repo.
2. Add a file called credentials.h. Type something into the file and save it.
3. Type git status. What do you see?
4. Now add a file called .gitignore.
5. In the .gitignore file, type credentials.h and save.
6. Type git status. What do you see?
7. Commit and push to GitHub.

Going forward, make sure every .gitignore file contains the following two lines:

```
credentials.h
target/
```

What to do if you accidentally COMMIT a sensitive file

Suppose you accidentally **COMMIT** the sensitive file anyway?

`https://www.atlassian.com/git/tutorials/undoing-changes`

Note: there are many different scenarios. So if you run into a situation where you need to revert a commit, check the above documentation and/or check procedures with your company to see how they want to resolve.

What to do if you accidentally PUSH a sensitive file

Suppose you accidentally **PUSH** the sensitive file anyway?

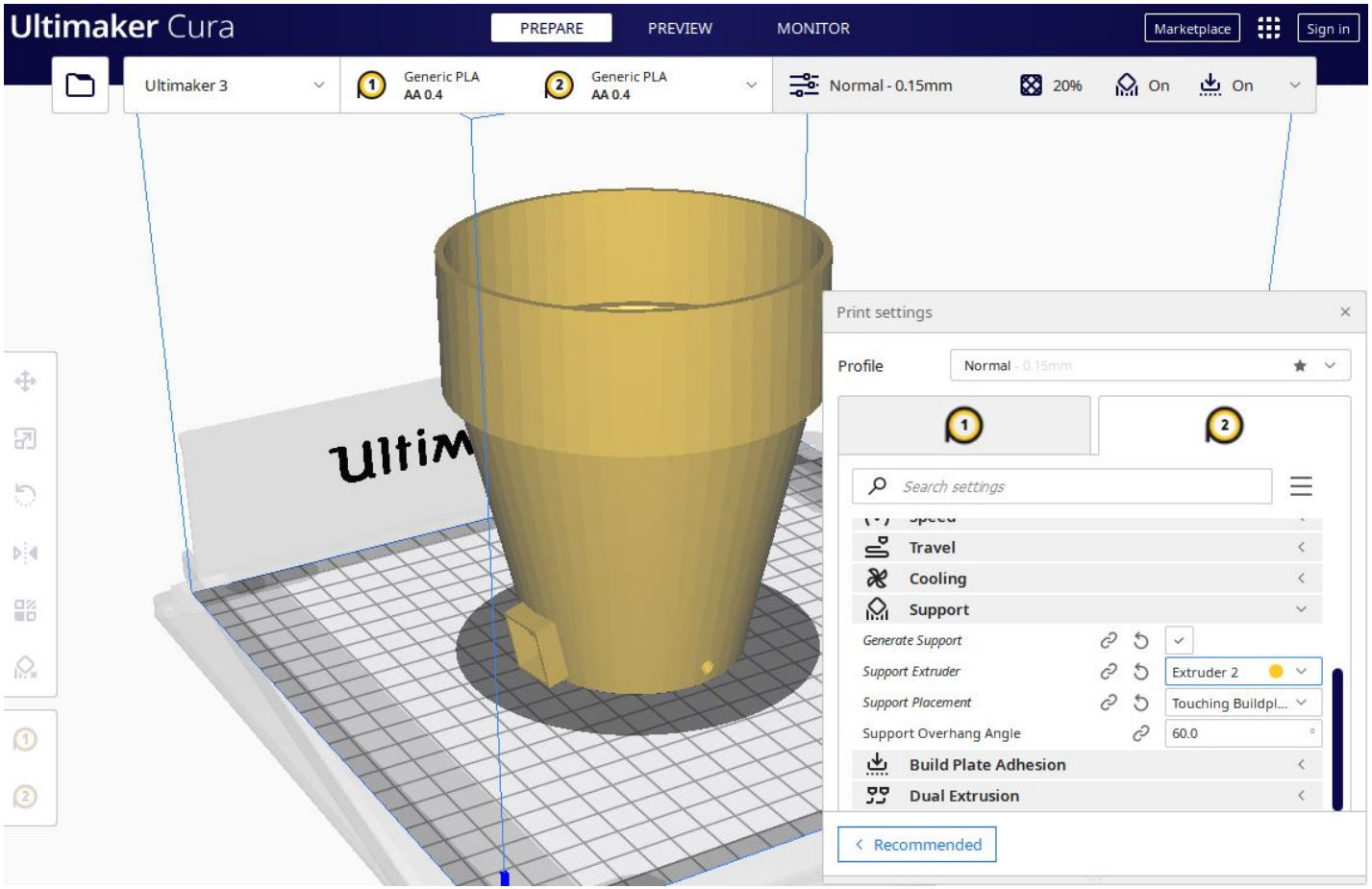
FIRST CHANGE THE PASSWORD OR INFORMATION IMMEDIATELY.

Then follow steps to remove the file from your git repository. *Note: deleting the file from your repo does not remove it completely.*

- If you are a solo-preneur or hobbyist, then you can decide how to resolve the issue.
- If you are working for a company, then check with your boss to see how to resolve.

Midterm 1 - Smart Room Controller

Printing IoT Flowerpot



It is important to change Support Placement to Touching Buildplate.

Midterm Project - Smart Room Controller

1. Determine functionality of your Smart Room Controller.
   - Use the components that we have learned over the last 3 weeks.
   - Get minimum requirements from the Instructor.
   - Sketch out the basic layout of your room controller in your lab notebook.
   - Draw flowcharts of the main functions you plan to implement.
   - Get feedback from at least 3 peers on your planned functionality.
2. Layout your circuitry in Fritzing along with a legible schematic.
3. Wire up your circuitry as if you're going to demo your controller for a perspective customer.
4. Code, debug, test.
5. Documentation and Demonstration:
   - Ensure all files are uploaded to GitHub with an appropriate README.md.
   - Upload your project to hackster.io.
   - Prepare a presentation/demonstration for the class on your controller.
   - Participate in class demonstrations (Friday morning - Week 4).

Smart Room Controller - Minimum Requirements

You are developing a Smart Room Controller prototype that you are going to pitch to a perspective investor.

1. Control all the Hue lights in the classroom
2. Control at least two Wemo outlets in the classroom
3. Display dynamic messages on the OLED display
4. Use at least three additional components (LEDs, buttons, NeoPixels, Encoders, BME280, uSD card, Servo motor, etc.)
5. Device has at least two modes
   - Manual - user controls lights and outlets
   - Automatic - external source triggers response of lights and/or outlet(s)
6. 3D design and print at least one part (button cover, knob, logo, etc.)
7. A component made at FUSE - laser, wood, metal (case, stand, etc.)
8. Extra Credit: use a component that we haven't learned in class
9. Extra Extra Credit: create a custom bitmap for your display