IoT Product Design and Coding Bootcamp

Brian Rashap, Ph.D.

15-FEB-2021

IoT Fun


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

Introduction

Brian Rashap, Ph.D.

- Proud husband of Krista and father of Shelby (22) and Ethan (18)
- Electrical Engineer with 25 years industrial experience
- High School track coach
- Hobbies: running, cycling, reading, spending time with family

Introductions

INTRODUCTIONS

Class Rules

- Respect Each Other, Help Each Other
- Ask Questions
- Be On Time (let 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!

Grading

Class assignments will total 1000 points. You will need to earn at least 750 total points and at least 225 points on your Capstone to graduate.

1. IoT assignments + Lab Notebooks: 300 pts
2. Fusion 360 assignments: 100 pts
3. Weekly quizzes: 100 pts
4. Midterm Projects: Smart Room Controller/Plant Watering System: 200 pts
5. Team Capstone Project: 300 pts

More information later today on how assignments are turned in

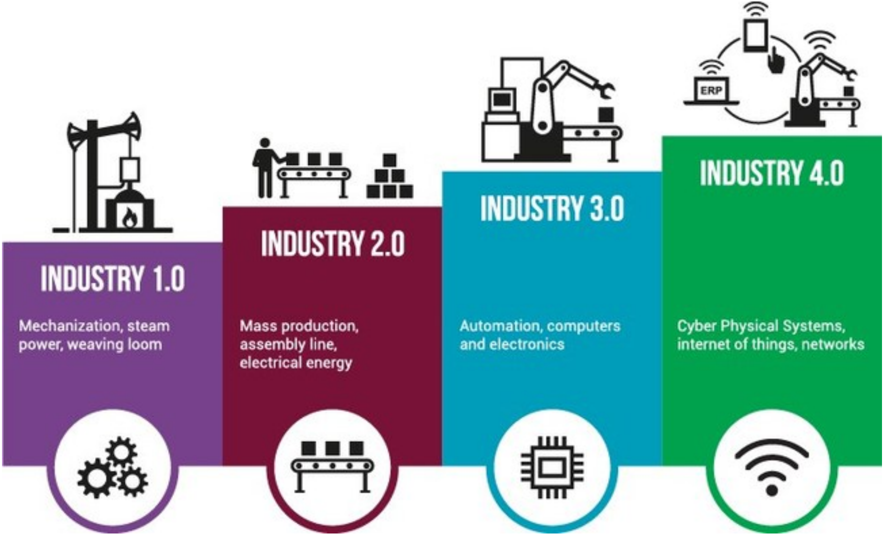
Credit for Prior Learning (CPL)

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*

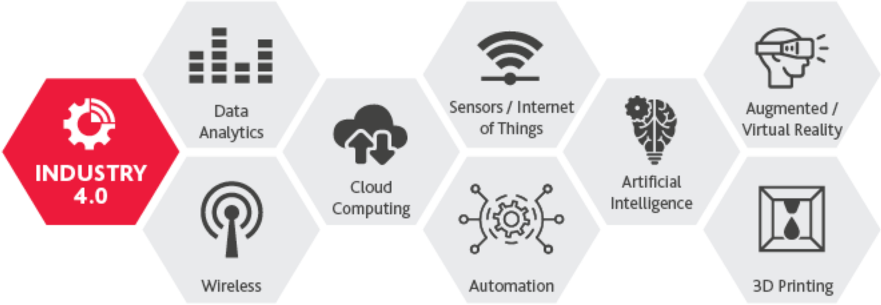
* CSIS 1151 credit requires appropriate math prerequisites

Under Review for CPL	
RPID 1005	3 Dimensional CAD
RPID 1010	Design and Simulation
RPID 1015	Prototype Fabrication I
RPID 1020	Prototype Fabrication II

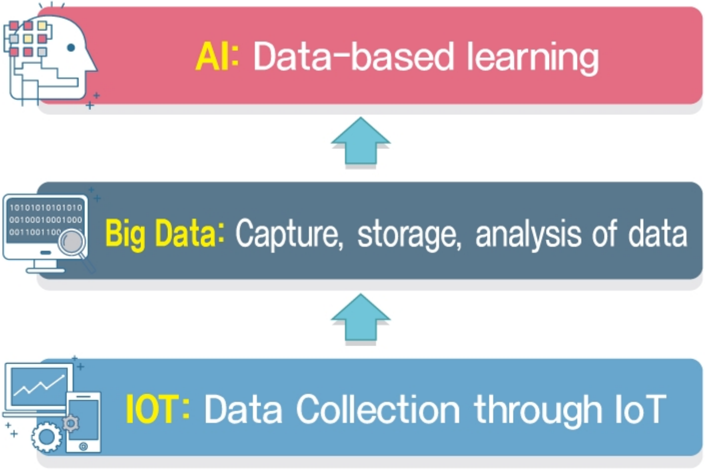
Evolution of Industry



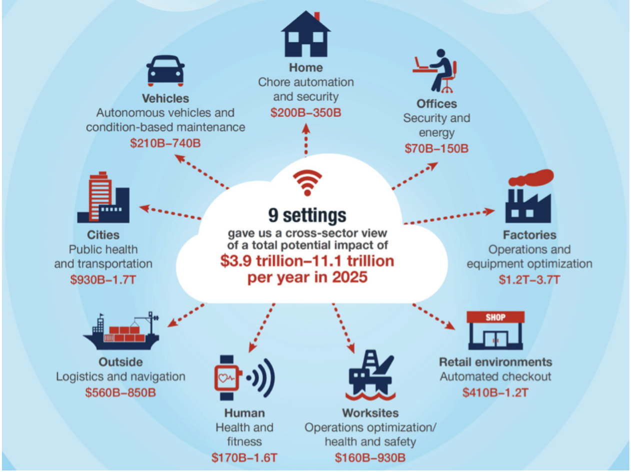
Components of Industry 4.0



IoT and Data Science



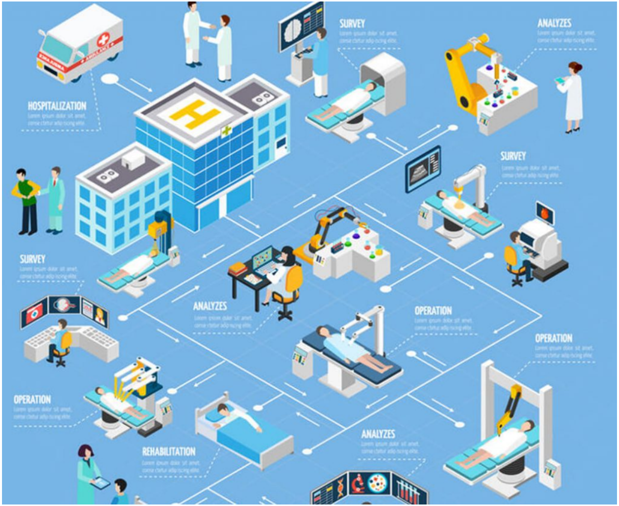
IoT 2025



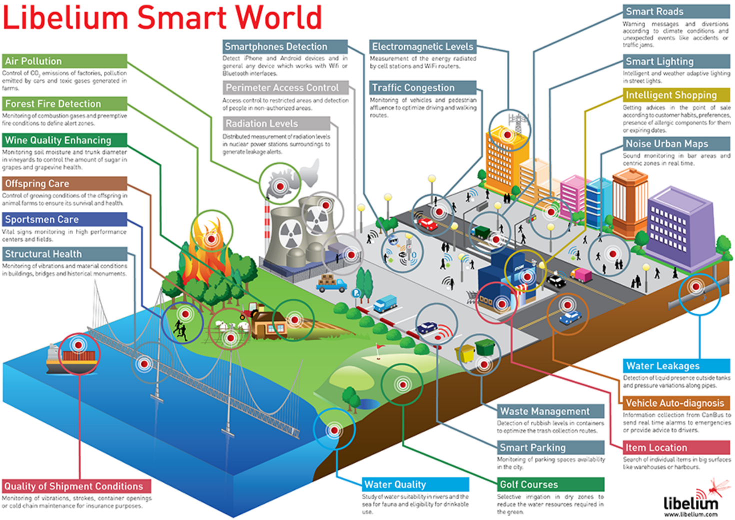
Home
Chore automation and security
$200B–350B
Vehicles
Autonomous vehicles and condition-based maintenance
$210B–740B
Offices
Security and energy
$70B–150B
Cities
Public health and transportation
$930B–1.7T
9 settings
gave us a cross-sector view of a total potential impact of
$3.9 trillion–11.1 trillion per year in 2025
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



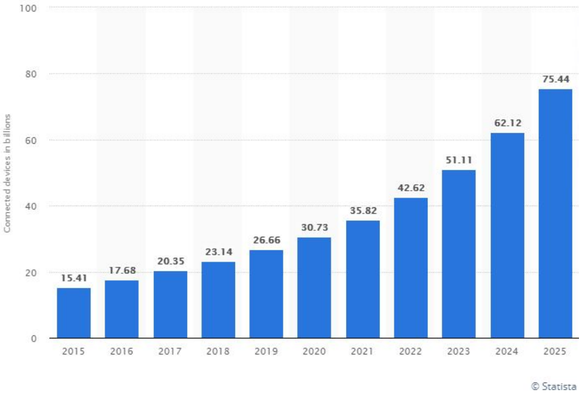
Smart World



Libelium Smart World
Air Pollution
Control of CO_2 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.
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.
Smart Roads
Warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.
Smart Lighting
Intelligent and weather adaptive lighting in street lights.
Intelligent Shopping
Getting advices in the point of sale according to customer habits, preferences, presence of allergic components for them or expiring dates.
Noise Urban Maps
Sound monitoring in bar areas and centric zones in real time.
Water Leakages
Detection of liquid presence outside tanks and pressure variations along pipes.
Vehicle Auto-diagnosis
Information collection from CanBus to send real time alarms to emergencies or provide advice to drivers.
Item Location
Search of individual items in big surfaces like warehouses or harbours.
Waste Management
Detection of rubbish levels in containers to optimize the trash collection routes.
Smart Parking
Monitoring of parking spaces availability in the city.
Golf Courses
Selective irrigation in dry zones to reduce the water resources required in the green.
Water Quality
Study of water suitability in rivers and the sea for fauna and eligibility for drinkable use.
Quality of Shipment Conditions
Monitoring of vibrations, strokes, container openings or cold chain maintenance for insurance purposes.
libelium
www.libelium.com

And Out of This World

IoT Growth

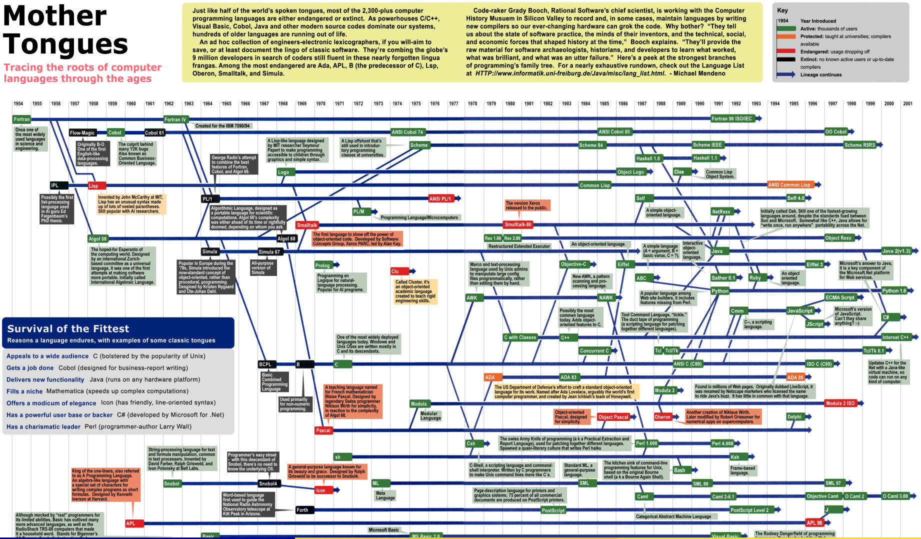


How ubiquitous is the Internet of Things?

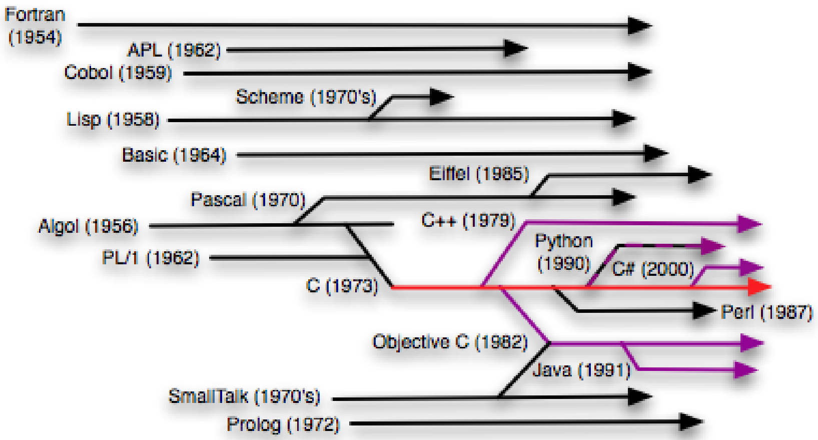
- There are approximately 31 billion IoT devices today
- 127 new IoT devices are connected to the internet every SECOND.
- This morning, 1,828,800 IoT devices will be added to the internet.

Let’s Begin Our Journey

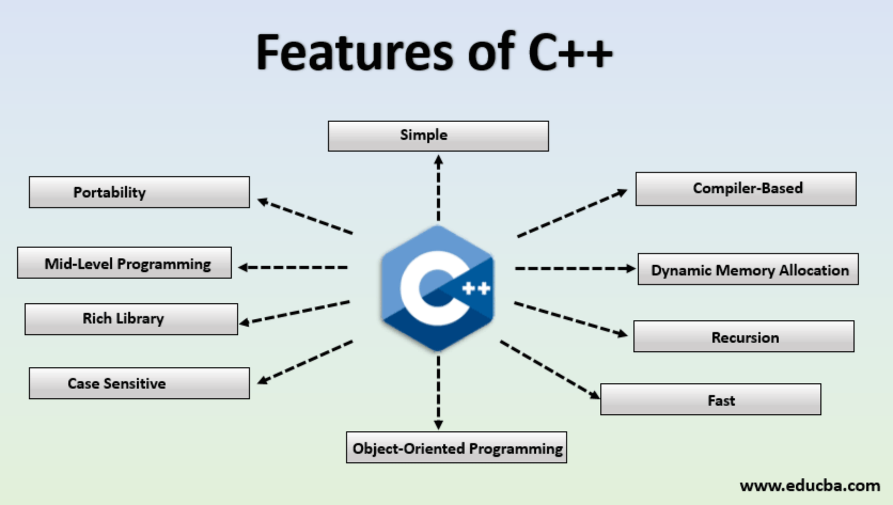
Computer Languages



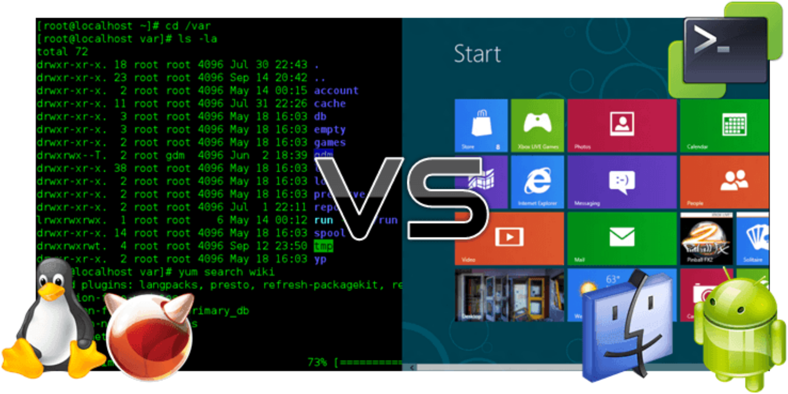
Computer Languages



Why C++


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

CLI vs GUI


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

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 system, 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: It is used to delete a directory if it is empty.
- cp: This command will copy the files and directories from source path to destination path. It can copy a file/directory with new name to the destination path. It accepts source file/directory and destination file/directory.
- mv: Used to move the files or directories. This command's working is almost similar to cp command but it deletes copy of file or directory from source path.
- rm: Used to remove files or directories.
- touch: Used to create or update a file. (PowerShell New-Item).

Command Line Interface - Displaying the file contents

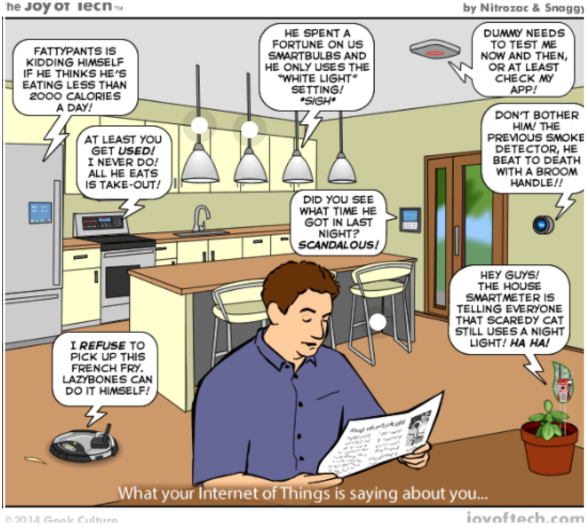
- cat: It is generally used to concatenate the files. It gives the output on the standard output.
- more: It is a filter for paging through text one screenful at a time.

The below commands are not available in PowerShell:

- less: It is used for viewing the files instead of opening the file. Similar to more command but it allows backward as well as forward movement.
- head: Used to print the first N lines of a file. It accepts N as input and the default value of N is 10.
- tail: Used to print the last N-1 lines of a file. It accepts N as input and the default value of N is 10.

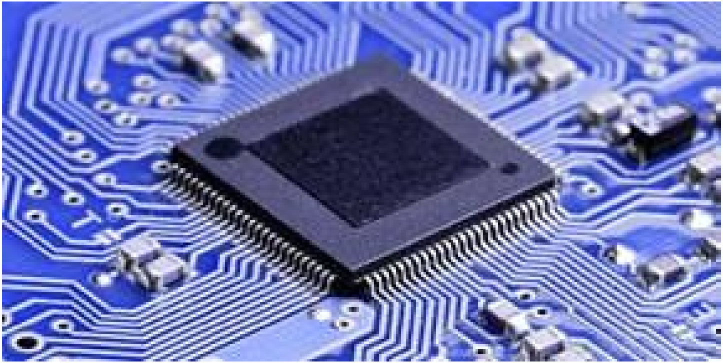
On all systems, commands can be "piped" together: ls | more < *file* >

IoT Fun

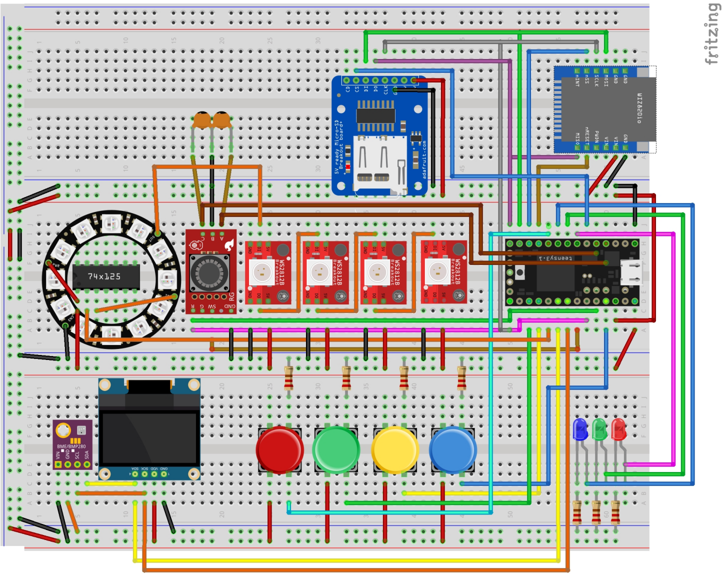

by Nitrozac & Snaggy
FATTYPANTS IS KIDDING HIMSELF IF HE THINKS HE'S EATING LESS THAN 2000 CALORIES A DAY!
AT LEAST YOU GET USED! I NEVER DO! ALL HE EATS IS TAKE-OUT!
HE SPENT A FORTUNE ON US SMARTBULBS AND HE ONLY USES THE "WHITE LIGHT" SETTING! *SIGH*
DUMMY NEEDS TO TEST ME NOW AND THEN, OR AT LEAST CHECK MY APP!
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

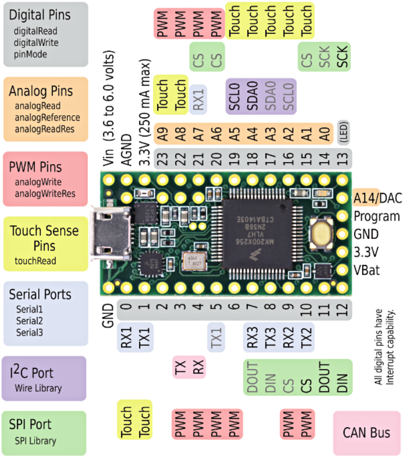


Smart Room Controller



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



Arduino IDE for Teensy

We are going to start off using the Arduino IDE[1]. The Arduino IDE is programmed essentailly using C++ code, but 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

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

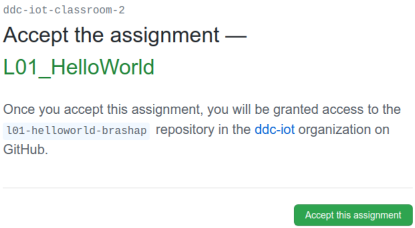
Other Software

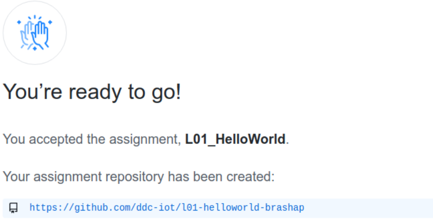
1. Git
   - https://git-scm.com/downloads
2. Fritzing
   - IoT Bootcamp Teams Site
3. Drawio
   - https://app.diagrams.net/
4. Fusion 360
   - Instructional Videos - Teams Site
5. Formlab's Preform
   - https://formlabs.com/software/
6. Ultimaker's Cura
   - https://ultimaker.com/software/ultimaker-cura

GITHUB Simplified

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

// From the repository directory, get updates
  git pull

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

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

GITHUB First Clone



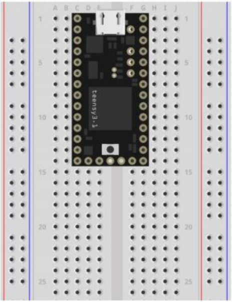



```
brian:~$ cd Documents/
brian:Documents$ mkdir IoT
brian:Documents$ cd IoT
brian:IoT$ git clone https://github.com/ddc-iot/L01_helloWorld-brashap
Cloning into 'L01_helloWorld'...
Username for 'https://github.com': brashap
Password for 'https://brashap@github.com':
  remote: Enumerating objects: 4,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

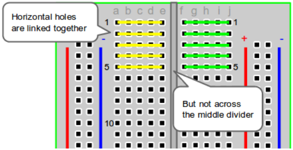
Second Clone: https://github.com/ddc-iot/class_slides

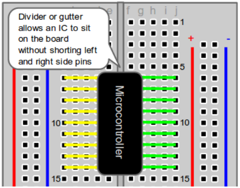
L01_HelloWorld

Teensy on Breadboard

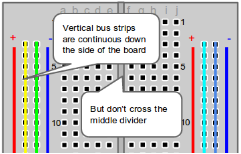


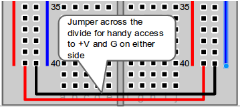
Horizontal Rows





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"
```

Unix Shell

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

echo Hello World
```

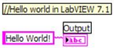
Assembler (Intel)

```
; 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}
```

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

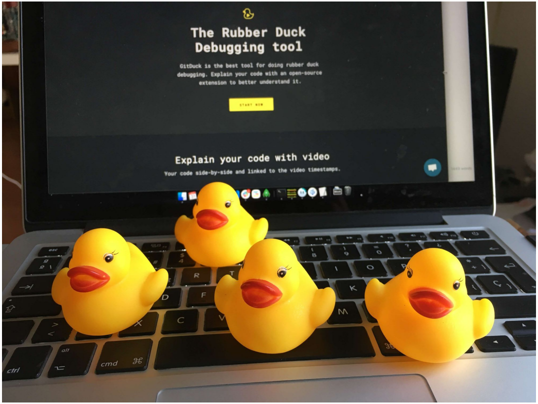
We will write our first program together as a class, using:

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

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

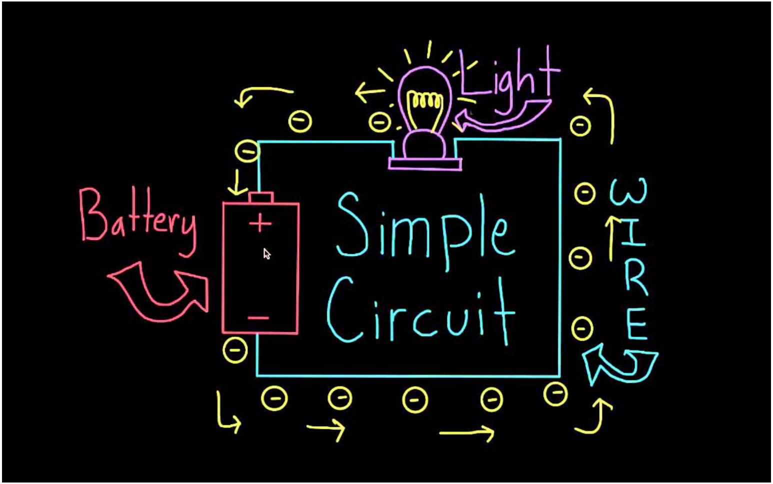
L02_HelloLED

Rubber Ducking - An Odd but Brilliant Tool

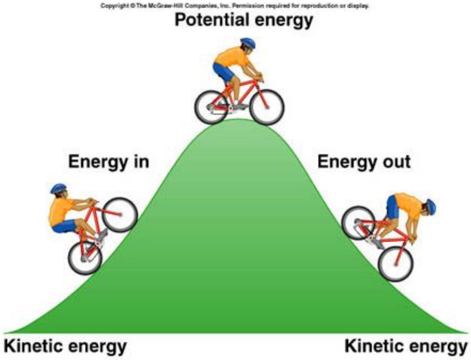


Rubber ducking is simply a method of debugging code. Programmers carry around a rubber duck with them, When they get stuck, they explain their code line-by-line to the rubber duck.

Introduction to Electrical Circuits

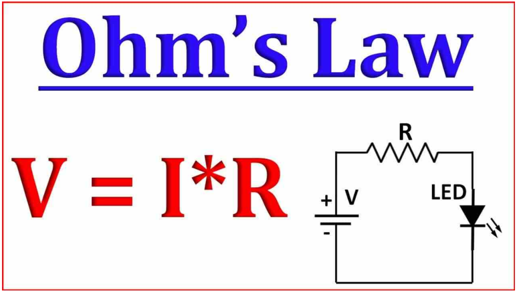


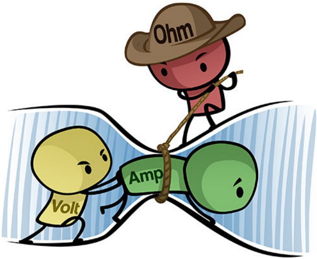
Energy


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

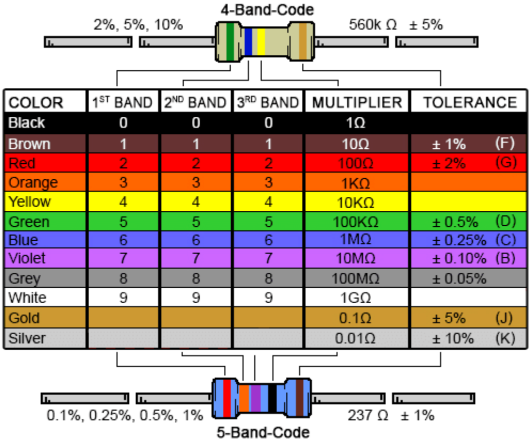
Ohm's Law

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



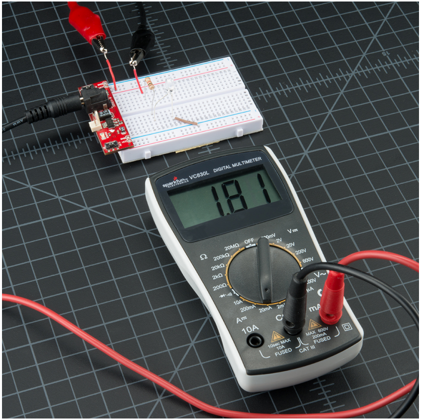


Resistor Color Bands

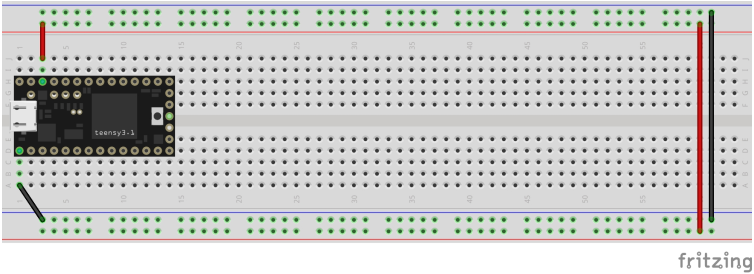


COLOR	1^{ST} BAND	2^{ND} BAND	3^{RD} BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8	100MΩ	± 0.05%
White	9	9	9	1GΩ	
Gold			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

Measuring Voltage, Current, and Resistance



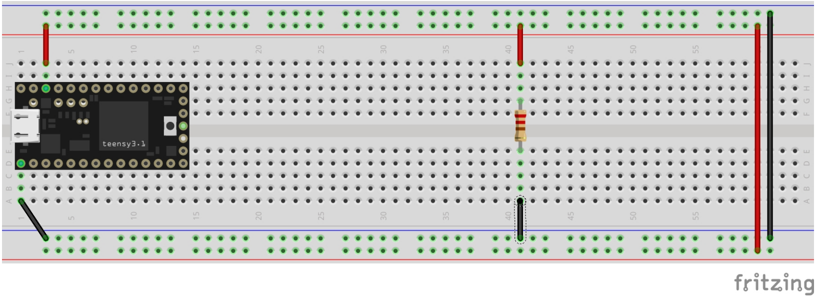
Power from the Teensy 3.2



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

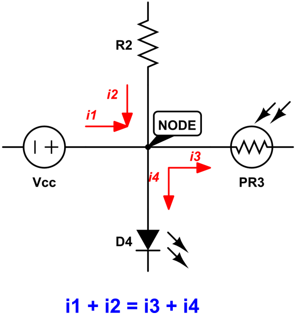


Using your multimeter, measure the voltage "across" and current "through" the resistor.

Kirchhoff's First Law

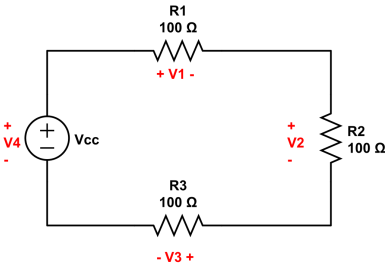
Gustav Robert Kirchhoff (12 March 1824 – 17 October 1887) was a German physicist who contributed to the fundamental understanding of electrical circuits. His first law:

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



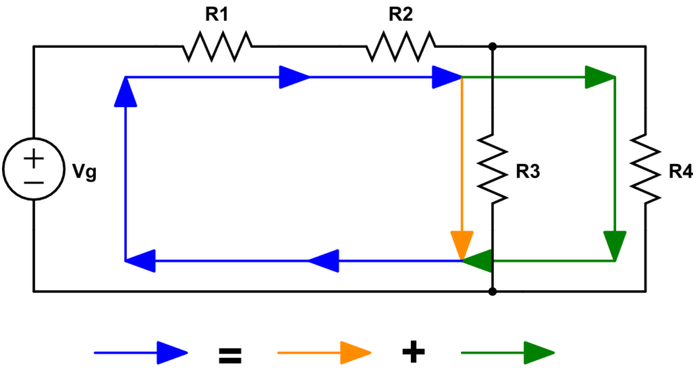
Kirchhoff's Second Law

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

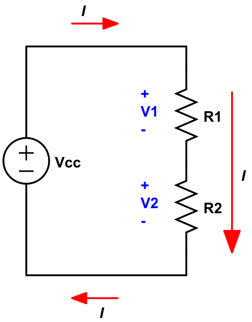


$$V4 - (V1 + V2 + V3) = 0$$

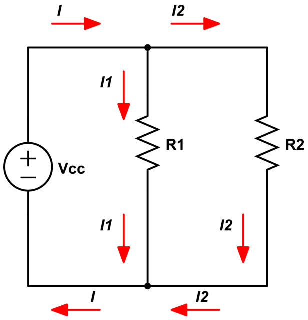
Resistors in Series and Parallel



Resistors in Series and Parallel

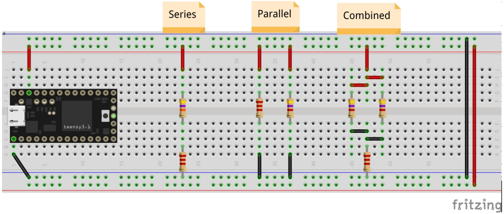


$$R_{eq} = R_1 + R_2$$



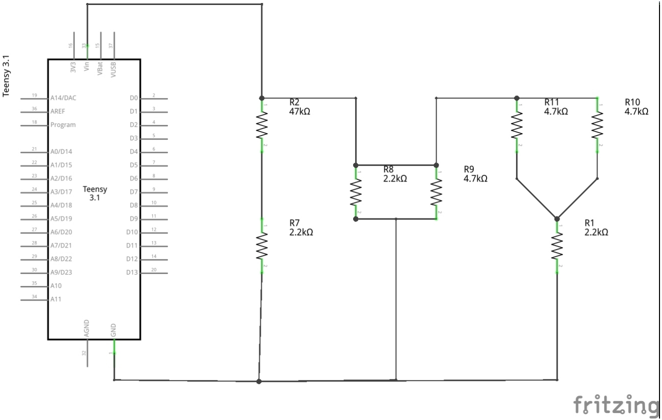
$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2}$$

Assignment: L02_00_Resistors



- In your lab notebook, draw the circuit diagrams
  1. Series: $4.7k\Omega$ and $2.2k\Omega$
  2. Parallel: $4.7k\Omega$ and $2.2k\Omega$
  3. Combined: Two $4.7k\Omega$ in series with $2.2k\Omega$
- Calculate the combined resistance and the voltage at each node, as well as the current through each component.
- Create Fritzing diagram
- Build **(one at a time)** on your breadboard and test your calculations with a multimeter.

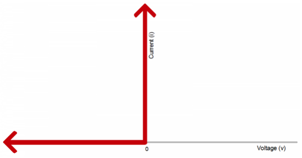
Schematics in Fritzing

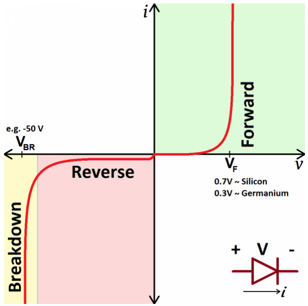


In Fritzing, go to the Schematic tab and layout your resistors.
NOTE: The schematic for the Teensy does not match it's physical layout.

Diodes

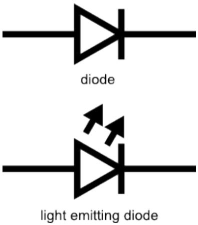
The key function of a diode is to control the direction of current-flow. Current passing through a diode can only go in one direction, called the forward direction. Current trying to flow the reverse direction is blocked.





Light Emitting Diodes

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

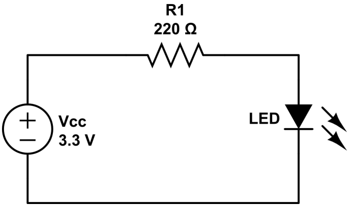


Current Limiting Resistors

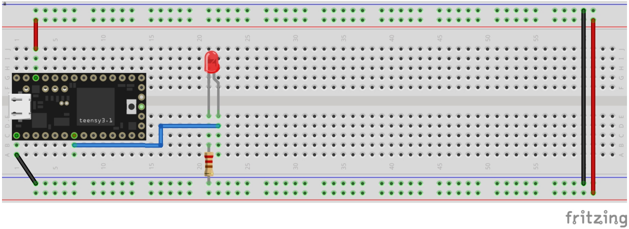
As a LED has very little resistance, when it is connected directly to a power supply, the current draw will exceed its specs and it will burn out.

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

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



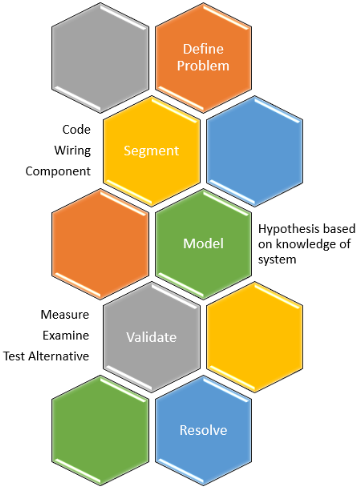
Assignment L02_01_helloLED



- Using Pin 5 as an output and the appropriate current limiting resistor, blink the LED once per second.
- Measure the voltage at both leads of the LED and record 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

Model Based Troubleshooting



Constants and Variables

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

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

The use of Constants and Variables has several advantages:

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

The first two Data Types that we will be using:

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

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;
}
```

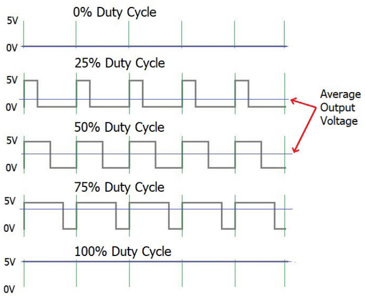
Assignment L02_02_helloLEDvar

- Convert L02_01_helloLED with constants and/or variables.

Pulse Width Modulation

Software Configurable:

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



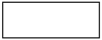
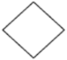
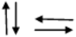
Assignment L02_03_helloLEDanalog

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

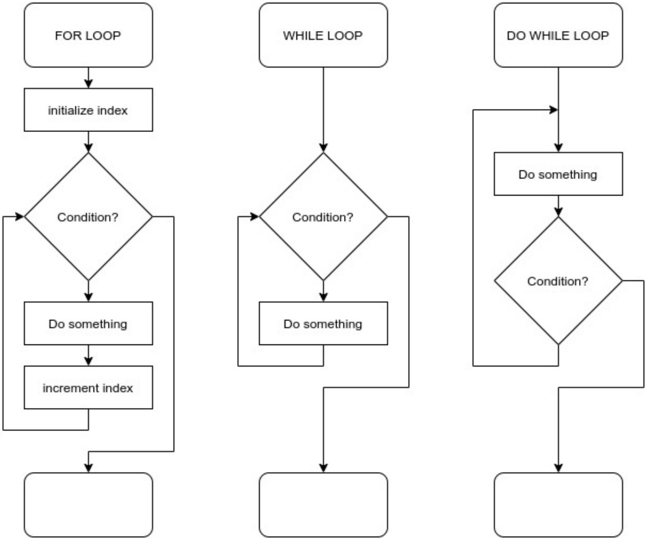
- 255
- 63
- 171
- 16

Measure the voltage with your multimeter at each value.

Flowcharts

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

Loops


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

FOR Loop syntax

```
// FOR loop syntax
for (initialization; condition; increment) {
  // statement(s);
}

// EXAMPLE
for (j=0; j <= 255; j++) {
  analogWrite(LEDPIN, j);
}
```

WHILE loop syntax

```
// WHILE loop syntax
while (condition) {
  // statement(s)
}


// EXAMPLE
while (button == HIGH) {
  digitalWrite(LEDPIN, HIGH);
} //continue this loop until button is released
```

For vs While Loops

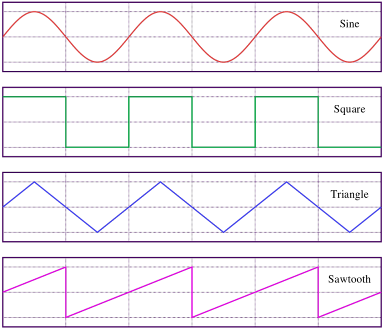
For vs While Loop

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.

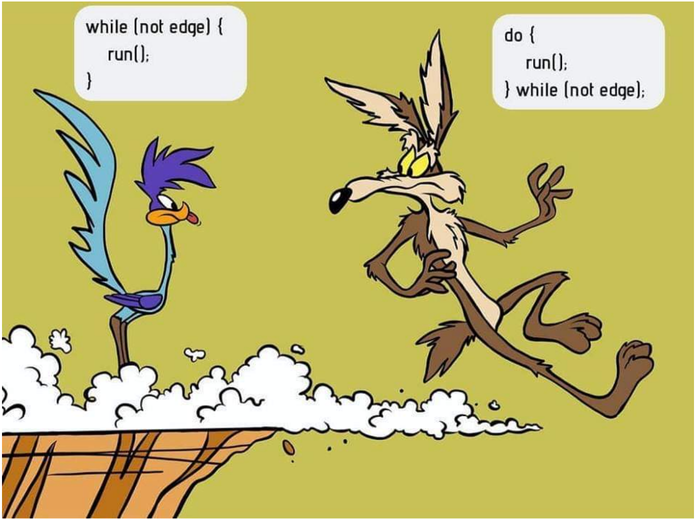
DifferenceBetween.net

Assignment L02_04_helloLEDtri

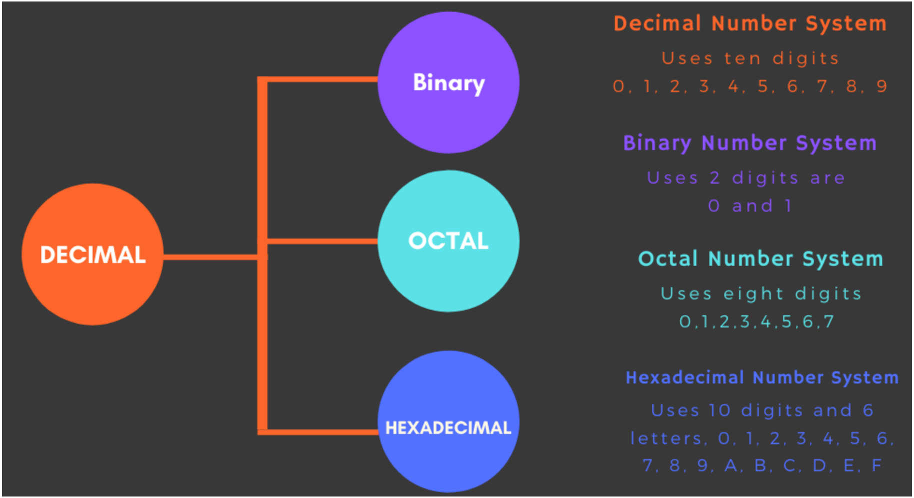


Using a FOR Loop, have the LEDs follow a Triangle Wave function from off to full brightness with a period of 10 seconds.

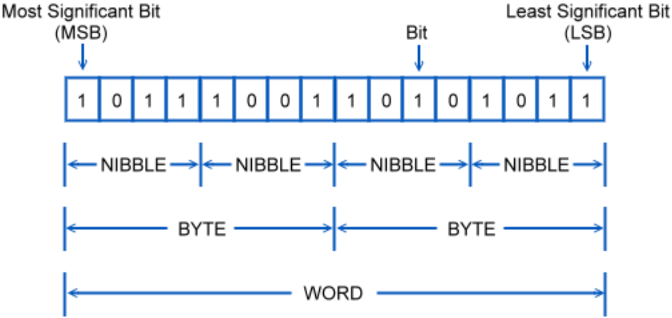
While or Do While



Number Systems



Bits, Nibbles, Bytes, and Words



Data Types: Numbers

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

Header Files

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

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

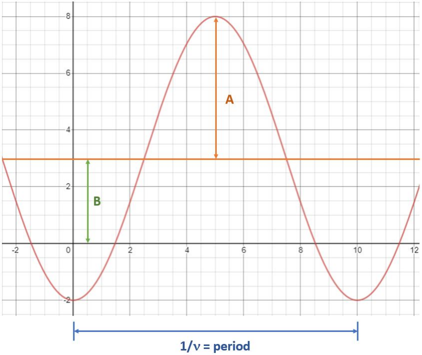
- #include $< file.h >$ for system header files.
- #include "*file.h*" for user created header files in the directory that contains the current code.

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

Basic Structure of Arduino Sketch Revisited

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

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

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

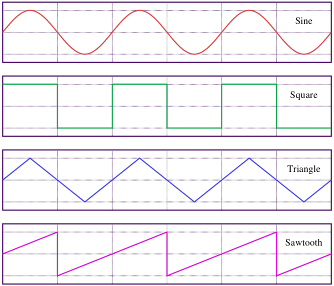
Sine Waves



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

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

Assignment L02_05_helloLEDsin



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

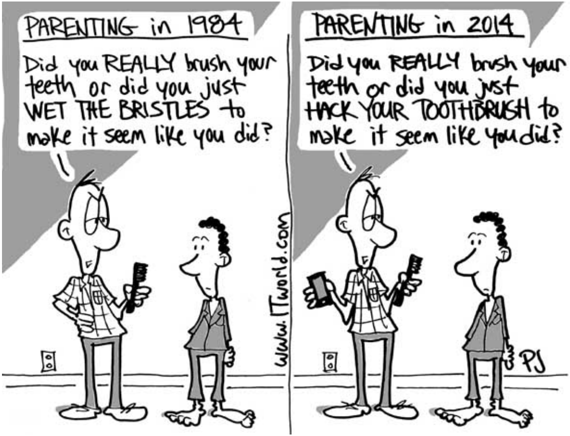
- Use math.h
- Function sin takes a double as an input and returns at double.
- Set the period to 5 seconds.

The function millis() returns milliseconds since 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?
www.ITworld.com
PARENTING in 2014
Did you REALLY brush your teeth or did you just HACK YOUR TOOTHBRUSH to make it seem like you did?
PJ

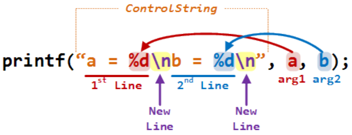
Displaying to the Screen: The Serial Monitor

```
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



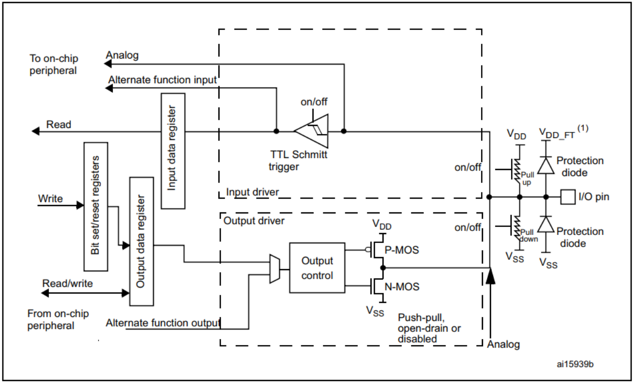
specifier	Output	Example
d *or* i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation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

```
int count = 42;
float value = 3.14159;
Serial.printf("Print an integer %i and a float %0.4f\n",count,value);

\\ 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();

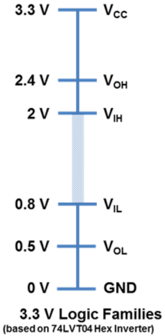
One Pin - Many Functions



Software Programmable: Input or Output and Digital or Analog.

Digital Input/Output

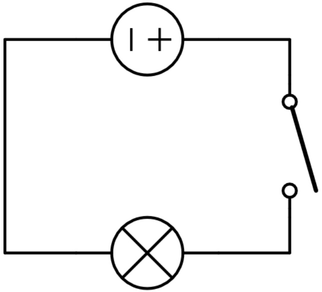
Digital electronics rely on binary logic to store, process, and transmit data or information. Binary Logic refers to one of two states – ON or OFF. This is commonly translated as a binary 1 or binary 0. A binary 1 is also referred to as a HIGH signal and a binary 0 is referred to as a LOW signal.



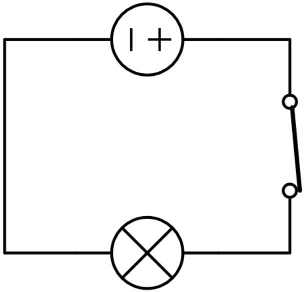
- digitalWrite(pin,value);
- inputValue = digitalRead(pin);

where, value equals HIGH or LOW.

Switches

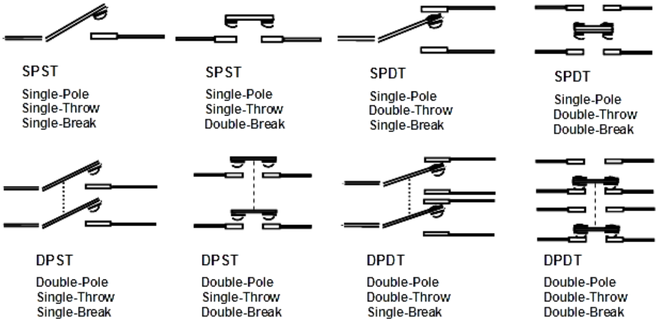


Lamp Off

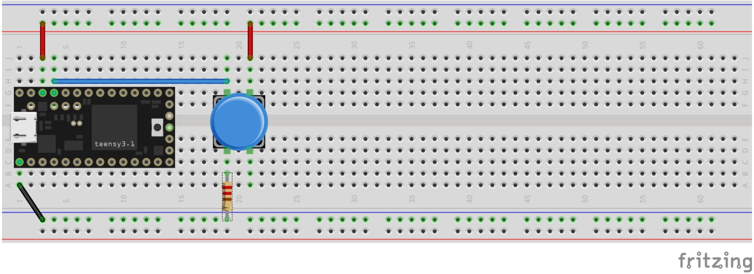


Lamp On

Types of Switches



Our First Button and Pull Down Resistors



Wait - why is there a resistor connected to ground?

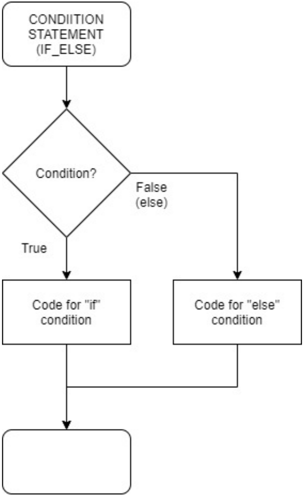
Assignment: Buttons

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

Connect a button to Pin 23 and a multimeter to measure the voltage at the pin. Using digitalRead():

1. L03_01_button
    - 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);

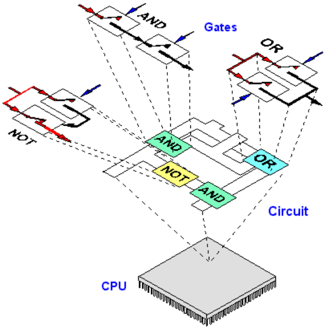
IF-ELSE Statements



IF-ELSE Statements

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

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

Data Types: Boolean and Boolean Logic



Boolean datatype (bool) holds either a TRUE or FALSE

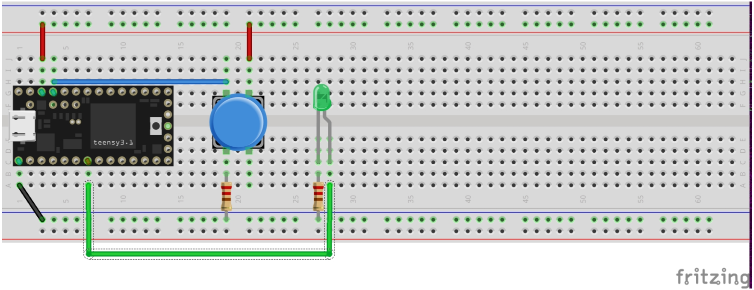
Boolean Logic Operations (condition statements)

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
   - $z = x \parallel y$

Boolean Logic In Action

```
bool x;
bool buttonState1, buttonState2;
int y, z;

x = !x;        // invert x

if (!x) {      // if x is false
  // do something
}

// LOGICAL AND: if both pins are pressed
buttonState1 = digitalRead(PIN1);
buttonState2 = digitalRead(PIN2);
if ((buttonState1) && (buttonState2) {
  // do something
}

// LOGICAL OR: if either value is greater than zero
if (y > 0 || z > 0) {
  // do something
}
```

Button and LED

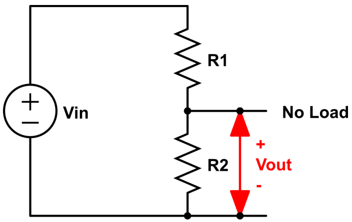


Assignment: Buttons and LEDs

1. L03_03_buttonLED
   - Add an LED to Pin 5 and use the button to turn the LED on and off.
   - Also, print button state to the screen
2. L03_04_twobuttonLED
   - Add a second button (Pin 16) and LED (Pin 6)
   - Have each button control one LED
   - Also, print button states to the screen

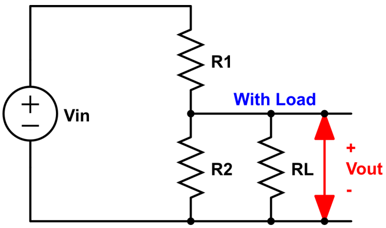
Resistors in Series and Parallel

Open Circuit Behavior



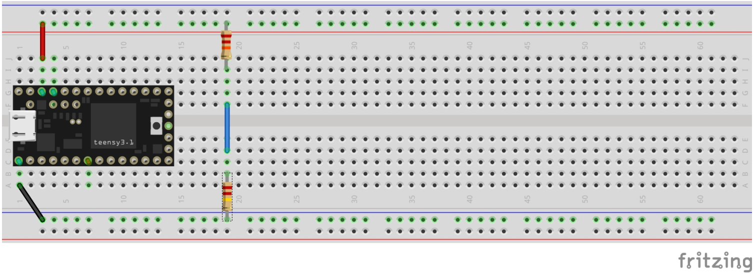
$$V_{out} = V_{in}\frac{IR_2}{I(R_1+R_2)} = \frac{R_2}{R_1+R_2}V_{in}$$

Behavior Under Load



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

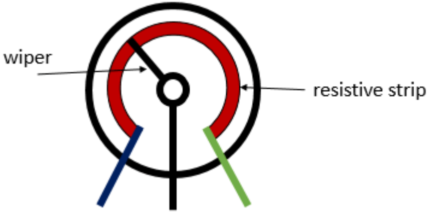
Voltage Dividing



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

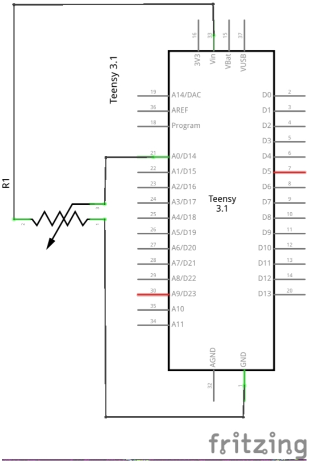
- Use various combinations of resistors between $1k\Omega$ and $22k\Omega$.
- Calculate the Series resistance and the voltage between the two resistors in your Lab Notebook.
- Measure with your multimeter and compare.

Potentiometer - Variable Resistor



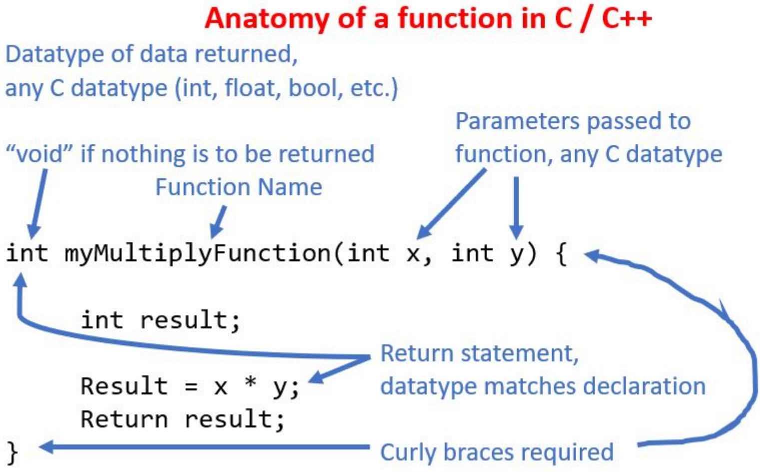
A potentiometer has 3 pins. Two terminals (the blue and green) are connected to a resistive element and the third terminal (the black one) is connected to an adjustable wiper.

Assignment L03_05_AnalogInput



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

Anatomy of a Function



Types of Variables

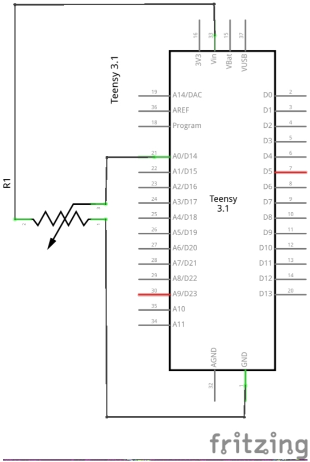
```
void setup() {
  x = 1;
}
void loop() {
  x = addx();
}
int addx() {
   int y;
   static int z;

   y = y + x;
   z = z + x;
   return y;
}
```

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.
   - Destroyed only when program is terminated.

Basic Structure of Arduino Sketch Revisited

```

int num, doubleNum;

void setup() {
  Serial.begin(9600);  // Turn on Serial Monitor
  while(!Serial);      // Wait for Serial Monitor to be running
  num = 1;             // Initialize num;
}

void loop() {
  doubleNum = twotimes(num);  // call the funciton twotimes
  num = doubleNum;
  Serial.printf("The number is now %i \n", num);
  Serial.printf("The number in HEX is 0x%X \n", num);
  delay(1000);
}

int twotimes(int number) {
  int answer;              // declare answer as a local variable
  answer = 2 * number;
  return answer;
}
```

Assignment L03_05_AnalogInput Revisited



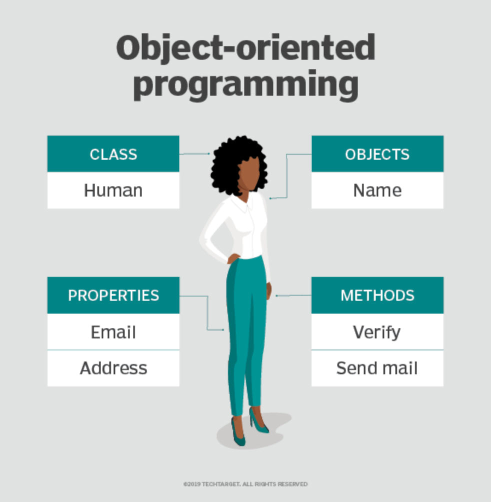
1. Modify your code by adding a function, in2volts(), that converts the analog input value to voltage.
2. Print both the raw analogInput and the associated voltage to your screen.

L04_oneButton

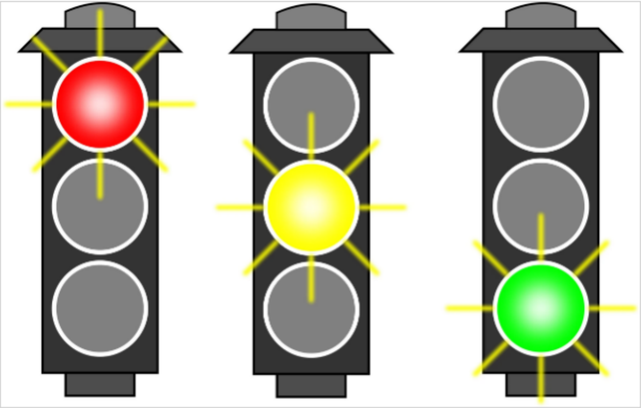
IoT Humor

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

Objects

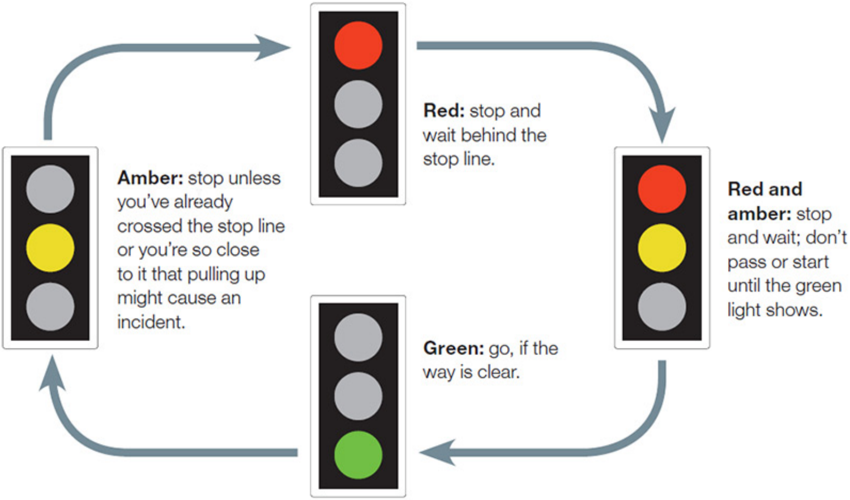

Object-oriented programming
CLASS
Human
OBJECTS
Name
PROPERTIES
Email
Address
METHODS
Verify
Send mail
©2019 TECHTARGET. ALL RIGHTS RESERVED

Traffic Light



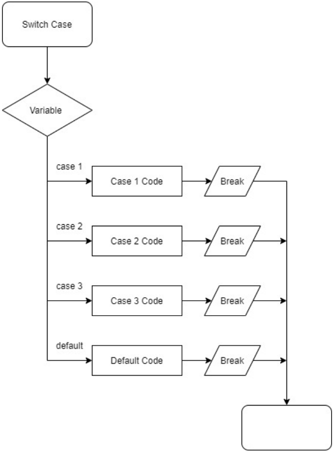
Let's use the traffic light to build our own Objects

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)

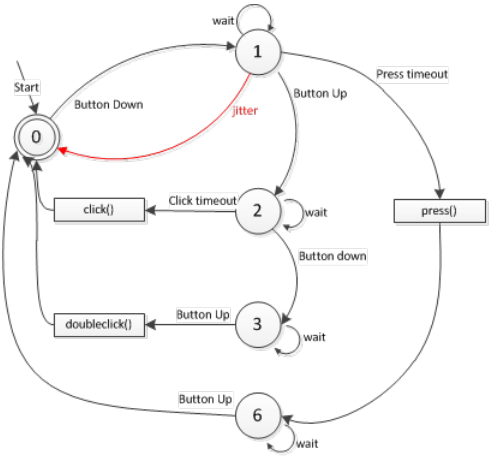
The C-language has a user defined datatype, enum, which allows the user to create a variable with various names for each of its states.

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

```c
// Datatype State: the four traffic light states
enum State{
  GREEN,
  YELLOW,
  RED,
  RED_YELLOW
};  //note the ; after the }
```

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

OneButton Library



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

Basic Structure of Arduino Sketch - Revisited

```
// the "header" is used for GLOBALS
#include <OneButton.h>              // system library files
#include "TrafficLight.h"           // local library files

OneButton button1(23, false);     // define objects

const int GREENLED = 23;          // declare global constants
bool greenState;                  // declare global variables

// setup() is used for code that runs once at the beginning
void setup() {
  Serial.begin(9600);             // begin processes
  pinMode(GREENLED,OUTPUT);       // define input/output modes
  button1.attachClick(click);     // initialize objects
  greenState = false;             // set variables
}

// loop() is used for code that runs continuously
void loop() {
  tick();
  digitalWrtie(GREENLED,greenState);
}

// user defined functions
void click() {
  greenState = !greenState;
}
```

OneButton Declarations

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

Using OneButton

```
void loop() {
  button1.tick();// check the state of the button
}

void click1() {
  Serial.println("Hi, my name is Brian.");
}

void doubleClick1() {
  Serial.println("Hope your day is going well.");
}
```

Assignment: OneButton

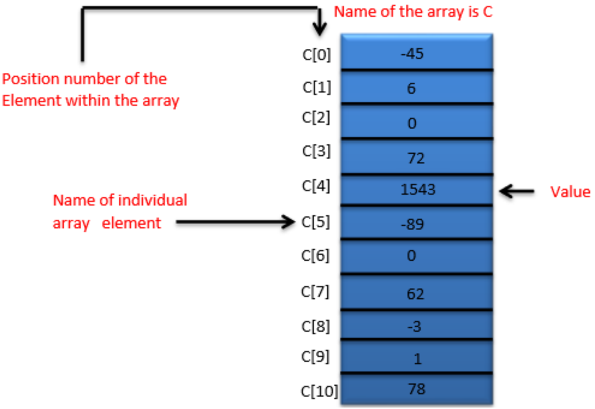
- Notebook: flowchart
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L04_01_oneButton
   - Use OneButton libary and button on Pin 23.
   - Click() - toggle bool variable buttonState.
   - doubleClick() - toggle bool variable blinker.
2. L04_02_oneButtonLED
   - Single Click: toggle LED on/off using buttonState variable.
   - Double Click: when LED is on, toggle between solid and rapidly blinking (50ms) using blinker variable.
   - What happens if blinking is made slower (try > 1 second)?

Avoiding Delays

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

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

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

Arrays



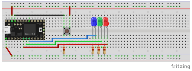
- Syntax: datatype var[ ] = {element 1, element 2, element 3};
- Example: int ledArray[ ] = {greenPin, yellowPin, redPin};

Using Arrays

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

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

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

Assignment: OneButton



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

1. L04_04_oneButtonArray
   - Use 3 LEDs and one Button
   - 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
   - Long Press (end) - light up the three LEDs in reverse order

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()

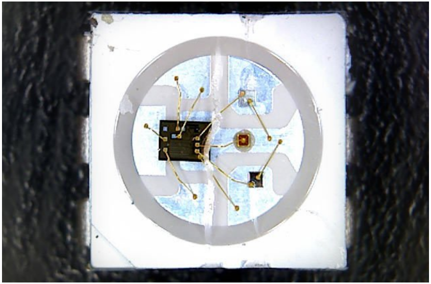
```
String input;
int value;

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

void loop() {
  input = "";
  while(input=="") {
    input = Serial.readStringUntil('\n');
  }
  value = input.toInt();
  Serial.printf("The number you entered is %i \n",value);
}
```

Assignment: Serial Read

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

L05_NeoPixels

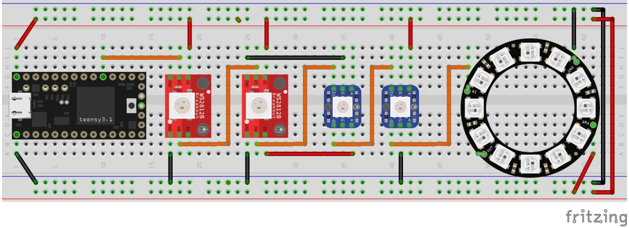
NeoPixels

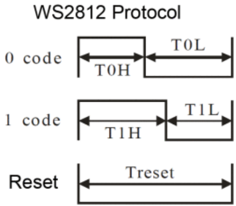


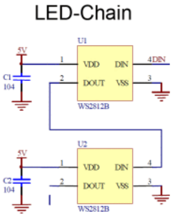
NeoPixels are:

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

NeoPixel Programming



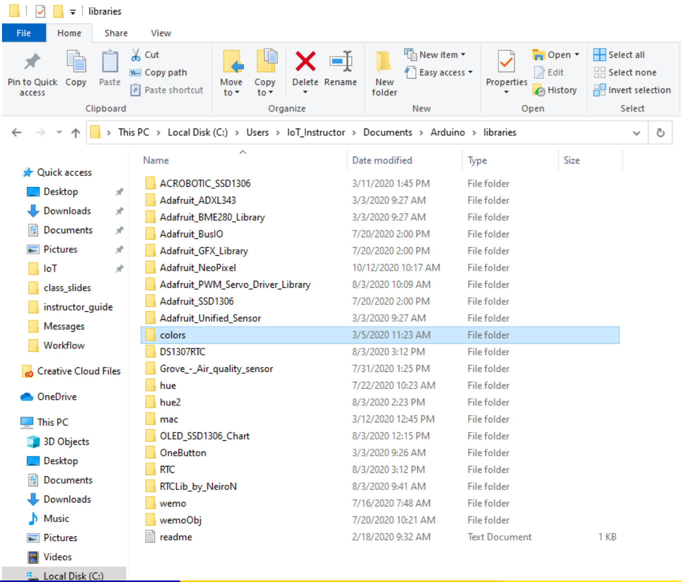




Using NeoPixel Class and Methods

```
#include <Adafruit_NeoPixel.h>

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

Adafruit_NeoPixel pixel(LED_COUNT, LED_PIN, 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?



Generating Random Numbers

```
/*
 * The random() 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.println(randNumber);

  // print a random number from 10 to 19
  randNumber = random(10, 20);
  Serial.println(randNumber);
```

Assignment: NeoPixels

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

1. L05_01_neoPixel
   - With a FOR loop, light up 4 pixels and ring. Small delay between pixels.
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 6 segments of different colors using the fill() method

L06_Encoders

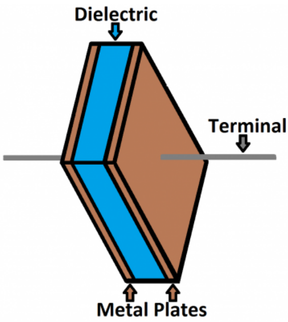
IoT Humor


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

Capacitors

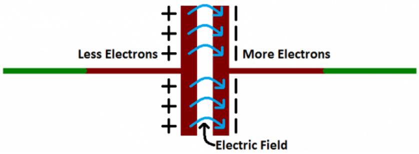
A capacitor is created out of two metal plates and an insulating material called a dielectric. The metal plates are placed very close to each other, in parallel, but the dielectric sits between them to make sure they don't touch.

- 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

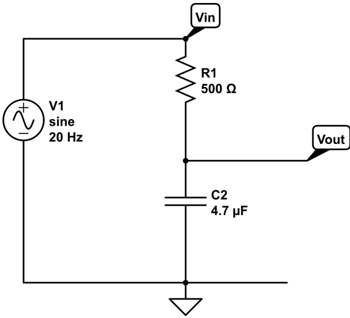
When current flows into a capacitor, the charges get "stuck" on the plates because they can not get past the insulating dielectric. Electrons build up on one of the plates, and it becomes overall negatively charged. The large amount of negative charges pushes away like charges on the other plate, making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



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

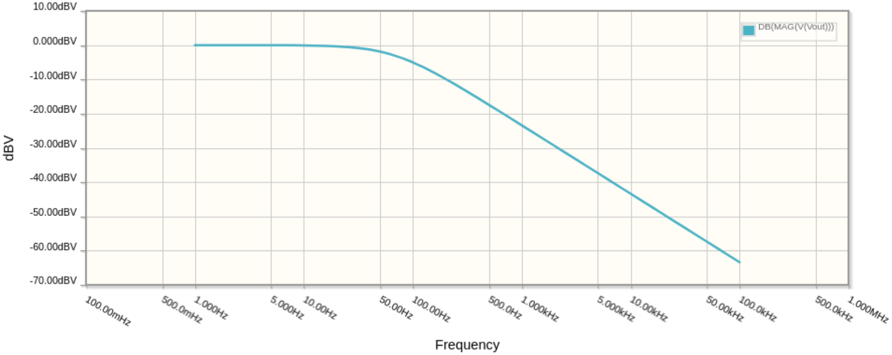
Low Pass Filter - cutoff frequency f_c

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



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

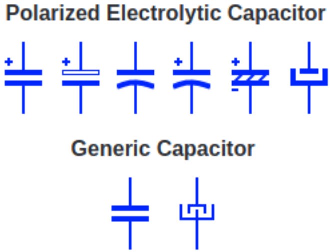
Low Pass Filter Response



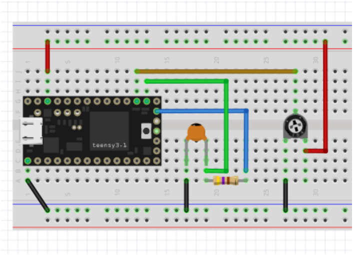
$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi(500)(4.7x10^{-6})} = 67.5678Hz$$

Capacitors - does it matter how they are placed

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



Assignment: Low Pass Filters

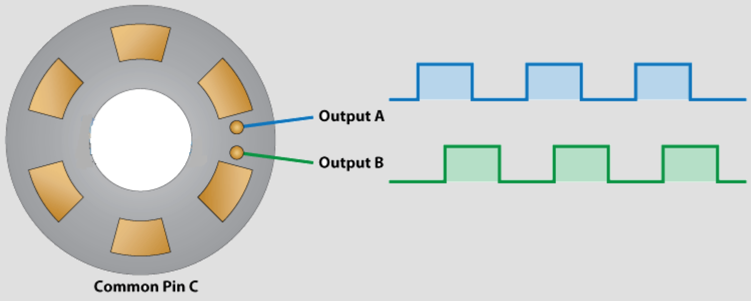


Using Serial Plotter:

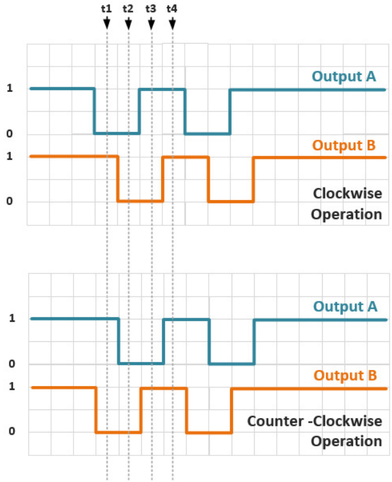
- Serial.begin(9600);
- Serial.printf("%i , %i, %0.3f \n",d1,d2,d3);
- Close Serial Monitor

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

Encoders



Encoders



Clockwise Sequence

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

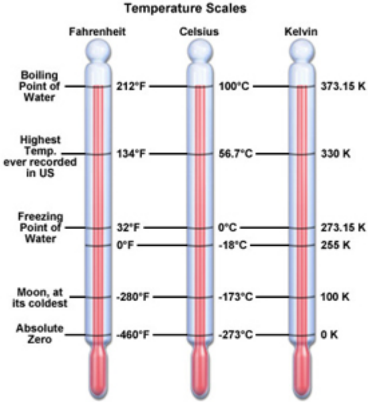
Counter-Clockwise Sequence

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

The Encoder Class

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

void setup() {
}

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

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

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);

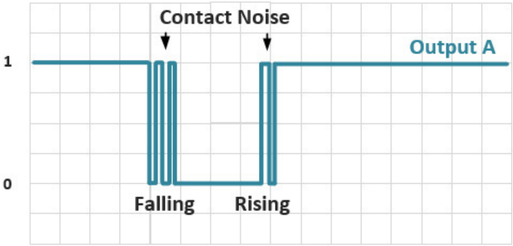
Assignment: Encoders

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

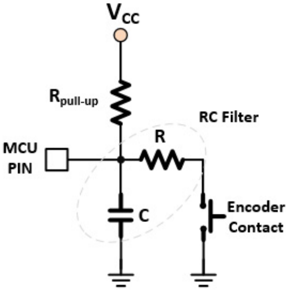
1. L06_01_encoder
   - Display the encoder position to the screen
2. L06_02_encoderScaled
   - The encoder has 96 positions. Mathematically map (without using the map() function) the encoder to 12 pos (0-7 = 0,8 - 15 = 1, etc.). Show your work to an instructor before moving on to the map() function.
   - Next, use the map() function.
3. L06_03_encoder_NeoPixel
   - Use the encoder to light up the pixel ring

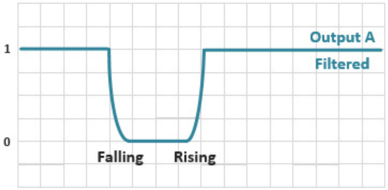
Reminder of the syntax of the map() function:
newVal = map(value, fromLow, fromHigh, toLow, toHigh)

Encoder Jitter



Encoder - Low Pass Filter





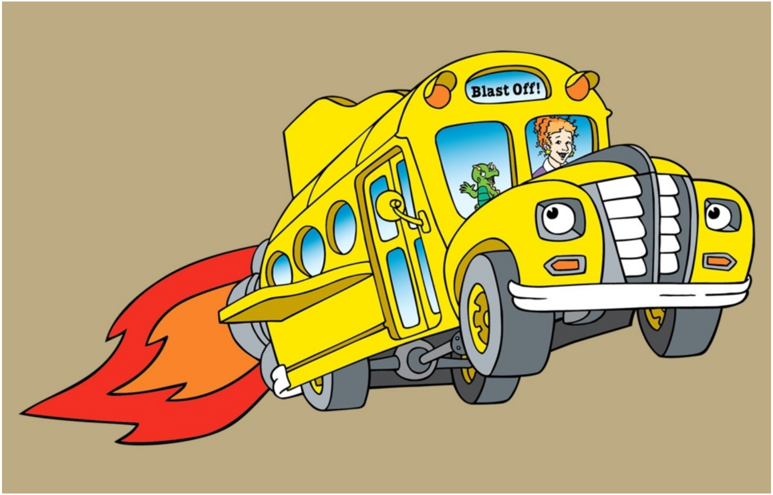
Assignment: Encoders

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

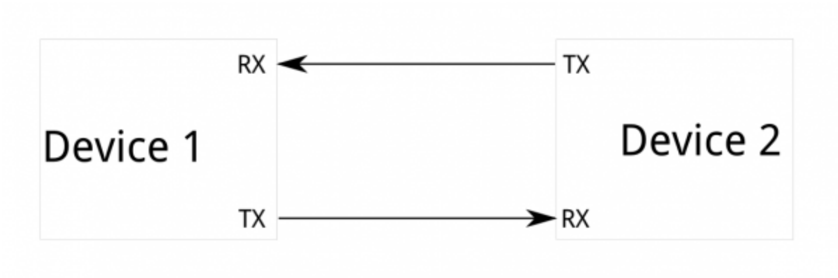
1. L06_04_encoder_switch
   - Connect your microcontroller to the encoder switch and LEDs
   - Use the switch to turn on/off the NeoPixels.
   - Also, the encoder LED should be red for off and green for on.
2. L06_05_rainbow1 - extra credit
   - Without OneButton: Use a button to cycle the NeoPixel ring colors through the colors of the rainbow. (i.e., one color change each time button is pressed).
3. L06_06_rainbow2 - extra credit
   - With OneButton: Have it continuously cycle (i.e, while pressed colors change every one second).

L07_SPI

Buses and Interfaces

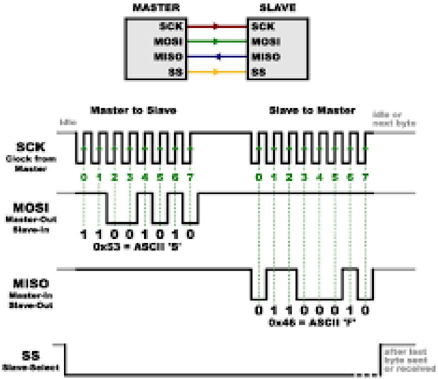


UART



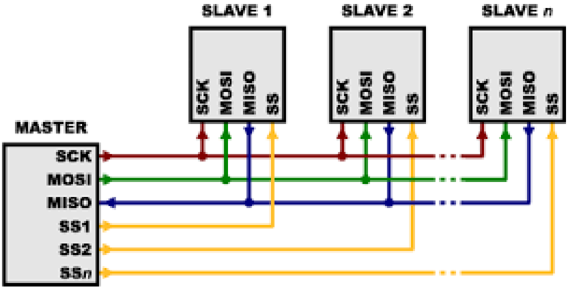
Universal Asynchronous Receiver/Transmitter

Serial Peripheral Interface



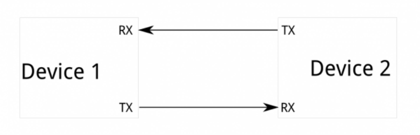
- Master Out, Slave In (MOSI) connects to Data In
- Master In, Slave Out (MISO) connects to Data Out

Serial Peripheral Interface

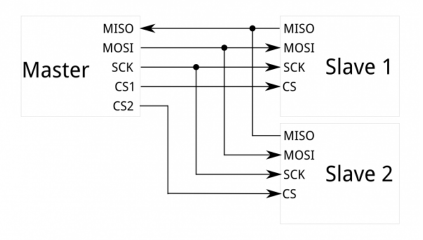


Serial +/−

UART



SPI



Assignment: L07_01_dataLogger

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

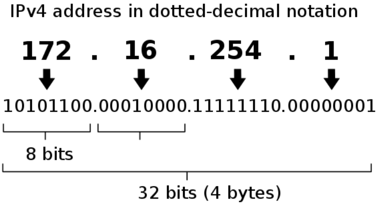
1. The starter code details the pin assignments for SPI, use these for your schematic and Fritzing.
2. Modify the starter code to:
   - Read two inputs:
     1. The value of the encoder (unbounded)
     2. The Pin 22, left floating (connected to nothing)
   - Write to μSD Card a timestamp and the two input values every 5 seconds.

L08_Ethernet

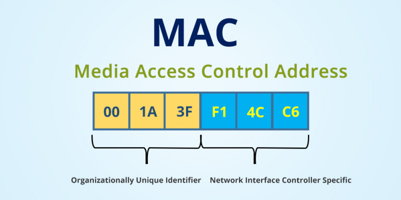
The Internet

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 quadrilliard
- In Powershell, try: *ipconfig/all*
- In Terminal (MAC), try: ipconfig getifaddr en0



MAC Address



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

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

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

Assignment: Wemo

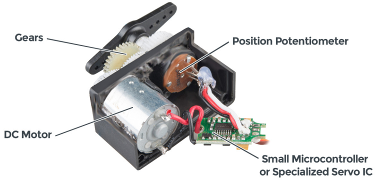
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

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
   - From the wemo.h library determine how the functions are called.
   - Create code to turn on/off multiple Wemo Outlets in the classroom.
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
   - Modify the wemo.h library to be a Class and Methods.
   - Modify your wemo code to create and use a wemo 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
3. void loop()
   - myServo.write(angle) - move servo to angle (in degrees)
   - myServo.read() - returns the current angle of the servo

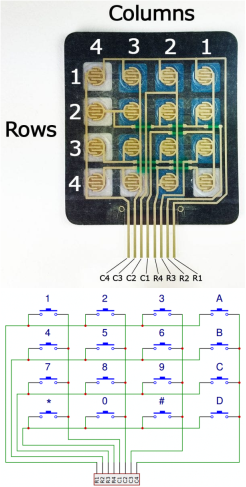
Assignment: L09_Servo

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

1. L09_01_Servo
   - Connect the servo and a button to the Teensy
   - Have the servo oscillate between 0 and 180 degrees using a sine wave pattern.
   - Use the button to start and stop the motion.

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

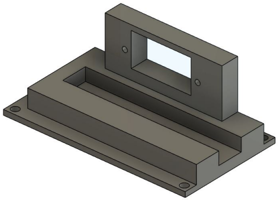


- The Keypad is an two dimensional array of buttons.
- Pushing a button connects one row pin with one column pin.
- We will use the Keypad.h library to access the Keypad.

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};
byte colPins[COLS] = {5, 4, 3, 2};

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

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

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

  if (customKey){
    Serial.println(customKey);
  }
}
```

Assignment: L09_02_Lock

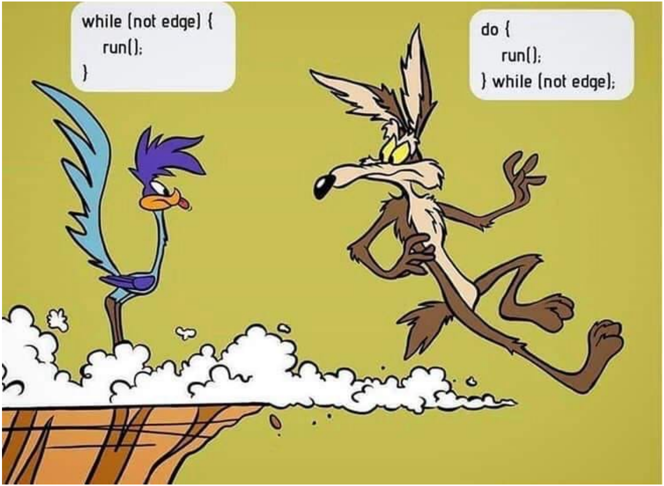


- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

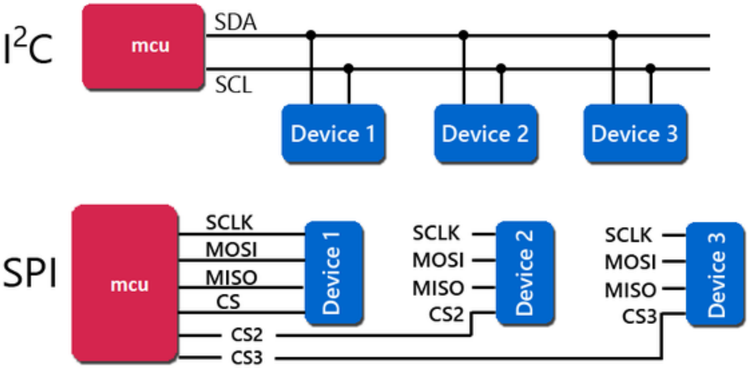
1. Design and print a lockholder based on the class example.
   - Optional: design and print your own gear and lock slider
2. Using the keypad, create an digital lock
   - Implement 4-digit code to lock and unlock
   - Use the servo to lock and unlock based on a correctly entered code
   - Green LED and lock disengaged when unlocked
   - Red LED and lock engaged when locked.

L10_I^2C

IoT Humor



Inter-integrated Circuit (I^2C)



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

- Create a Fritzing diagram for adding the BME280 and the OLED to your Teensy.
- 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
- Wire up your schematic and determine the I2C addresses of each device.

Char Datatype

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	énye, 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)

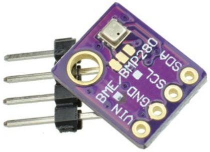
```
const char degree = 0xF8;  // Decimal 248 = 0xFB
float temp = 98.6;
void setup() {
  Serial.begin(9600);
  //NOTE: extended ASCII characters don't always print correct to Serial Monitor
  Serial.printf("My temperature is %0.1f %c",temp,degree);
}
```

Assignment: I^2C

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

1. L10_01_OLEDWrite
   - Install Adafruit_SSD1306 library
   - Review and run SSD_1306_128x32_i2c example
   - Create variables for your birthday. Using the syntax from testdrawstyles() print using printf():
     - Hello World
     - Your Name using spanish honorific (señor, señora, señorita).
     - Your Birthday
   - Experiment with rotating the screen using the setRotation(int) method.

Assignment: BME280



1. Header - define BME280 object
   - I2C: Adafruit_BME280 bme();
   - SPI: Adafruit_BME280 bme(BME_CS);
2. Setup - start BME280
   - status = bme.begin(hex address);
   - if(status==false) → initialization failed
3. Loop - read from sensor
   - tempC = bme.readTemperature();
   - pressPA = bme.readPressure()/100.0;
   - humidRH = bme.readHumidity();

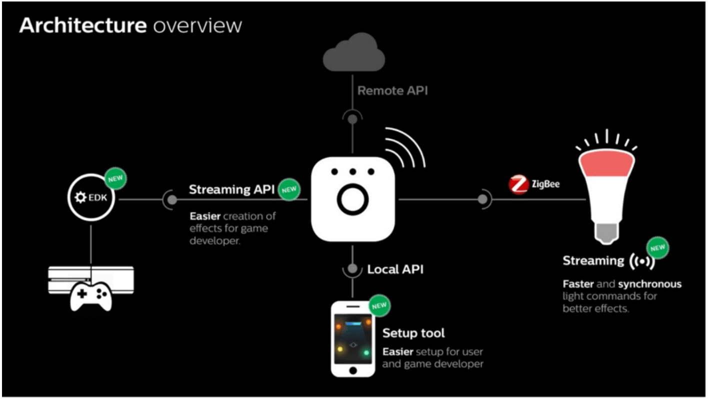
Assignment: I^2C

- 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
2. L10_03_BME280_OLED
   - Print data to the OLED display
3. L10_04_BME280_SDMicro
   - Add in saving data to the μSD card
   - Use your NeoPixels to give a visiual indication of room conditions

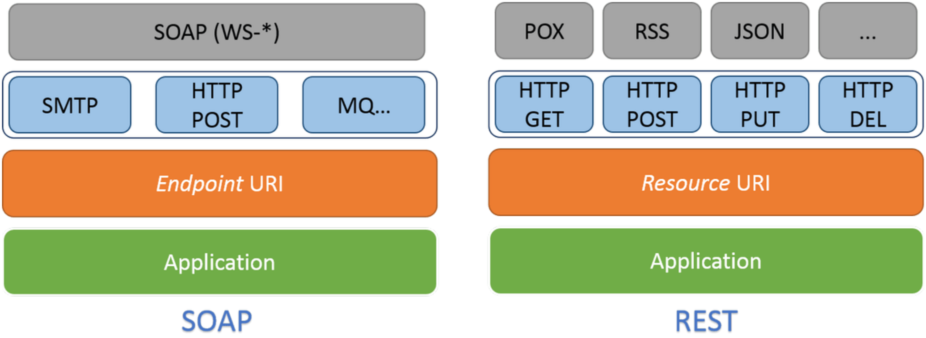
L11_Hue

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

Midterm 1 - Smart Room Controller

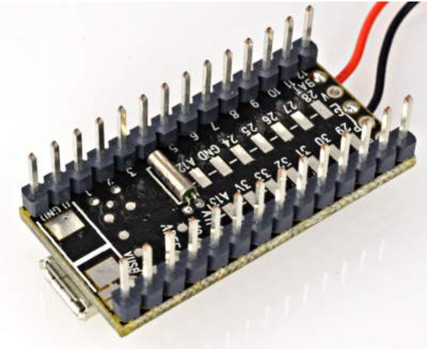
Getting with GIT

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)

Supplemental - Real Time Clock

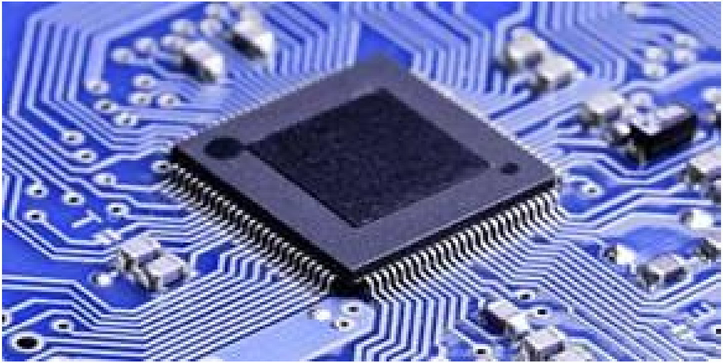
To use the Teensy 3.2 RTC, you need to add a 32.768 kHz, 12.5 pF crystal to the bottom side of the board.



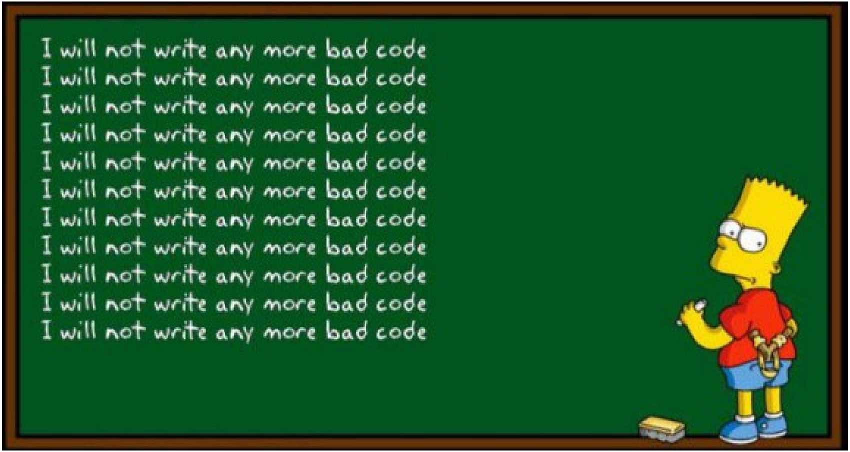
Example can be found at FILE → Examples → Time → TimeTeensy3.

Particle Argon

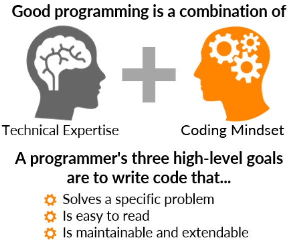
Our Second Microcontroller



Expectations for the rest of the course



Expectations for the rest of the course



1. Most important: Be Consistent
2. Proper Indentation
3. Brace placement: K&R or BSD
4. Do not check boolean for equality.
5. A variable's name (noun) should describe its contents.
6. A function's name (verb) should describe the set of actions it performs.
7. Reduce duplication / Modularize
8. So, what about capitalization?
    - variableNames
    - nameFunction
    - CONSTANTS
    - Classes and Enum

Previous Capstones

Previous Capstone Playlist

https://www.youtube.com/watch?v=s4TslpITeVw&list=PL0t2Pk5ETDgxfVptdyr6xbL6MW1-5CJey

Particle Argon

Main processor:

Nordic Semiconductor nRF52840 SoC

- ARM Cortex-M4F 32-bit processor @ 64MHz
- 1MB flash, 256KB RAM
- Bluetooth LE (BLE) central and peripheral support
- 20 mixed signal GPIO (6 x Analog, 8 x PWM), UART, I2C, SPI
- Supports DSP instructions, HW accelerated Floating Point Unit (FPU) calculations
- ARM TrustZone CryptoCell-310 Cryptographic and security module
- Up to +8 dBm TX power (down to -20 dBm in 4 dB steps)
- NFC-A radio

Argon Wi-Fi network coprocessor:

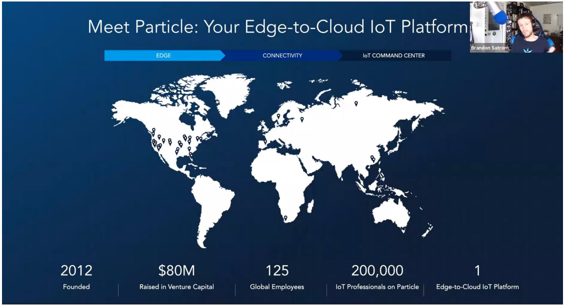
Espressif ESP32-D0WD 2.4 GHz Wi-Fi coprocessor

- On-board 4MB flash for the ESP32
- 802.11 b/g/n support
- 802.11 n (2.4 GHz), up to 150 Mbps

Argon general specifications:

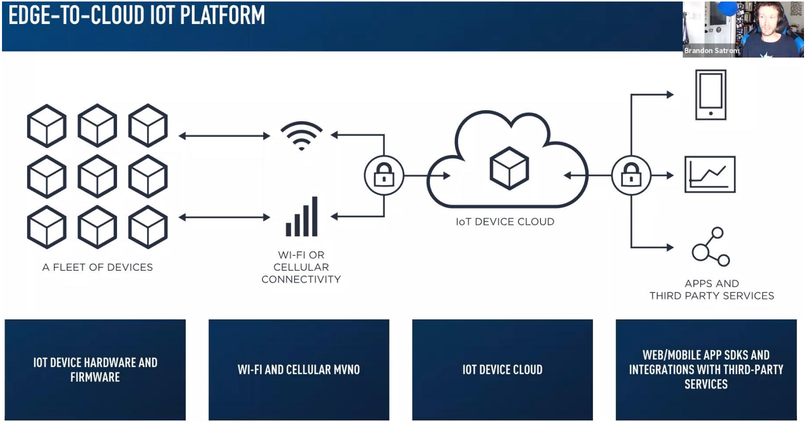
- On-board additional 4MB SPI flash
- Micro USB 2.0 full speed (12 Mbps)
- Integrated Li-Po charging and battery connector
- JTAG (SWD) Connector
- RGB status LED
- Reset and Mode buttons
- On-board 2.4GHz PCB antenna for Bluetooth (does not support Wi-Fi)
- Two U.FL connectors for external antennas (one for Bluetooth, another for Wi-Fi)
- Meets the Feather specification in dimensions and pinout
- FCC, CE and IC certified
- RoHS compliant (lead-free)

Why Particle



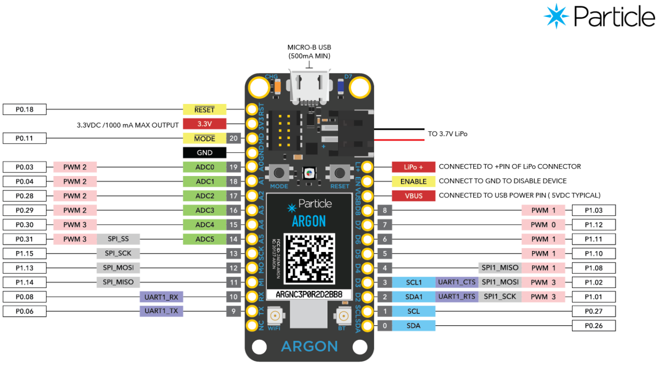
- Global reach with over 200,000 IoT professionals
- Edge-to-Cloud infrastructure
- Prototyping to Production with same code
- WI-FI, Bluetooth, and Cellular
- Secure Device OS
- Built in cloud communication
- Real-Time OS that works across all products

Particle: Edge to Cloud



Particle: Prototyping to Production

Particle Argon Pin Layout



PWM 0	GROUP 0 PWM PINS	ADC	ANALOG TO DIGITAL INPUT PINS	UART1	SERIAL COMMUNICATION PINS (SERIAL1)
PWM 1	GROUP 1 PWM PINS	SDA/SCL	I2C COMMUNICATION PINS (WIRE)	SDA1/SCL1	I2C COMMUNICATION PINS (WIRE1)
PWM 2	GROUP 2 PWM PINS	SPI	SPI COMMUNICATION PINS (SPI)	SPI1	SPI COMMUNICATION PINS (SPI1)
PWM 3	GROUP 3 PWM PINS	PX.XX	nRF52840 PIN NAMES		

v1.0

Please Do Not Skip Ahead During Setup

Particle Software - ParticleCLI and Visual Studio Code

1. Create Particle login: `https://login.particle.io/signup`
2. Install Particle Command Line Interface. Download from `https://docs.particle.io/tutorials/developer-tools/cli/`.
    - On Windows, you need to run this by right-clicking on it and selecting "Run As Administrator."
    - Test that the Particle CLI installed correctly by going to PowerShell or Terminal and type particle
3. Download Particle Workbench / Visual Studio Code `https://docs.particle.io/quickstart/workbench/`.
    - Select all default values during install.
    - **Do NOT install Azure IoT.**
    - After it is installed, when you launch it, it may ask you to Install Dependencies. If so, select yes.

Particle Setup

1. Attach the Wi-Fi antenna to your Argon. Use the correct connector, there are 3 U.FL connectors: WiFi, BT, and NFC.
2. Plug the Argon into a USB port. It should begin blinking blue.
3. Open PowerShell or Terminal.
4. Login into your Particle Account

```
particle login
```

5. Ensure you have the latest Particle CLI

```
particle update-cli
```

6. Put the Argon in DFU mode (blinking yellow) by holding down MODE. Tap RESET and continue to hold down MODE. The status LED will blink magenta (red and blue at the same time), then yellow. Release when it is blinking yellow.

Updating your Argon to latest Device OS

1. Update the device by running the following two commands. If the device goes out of blinking yellow after the first command, put it back into DFU mode. See Note [2].

```
particle update
particle flash --usb tinker
```

2. When the command reports Flash success!, reset the Argon. It should go back into listening mode (blinking dark blue).
3. Verify that the update worked by running the following command:

```
particle serial identify

Your device id is e00fce681fffffffc08949b
Your system firmware version is 1.5.2
```

[2] paticle flash –usb tinker can be used for device troubleshooting

Setting Up WiFi

- Set your Argon into Listening Mode by holding the MODE button for three seconds, until the RGB LED begins blinking blue.
- Execute the command: particle serial wifi

```
brian:~$ particle serial wifi
? Should I scan for nearby Wi-Fi networks? No
? SSID DDCIOT
? Security Type WPA2
? Cipher Type AES+TKIP
? Wi-Fi Password ddcIOT2020
Done! Your device should now restart.
```

After setting, you Argon should go through the normal sequence of blinking green, blinking cyan (light blue), fast blinking cyan, and breathing cyan.

Claim Your Device

1. Claim the device to your account. This can only be done if it's breathing cyan. Replace e00fce681ffffffffc08949b with the device ID you got earlier from particle serial identify. Then, rename it to the name of your choice.

```
particle device add e00fce681ffffffffc08949b
particle device rename e00fce681ffffffffc08949b
    myArgon
```

2. Ensure that your setup flag is marked as done

```
particle usb setup-done
```

3. You have successfully set up your Argon!

Useful Particle CLI Commands

1. Enter DFU mode from the CLI

```
particle usb dfu
```

2. If Argon won't enter DFU mode or is otherwise acting strangely, restore the base firmware

```
particle flash --usb tinker
```

3. Get a list of your Particle devices and their connection status

```
particle list
```

4. Search for available libraries

```
particle library search <search term>
```

5. Link for the Particle Setup procedures: Particle Setup via CLI

Argon LED Modes

Mode	LED Status
Connected	Breathing Cyan
OTA Firmware Update	Blinking Magenta (red and blue together)
Looking for Internet	Blinking Green
Connecting to Cloud	Rapid Blinking Cyan
Listening Mode	Blinking Blue
Network Reset	Rapid Blinking Blue
WiFi Off	Breathing White
Safe Mode	Breathing Magenta (red and blue together)
DFU (Device Firmware Upgrade	Blinking Yellow
Restore Factory Firmware	Rapid Blinking Yellow
Factory Reset	Rapid Blinking White
Decryption Error	Blinking Cyan followed by 1 Orange Blink
No Internet	Blinking Cyan followed by 2 Orange Blink
No Particle Cloud	Blinking Cyan followed by 3 Orange Blink
Authentication Error	Blinking Cyan followed by 1 Magenta Blink
Handshake Error	Blinking Cyan followed by 1 Red Blink
Decryption Error	Blinking Cyan followed by 1 Orange Blink
SOS - Firmware Crash	Blinking Red - 3 short, 3 long, 3 short, error code

Directory Structure - VERY IMPORTANT

EXPLORER

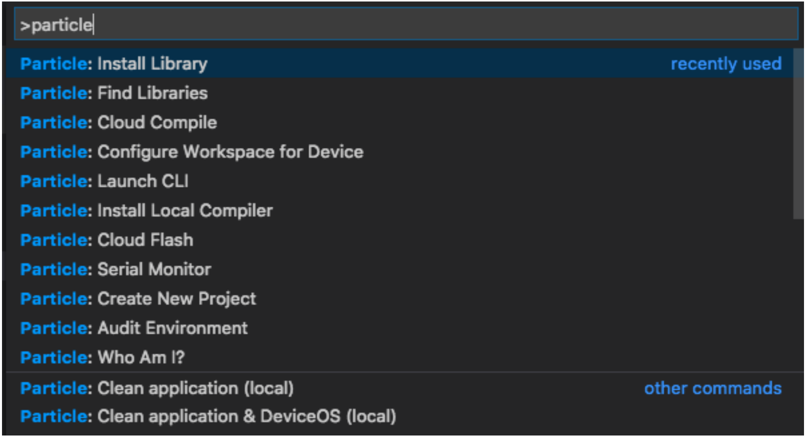
- OPEN EDITORS 1 UNSAVED
  - PM25_Test.ino src 3
- PM25_TEST
  - .vscode
    - launch.json
    - settings.json
  - lib \ Seeed_HM330X
    - examples \ basic_demo
      - basic_demo.ino
    - src
      - HM330XErrorCode.h
      - I2COperations.cpp
      - I2COperations.h
      - Seeed_HM330X.cpp
      - Seeed_HM330X.h
  - src
    - PM25_Test.cpp
    - PM25_Test.ino 3
  - target \ 1.5.0 \ argon
  - project.properties
  - README.md

PM25_Test.ino ●

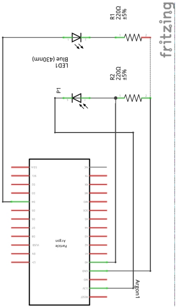
src > PM25_Test.ino > ...

```cpp
/*
 * Project PM25
 * Description: 2.5um Particle Measurement with H3301 Sensor
 * Author:  Brian Rashap
 * Date:    17-APR-2020
 */
#include <Particle.h>
#include <Seeed_HM330X.h>
#include <Wire.h>

/***************SetUp HM330X**************/
HM330X sensor;
uint8_t buf[30];
int PM25;

const char* str[] = {"sensor num: ", "PM1.0 concentration(CF=1,Standard particulate matter,
                     "PM2.5 concentration(CF=1,Standard particulate matter,unit:ug/m3): ",
                     "PM10 concentration(CF=1,Standard particulate matter,unit:ug/m3): ",
                     "PM1.0 concentration(Atmospheric environment,unit:ug/m3): ",
                     "PM2.5 concentration(Atmospheric environment,unit:ug/m3): ",
                     "PM10 concentration(Atmospheric environment,unit:ug/m3): ",
                    };

HM330XErrorCode print_result(const char* str, uint16_t value) {
    if (NULL == str) {
        return ERROR_PARAM;
    }
    Serial.print(str);
    Serial.println(value);
    return NO_ERROR1;
```

Command Palette - Ctrl-Shift-P



Assignment: L12_HelloParticle

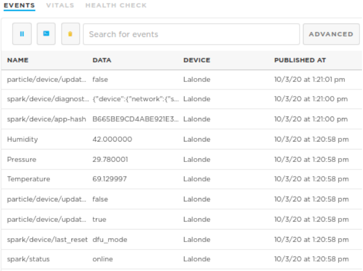


- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

1. L12_01_HelloParticle
   - Blink the onboard LED (Pin D7)
2. L12_02_HelloNightLight
   - The anode of the photodiode is connected to Pin A0. Note, unlike an LED, the cathode (short pin) of the photodiode is connected to 3.3V.
   - The LED anode to Pin D4.
   - Using analogRead/digitalWrite, turn on the LED when photodiode is dark.
   - Using analogWrite, turn on LED slowly as room darkens
3. L12_03_HelloRes
   - Create code to determine the resolution (in bits) of read / write.
   - Use Serial Monitor via Command Palette

Particle Publish

One of the advantages of the Argon is seamless publishing to the Cloud.



NAME	DATA	DEVICE	PUBLISHED AT
particle/device/updat...	false	Lalonde	10/3/20 at 1:21:01 pm
spark/device/diagnost...	{"device":{"network":{"s...	Lalonde	10/3/20 at 1:21:00 pm
spark/device/app-hash	B665BE9CD4ABE921E3...	Lalonde	10/3/20 at 1:21:00 pm
Humidity	42.000000	Lalonde	10/3/20 at 1:20:58 pm
Pressure	29.780001	Lalonde	10/3/20 at 1:20:58 pm
Temperature	69.129997	Lalonde	10/3/20 at 1:20:58 pm
particle/device/updat...	false	Lalonde	10/3/20 at 1:20:58 pm
particle/device/updat...	true	Lalonde	10/3/20 at 1:20:58 pm
spark/device/last_reset	dfu_mode	Lalonde	10/3/20 at 1:20:58 pm
spark/status	online	Lalonde	10/3/20 at 1:20:58 pm

```
float temp, prs, hum;    // BME280 variables
String Temp, Prs, Hum;   // Strings to hold BME280 values

void loop() {
  Temp = String(temp);
  Prs = String(prs);
  Hum = String(hum);

  Particle.publish("Temperature", Temp, PRIVATE);
  Particle.publish("Pressure", Prs, PRIVATE);
  Particle.publish("Humidity", Hum, PRIVATE);
}
```

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

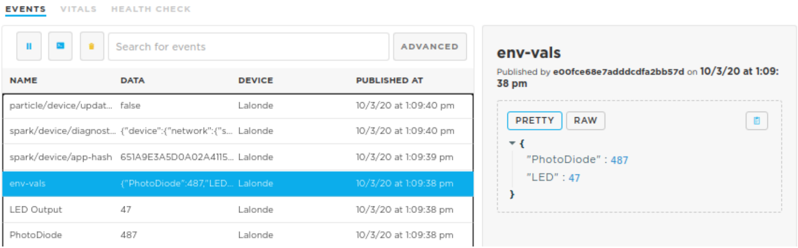
- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

JSON Parser Generator

Creating objects in JSON are straightforward but can be tedious. There is a JSON Parser available to simplify the process.

```
#include <JsonParserGeneratorRK.h>

void createEventPayLoad(float tempValue, float presValue, float humValue) {
  JsonWriterStatic<256> jw;
  {
    JsonWriterAutoObject obj(&jw);

    jw.insertKeyValue("Temperature", tempValue);
    jw.insertKeyValue("Pressure", presValue);
    jw.insertKeyValue("Humidity", humValue);
  }
  Particle.publish("env-vals",jw.getBuffer(), PRIVATE);
}
```

Assignment: L12_HelloParticle



1. L12_04_HelloPublish
   - Using Particle.publish() send the photodiode and LED values to the Particle Cloud.
   - Use the Command Palette to Install Library JsonParserGeneratorRK.
   - Use this library to send the same data using the JSON Generator.

SYSTEM_MODE

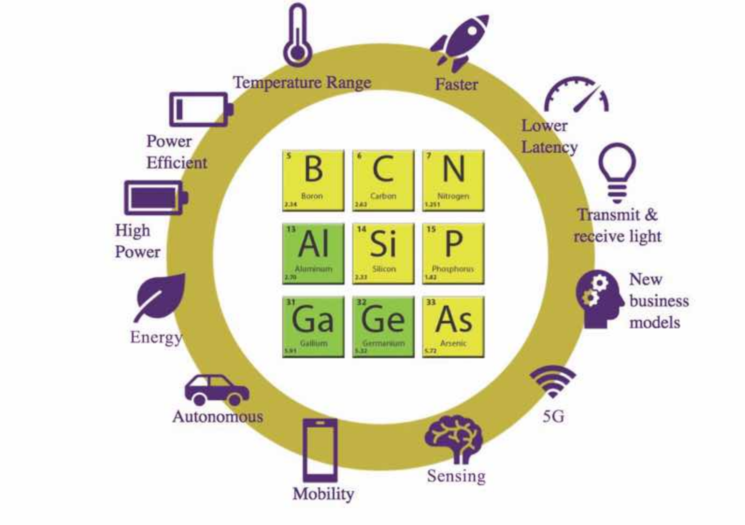
System modes help control how the device manages the connection with the cloud. By default, the device connects to the Cloud and processes messages automatically. However there are many cases where a user will want to take control over that connection. There are three available system modes:

- AUTOMATIC,
- SEMI_AUTOMATIC,
- MANUAL.

```

//The below is placed in the header before Void Setup()

// SYSTEM_MODE(AUTOMATIC);       // Default if no SYSTEM_MODE included
   SYSTEM_MODE(SEMI_AUTOMATIC); // Uncomment if using without Wifi
// SYSTEM_MODE(MANUAL);          // Fully Manual
```

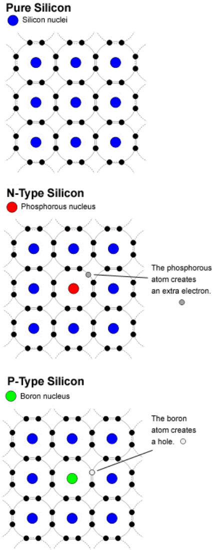
MOVE THIS SECTION: Semiconductors

Semiconductors



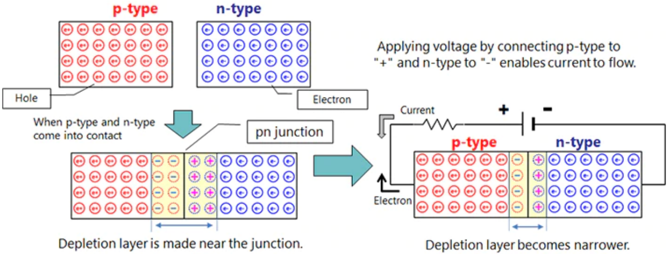
Semiconductor

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



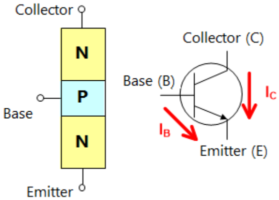
pn junction diode

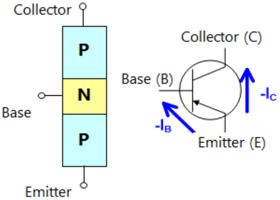
- When p-type and n-type semiconductors are bonded, holes and free electrons are attracted, combine, and disappear near the boundary. Since there are no carriers in this area, it is called a depletion layer and it is an insulator.
- A positive voltage applied to the p-type region causes electrons to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



Bipolar Junction Transistor

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

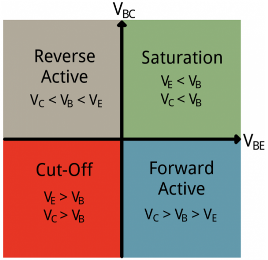




Bipolar Junction Transistor - Modes of Operation

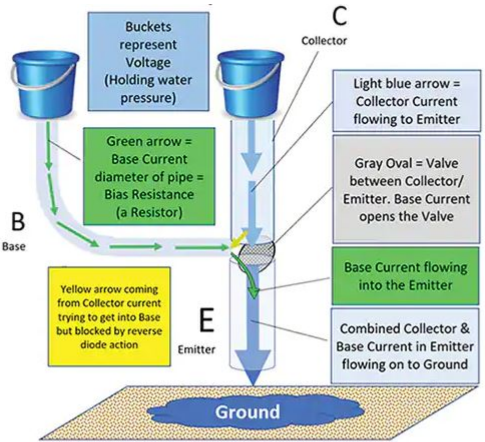
- **Saturation:** Current freely flows from collector to emitter. (ON Switch)
- **Cut-off:** No current flows from collector to emitter. (OFF Switch)
- **Active:** The current from collector to emitter is proportional to the current flowing into the base. (Amplifier)
- **Reverse-Active:** Like active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

NPN Modes



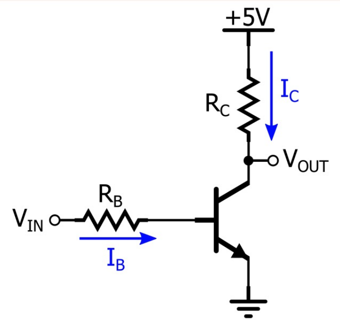
Voltage relations	NPN Mode	PNP Mode
$V_E < V_B < V_C$	Active	Reverse
$V_E < V_B > V_C$	Saturation	Cutoff
$V_E > V_B < V_C$	Cutoff	Saturation
$V_E > V_B > V_C$	Reverse	Active

Water Analogy



Active Mode NPN Transistor Circuit

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

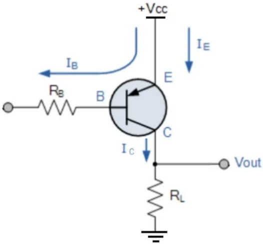


NOTE: V_{OUT} is an amplified but inverted signal of V_{IN}.
This simple circuit will step-up a 0 - 3.3V output from the microcontroller to 0 - 5.0V (inverted). The low impedance of the output will also provide sufficient current to drive a higher current device (e.g., a relay).

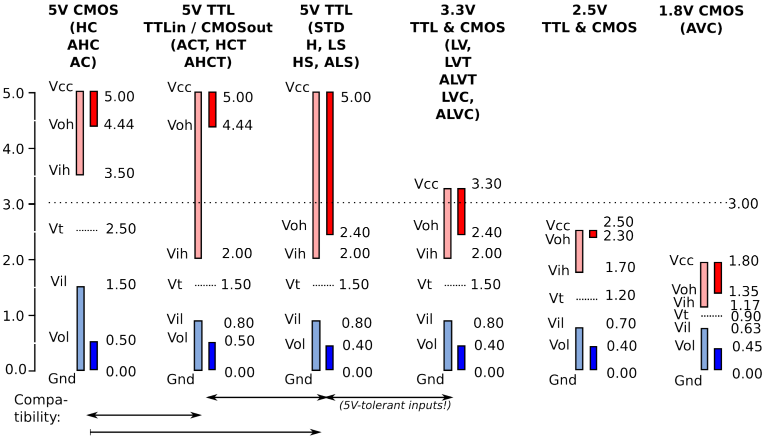
PNP Transistor

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

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



Logic Voltage Level Standards

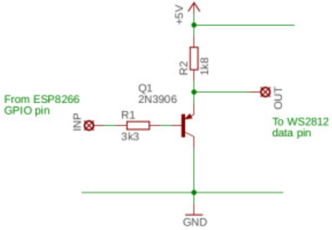


Data source: EETimes, A brief recap of popular logic standards (Mark Pearson, Maxim).

Or, what is wrong with the NeoPixels.

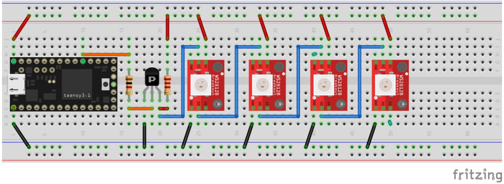
Emitter Follower

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

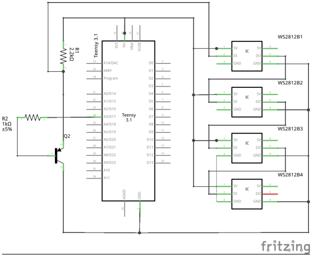


Emitter Follower Layout

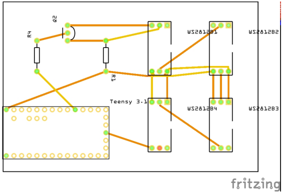
Fritzing



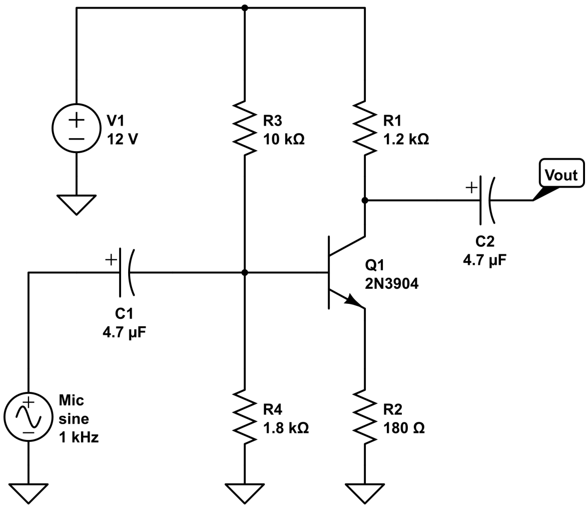
Schematic



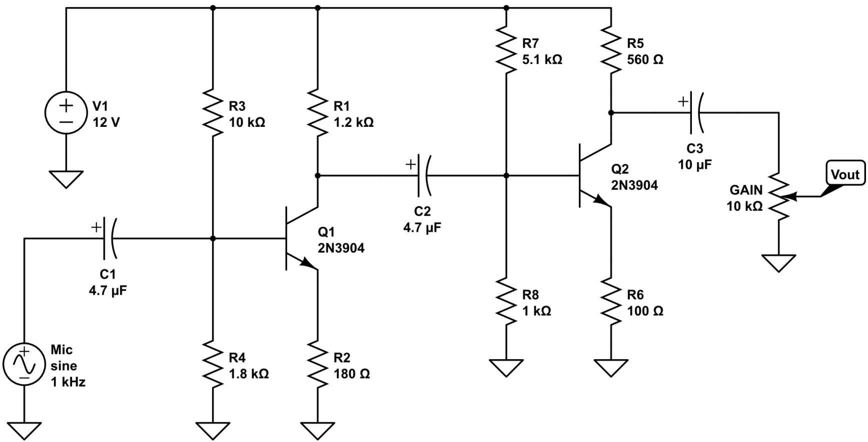
PCB Layout



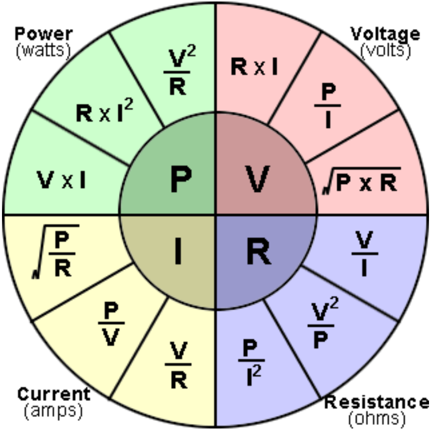
NPN Pre-Amplifier Circuit



Two Stage Pre-Amplifier



Ohm's Law - Revisited



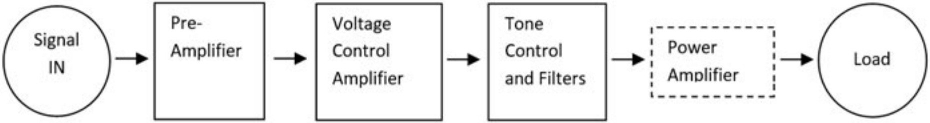
PreAmp vs PowerAmp

PreAmp:

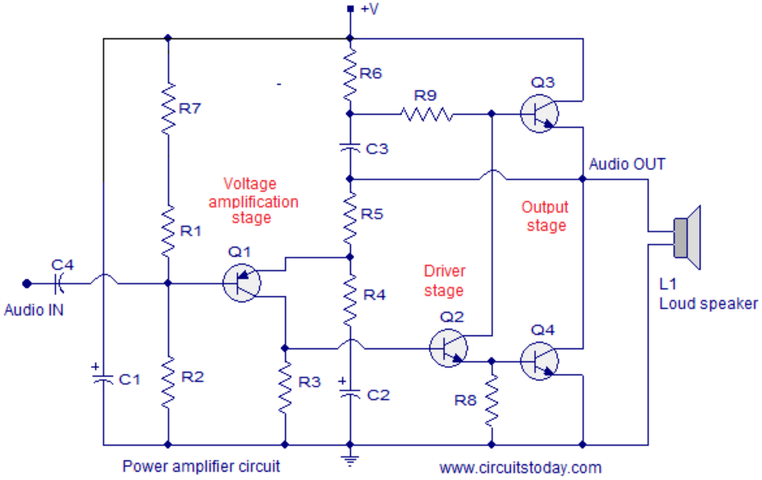
- A preamp boosts the signal up to ‘line level’.
- Guitar PreAmp
  - A pure guitar signal typically sounds weak and anaemic, as is seen if a guitar is directly plugged PA system.
  - A preamp is able to raise a guitar’s signal up to an audible volume.
  - It can also be used to affect the audio characteristics.

PowerAmp:

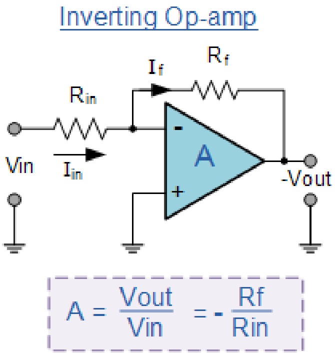
- A power amp boosts that line level signal even more – so that it can be projected through speakers.

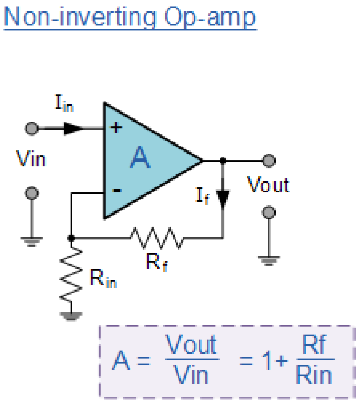


Power Amplifier

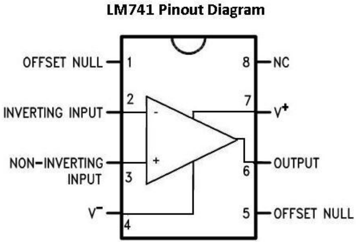


Op Amp Lesson - Under Construction





Assignment: Amplifiers



1. L09_01_NeoPixel
    - Add a Emitter-Follower into your NeoPixel circuit to boost the pixel commands to 5V.
2. L09_02_NPNAmp (no Teensy)
    - Make NPN Preamp
    - Test your NPN Preamp with the signal generator and oscilloscope.
3. L09_03_OpAmp
    - Create a preamp using your LM741 OpAmp

More DataTypes - Strings, strings, and char[]

```
// A string is an array of characters
char firstName[6];
char *name = {"B","R","I","A","N"};  //The "*" indicates a pointer
char *myName = "BRIAN";              //We will learn pointers later

// A String is a Class that holds a character array
String instructor = "BRIAN RASHAP";


// Because String is a Class it has methods
instructorName.substring(0,4);
instructorName.toCharArray(firstName,5);

// Using Serial.print and strings
// The %s in formatted print inserts an array of char
Serial.println(instructorName);
Serial.printf("My name is %s \n",firstname);
```

To learn more about the String Class: `https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/`

Too Much Time On My Hands

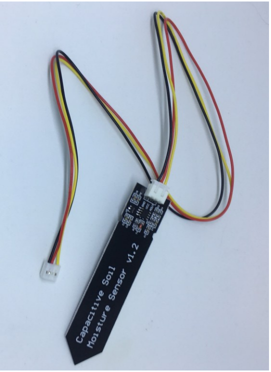
When the Particle Argon connects to the Particle Cloud, it synchronizes its clock to the current time.

```
// Declare Global Variables in Header
String DateTime, TimeOnly;
char currentDateTime[25], currentTime[9];

void setup() {
  Time.zone(-7);            // MST = -7, MDT = -6
  Particle.syncTime();      // Sync time with Particle Cloud
}

void loop() {
  DateTime = Time.timeStr();             //Current Date and Time from Particle Time class
  TimeOnly = DateTime.substring(11,19); //Extract the Time from the DateTime String

  // Convert String to char arrays - this is needed for formatted print
  DateTime.toCharArray(currentDateTime,25);
  TimeOnly.toCharArray(currentTime,9);

  //Print using formatted print
  Serial.printf("Date and time is %s\n",currentDateTime);
  Serial.printf("Time is %s\n",currentTime);

  delay(10000); //only loop every 10 seconds
}
```

Soil Moisture Sensors

Resistive Sensor

Capacitive Sensor



Installing Libraries

Using the Command Palette within VSCode:

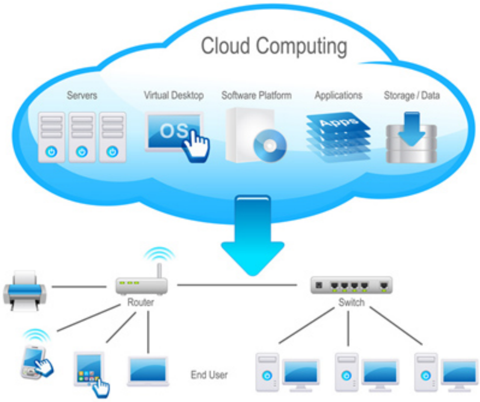
- ctrl-shift-p → Particle: Find Libraries
- ctrl-shift-p → Particle: Install Library

Assignment: L14_SoilMoisture

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

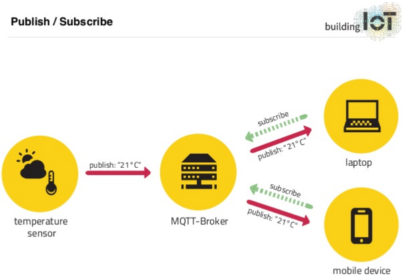
1. L14_01_OLED
   - Install the Adafruit_SSD1306 library
   - Use I2C_Scan to get the OLED I2C address
   - Create sample code displaying your name and time to the OLED
2. L14_02_Moisture
   - Using the Capacitive Soil Moisture probe. In your notebook note the moisture readings when:
     - Empty Cup
     - Submerged in water to the notch
     - Dry Soil
     - Soil after watered
   - Display the moisture to the OLED with a Time-stamp

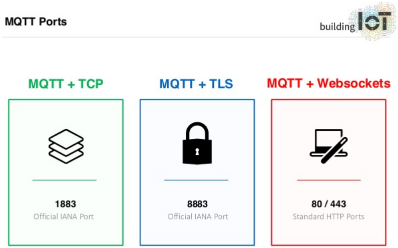
The Cloud



Cloud Computing
Servers
Virtual Desktop
Software Platform
Applications
Storage / Data
OS
Apps
Router
Switch
End User

MQTT: MQ Telemetry Transport





Adafruit.io

Let's create an Adafruit.io account

Get Started	Power Up
FREE forever	**$10** per month or **$99** per year
30 data points per minute	**60** data points per minute
30 days of data storage	**60** days of data storage
Triggers every **15 minutes**	Triggers every **5 seconds**
5 feed limit	**Unlimited feeds**
	Learn more about IO+
Sign Up Now	Sign Up Now

MQTT Elements explained

- TheClient - object that defines the TCP (Transmission Control Protocol) connection over WiFi.
- mqtt - object that defines the MQTT connection using the WiFi object, the MQTT server/port, and user name/password.
- FeedName - a "variable" located on Adafruit.io that can be subscribed or published to. There can be many of these.
- mqttObj - object that will be used in the C++ code that will be used to publish or subscribe to an Adafruit.io feed. There needs to be one object for each feed.
- value - Variable in the C++ code that stores information to be published or to receive information from a feed that is subscribed to.

NOTE: FeedName, mqttObj, and value should be given descriptive "names" similar to the naming convention for all variables and objects in the C++ code.

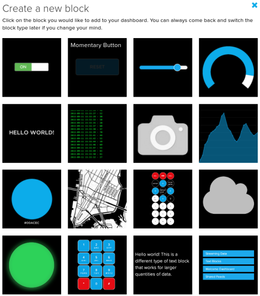
MQTT Elements in VSCode

```
#include <Adafruit_MQTT.h>

#include "Adafruit_MQTT/Adafruit_MQTT.h"
#include "Adafruit_MQTT/Adafruit_MQTT_SPARK.h"
#include "Adafruit_MQTT/Adafruit_MQTT.h"

/************ Global State (you don't need to change this!) *****************/
TCPClient TheClient;

/************************* Adafruit.io Setup *********************************/
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
#define AIO_USERNAME    "<username>"
#define AIO_KEY         "<key>"

// Setup the MQTT client class with the WiFi client, MQTT server and login details.
Adafruit_MQTT_SPARK mqtt(&TheClient,AIO_SERVER,AIO_SERVERPORT,AIO_USERNAME,AIO_KEY);

/***************************** Feeds ***************************************/
// Setup Feeds to publish or subscribe
// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>
Adafruit_MQTT_Subscribe mqttObj1 = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/
     FeedNameA");
Adafruit_MQTT_Publish mqttObj2 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/
     FeedNameB");

/************Declare Variables*************/
float value1;   //variable that will hold data received from adafruit.io feed
float value2;   //varaible that will hold data to be published to adafruit.io feed
```

MQTT Publish and Subscribe

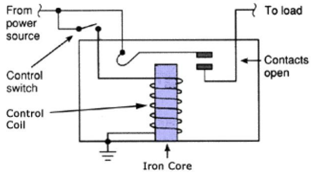
```
// Publishing to a MQTT feed
if(mqtt.Update()) {   //if mqtt object (Adafruit.io) is available to receive data
  Serial.printf("Publishing %0.2f to Adafruit.io feed FeedNameB \n",value2);
  mqttObj2.publish(value2);
}


// Two new functions that will be useful
// atof() - ASCII to Float: converts an ASCII string to a floating point number
// atoi() - ASCII to Integer: converts an ASCII string to an integer


// Receive data from a subscription to an MQTT feed
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(10000))) { //wait 10 sec for new feed data
  if (subscription == &mqttObj1) {         // assign new data to appropriate variable
    value1 = atof((char *)mqttObj1.lastread); //value1 equals data from MQTT subscription
    Serial.printf("Received %0.2f from Adafruit.io feed FeedNameA \n",value1);
  }
}
```

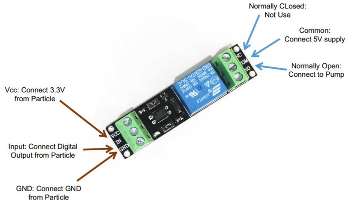
Assignment: L14_03_SubscribePublish



1. Modify the starter code for your Adafruit.io
2. Subscribe
   - Add a button to your Adafruit.io dashboard and connect it to a feed called buttonOnOff
   - Subscribe to the buttonOnOff and turn on the on board LED when pressed.
3. Publish
   - Publishing a random number to a feed once per minute (do not use a delay).
   - Create a line chart on your dashboard to display the random number.
4. Experiment with other blocks
   - Replace the button with a slider.
   - Control the brightness of an LED
   - Display data with other dashboard blocks.

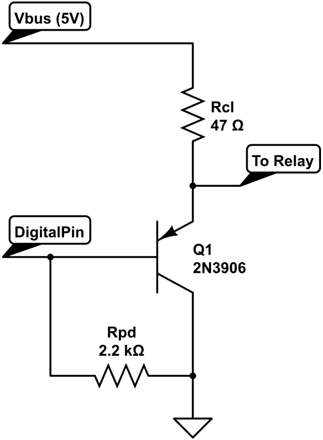
Relays





- When a device (e.g. a pump) requires higher voltage ($> 5V$) or higher current, then a relay can be used as a switch for the device
- The relay is activated by a digital pin from the microcontroller.
    - However, as the relay requires 100mA from the digital pin, we will use a transistor switch to draw power directly from the USB connection.

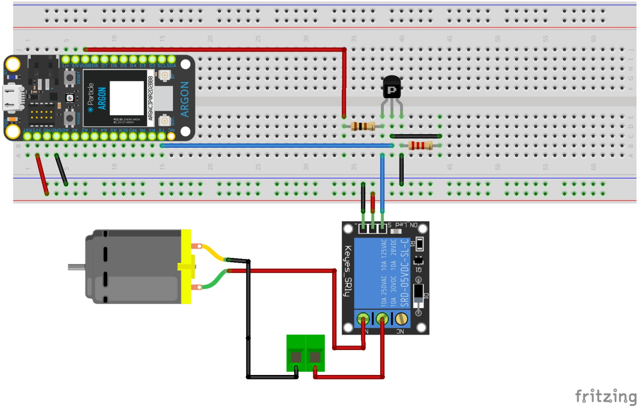
Assignment: L14_04_PlantWater



PNP Emitter Follower

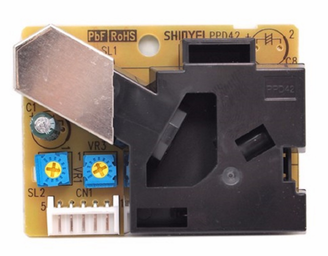
1. Integrate an 2N3906 Emitter Follower and Relay, as well as a BME280 and Display.
2. Publish soil moisture and room environmental data to a new dashboard
3. Automatically water your plant when the soil is too dry
   - Only turn on the pump for a very short period of time ($\frac{1}{2}sec$).
4. Integrate a button into your dashboard that manually waters the plant.

Relay and Pump Fritizng Diagram

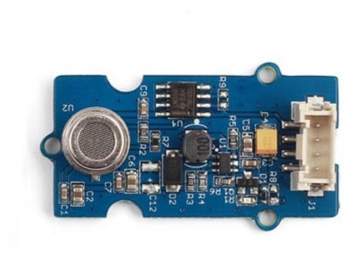


NOTE: The relay in Fritzing pins are in a different order than the relay in your kit

Seeed Sensors



Seeed Grove - Dust Sensor

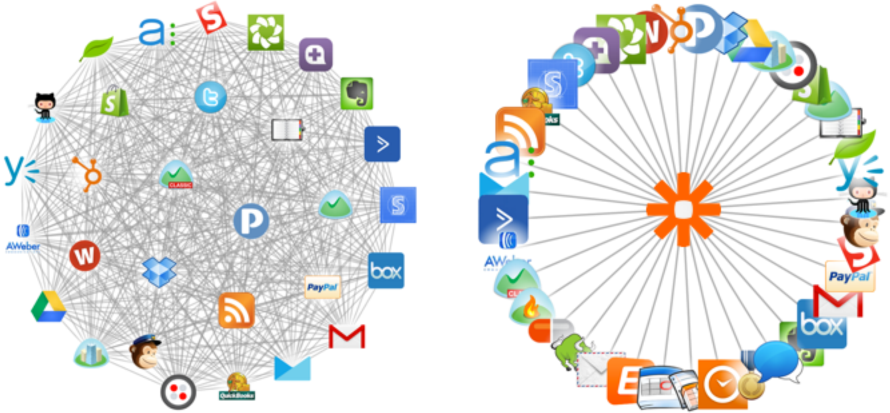


Seeed Grove - Air Quality Sensor v1.3

Assignment: More L14

1. L14_04_PlantWater (revisted): Integrate into your Plant Water project both Seeed sensors.
   - Look up the Seeed sensors online to see how they work.
   - Do not blindly copy the examples, only use the code you need.
   - By looking at the .cpp code, determine how to get a quantitative value for air quality, in addition to the qualitative level.
2. Include Hackster.io story

Using Zapier



Zapier is an online automation tool that connects your favorite apps, such as Outlook, Slack, Mailchimp, and more. You can connect two or more apps to automate repetitive tasks.

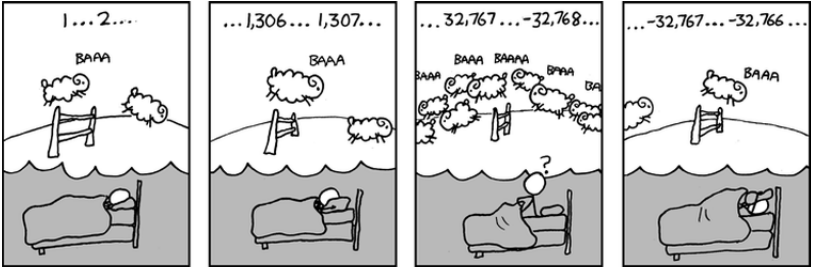
Optional Assignment: L14_04_PlantWater

zapier

```
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(10000))) {
  if (subscription == &gotmail) {
    Serial.printf("You Got Mail \n");
    digitalWrite(D7, HIGH);
    flagServo.write(90);
    delay(10000);
    digitalWrite(D7, LOW);
    flagServo.write(5);
  }
}
```

1. Use the Zapier instructions in Class_Slides to light up your D7 LED when new mail arrives in your cnm.edu outlook account.
2. Add to your L14_04_PlantWater project to send you an SMS text whenever your soil is too dry. In Zapier:
   - Create Feed Trigger from Adafruit.io
   - Add a "Only Continue If..."
   - Add a Send to SMS action

Counting Sheep



Negative Numbers

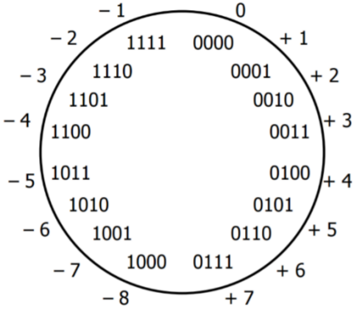
Question: How are negative numbers represented in binary?

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Answer: Left-most bit is 1 to signify negative. But wait...

2's Compliment

2's compliment is used as it makes the math consistent.



Integer		2's Complement
Signed	Unsigned	
5	5	0000 0101
4	4	0000 0100
3	3	0000 0011
2	2	0000 0010
1	1	0000 0001
0	0	0000 0000
-1	255	1111 1111
-2	254	1111 1110
-3	253	1111 1101
-4	252	1111 1100
-5	251	1111 1011

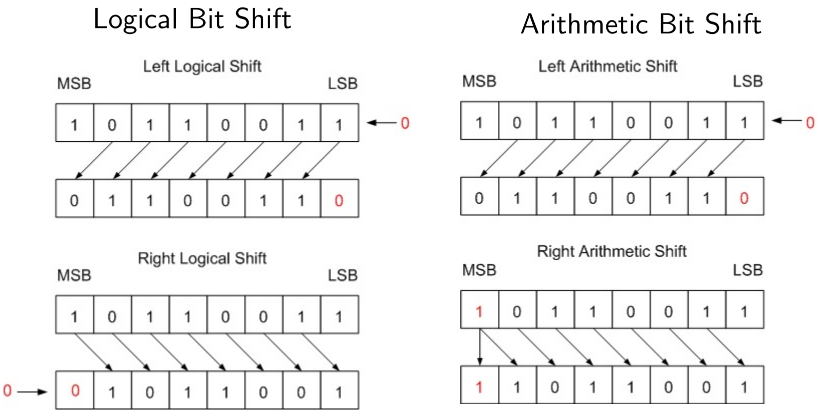
The negative plus the positive equals zero.

Bitwise Operations

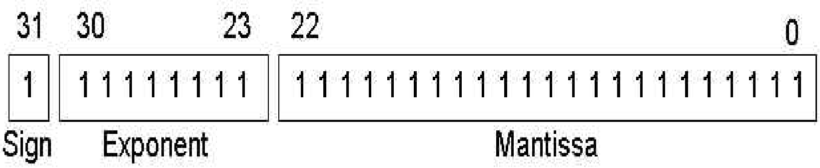
The following table lists the Bitwise operators supported by C. Assume variable 'A' holds 60 and variable 'B' holds 13, then −

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = ~(60), i.e,. 1100 0011
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

Bit Shifting



BONUS: But what about Floating Point



Example:
$-36382.366 = -3.6382366 * 10^4 = 1000001011101111000010001011110$

Electrically Erasable Programmable Read Only Memory

EEPROM emulation allows small amounts of data to be stored and persisted even across reset, power down, and user and system firmware flash operations. Since the data is spread across a large number of flash sectors, flash erase-write cycle limits should not be an issue in general.

```
len = EEPROM.length(); //available EEPROM bytes
// Argons have 4096 bytes of emulated EEPROM.
// Addresses 0x0000 through 0x0FFF

addr = 0x00AE;    //addr between 0 and len-1

val = 0x45;
EEPROM.write(addr, val);

value = EEPROM.read(addr);
```

Interrupts

Interrupts are a way to write code that is run when an external event occurs. As a general rule, interrupt code should be very fast, and non-blocking. This means performing transfers, such as I2C, Serial, TCP should not be done as part of the interrupt handler. Rather, the interrupt handler can set a variable which instructs the main loop that the event has occurred.

```
attachInterrupt(pin, function, mode);
```

Mode: defines when the interrupt should be triggered. Three constants are predefined as valid values:

- CHANGE to trigger the interrupt whenever the pin changes value,
- RISING to trigger when the pin goes from low to high,
- FALLING for when the pin goes from high to low.

RandomSeed()

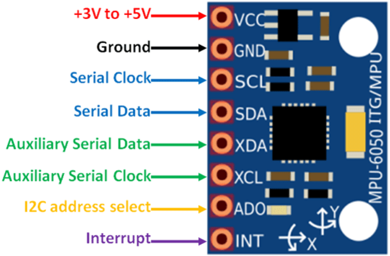
- The "pseudo" in pseudo-random refers it not truly being random.
- randomSeed() initializes the pseudo-random number generator, causing it to start at an arbitrary point in its random sequence. This sequence, while very long, and while appearing random, is always the same.
- If it is important for a 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 analogRead() on an unconnected pin.

```
// Leave an Analog Input (A0) floating
pinMode(A0,INPUT);
randomSeed(analogRead(A0));

// print a random number hex color value
randNumber = random(0x0000,0xFFFFFF);
Serial.println(randNumber);
```

L15_BEI Assignments

1. L15_01_ConvertStore
    - Convert random hex color into R, G, and B components using Bit Shifting and Bitwise AND.
        - First sketch out how this might be done in your lab notebook. Show to instructor.
        - Then, and only then, write the code.
    - Store the components of the color as bytes in the Argon's EEPROM
2. L15_02_RetrieveShow
    - Retrieve the color from EEPROM memory
    - Convert to Hex code
    - Display color in NeoPixel.
3. L15_03_Interrupt Create an interrupt Stop Button
    - Add a button to your Argon
    - Using delays, blink your NeoPixel GREEN (once per second)
    - Using an Interrupt, when the button is pressed, the NeoPixel turns to rapidly flashing RED.

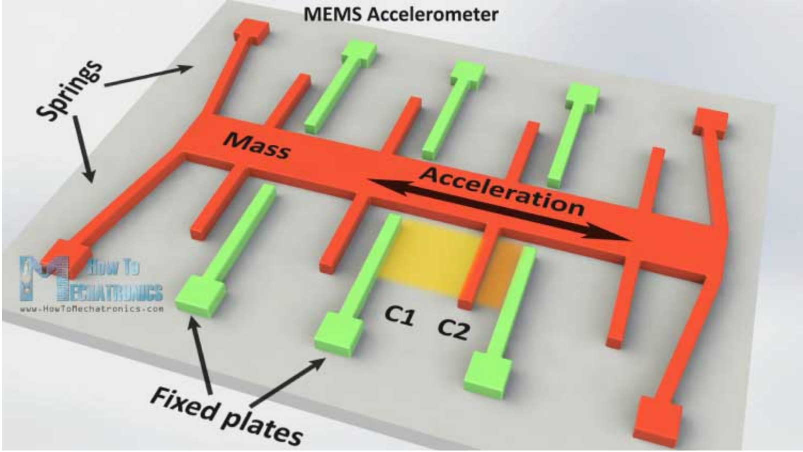
MPU6050 Accelerometer



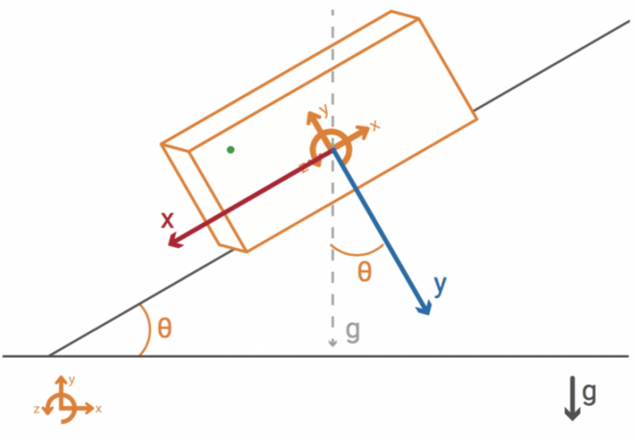
- Data Output - 16 Bit
- Gyros range: $\pm$ 250 500 1000 2000 $^{\circ}/s$
- Accel range: $\pm$ 2 $\pm$ 4 $\pm$ 8 $\pm$ 16g

- The interrupt pin lets the MPU be notified about available data. To reduce power consumption the processor can go into sleep mode, the interrupt can be used as a wake up.
- XDA and XCL refer to the I2C bus that the MPU-6050 controls, so it can read from slave devices such as magnetometers etc.

Accelerometers



Gravity and Orientation



$$a_y = g * cos(\theta)$$

REMINDER - 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.

Initializing MPU6050

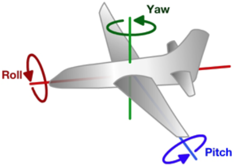
```
// Initialize the MPU in the void setup()
void setup() {
  // Begin I2C communications
  Wire.begin();

  // Begin transmission to MPU-6050
  Wire.beginTransmission(MPU_ADDR);

  // Select and write to PWR_MGMT1 register
  Wire.write(0x6B);
  Wire.write(0); // set to 0 (wakes up MPU-6050)

  // End transmission and close connection
  Wire.endTransmission(true);
}
```

Reading Acceleration Data from the MPU-6050

```
// Declare variables
byte accel_x_h, accel_x_l;    //variables to store the individual btyes
int16_t accel_x;              //variable to store the x-acceleration

// Set the "pointer" to the 0x3B memory location of the MPU and wait for data
Wire.beginTransmission(MPU_ADDR);
Wire.write(0x3B); // starting with register 0x3B
Wire.endTransmission(false); // keep active.

// Request and then read 2 bytes
// Syntax:
//    Wire.requestFrom(I2C_addr, quantity, stop);
//    Wire.read(); //repeat this for each byte to be read

Wire.requestFrom(MPU_ADDR, 2, true);
accel_x_h = Wire.read(); // x accel MSB
accel_x_l = Wire.read(); // x accel LSB

accel_x = accel_x_h << 8 | accel_x_l;     // what happens if declared int instead?
Serial.printf("X-axis acceleration is %i \n",accel_x);
```

Note, the data is stored in Big Endian Byte Order. The most significant byte (the "big end") of the data is placed at the byte with the lowest address. The rest of the data is placed in the next byte.

Assignment: L16_Motion



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

1. L16_01_MPU6050
   - Read the memory addresses for X, Y, and Z acceleration
   - Convert the returned acceleration value to standard gravity units (e.g.,when flat on the table, a_z = -1G).
2. L16_02_AutoRotate
   - Create a Clock on an OLED Display
   - Use GY-521 to auto-rotate the OLED
3. L16_03_Airplane
   - Calculate pitch $\theta = -asin(a_x)$
   - Calculate roll $\phi = atan2(a_y/-a_z)$
4. L16_04_Shock
   - Store a_{tot} in an array every 10ms for 5s
   - Find and print the max value
   - Repeat

Pitch and Roll Improved

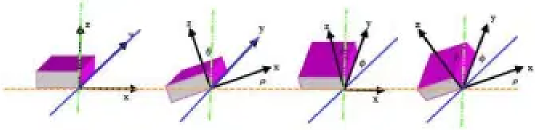


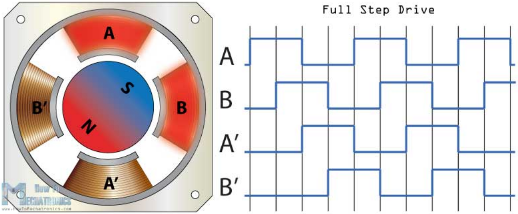
Figure 8. Three Axis for Measuring Tilt

$$\rho = \arctan\left(\frac{A_X}{\sqrt{A_Y^2 + A_Z^2}}\right)$$

$$\phi = \arctan\left(\frac{A_Y}{\sqrt{A_X^2 + A_Z^2}}\right)$$

$$\theta = \arctan\left(\frac{\sqrt{A_X^2 + A_Y^2}}{A_Z}\right)$$

Stepper Motors



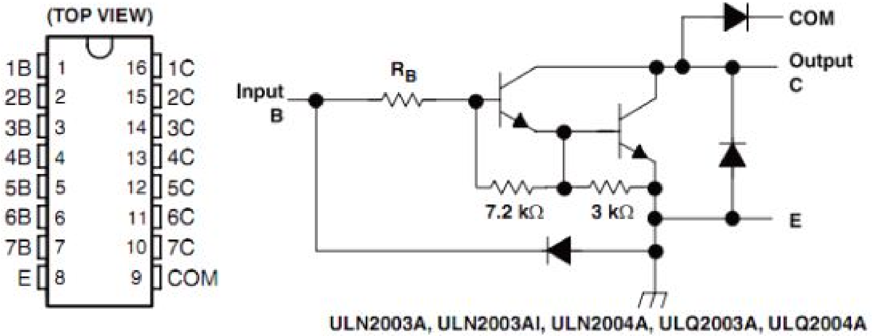
28BYJ Stepper Motor

- 2048 steps per revolution
  - 32 steps per rotor revolution
  - Gear ratio 1:64
- Capable of 10-15 RPM (at 5V)
- ULN2003 Darlington Array driver

Stepper.h

- Stepper myStepper (spr,IN1,IN3,IN2,IN4)
- myStepper.setSpeed(speed)
- myStepper.step(steps)

ULN2003 Darlington Array



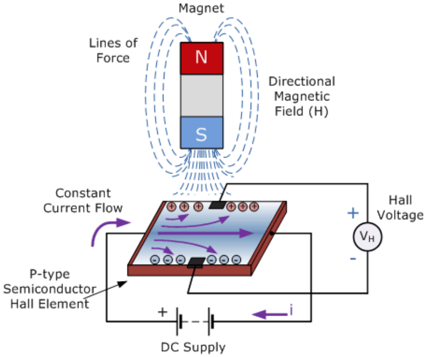
A Darlington Array is an is a set of current amplifying circuits that take outputs from the microcontroller and boost the current used to drive the motor.

Assignment: L16_Motion

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

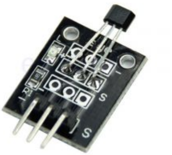
1. L16_05_StepperGyro
   - Wire the Stepper Motor noting the order: IN1, IN3, IN2, IN4
   - Move the motor 2 rotations clockwise, pause, 1 rotation counter-clockwise, repeat.
2. L16_06_Gyro
   - Connect the GY-521 to your system
   - Obtain the z-axis rotation from the appropriate register on the GY-521.
   - Map the change in gyro position (signed 16-bit) to one revolution of the Stepper (signed 12-bit)
   - Move the stepper the amount of steps corresponding to the mapped gyro change.

Hall Effect Sensor



Discovered by Edwin Hall in 1879, the Hall Effect is the production of a voltage difference (the Hall voltage) across an electrical conductor, transverse to an electric current in the conductor and to an applied magnetic field perpendicular to the current.

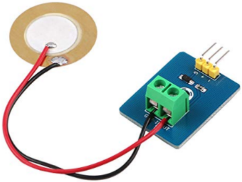
Assignment: L16_Motion



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

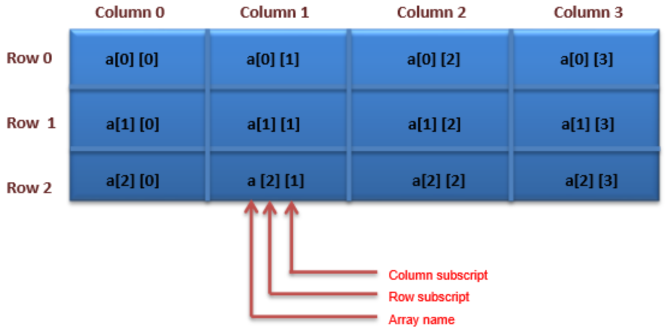
1. L16_07_Alarm
   - Connect Hall Effect Sensor, button, and Neopixel
   - Use the button to enable / disable alarm
   - When armed:
     - Green when magnet detected
     - Blinking Red when magnet not detected
2. L16_08_RPM
   - Place magnet on shaft of Lathe
   - Using Hall Sensor, measure time per rotation
   - Convert to rotations per minute
   - Display on Adafruit.io databoard
   - EXTRA: Create a jig to hold the sensor
   - EXTRA: Measure RPM on other equipment at FUSE.

Piezoelectric Elements



- The piezoelectric effect is the appearance of electrical potential (voltage) across the side of a crystal when subject to mechanical stress.
- Conversely, a crystal becomes mechanically stressed (deformed in shape) when a voltage is applied across opposite faces.
- By utilizing an analogRead(), the vibration (change in mechanical stress) can be monitored over time.

2-dimensional arrays



- Declare Array: int a[3][4] = {{0,1,2,3},{4,5,6,7},{8,9,10,11}};
- Set a Cell: a[1][2] = 7;
- Access a Cell: x = a[0][0];

FAT File System - SDCard Argon Project

Feature	FAT32	NTFS
Maximum Partition Size	2TB	2TB
Maximum File Size	4GB	16TB
Maximum File Name	8.3 Characters	255 Characters
File/Folder Encryption	No	Yes
Fault Tolerance	No	Auto Repair
Security	Network Only	Local and Network
Compression	No	Yes
Compatibility	Win 95/98/2000/XP and the derivations	Win NT/2000/XP/Vista/7 and the later versions

The FAT (File Allocation Table) file system, originally designed in 1977 for Floppy Disks, is simple and robust. It offers good performance in very light-weight implementations, but doesn't deliver the performance, reliability and scalability as modern file systems (such as NTFS or exFAT).

Note: FAT file name follows a 8.3 format (e.g., ABCDEFGH.txt). We will be appending two digits, so our base name can be up to 6 characters (e.g. FILE_BASE_NAME = "mydata" → mydata42.csv).

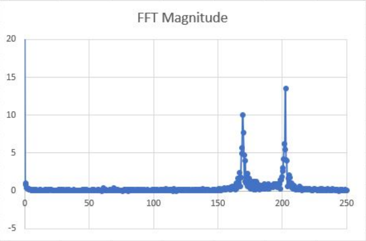
Galaxy S9+ Vibration Analysis

Using the FTT Tutorial in Class_Slides

Time Stamp	Piezo Data	Frequency	FFT Magnitude	Complex FFT
0	0	0	34.01318359	69659
495.795197	0	0.48828	0.834211076	-1688.50727764188+260.371627432835i
495.795776	0	0.97656	0.990540712	1160.28073176531+1664.05458422262i
495.796356	0	1.46484	0.335296983	470.271617580932+500.38517069302i
495.796936	22	1.95312	0.350903895	718.458646378537+16.6339444111701i
495.797516	42	2.4414	0.296405963	-347.540356147162+497.707293053373i
495.798096	45	2.92968	0.188787007	66.5063987685909+380.872856200835i
495.798645	38	3.41796	0.23257935	21.7131959975456+475.827353423851i
495.799255	0	3.90624	0.181340831	-361.737320108222-84.105221575463i
495.799805	0	4.39452	0.219111001	-151.891749008158+422.250971182262i
495.800415	0	4.8828	0.185155013	-343.539789040933+160.533895023591i
495.800964	0	5.37108	0.027207649	3.29352891642554-55.6238447102985i
495.801544	0	5.85936	0.094048245	168.013999708849+94.1818379444134i
495.802155	0	6.34764	0.107193183	-12.442997448963-219.178721949085i
495.802704	18	6.83592	0.201162273	210.468430396862+354.162160126848i
495.803314	45	7.3242	0.123920667	163.758018048072+193.887687505018i
495.803864	45	7.81248	0.071023469	60.4052139160327-132.320356144062i
495.804474	36	8.30076	0.087211846	89.2879735866281+154.690466161755i

stop = 1/.0005 = 2000

step = 2000/4096 = 0.48828



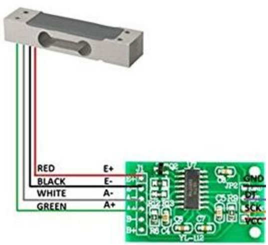
This particular phone vibrates at a combination of 174 and 202 Hz.

Assignment: L16_Motion

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

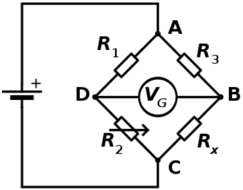
1. L16_09_Vibration
   - Connect the Piezo, a Button, and a μSD module to your Argon
   - Executing a loop 4096 times:
     - Every 500μsec, collect piezoelectric data (without using a delay)
     - Save the pizeo data and a timestamp (converting micros() to seconds) to a 2-dimensional array
   - When the loop is complete, write the timestamp and data to a file.
   - Collect vibration data from Lathe, cellphone vibration, other machines at FUSE
   - Use Excel and the FFT Tutorial (Class_Slides) to resample graph data in frequency domain (this process will be reviewed as a class).

Load Cells



1. A load cell is a force transducer. It converts a force such as tension, compression, pressure, or torque into an electrical signal that can be measured and standardized. As the force applied to the load cell increases, the electrical signal changes proportionally. The most common types of load cells used are hydraulic, pneumatic, and strain gauge.
2. HX711 module is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.

Wheatstone Bridge

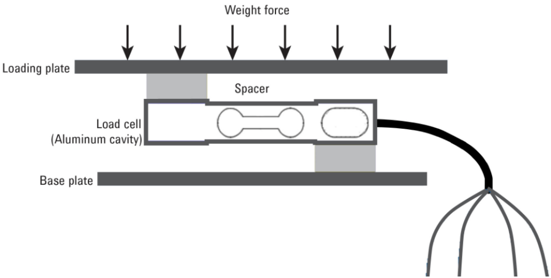


1. The Wheatstone bridge was invented by Samuel Hunter Christie in 1833 and improved and popularized by Sir Charles Wheatstone in 1843.
2. A Wheatstone bridge is an electrical circuit used to measure an unknown electrical resistance by balancing two legs of a bridge circuit, one leg of which includes the unknown component.
3. The primary benefit of the circuit is its ability to provide extremely accurate measurements (in contrast with something like a simple voltage divider)

HX711A Library

```
// From the Command Palette install the HX711A library, that will give you HX711.h
#include "HX711.h"
HX711 myScale(DT,CLK);     // any two digital pins

const int cal_factor=1000; //changing value changes get_units units (lb, g, ton, etc.)
const int samples=10; //number of data points averaged when using get_units or get_value

float weight, rawData, calibration;
int offset;

void setup() {
  myScale.set_scale();              // initialize loadcell
  delay(5000);                      // let the loadcell settle
  myScale.tare();                   // set the tare weight (or zero)
  myScale.set_scale(cal_factor); //adjust when calibrating scale to desired units
}

void loop() {
  // Using data from loadcell
  weight = myScale.get_units(samples); // return weight in units set by set_scale();
  delay(5000)                          // add a short wait between readings

  // Other useful HX711 methods
  rawData = myScale.get_value(samples);// returns raw loadcell reading minus offset
  offset = myScale.get_offset();       // returns the offset set by tare();
  calibration = myScale.get_scale();   // returns the cal_factor used by set_scale();
}
```

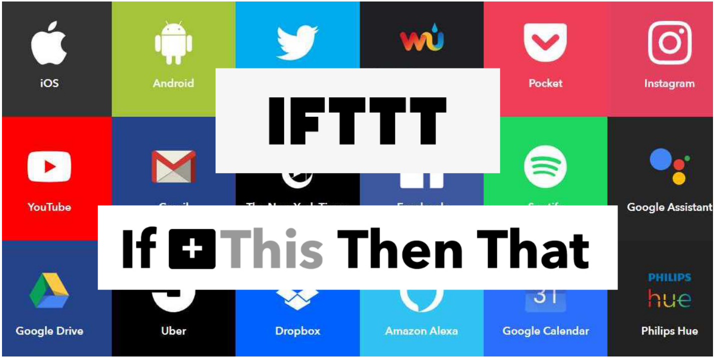
Assignment: L16_Motion (Learn to Calibrate)



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

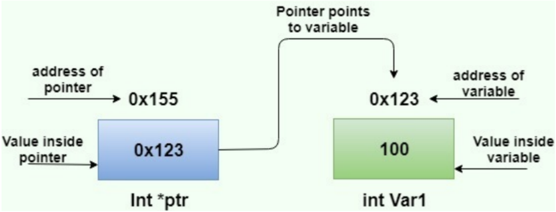
1. L16_10_Scale
   - Set initial cal_factor to 1000 and measure calibration_weight
   - Revise cal_factor to get proper measurement in grams
   - Post data to Adafruit.io (once / min)
   - Optional: Send text via IFTTT

IFTTT



If This Then That, also known as IFTTT, is a freeware web-based service that creates chains of simple conditional statements, called applets. An applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, or Pinterest.

Pointers



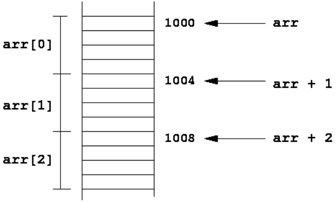
1. A pointer is a variable whose value is the address of another variable.
2. When you declare a pointer, the $*$ symbol denotes that this variable is a pointer variable. For example:
   - Pointer to an Integer: int *ptr;
3. Reference operator (&) gives the address of a variable.
4. To get the value stored in the memory address, we use the dereference operator ($*$).

Pointers

```
int data = 13;
int data2;
int *ptr;

void setup() {
  Serial.begin(9600);
  delay(1000);
  ptr = &data;           //point ptr to the memory location of data
  data2 = *ptr;          //set data2 to value of data (13)

  // Print the Value and Address of the Variables
  Serial.printf("Variable         Value      Address \n");
  Serial.printf("   data           %i        0x%X    \n",data, &data);
  Serial.printf("   ptr          0x%X        0x%X    \n",ptr, &ptr);
  Serial.printf("   data2          %i        0x%X    \n",data2,&data2);
}
```

```
Serial monitor opened successfully:
Variable        Value       Address
  data          13          0x2003E380
  ptr           0x2003E380  0x2003E3F4
  data2         13          0x2003E3F0
```

Pointers and Arrays




```
int arr[] = {100, 200, 300};

void loop() {
  // Compiler converts below to *(arr + 2).
  Serial.printf("%i \n", arr[2]);

  // So below also works.
  Serial.printf("%i \n", *(arr + 2));
}
```

When an array (arr[]) is declared, the variable is a pointer to the first element of a continuous block of memory. In this case 3 elements, 4-bytes each in size, so a total of 12-bytes.

Finding Average of an Array

```
// This function finds the average of an array
// The array is passed to it as a pointer

float getAve(int *array,int size) {
  int j;
  float total=0;
  for(j=0;j<size;j++) {
    total += array[j];
  }
  return total/size;
}
```

Finding Average of Arrays in Action

```
int xArray[4], yArray[256];
int *pointerX, *pointerY;
float average;
int i, sizeX, sizeY;

void setup() {
  pointerX=&xArray[0];
  sizeX=sizeof(xArray)/4;
  pointerY=&yArray[0];
  sizeY=sizeof(yArray)/4;
  for(i=0;i<sizeX;i++) {
    xArray[i] = random(0,255);
  }
  for(i=0;i<sizeY;i++) {
    yArray[i] = random(256,512);
  }
  average = getAve(pointerX, sizeX);
  average = getAve(pointerY, sizeY);
}
```

```
Array X Average = 162.50
Array Y Average = 388.35
xArray[0] value: 173, *pointerX:      173, pointerX:   0x2003E3DC
xArray[1] value: 179, *(pointerX+1): 179, pointerX+1: 0x2003E3E0
xArray[2] value: 110, *(pointerX+2): 110, pointerX+2: 0x2003E3E4
xArray[3] value: 188, *(pointerX+3): 188, pointerX+3: 0x2003E3E8
```

Array of Functions via Pointers

```
//Array of pointers to each of the functions
int (* funky[4])(int x, int y) = {add,sub,mult,divi};

int a,b,answer,i;

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

void loop() {
  a = random(0,100);
  b = random(0,100);
  for(i=0;i<4;i++) {
    answer = funky[i](a,b);
    Serial.printf("For function %i: a = %i and b = %i equals %i \n",i,a,b,answer);
    delay(250);
  }
  Serial.printf("\n\n\n");
  delay(3000);
}

// The Functions
int add(int x,int y) {return x+y;}

int sub(int x,int y) {return x-y;}

int mult(int x,int y) (return x*y;}

int divi(int x,int y) {return x/y;}
```

Returning Multiple Values from a Function

Arguments can be passed to a function by reference; thus, allowing multiple parameters to be returned by a function.

```
void setup() {
  x = 4;
  y = 2;
}

void loop() {
  swap(&x, &y);
  Serial.printf("%i%i\n",x,y);
  delay(1000);
}

void swap(int *x, int *y) {
  int temp;

  temp = *x;
  *x = *y;
  *y = temp;
}
```

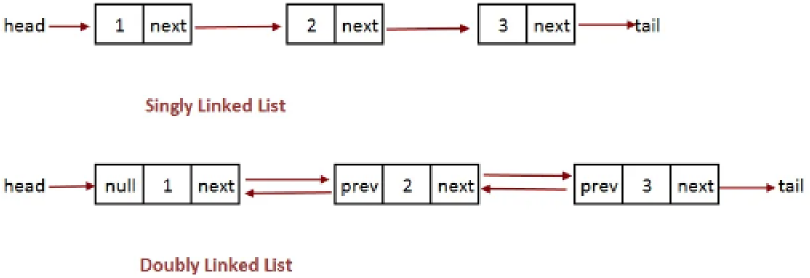
Struct datatype and Member Operators

```
struct geo {
  float lat;
  float lon;
  int alt;
  };
geo myLoc;
geo *ptLoc;

void setup() {
  Serial.begin(9600);
  delay(500);

  ptLoc = &myLoc;
  myLoc.lat = 35.120606;
  myLoc.lon = -106.65818;
  myLoc.alt = 1517;

  Serial.printf("Location: lat %f, lon %f, alt %i  \n",myLoc.lat,myLoc.lon,myLoc.alt);
  Serial.printf("Location: lat %f, lon %f, alt %i \n",ptLoc->lat,ptLoc->lon,ptLoc->alt);
}
```

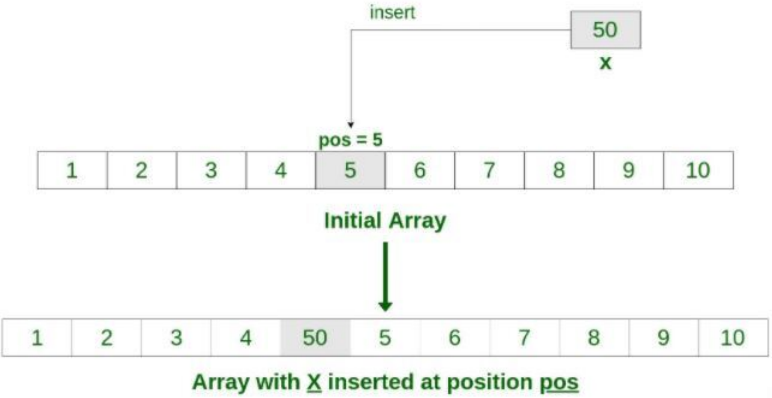
The . (dot) operator and the $- >$ (arrow) operator are used to reference individual members of structures.

- The dot operator is applied to the actual object.
- The arrow operator is used with a pointer to an object.

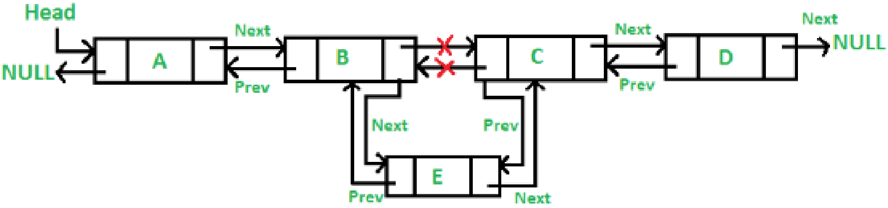
Linked Lists and Doubly Linked Lists



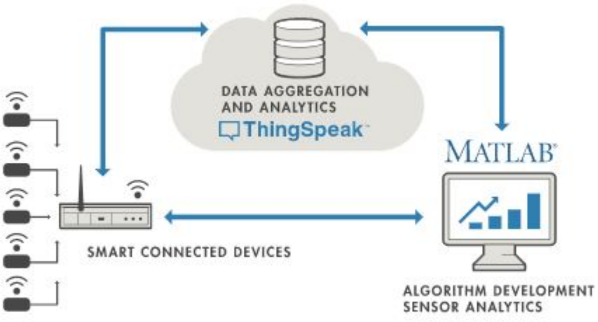
Inserting a "cell" into an Array



Inserting a "cell" into a Linked List



ThingSpeak Dashboard and Matlab Analysis



ThingSpeak™is an IoT analytics platform service from MathWorks®, the makers of MATLAB® and Simulink®. ThingSpeak allows you to aggregate, visualize, and analyze live data streams in the cloud.

ThingSpeak.h

```
#include "ThingSpeak.h"
TCPClient client;

// Change the below to your channel number and APIKeys
unsigned int myChannelNumber = 31461;
const char * myWriteAPIKey = "LD79E0AAWRVYF04Y";
const char * myReadAPIKey = "NKX4Z5JG04M5I18A";

void setup() {
  ThingSpeak.begin(client);
}

void loop() {

  float voltage = ThingSpeak.readFloatField(myChannelNumber, 1, myReadAPIKey);

  pinVoltage = analogRead(A1) * (3.3 / 4095.0);
  ThingSpeak.setField(1,pinVoltage);

  ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

}
```

Information on ThingSpeak channels can be found at: https://community.thingspeak.com/tutorials/thingspeak-channels/

Capstone Projects

Intent:

- Based on a direct observation or a need expressed by one of our guest speakers.
- Original work demonstrating the skills obtained in this class.
- Demonstrate the ability to work as part of a team.
- A pitch to potential employers or investors.

Guidelines:

- Break class into 3 or 4 teams.
- Practical application of smart home, manufacturing, city environment or immersive entertainment.
- Needs to include a Cloud Dashboard component.
- Project will include a video presentation as well as a Hackster.io post.

Previous capstone projects can been found at: `https://www.youtube.com/playlist?list=PL0t2Pk5ETDgxfVptdyr6xbL6MW1-5CJey`

The Big Question: What about main()

```
#include <Arduino.h>

extern "C" int main(void)
{
#ifdef USING_MAKEFILE

  // To use Teensy 3.0 without Arduino, simply put your code here.
  // For example:

  pinMode(13, OUTPUT);
  while (1) {
    digitalWriteFast(13, HIGH);
    delay(500);
    digitalWriteFast(13, LOW);
    delay(500);
  }


#else
  // Arduino's main() function just calls setup() and loop()....
  setup();
  while (1) {
    loop();
    yield();
  }
#endif
}
```

Hello World - What a microcontroller sees

HelloWorld.ino

```
void setup() {
  Serial.begin(9600);
  Serial.printf("Hello World! \n");
}

void loop() {}
```

Hex Code and Assembly Language

```
HelloWorld.bin:     file format binary
Disassembly of section .data:
00000000 <.data>:
    101c: bd10        pop {r4, pc}
    101e: 4402        add r2, r0
    1020: 4603        mov r3, r0
    1022: 4293        cmp r3, r2
    1024: d002        beq.n 0x102c
    1026: f803 1b01   strb.w  r1, [r3], #1
    102a: e7fa        b.n 0x1022
    102c: 4770        bx  lr
    102e: 0000        movs  r0, r0
    1030: b538        push  {r3, r4, r5, lr}
```

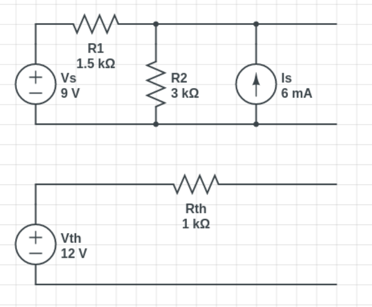

WORK
IN PROGRESS

Thevenin Equivalent Circuits

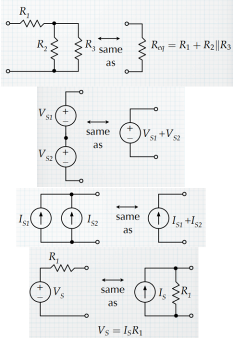
Léon Charles Thévenin (30 March 1857 – 21 September 1926) was a French telegraph engineer who extended Ohm's law to complex circuits.

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

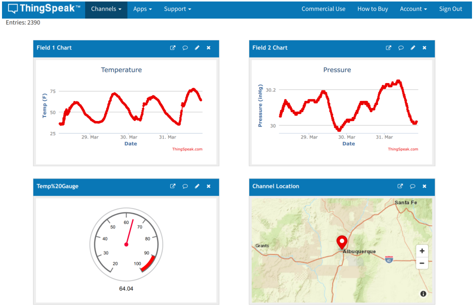
Equivalent Circuit



Useful relationships



ThingSpeak Via Webhooks



We will be learning how to use Particle Webhooks to publish between cloud services. For this, you will need:

1. Particle Argon
2. BME280

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.
JSON is built on two structures:

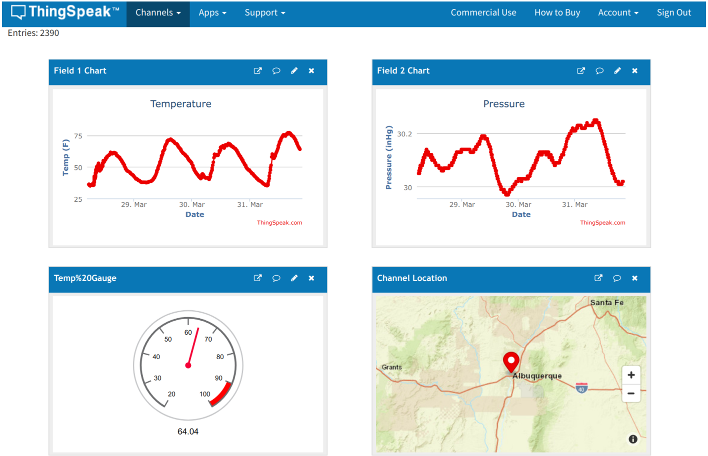
- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

JSON Parser Generator

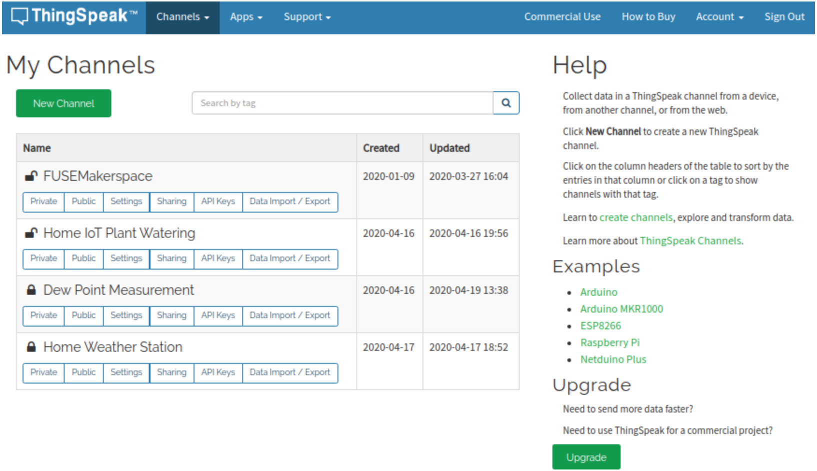
Creating objects in JSON are straightforward but can be tedious. There is a JSON Parser available to simplify the process.

```
#include <JsonParserGeneratorRK.h>

void createEventPayLoad(int moistValue, float tempValue, float presValue, float humValue, int waterED) {
  JsonWriterStatic<256> jw;
  {
    JsonWriterAutoObject obj(&jw);

    jw.insertKeyValue("Moisture", moistValue);
    jw.insertKeyValue("Temperature", tempValue);
    jw.insertKeyValue("Pressure", presValue);
    jw.insertKeyValue("Humidity", humValue);
    jw.insertKeyValue("Plant Watered", waterED);
  }
  Particle.publish("env-vals",jw.getBuffer(), PRIVATE);
}
```

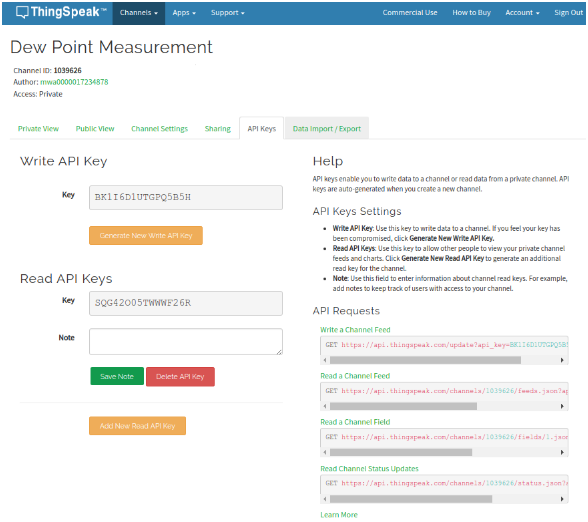
ThingSpeak Dashboard



Step 1 - Create ThingSpeak Channel

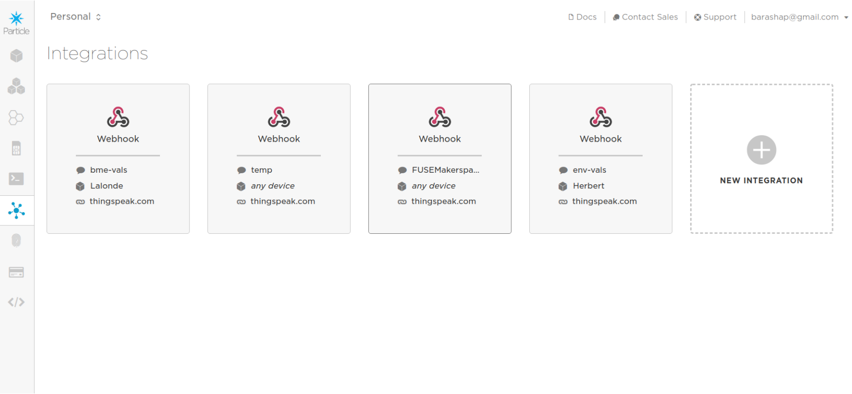


Step 2 - Get API Key

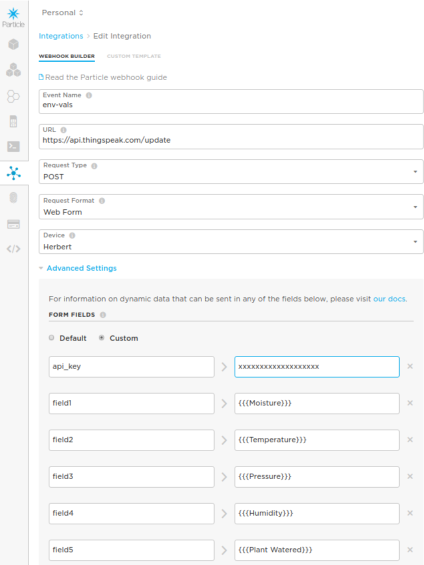


Step 3 - Create Webhook

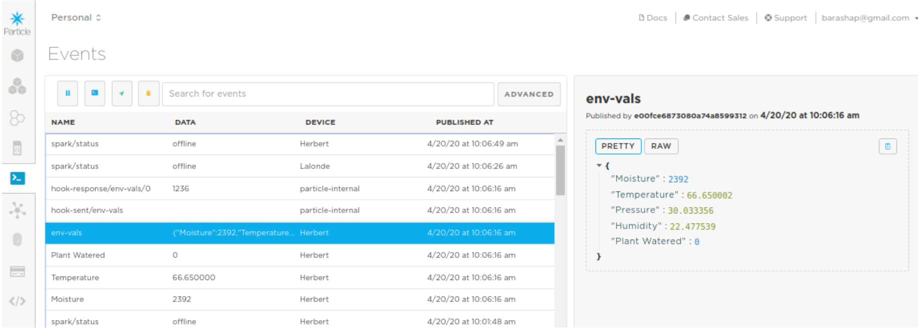
From `console.particle.io`:



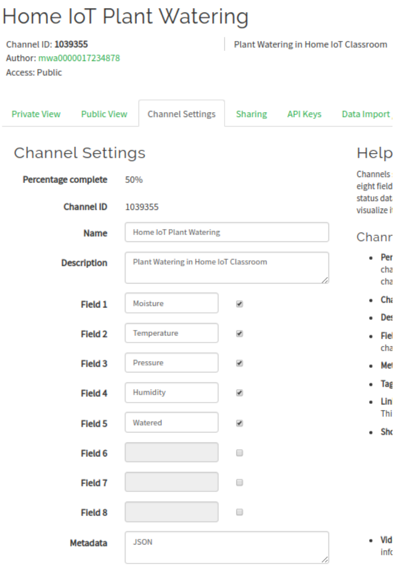
Step 4 - Add JSON Data



Step 5 - Particle Cloud Events

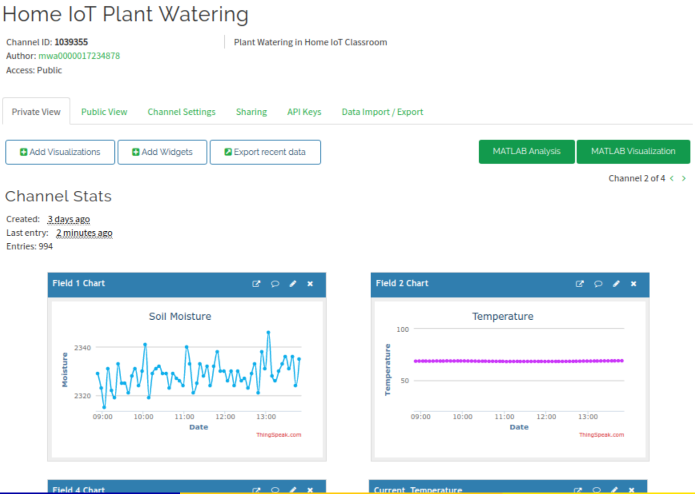


Step 6 - Create Channel

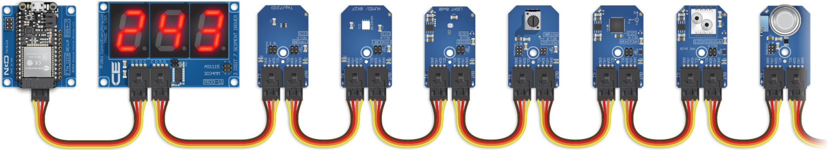


Step 7 - Create Dashboard

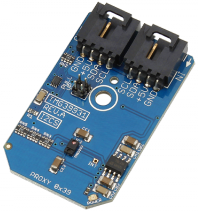
You can change the colors of your lines, by editing the graph. The hex codes are found at `https://htmlcolorcodes.com/color-picker/`



NCD.io ControlEverywhere



NCDio TMG3993 Proximity/Color Sensor



- From Address 0x94
- Request 9 bytes
    - Byte 0 / 1 - LSB / MSB of Infrared Luminance
    - Byte 2 / 3 - LSB / MSB of Red Luminance
    - Byte 4 / 5 - LSB / MSB of Green Luminance
    - Byte 6 / 7 - LSB / MSB of Blue Luminance
    - Byte 9 - Proximity

TMG3993 Initialization

```
  Serial.println("Initializing TMG3993");
  Wire.beginTransmission(Addr);
  // Select Enable register
  Wire.write(0x80);
  // Power ON, ALS enable, Proximity enable, Wait enable
  Wire.write(0x0F);
  Wire.endTransmission();

  Wire.beginTransmission(Addr);
  // Select ADC integration time register
  Wire.write(0x81);
  // ATIME : 712ms, Max count = 65535 cycles
  Wire.write(0x00);
  Wire.endTransmission();

  Wire.beginTransmission(Addr);
  // Select Wait time register
  Wire.write(0x83);
  // WTIME : 2.78ms
  Wire.write(0xFF);
  Wire.endTransmission();

  Wire.beginTransmission(Addr);
  // Select control register
  Wire.write(0x8F);
  // AGAIN is 1x
  Wire.write(0x00);
  Wire.endTransmission();
}
```

NCDio ACD121C MQ9 CO Sensor

12-Bit Analog to Digital Conversion

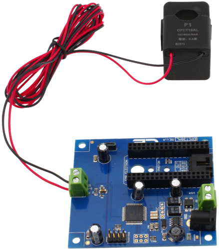
- From Address 0x00
- Request 2 bytes (raw_adc)
    - Byte 0 - MSB
    - Byte 1 - LSB
- CO (ppm) $= \frac{1000}{4096} * raw\_adc + 10$

NCDio ACD121C MQ131 Ozone Sensor

12-Bit Analog to Digital Conversion

- From Address 0x00
- Request 2 bytes (raw_adc)
  - Byte 0 - MSB
  - Byte 1 - LSB
- O_3 (ppm) $= \frac{1.99}{4095} * raw\_adc + 0.01$

NCDio PECMAC Current Sensor



- 1 to 8 channels
- Full range between 10 and 100 amps
- Simple to use. Simply run an AC power wire through the opening of the current sensor. This controller will read the magnetic field inducted onto the current sensor and provide you with a real-world current measurement value that is 98 percent accurate (prior to calibration).