

IoT Product Design and Rapid Prototyping

Brian Rashap

January 2023

Class Logistics



Brian Rashap, Ph.D.

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





Christian Chavez, CEO, Vital Grow Inc

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





Introductions

INTRODUCTIONS



Class Rules

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

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

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



Grading

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

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

Point distribution

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

³All coding assignments must follow style-guide



Credit for Prior Learning (CPL)

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

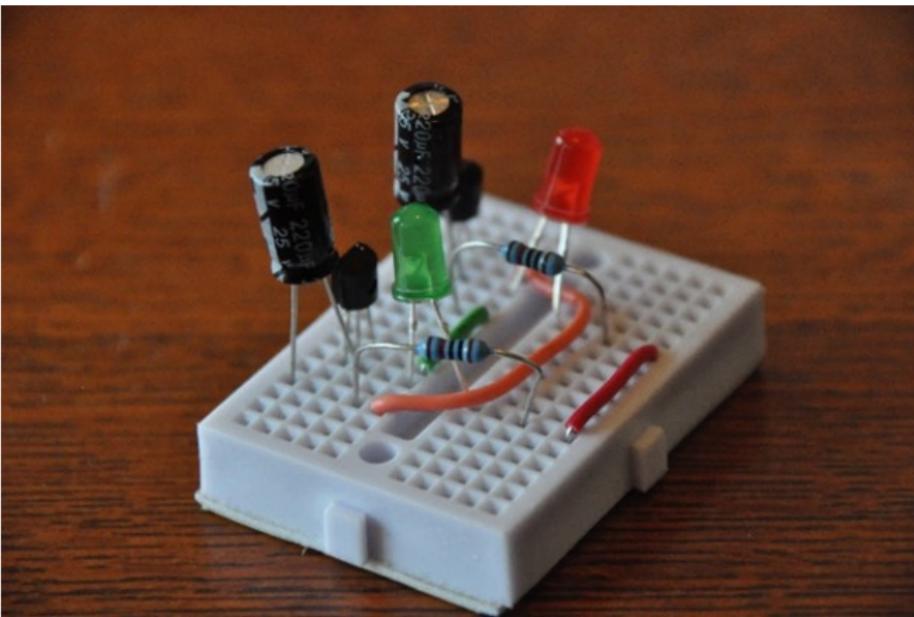
Introduction



"THE CHANGING OF THE PASSWORDS
HAS BECOME A DAILY EVENT."

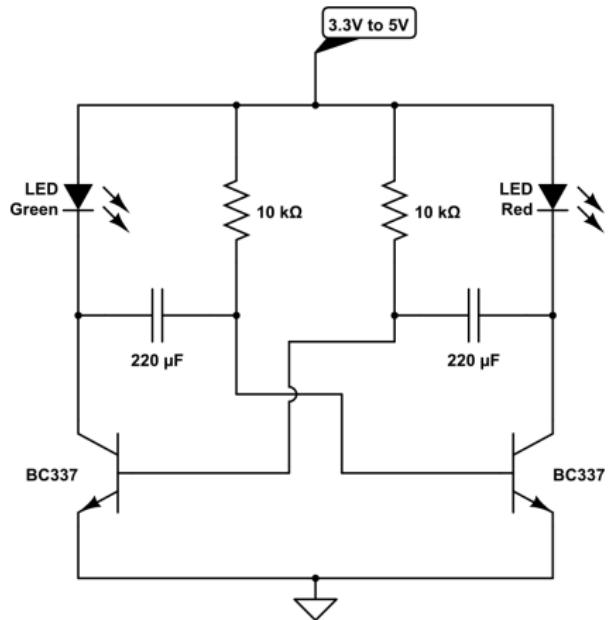


Let's Build Something





Oscillator: Flip Flop

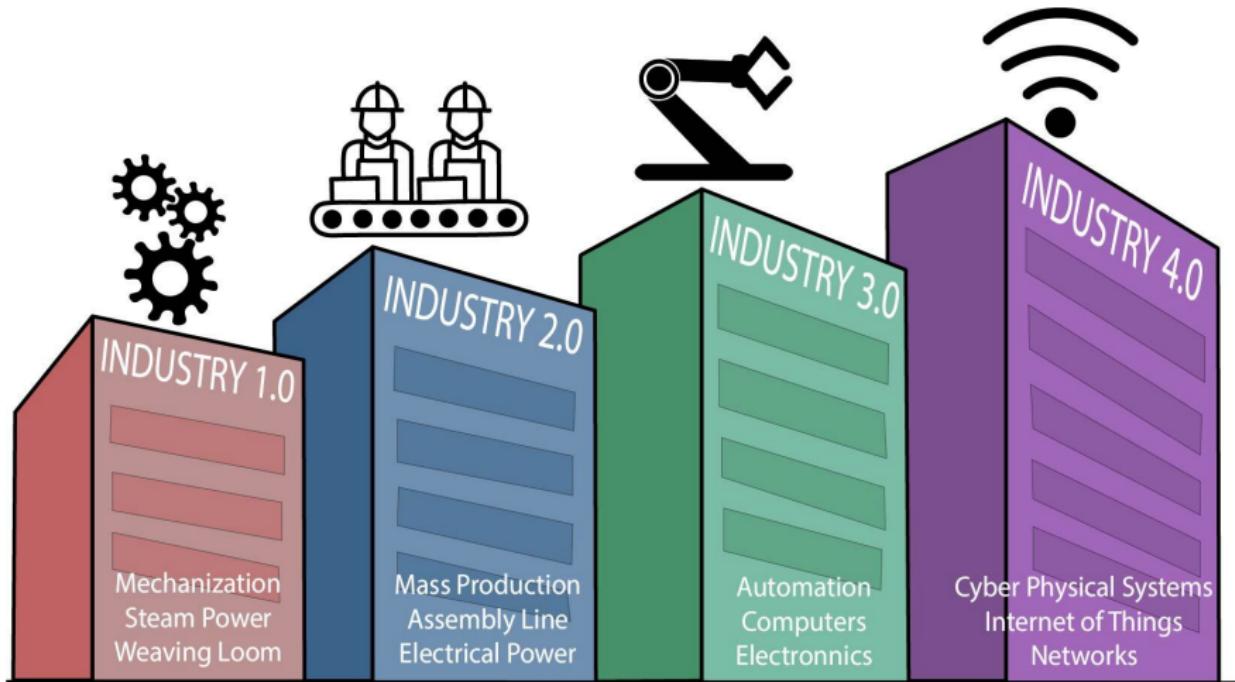


Components:

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

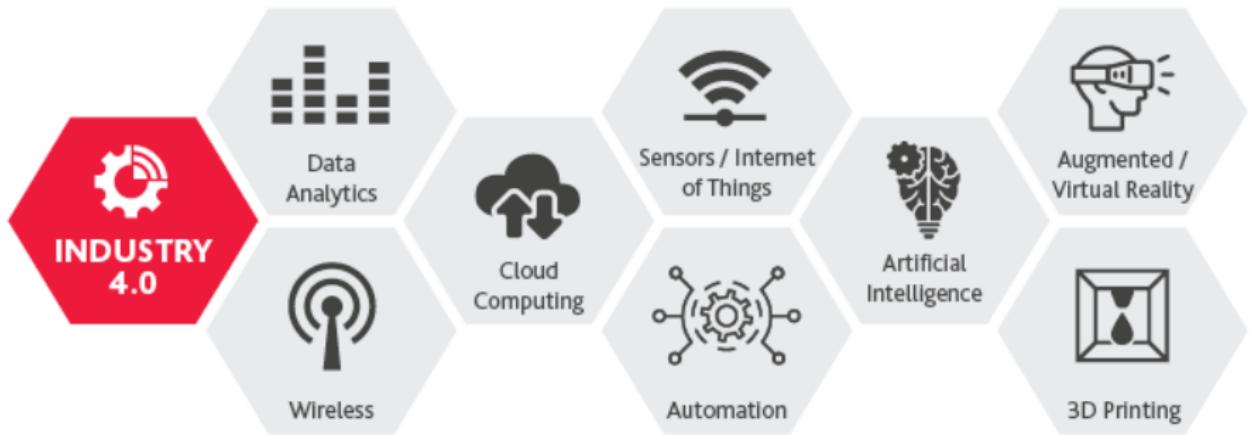


Evolution of Industry



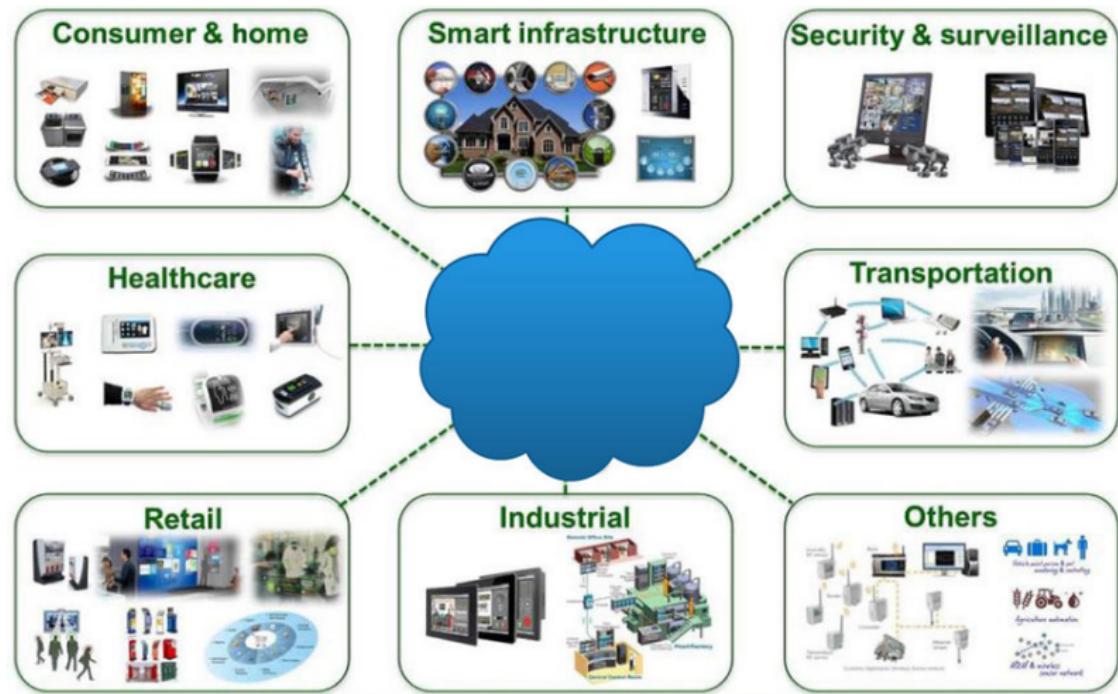


Components of Industry 4.0



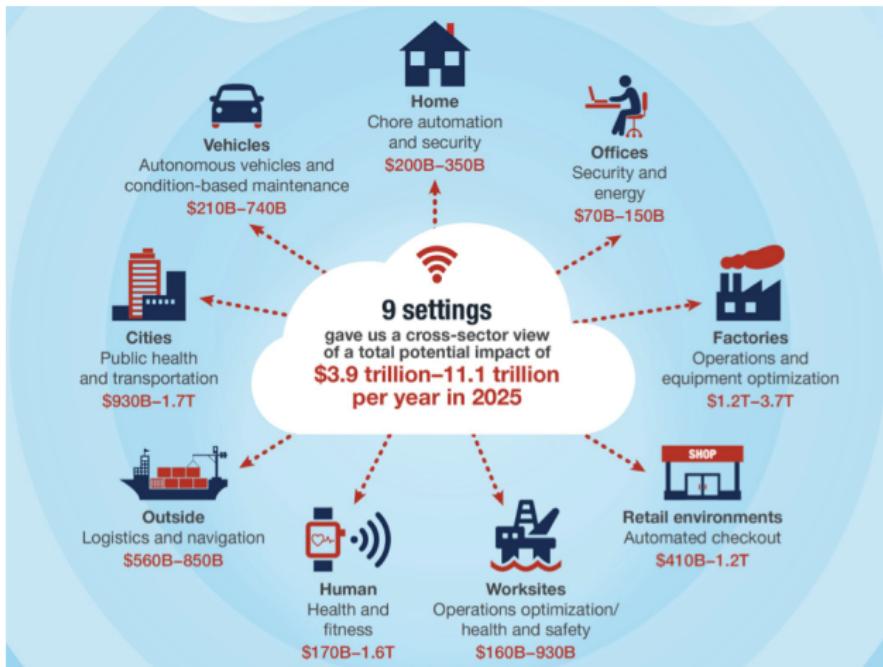


What Is the Internet of Things





IoT 2025



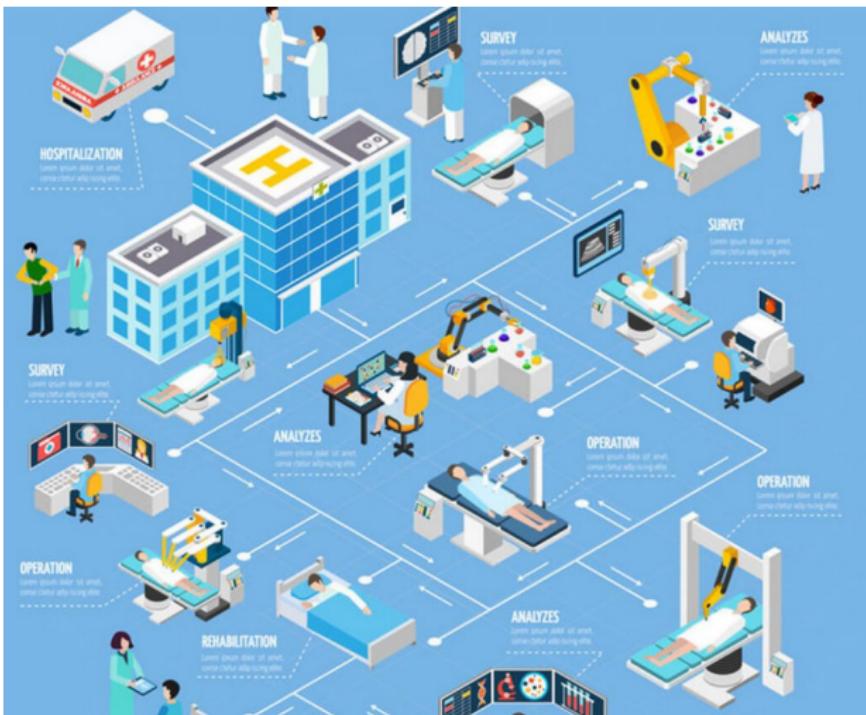


Smart Facilities





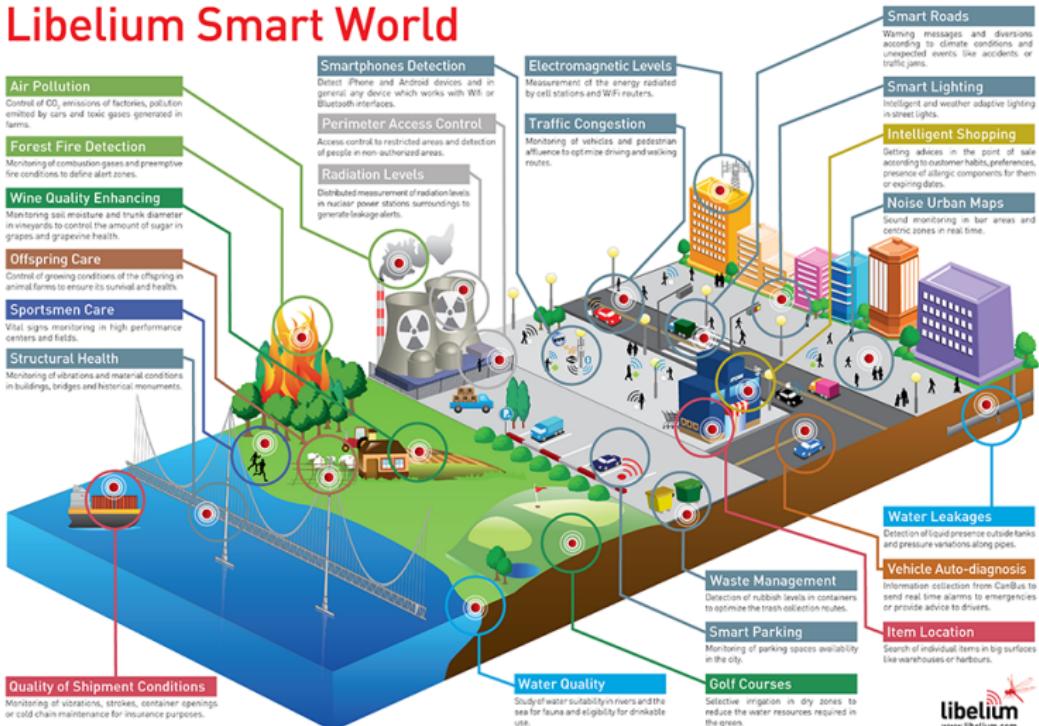
Healthcare 2025





Smart World

Libelium Smart World





And Out of This World





IoT and Data Science



AI: Data-based learning



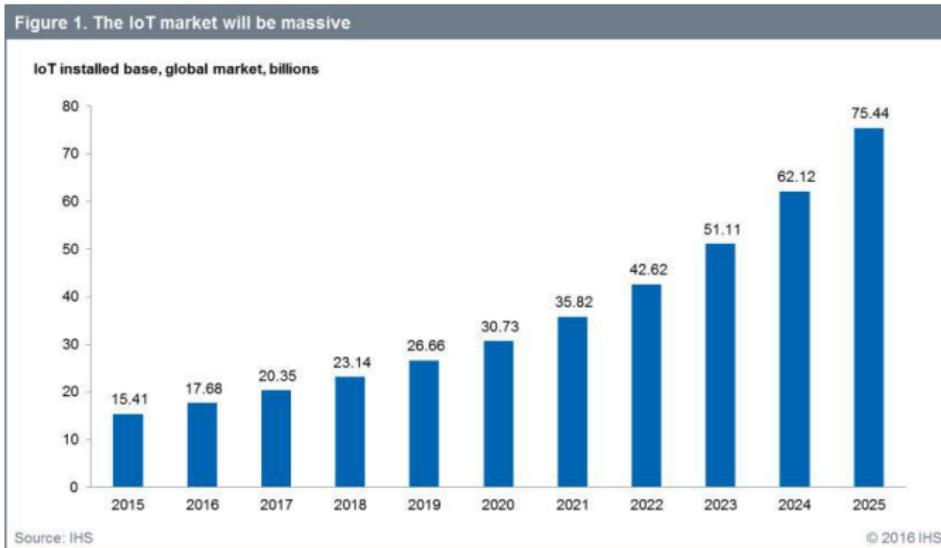
Big Data: Capture, storage, analysis of data



IOT: Data Collection through IoT



How Big Will IoT Be



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.

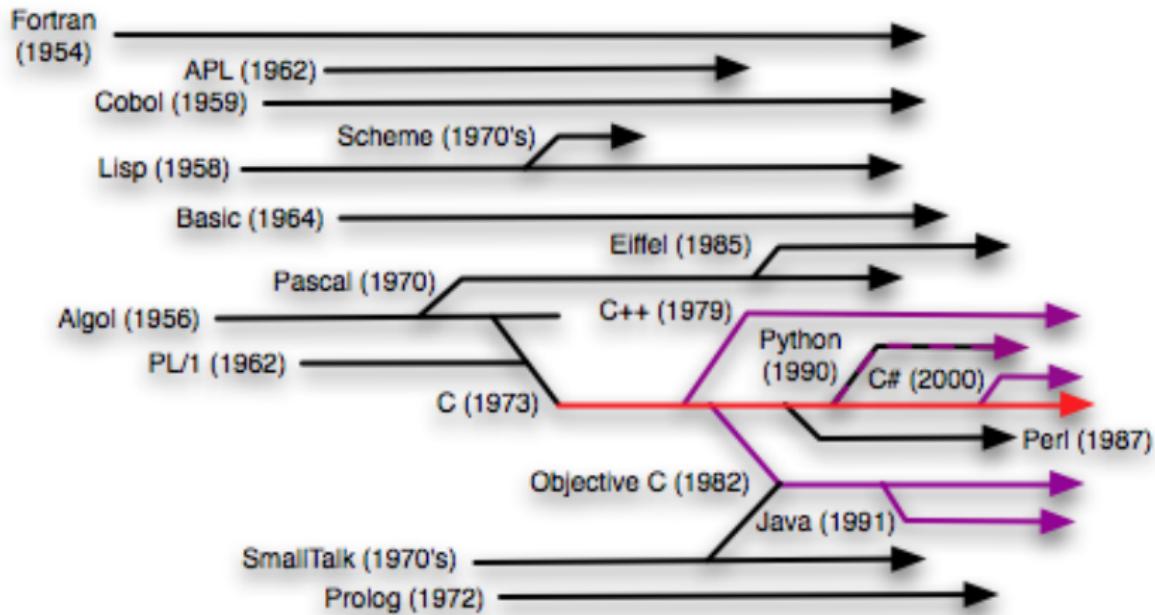


Let's Begin Our Journey





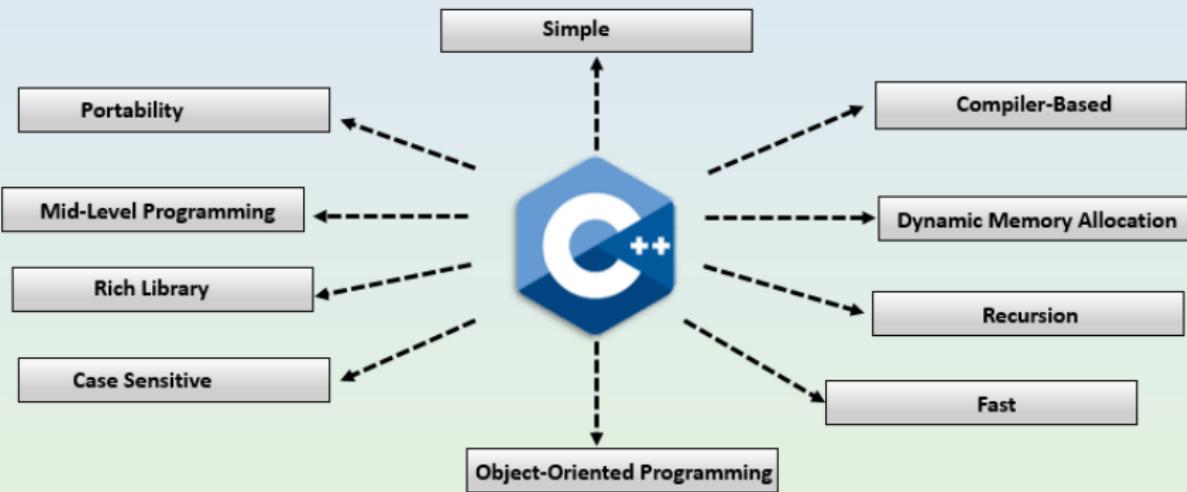
Computer Languages





Why C++

Features of C++



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 pdfs  
drwxr-xr-x. 38 root root 4096 May 18 16:03  
drwxr-xr-x. 2 root root 4096 May 18 16:03 log  
drwxr-xr-x. 2 root root 4096 May 18 16:03 private  
drwxr-xr-x. 2 root root 4096 Jul 1 22:11 repodata  
lrwxrwxrwx. 1 root root 6 May 14 00:12 run -> run  
drwxr-xr-x. 14 root root 4096 May 18 16:03 spool  
drwxrwxrwt. 4 root root 4096 Sep 12 23:58 tmp  
drwxr-xr-x. 2 root root 4096 May 18 16:03 yp  
[root@localhost var]# yum search wiki  
No matches found.  
[root@localhost var]# yum install mediawiki  
[root@localhost var]# rpm -q mediawiki  
mediawiki-1.32.1-1.el8.x86_64
```



VS

Start





Command Line Interface - Basic Navigation

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

- macOS or Linux: Terminal
- Windows: PowerShell

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

- `pwd`: Show the present working directory.
- `ls`: To get the list of all the files or folders.
- `cd`: Used to change the directory.
- `du`: Show disk usage. (not available in PowerShell).
- `man`: Used to show the manual of any command.



Command Line Interface - File and Directory Manipulation

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



Command Line Interface - Displaying the file contents

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

The below commands are not available in PowerShell:

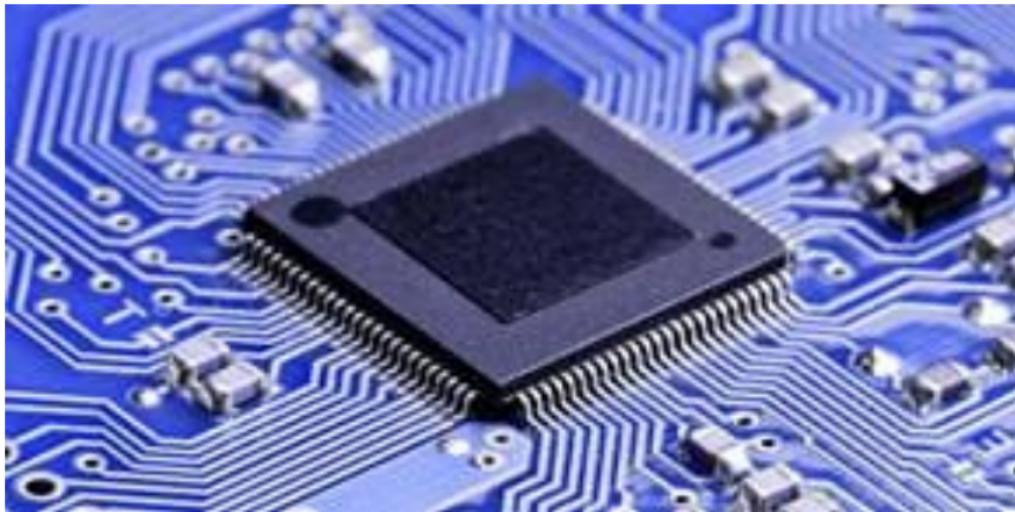
- less: Used for viewing files instead of opening the file. Similar to the "more" command but it allows backward as well as forward movement.
- head: Used to print the first N lines of a file. It accepts N as input and the default value of N is 10.
- tail: Used to print the last N-1 lines of a file.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

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

Particle Argon

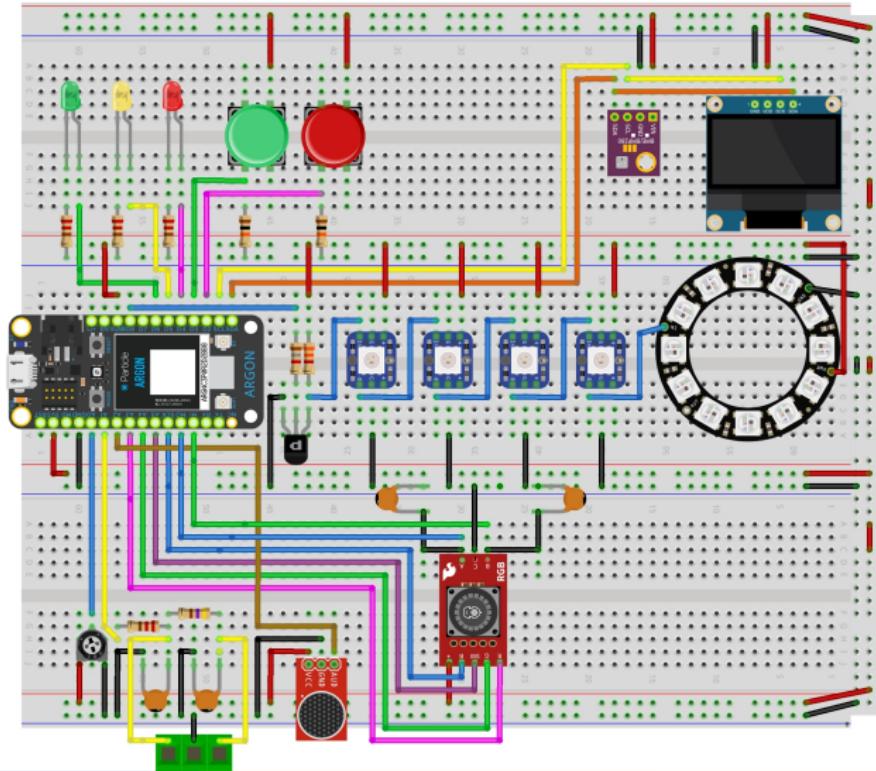


Our Microcontroller



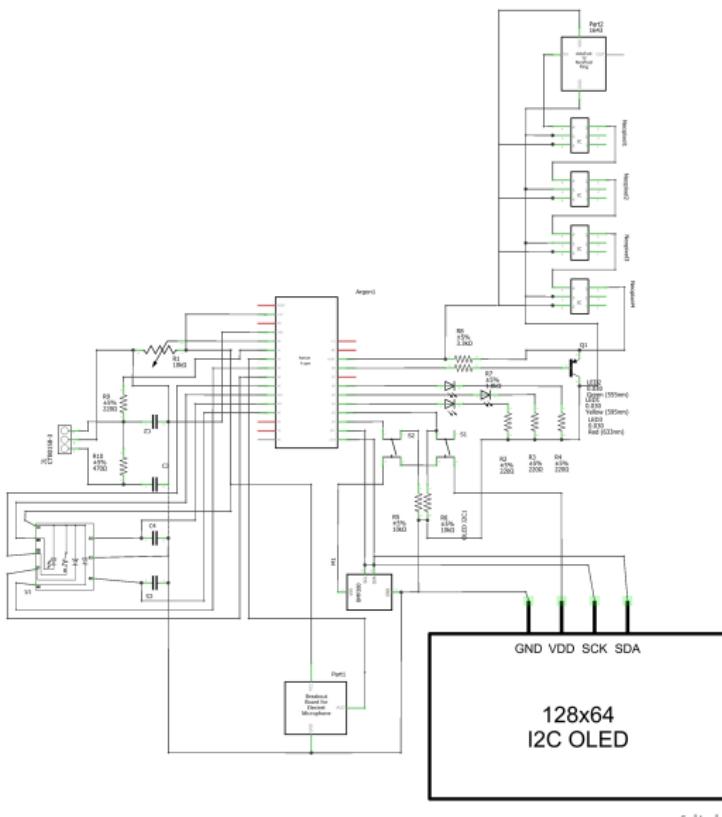


Smart Room Controller





Smart Room Controller Schematic





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

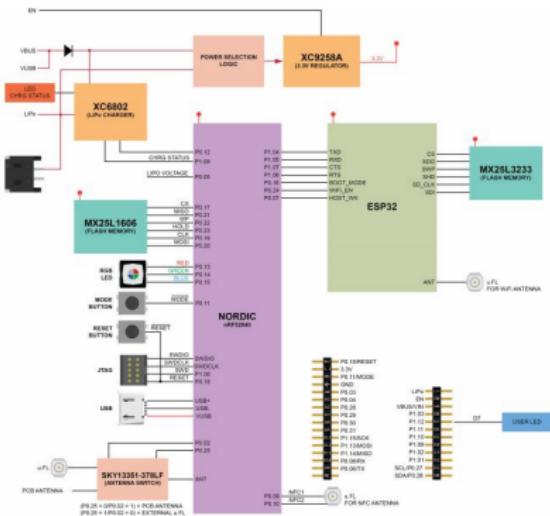


Argon general specifications:

- 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)



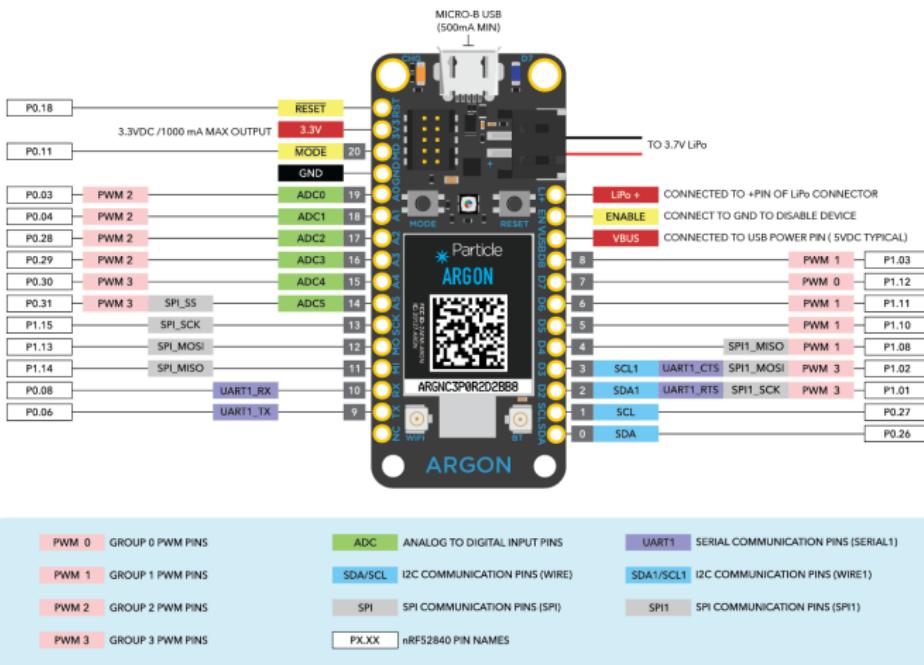
Particle Argon Block Diagram



- nRF52840 (64 MHz ARM M4 Cortex with BLE and NFC)
 - ESP32 (Wifi Coprocessor)
 - 20 GPIO pins
 - Additional 4MB SPI Flash
 - Integrated LiPo battery charging
 - Adafruit Feather pinout



Argon Pinout



v1.0



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, 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.
- ④ On the Mac, install dfu-util per the instructions on the website.



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.

```
1 particle login
```

- ⑤ Ensure you have the latest Particle CLI.

```
1 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 ⁴.

```
1 particle update  
2 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:

```
1 particle serial identify  
2  
3 Your device id is e00fce681fffffffffc08949b  
4 Your system firmware version is 1.5.2
```

⁴particle 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, your 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 e00fce681fffffffffc08949b with the device ID you got earlier from particle serial identify. Then, rename it to the name of your choice.

```
1 particle device add e00fce681fffffffffc08949b  
2 particle device rename e00fce681fffffffffc08949b  
    myArgon
```

- ② Ensure that your setup flag is marked as done.

```
1 particle usb setup -done
```

- ③ You have successfully set up your Argon!



Useful Particle CLI Commands

- ① Enter DFU mode from the CLI.

```
1 particle usb dfu
```

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

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

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

```
1 particle list
```

- ④ Search for available libraries.

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

- ⑤ Link for the Particle Setup procedures: [Particle Setup via CLI](#)



Particle Troubleshooting

- ① Enter DFU mode from the CLI.

```
1 particle usb configure
```



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

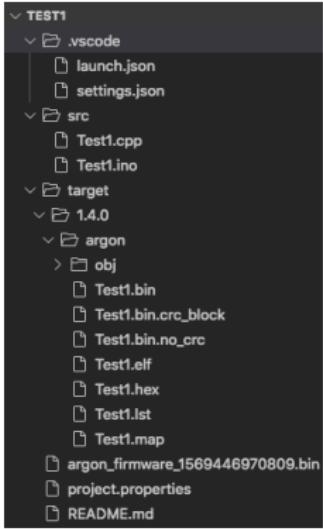
The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows the project structure:
 - OPEN EDITORS: 1 UNSAVED
 - PM25_Testino (selected)
 - 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.cpp
 - PM25_Test.ino (selected)
 - target\1.5.0\argon
 - project.properties
 - README.md
- EDITOR** view: Displays the content of the PM25_Testino.ino file:

```
src > PM25_Test.ino > ...
1 /*
2  * Project PM25
3  * Description: 2.5um Particle Measurement with HM3301 Sensor
4  * Author: Brian Rashap
5  * Date: 17-APR-2020
6 */
7 #include <Particle.h>
8 #include <Seeed_HM330X.h>
9 #include <Wire.h>
10
11 //*****SetUp HM330X*****
12 HM330X sensor;
13 uint8_t buf[30];
14 int PM25;
15
16 const char* str[] = {"sensor num: ", "PM1.0 concentration(CF=1,Standard particulate matter",
17 "PM2.5 concentration(CF=1,Standard particulate matter,unit:ug/m3): ",
18 "PM10 concentration(CF=1,Standard particulate matter,unit:ug/m3): ",
19 "PM1.0 concentration(Atmospheric environment,unit:ug/m3): ",
20 "PM2.5 concentration(Atmospheric environment,unit:ug/m3): ",
21 "PM10 concentration(Atmospheric environment,unit:ug/m3): ",
22 };
23
24 HM330XErrorCode print_result(const char* str, uint16_t value) {
25     if (NULL == str) {
26         return ERROR_PARAM;
27     }
28     Serial.print(str);
29     Serial.println(value);
30     return NO_ERROR;
```



Directory Structure - VERY IMPORTANT



- The .vscode directory contains your project-specific settings.
- The src directory contains your source file. The .ino file is transformed into a .cpp before compiling.
- Not pictured here is the lib directory, at the same level as src. This contains the source to libraries that you have included.
- The target directory contains local build output
- The *.bin file is the result of a cloud compile for this project.
- The project.properties file specifies all of the libraries that this project uses.
- README.md is where you could put documentation for your project.



Command Palette - Ctrl-Shift-P

>particle

- Particle: Install Library** recently used
- Particle: Find Libraries**
- Particle: Cloud Compile**
- Particle: Configure Workspace for Device**
- Particle: Launch CLI**
- Particle: Install Local Compiler**
- Particle: Cloud Flash**
- Particle: Serial Monitor**
- Particle: Create New Project**
- Particle: Audit Environment**
- Particle: Who Am I?**
- Particle: Clean application (local)** other commands
- Particle: Clean application & DeviceOS (local)**



Improve Compile Time

Anti-virus programs scan everything that VSCode is doing. This leads to long compile times. To reduce the compile times, you can exclude your particle code from real-time virus scans.

Manage exceptions

Add or remove items to be excepted from scan.

+ Add an Exception

All exceptions Antivirus Advanced Threat Defense

c:\users\ddcio\particle
On-access, On-demand, Embedded scripts

c:\users\ddcio\vscode
On-access, On-demand, Embedded scripts

c:\users\ddcio\documents\iot
On-access, On-demand, Embedded scripts



Go to Start > Settings > Update & Security > Windows Security > Virus & threat protection. Under Virus & threat protection settings, select Manage settings, and then under Exclusions, select Add or remove exclusions. Select Add an exclusion, and then select from files, folders, file types, or process.



How to exclude files and folders from Bitdefender Antivirus scan

1. Click Protection on the navigation menu on the Bitdefender interface.
2. In the ANTIVIRUS pane, click Open.
3. In the Settings window, click Manage Exceptions.
4. Click +Add an Exception.



To set a global exception:

1. Open AVG AntiVirus and go to ≡ Menu ▶ Settings.
2. Select General ▶ Exceptions, then click Add exception.
3. Add an exception in one of the following ways: Type the specific file/folder path or URL into the text box, then click Add exception.



1. Open your McAfee security software.
2. Click PC Security.
3. Click Real-Time Scanning.
4. Click Excluded Files.
5. Click Add file.
6. Browse to, and select, the file that you want to exclude from Real-Time scanning.



GitHub - Part 1



Git and GitHub: Your Version Control Friends



GitHub



What is a version control system?

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

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

Git is a VCS, or Version Control System.

It is similar to Backup on Windows or Time Machine on the Mac.



Installing Git

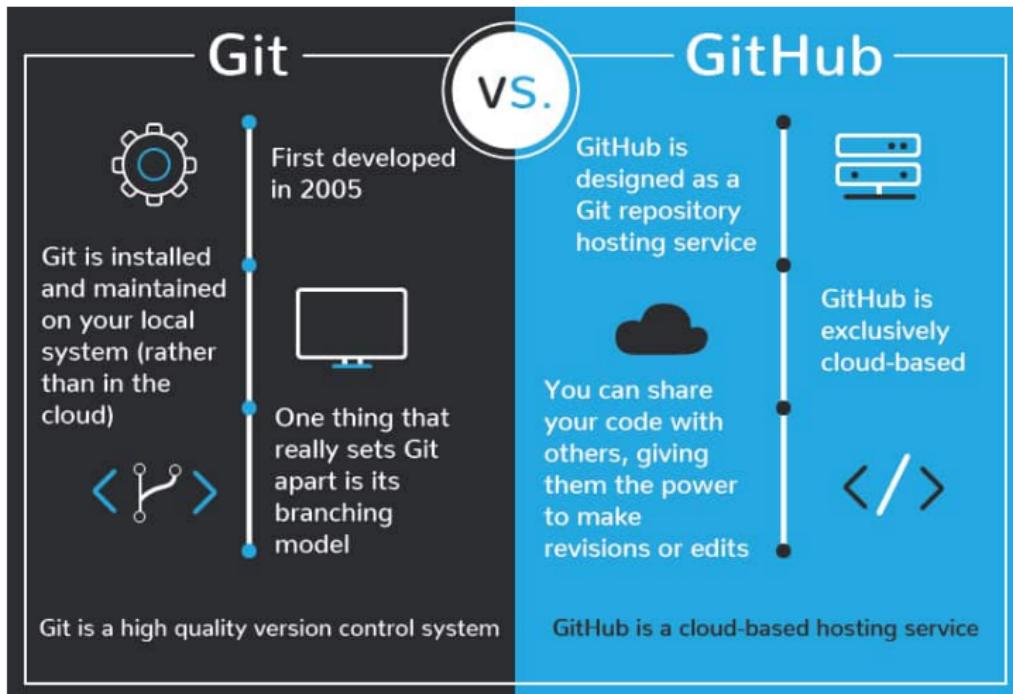
If you have not already done so, install Git on your computer.

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

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



Git vs. GitHub





Commit vs. Push

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

→ git commit

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

→ git push

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



When should you commit and push your work?

When to commit? **OFTEN!**

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

When to push? **OFTEN!**

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



Getting a GitHub account

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

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



GitHub Authentication: Using HTTPS and PAT

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

To create a PAT:

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

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



GitHub: Cloning and Pulling Code

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

git clone <URL of repository>

Let's get the Class Slide now, from your IoT directory:

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

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

git pull

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



GitHub: Getting Assignments

ddc-iot-classroom-2

Accept the assignment —

L01_HelloWorld

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



You're ready to go!

You accepted the assignment, L01_HelloWorld.

Your assignment repository has been created:

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

Accept this assignment

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



GitHub: Cheatsheet. Memorize this!

```
1 // In PowerShell go to ./Documents/IoT
2 // Get a repository that already exists and pull
   it into your local machine
3 git clone <URL of repository>
4
5 // Send your changes up to the repository
6 git add . //adds all changed files
7 git commit -m "some comment"
8 git push //send your changes to the cloud
9
10 // The first time you use git, you may get asked
    to enter your GIT username
11 git config --global user.email "you@example.com"
12
13 // From the repository directory, get updates
14 git pull
```



Other Software

① Fritzing

- IoT Bootcamp Teams Site
- Note: to extract faster, <https://www.7-zip.org/download.html>

② KiCad

- <https://www.kicad.org/download/>

③ Drawio

- <https://app.diagrams.net/>

④ Adobe Illustrator

- <https://www.adobe.com/creativecloud.html>

⑤ Formlab's Preform

- <https://formlabs.com/software/>

⑥ Ultimaker's Cura

- <https://ultimaker.com/software/ultimaker-cura>

⑦ Bookmark: <https://www.desmos.com/>



Solidworks

To install Solidworks (Windows only), go to

<http://www.SolidWorks.com/SEK>

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

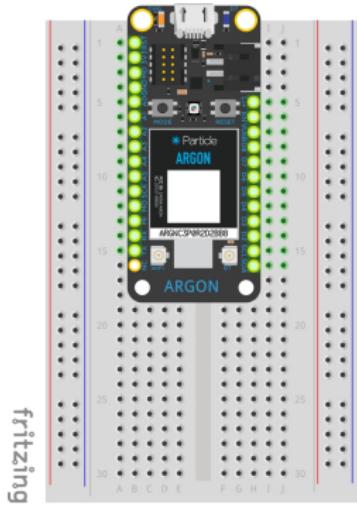
macOS and Linux users will use onShape:

<https://www.onshape.com/en/education/>

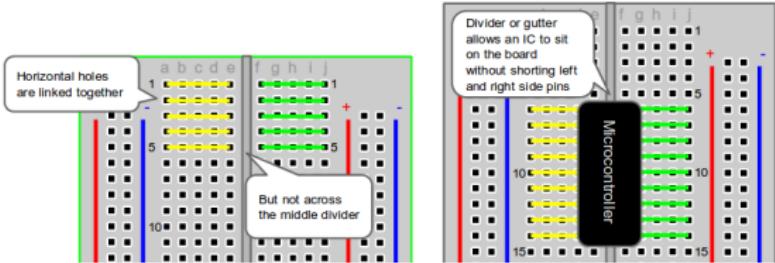
Module 1 - HelloWorld



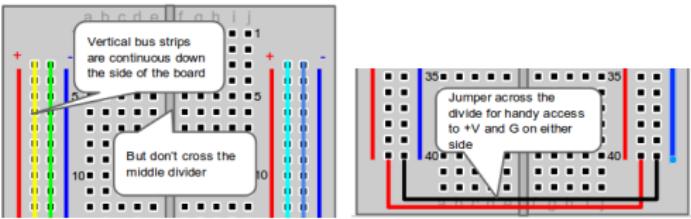
Argon on Breadboard



Horizontal Rows



Vertical Columns





Basic Structure of Arduino Sketch

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



Class Assignments

- ① Lab Notebook - flow chart
- ② Lab Notebook - schematic
- ③ Fritzing breadboard layout
- ④ Arduino code with comments

```
1 /*
2  * Project:      Title of Project
3  * Description: Description of Project
4  * Author:       Your Name
5  * Date:        Today's Date
6 */
7
8 // Single Line Comments
```



Hello World in some of the 603+ Coding Languages

Fortran

```
c      Hello world in Fortran
      PROGRAM HELLO
      WRITE (*,100)
      STOP
100 FORMAT (' Hello world! ' /)
      END
```

C (K&R)

```
/* Hello world in c, K&R-style */
main()
{
    puts("Hello world!");
    return 0;
}
```

Python 2

```
# Hello world in python_2
print "Hello world"
```

Assembler (Intel)

```
; Hello world for intel Assembler (MSDOS)
mov ax,cs
mov ds,ax
mov ah,9
mov dx, offset Hello
int 21h
xor ax,ax
int 21h
```

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

Powershell

```
# Hello World in Microsoft Powershell
'Hello world!'
```

LabVIEW

Hello world in LabVIEW 7.1

LaTeX

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

Lisp-Emacs

```
(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
```

Unix Shell

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



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?



Other Software

① Git

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

② Fritzing

- IoT Bootcamp Teams Site
- Note: to extract faster, <https://www.7-zip.org/download.html>

③ Drawio

- <https://app.diagrams.net/>

④ Adobe Illustrator

- <https://www.adobe.com/creativecloud.html>

⑤ Formlab's Preform

- <https://formlabs.com/software/>

⑥ Ultimaker's Cura

- <https://ultimaker.com/software/ultimaker-cura>

⑦ Bookmark: <https://www.desmos.com/>



Solidworks

To install Solidworks (Windows only), go to

<http://www.SolidWorks.com/SEK>

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

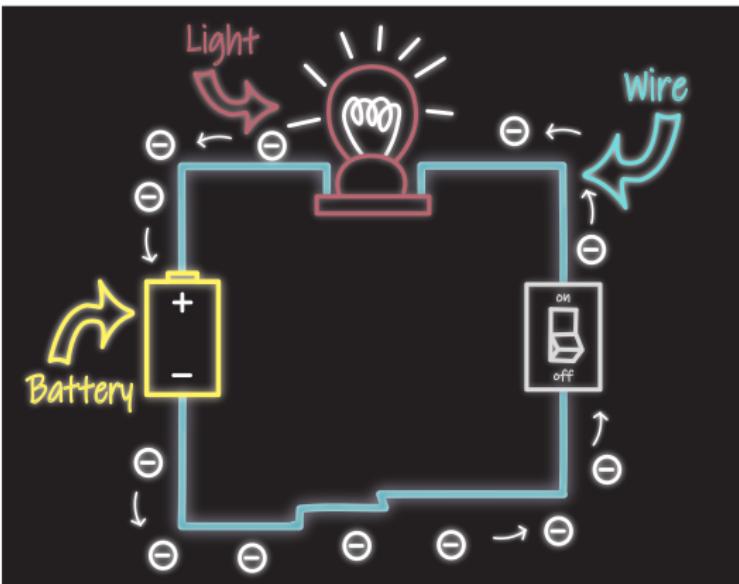
macOS and Linux users will use onShape:

<https://www.onshape.com/en/education/>

Module 2 - HelloLED

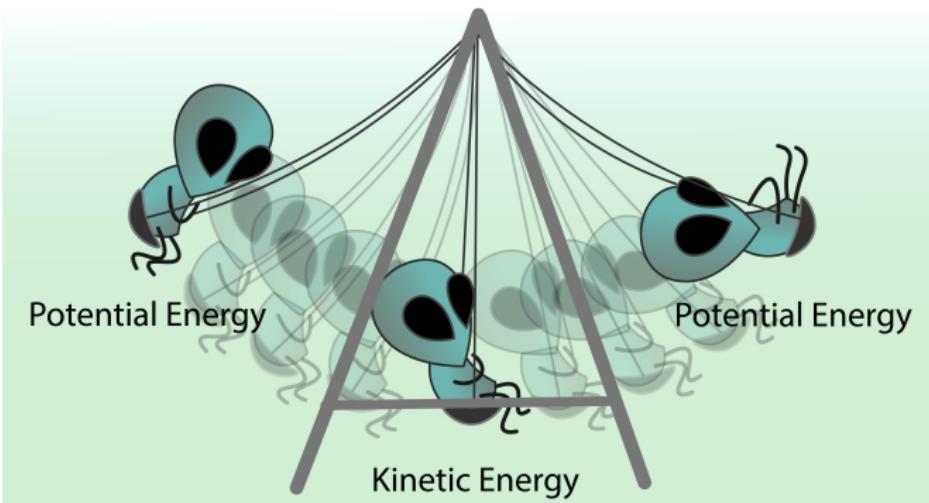


Introduction to Electrical Circuits





Energy



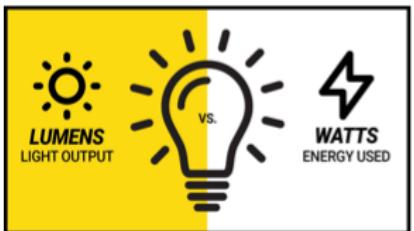
- Kinetic Energy - energy of motion
- Potential Energy - energy stored in an object



Electrical Circuit Terms

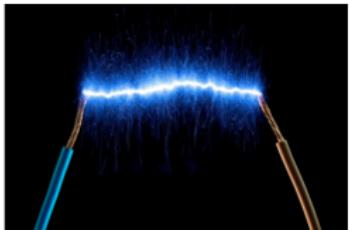


Voltage is **electric potential energy per unit charge** ($V = J/C$)

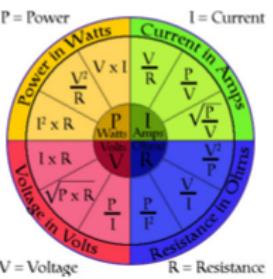


Power is the rate of doing work or **the rate of using energy**. ($W = J/S$)

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



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



$$\text{Power} = \text{Voltage} \times \text{Current}$$



Energy is the **amount of power produced or consumed over a given time**. ($J = W \times s$)

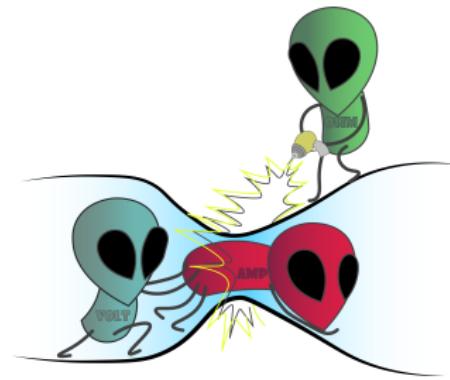
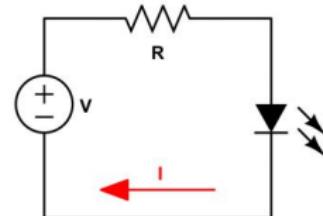


Ohm's Law

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

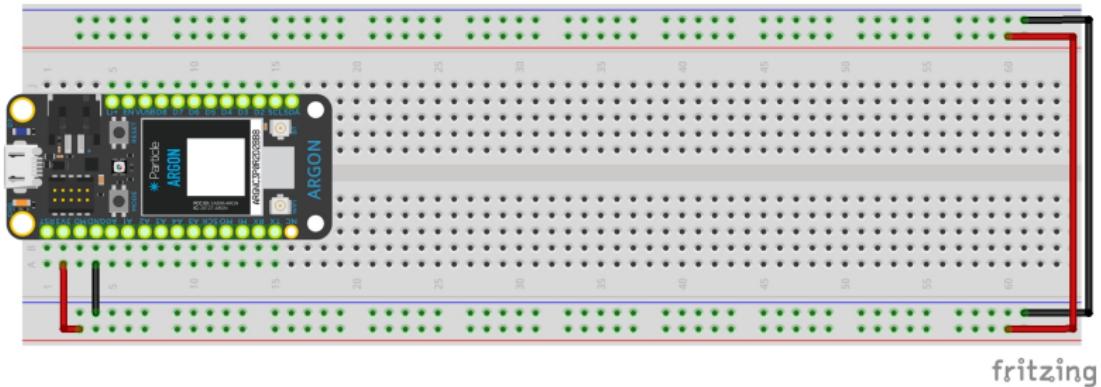
Ohm's Law

$$V = I * R$$





Power from the Argon

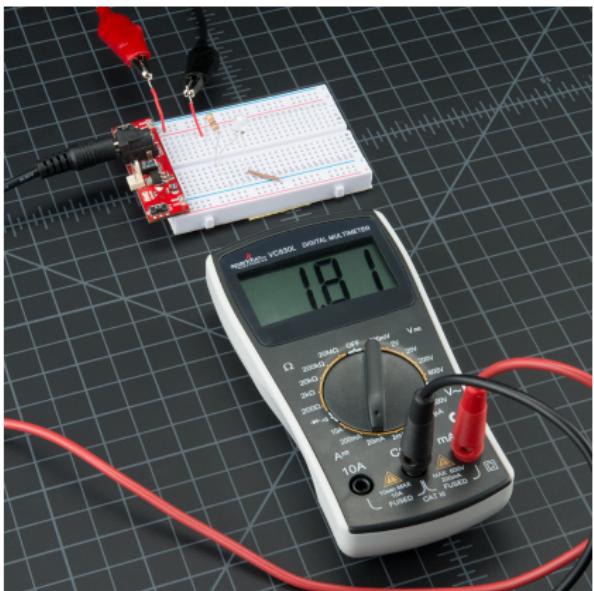


The Particle Argon has three pins related to power:

- 3.3V: 500mA of power to be used for most hardware
- V_{BUS} : 5V from the USB cable to power 5V hardware
- GND: The ground pin to close the electrical loop

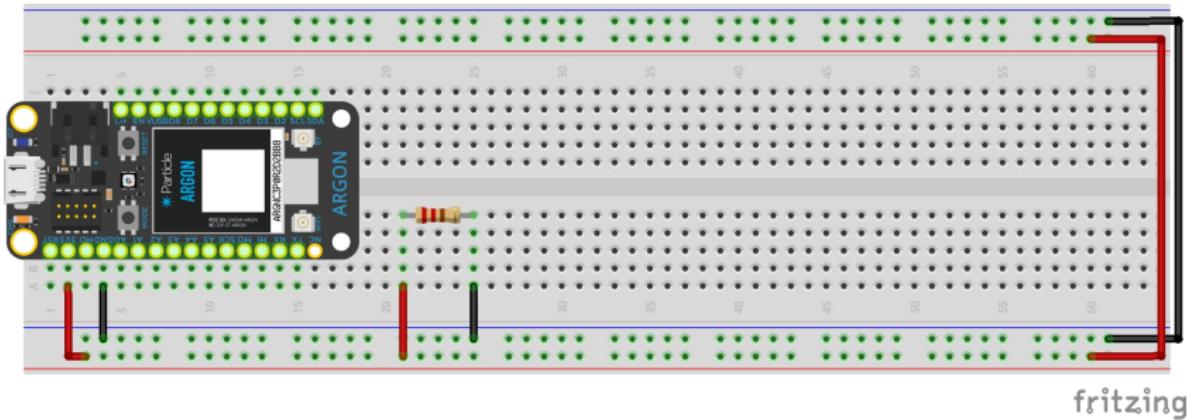


Measuring Voltage, Current, and Resistance





One Resistor

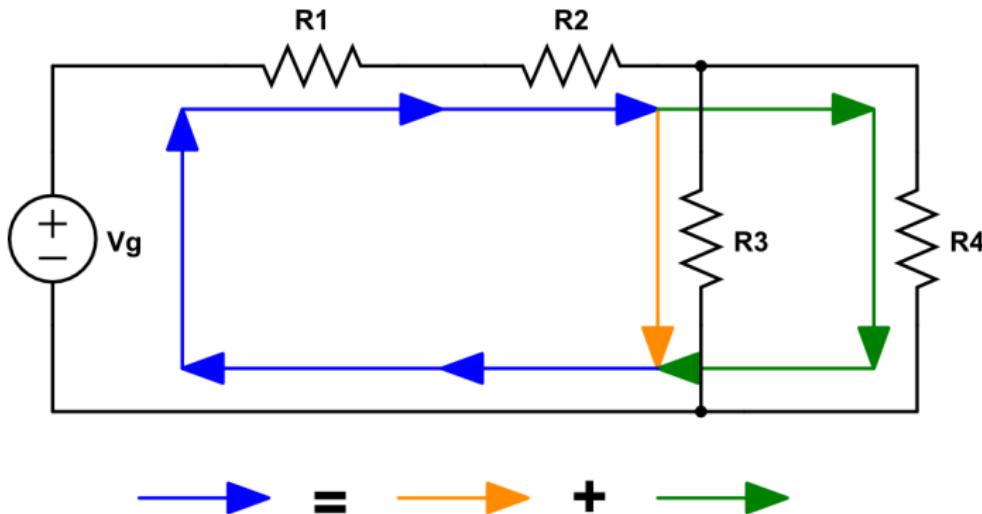


fritzing

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

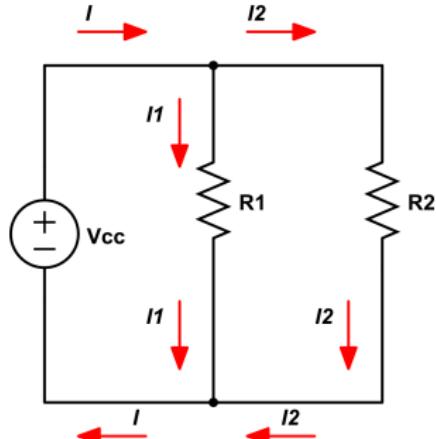
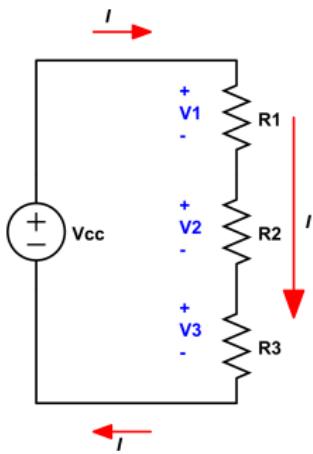


Resistors in Series and Parallel





Resistors in Series and Parallel

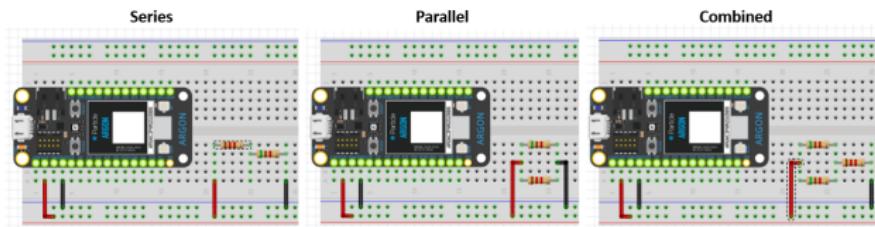


$$R_{eq} = R_1 + R_2 + R_3$$

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



Assignment: L02_00_Resistors

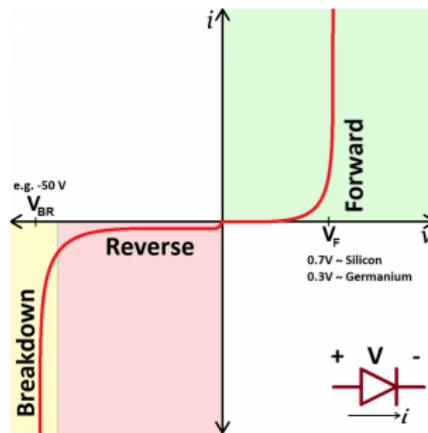


- In your lab notebook, draw the circuit diagrams
 - ① Series: $5.1\text{k}\Omega$ and $1.2\text{k}\Omega$
 - ② Parallel: $5.1\text{k}\Omega$ and $1.2\text{k}\Omega$
 - ③ Combined: Two parallel $5.1\text{k}\Omega$ in series with $1.2\text{k}\Omega$
- Calculate the combined resistance, the voltage at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



Diodes

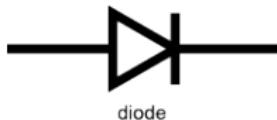
The key function of a diode is to control the direction of current-flow. Current passing through a diode can only go in one direction, called the forward direction. Current trying to flow the reverse direction is blocked.





Light Emitting Diodes

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



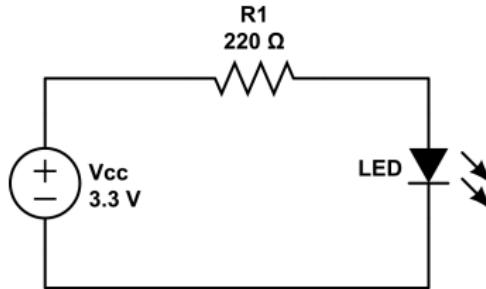


Current Limiting Resistors

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

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

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 .





SYSTEM_MODE

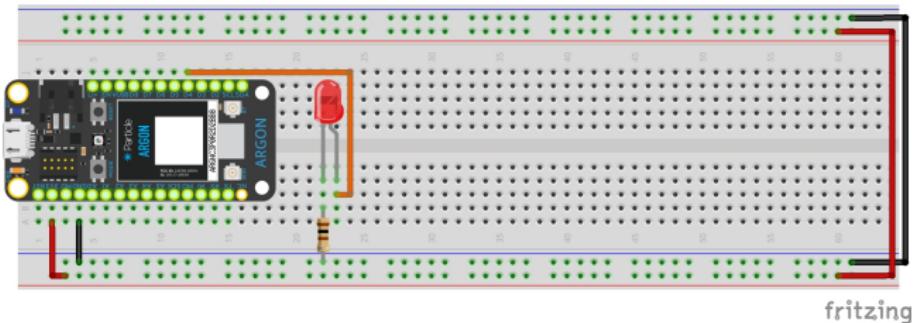
System modes help control how the device manages the connection with the cloud. By default, the device connects to the Cloud and processes messages automatically. However there are many cases where a user will want to take control over that connection. There are three available system modes:

- AUTOMATIC,
- SEMI_AUTOMATIC,
- MANUAL.

```
1 //The below is placed in the header before Void Setup()
2
3 // SYSTEM_MODE(AUTOMATIC);           // Default if no SYSTEM_MODE included
4 // SYSTEM_MODE(SEMI_AUTOMATIC);     // Uncomment if using without Wifi
5 // SYSTEM_MODE(MANUAL);            // Fully Manual
```



Assignment L02_01_helloLED



fritzing

- Using Pin D4 as an output and the appropriate current limiting resistor, blink the LED once per second.
- Measure the voltage at both leads of the LED and record the voltage in your notebook.
- Change the resistor to $1k\Omega$ and then $10k\Omega$. What happens to the brightness? Measure the voltage and current in each case. Record in your notebook.

REMEMBER: Lab notebook, Fritzing, breadboard, then code



Constants and Variables

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

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

The use of Constants and Variables has several advantages:

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

The first two Data Types that we will be using:

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



IoT Style Guide - camelCase



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



Operators

There are a number of operators that act on variables:

```
1 // Assignment
2 x = y;      // assign x to be equal to y
3
4 // Math Operators
5 sum = x + y;
6 difference = x - y;
7 product = x * y;
8 quotient = x / y;
9 remainder = x % y;
10
11 // Incrementing
12 i = i + 1;
13 i += 1;
14 i++;
15
16 // Decrementing
17 i = i - 1;
18 i -= 1;
19 i--;
20
21 // Comparison
22 (x == y); // true if x is equal to y
23 (x != y); // true if x is not equal to y
24 (x > y); // true if x is greater than y
25 (x >= y); // true if x is greater than or equal to y
26 (x < y); // true if x is less than y
27 (x <= y); // true if x is less than or equal to y
```

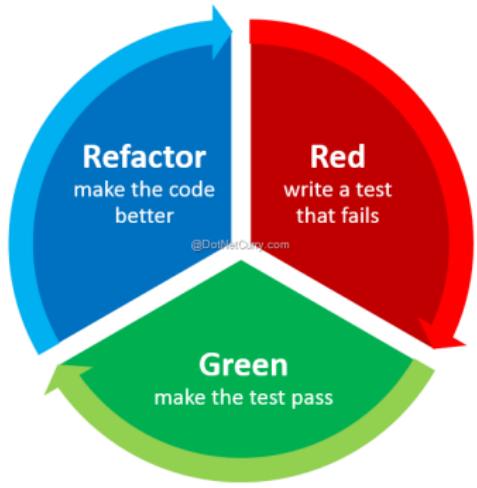


Constants and Variables Example

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



Assignment L02_02_helloLEDvar



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

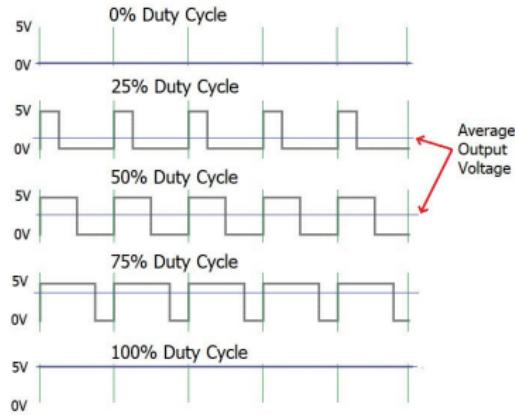
Note: refactoring code is when you rewrite portions of your code to make it more readable, or to make it run more efficiently.



Pulse Width Modulation

Software Configurable:

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





Assignment L02_03_helloLEDanalog



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

- 255
- 63
- 171
- 16

Measure the voltage with your multimeter at each value.

Syntax: `analogWrite(pin,value);`

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

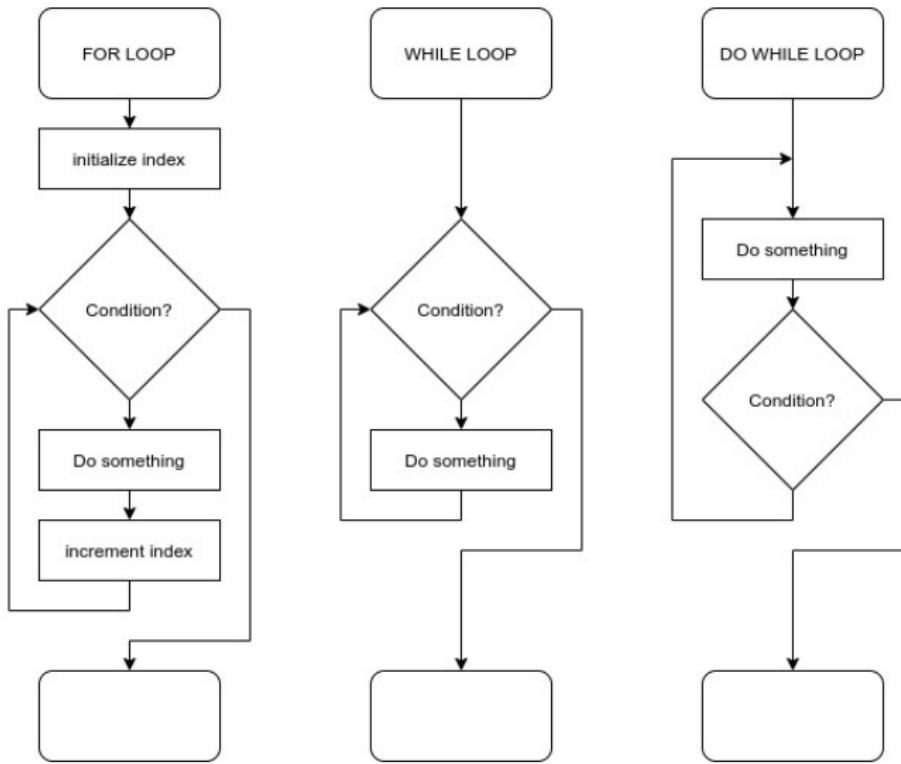


Flowcharts

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



Loops





FOR Loop syntax

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

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



WHILE loop syntax

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



For vs While Loops

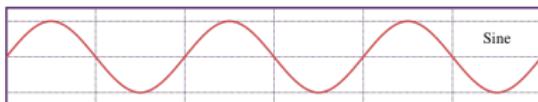
For VS While Loop

Comparison Chart

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



Assignment L02_04_helloLEDtri



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

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



Number Systems

Decimal

92₁₀

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

Binary

01011100₂

Digits: 0,1

Hexadecimal

5C₁₆

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

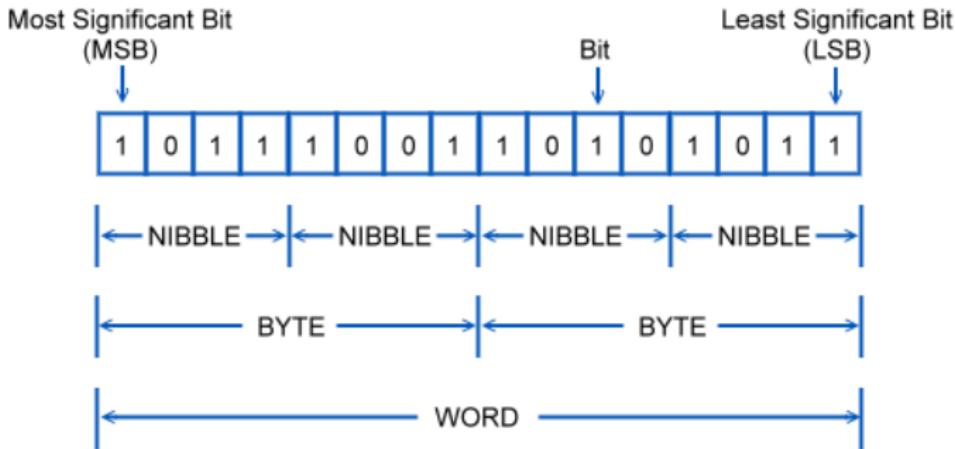
Octal

134₈

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



Bits, Nibbles, Bytes, and Words





Data Types: Numbers

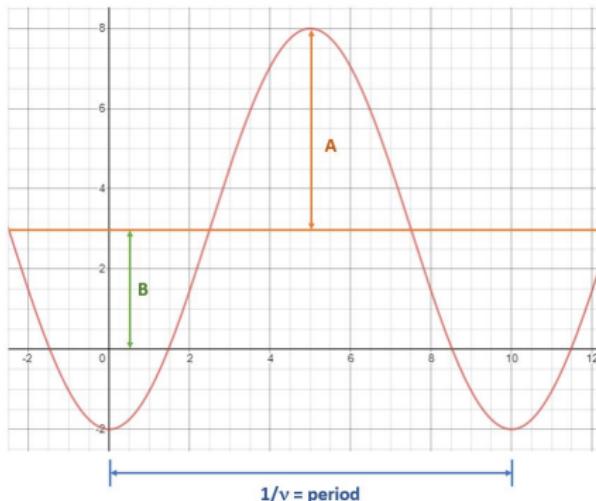
Data Type	8-bit AVR systems (Arduino Uno)			32-bit ARM systems (Teensy 3.2)		
	bytes	range (signed)	range (unsigned)	bytes	range (signed)	range (unsigned)
char	1	-128 to 127	0 to 255	1	-128 to 127	0 to 255
short	2	+/- 32,767	0 to 65,353	2	+/- 32,767	0 to 65,353
int	2	+/- 32,767	0 to 65,353	4	+/- 2,147,483,648	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 7.5×10^{18} grains of sand on Earth. A long long integer and the floating point numbers are larger than this.

MATH NOTE: An Integer divided by an Integer always returns an Integer



Sine Waves



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

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



Header Files

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

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

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

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

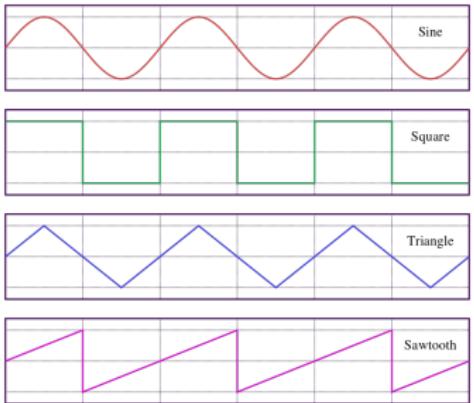


Basic Structure of Arduino Sketch Revisited

```
1 #include <math.h>          // include header files
2 const int LEDPIN = 5;      // declare constants
3 float value,n;            // declare variables
4
5 void setup() {             // runs once
6   pinMode(LEDPIN,OUTPUT);  // system settings
7   n = 0;                  // set variables
8 }
9
10 void loop() {              // loops indefinitely
11   value = sin(2*M_PI*n);
12   n = n+0.25;             // move a quarter around
13                           // the unit circle
14 }
```



Assignment L02_05_helloLEDsin



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

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

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

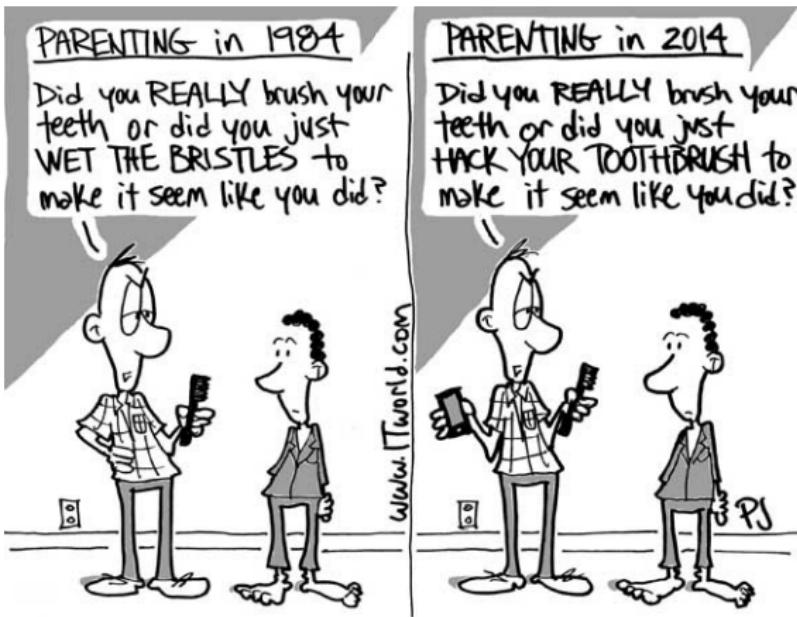
The function `millis()` returns milliseconds since the Argon has been powered on. For the sine equation, use $t = \text{millis()} / 1000.0$.

Why is adding the decimal after 1000 important?

Module 3 - Buttons



IoT Fun





Displaying to the Screen: The Serial Monitor

```
1 void setup() {  
2  
3     // Enable Serial Monitor  
4     Serial.begin (9600);  
5     waitFor(Serial.isConnected,10000); // wait for  
6     Serial monitor  
7     Serial.println ("Ready to Go");  
8 }  
9  
10 void loop() {  
11     for (i=0; i<=13; i++){  
12         Serial.print(i);  
13         delay(printDelay);  
14     }  
15 }
```



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

`printf("a = %d\nb = %d\n", a, b);`

specifier	Output	Example
d or i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7F8
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.90FEPE-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

```

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

```



Opening the Serial Monitor

Access the Command Palette - Ctrl-Shift-P

The screenshot shows the Particle IDE's Command Palette open. The search bar at the top contains the text '>particle'. Below the search bar is a list of recently used commands, each preceded by a blue 'Particle' icon. The commands listed are: 'Install Library', 'Find Libraries', 'Cloud Compile', 'Configure Workspace for Device', 'Launch CLI', 'Install Local Compiler', 'Cloud Flash', 'Serial Monitor', 'Create New Project', 'Audit Environment', 'Who Am I?', 'Clean application (local)', and 'Clean application & DeviceOS (local)'. A 'recently used' label is positioned to the right of the first few items, and an 'other commands' label is positioned to the right of the last item.

recently used
>particle
Particle: Install Library
Particle: Find Libraries
Particle: Cloud Compile
Particle: Configure Workspace for Device
Particle: Launch CLI
Particle: Install Local Compiler
Particle: Cloud Flash
Particle: Serial Monitor
Particle: Create New Project
Particle: Audit Environment
Particle: Who Am I?
Particle: Clean application (local)
Particle: Clean application & DeviceOS (local)

other commands

Or, open the Serial Monitor in Powershell or Terminal:

```
1 particle serial monitor --follow
```



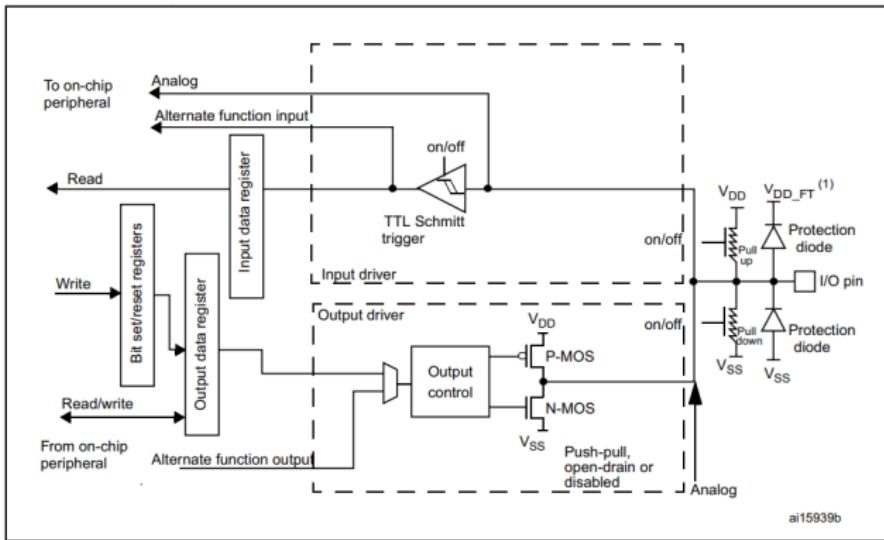
Assignment L03_00_SerialMonitor



- 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

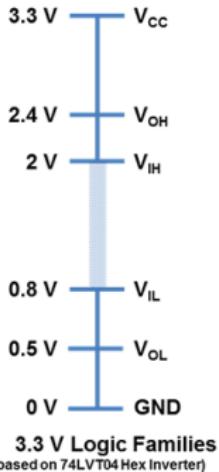


Software Programmable: Input or Output and Digital or Analog.



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.

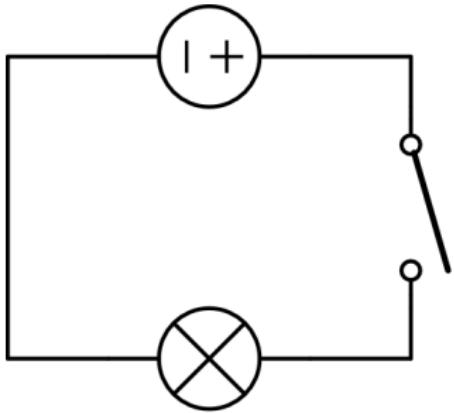


- `digitalWrite(pin,value);`
- `inputValue = digitalRead(pin);`

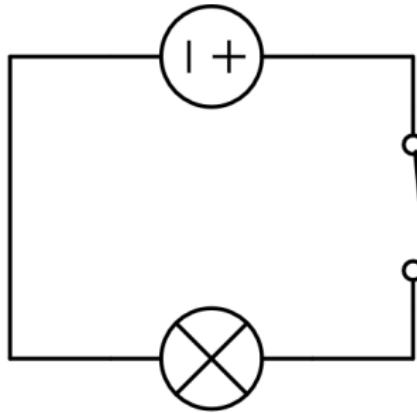
where, value equals HIGH or LOW.



Switches



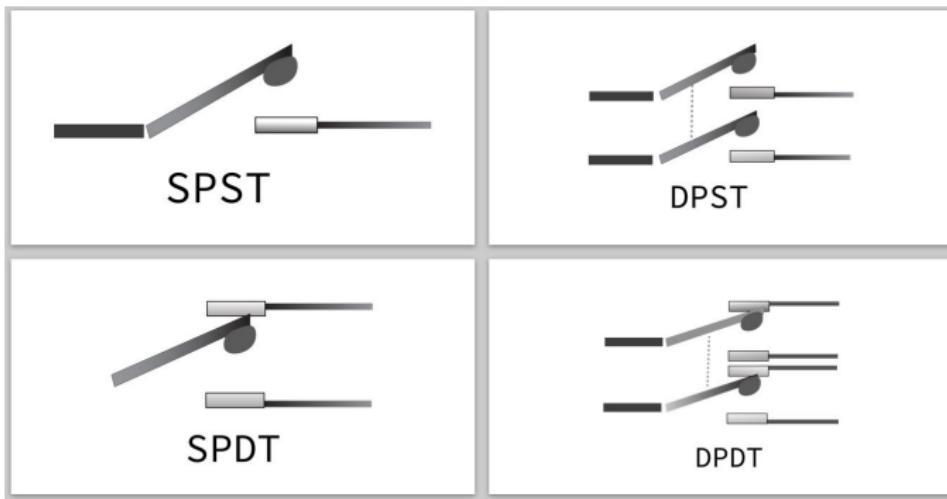
Lamp Off



Lamp On



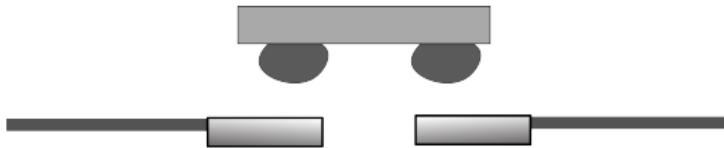
Poles and Throws



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



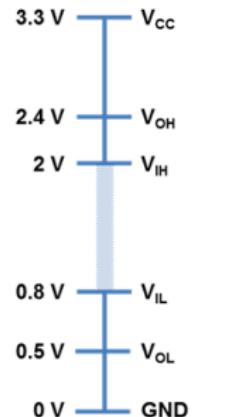
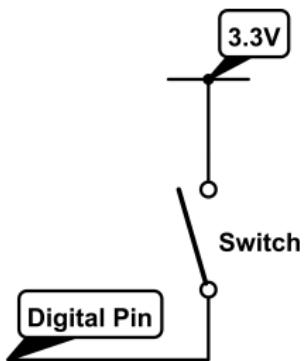
A Button - SPST



SPST
double break

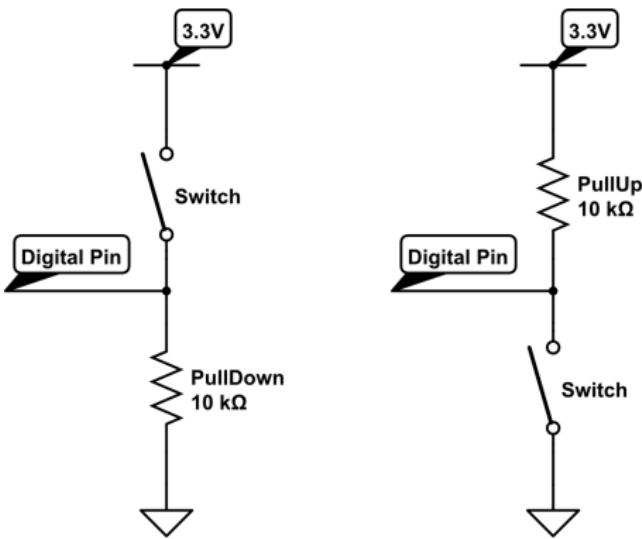


Floating Inputs





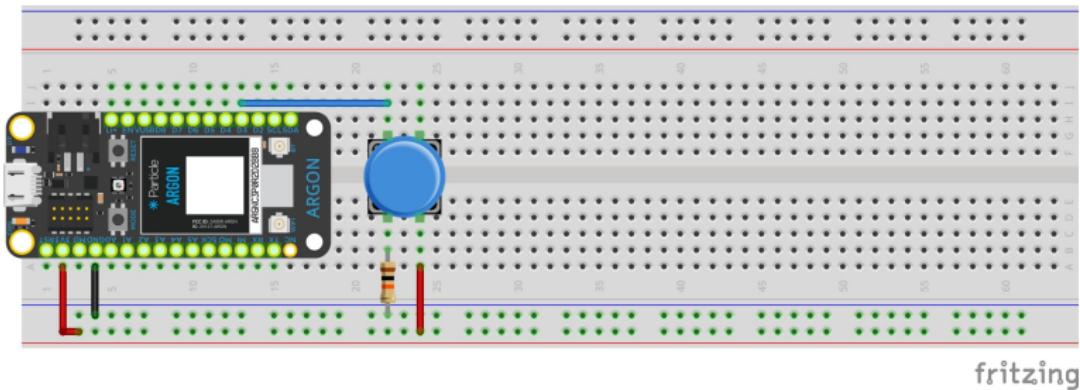
Pull Down and Pull Up Resistors



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



Our First Button and Pull Down Resistors

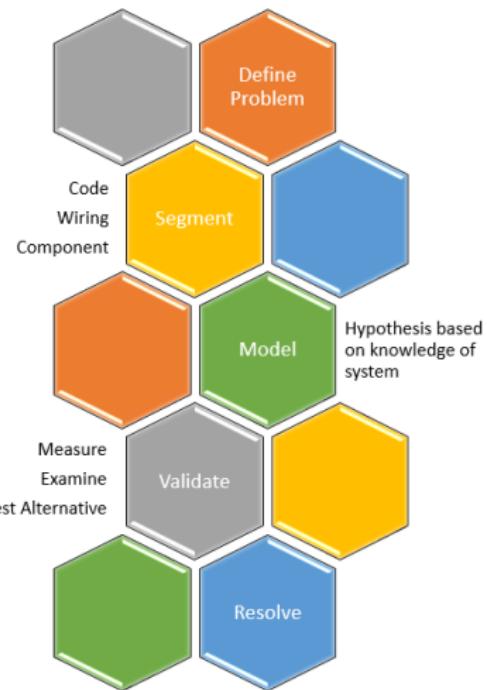


Reading from a digital pin:

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



Model Based Troubleshooting





Assignment: Buttons

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



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

① L03_01_button

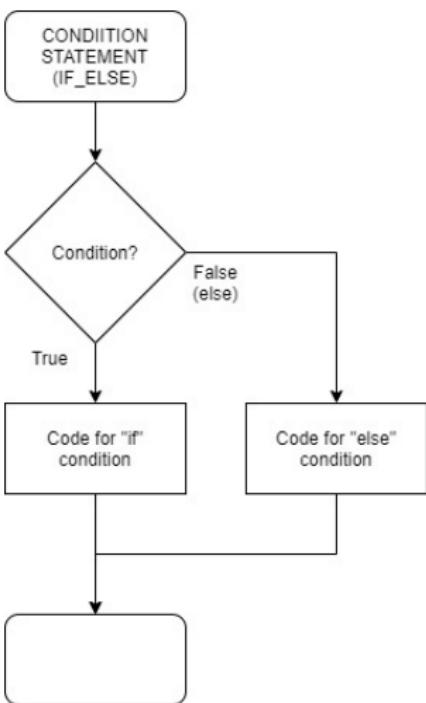
- Use digitalRead() to input the button state.
- Print button state to the screen.
- Remove the resistor. How does this affect the button state and the voltage?
- Replace the pull-down resistor with a pull-up. How does the logic change?
 - Not pressed: 3.3V
 - Pressed: GND

② L03_02_button_input_pullup

- Remove the pull-up resistor.
- Implement:
`pinMode(pin,INPUT_PULLUP);`



IF-ELSE Statements



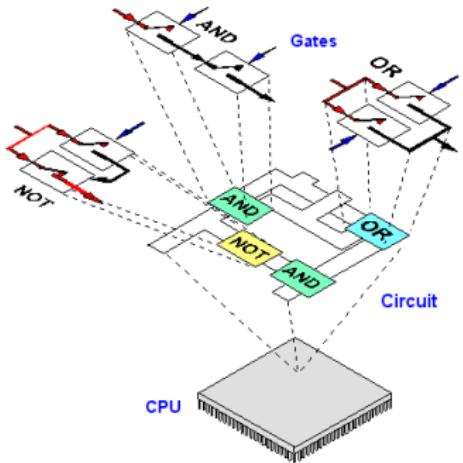


IF-ELSE Statements

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



Data Types: Boolean and Boolean Logic



Boolean datatype (bool)
holds either a TRUE or
FALSE

Boolean Logic Operations (condition statements)

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

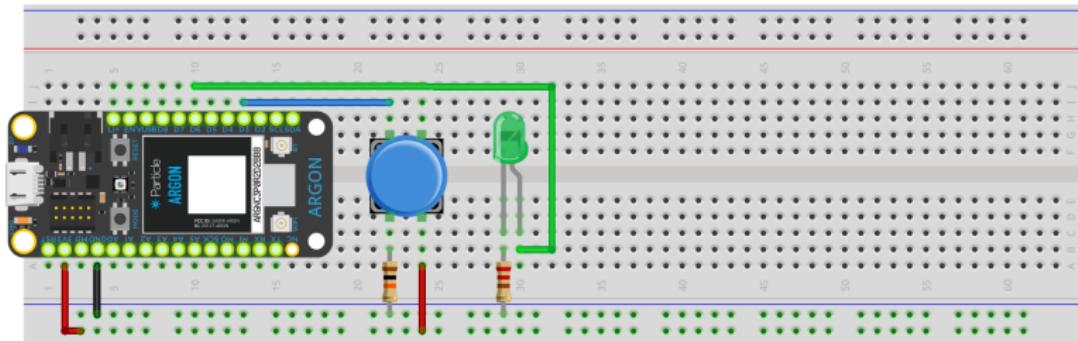


Boolean Logic In Action

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



Button and LED





Assignment: Buttons and LEDs



Update your
schematic and
Fritzing

① L03_03_buttonLED

- Add a Green LED to Pin 5 and use the button to turn the LED on/off.
- Also, print button state to the screen

② L03_04_twoButtonLED

- Add a second button (Pin 16) and Yellow LED (Pin 6).
- Have each button control one LED.
- Also, print button states to the screen.

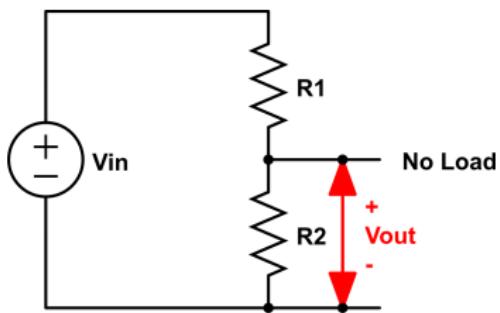
③ Extra Credit: Modify twoButton

- The Green LED lights up if both buttons are pressed
- The Yellow LED lights up if either button is pressed

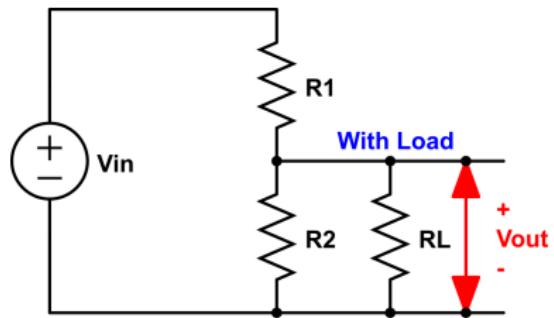


Resistors in Series and Parallel

Open Circuit Behavior



Behavior Under Load

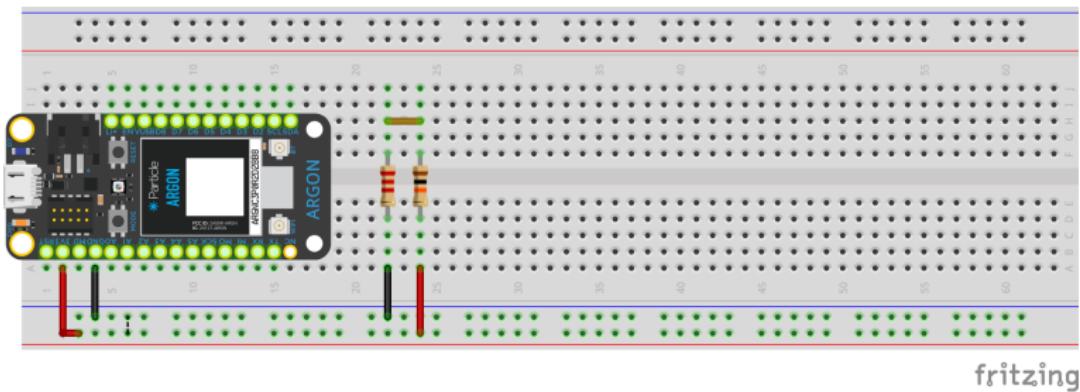


$$V_{out} = V_{in} \frac{IR_2}{I(R_1+R_2)} = \frac{R_2}{R_1+R_2} V_{in}$$

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



Voltage Dividing

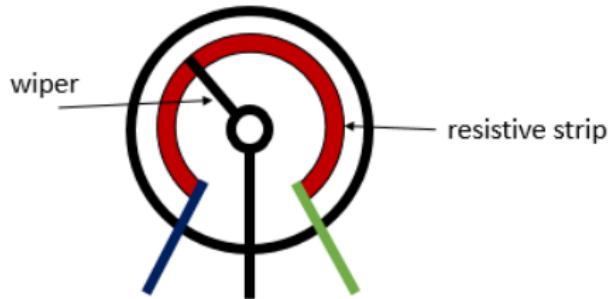


We are just using the Argon to provide Power and GND.

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



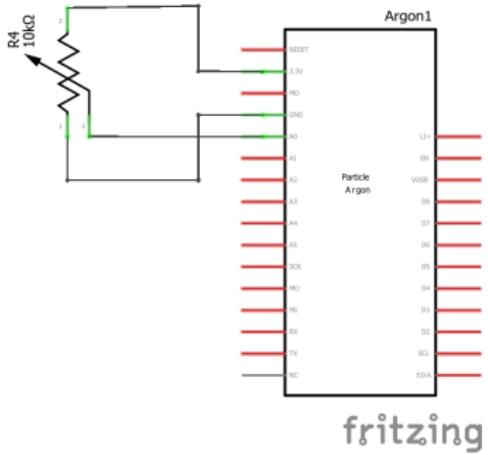
Potentiometer - Variable Resistor



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



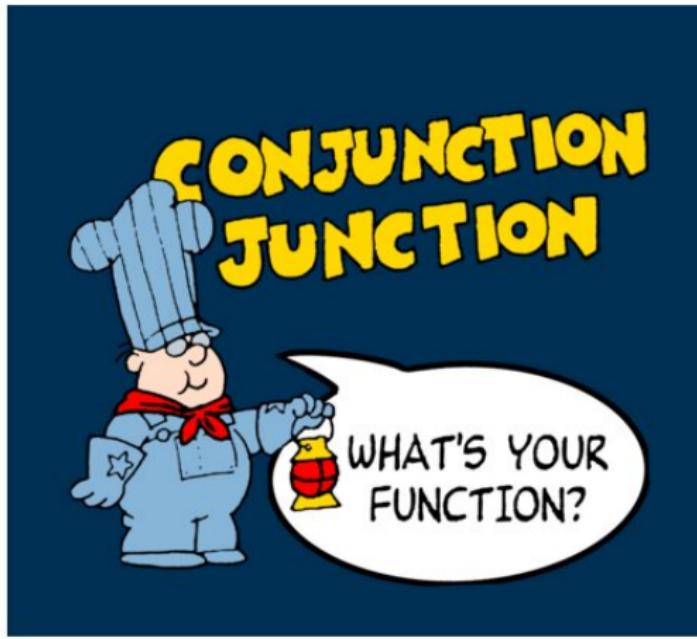
Assignment L03_05_AnalogInput



- ① Look up the syntax of `analogRead()` in the Arduino Reference.
- ② Utilize `analogRead()` to measure analog input across a potentiometer (voltage divider) using Pin 19.
- ③ Determine the range of the `analogRead()` across the entire range of the potentiometer.



Conjunction Junction





Anatomy of a Function

Anatomy of a C function

Datatype of data returned, any C datatype

"void" if nothing is returned

```
int myMultiplyFunction(int x, int y){  
    int result;  
    result = x * y;  
    return result;  
}
```

Function name

Parameters pass to function, any C datatype

Return statement, datatype matches declaration

Curly braces required



Functions in Action

```
1 int variable1, variable2, answer;
2
3 // Declare (or prototype) the function in the header
4 int myMultiplyFunction(int x, int y);
5
6 void setup() {
7     Serial.begin(9600);
8 }
9
10 void loop() {
11     // Call the function
12     answer = myMultiplyFunction(variable1, variable2);
13     Serial.printf("%i times %i = %i\n", variable1, variable2, answer);
14     variable2 = answer;
15 }
16
17 // Define the function
18 int myMultiplyFunction(int x, int y) {
19     int result;
20
21     result = x * y;
22     return result;
23 }
```

The function needs to be: Declared, Defined, and Called (at least once)



Types of Variables

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

① Global Variables

- Accessible throughout the program and all functions.

② Local Variables

- Accessible only in the function.
- Created when function is called. Destroyed when function is returned.

③ Static Local Variables

- Accessible only in the function.
- Maintains value across multiple calls of a function.

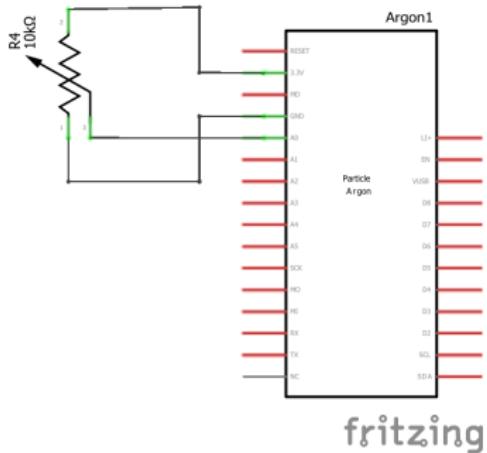


Basic Structure of Arduino Sketch with Functions

```
1  /*
2   * This is an example of how to use functions
3   * The code below converts inches to feet
4   */
5
6 const int INCHPIN=14;
7 int inches;
8 float feet;
9
10 void setup() {
11   Serial.begin(9600);    // Turn on Serial Monitor
12   while(!Serial);        // Wait for Serial Monitor to be running
13   pinMode(INCHPIN,INPUT);
14 }
15
16 void loop() {
17   inches = analogRead(INCHPIN);
18   feet = inchestoFeet(inches);
19   Serial.printf("%i inches equal %0.2f feet \n",inches , feet);
20   delay(1000);
21 }
22
23 float inchestoFeet(int measurement) {
24   float answer;           // declare answer as a local variable
25
26   answer = measurement / 12.0;
27   return answer;
28 }
```



Assignment L03_05_AnalogInput Revisited



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

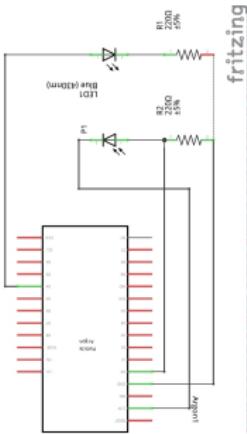


Photodiode

PARAMETERS	DIODE	PHOTODIODE
Definition	A diode is two terminal device which conducts when it is forwards biased.	A photodiode is a two terminal device which conducts when it is reversed biased.
Circuit symbol		
Main Function	Diode is mainly used as a switch.	Photodiode is used for conversion of light energy into electrical energy.
Material Used	Germanium or silicon, any of these two can be used.	Silicon is used for manufacturing photodiode. An anti-reflective layer of Silver Nitride is used for coating.
Applications	Used in clippers, clampers, rectifiers etc.	Used in optoelectronic device, camera, optocouplers etc.



Assignment: L03_06_HelloNightLight

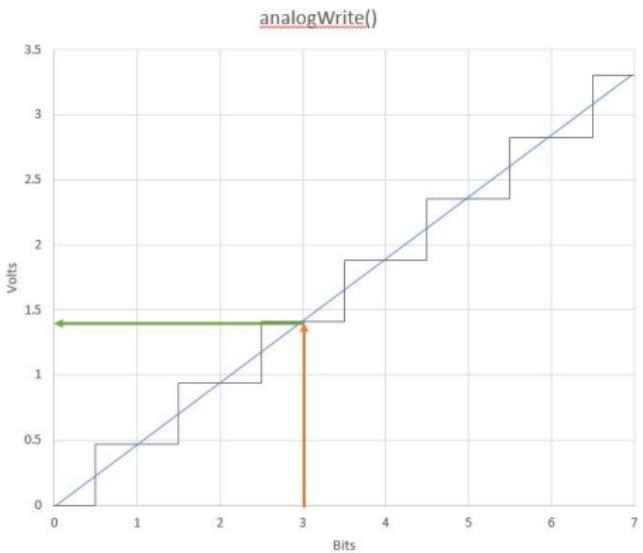
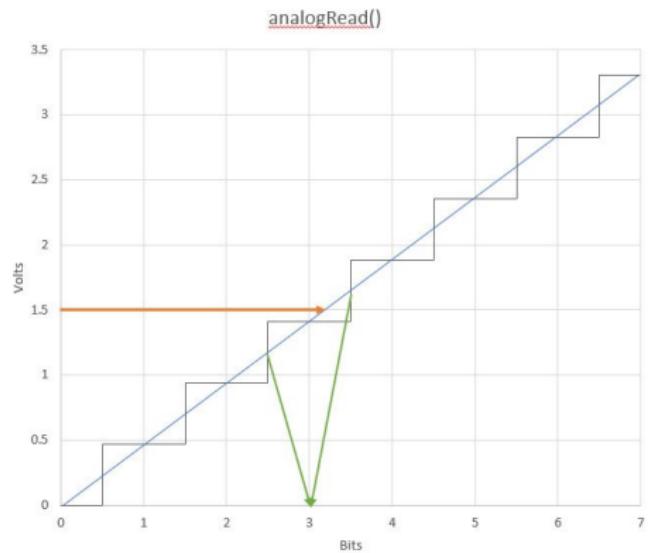


① L03_06_HelloNightLight

- Notebooks: schematic
- Fritzing diagram
- Wire your circuit
- Write the code
- 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 the photodiode is dark.
- Using analogWrite, turn on the LED slowly as the room darkens.

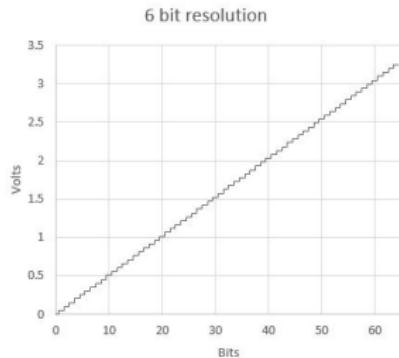
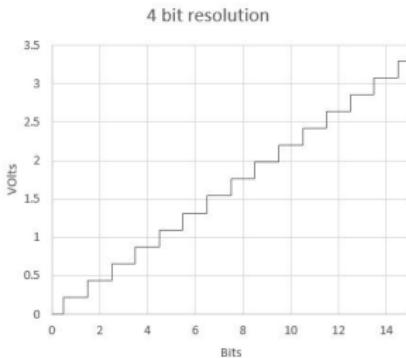
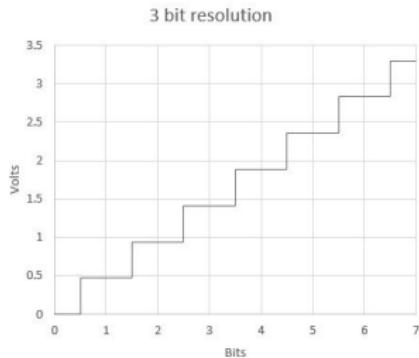


Analog Resolution - ADC





Analog Resolution - DAC

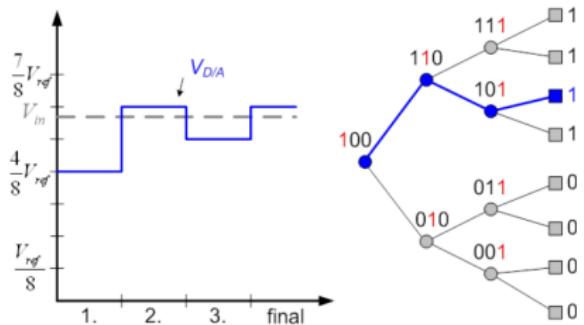
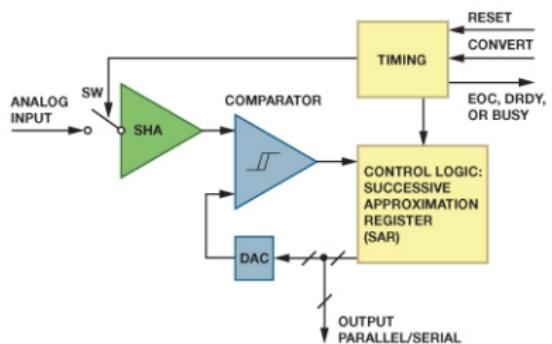


`analogWrite()` resolution can be modified between 2 and 31 bits.

```
1 analogWriteResolution(pin, bits);
```



Analog Resolution - ADC





Particle Publish

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

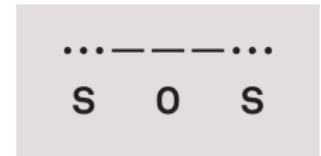
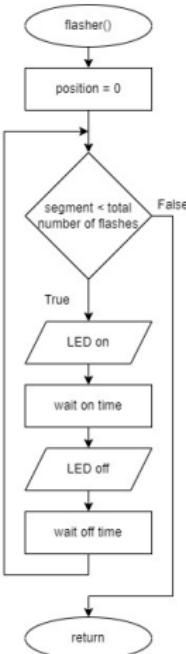
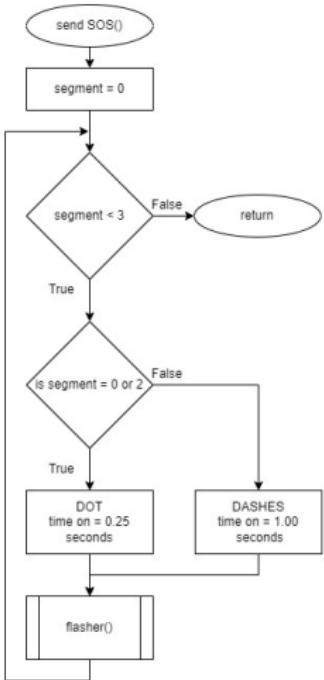
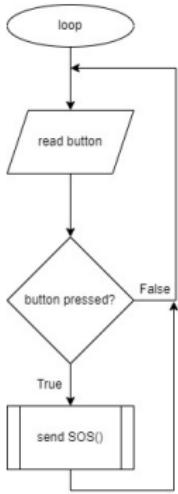
PARTICLE CLOUD			
EVENTS	VITALS	HEALTH CHECK	
		Search for events	ADVANCED
NAME	DATA	DEVICE	PUBLISHED AT
particle/device/update...	false	Lalonde	10/3/20 at 1:21:01 pm
spark/device/diagnos...	{"device": "network", "s...	Lalonde	10/3/20 at 1:21:00 pm
spark/device/app-hash	B665BEE9CD4A8E92IE3...	Lalonde	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/update...	false	Lalonde	10/3/20 at 1:20:58 pm
particle/device/update...	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

```
1 float temp, prs, hum;    // BME280 variables
2 String Temp, Prs, Hum;   // Strings to hold BME280 values
3
4 void loop() {
5     Temp = String(temp);
6     Prs = String(prs);
7     Hum = String(hum);
8     Particle.publish("Temperature", Temp, PRIVATE);
9     Particle.publish("Pressure", Prs, PRIVATE);
10    Particle.publish("Humidity", Hum, PRIVATE);
11 }
```

Modify L03_06_NightLight to publish the photodiode and LED values to the Particle Cloud once every 2 seconds.



Optional Week 1 Review: L03_00_SOS



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

Module 4 - NeoPixels

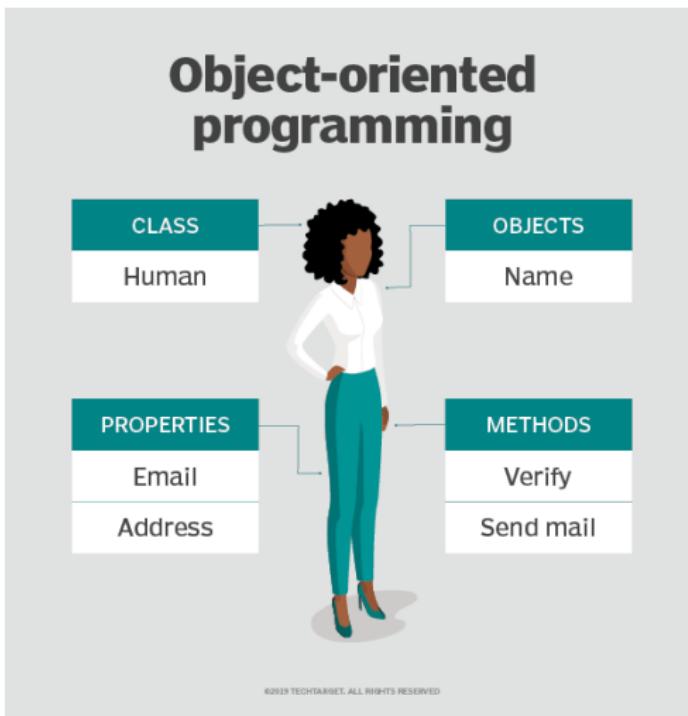


Smart Communities



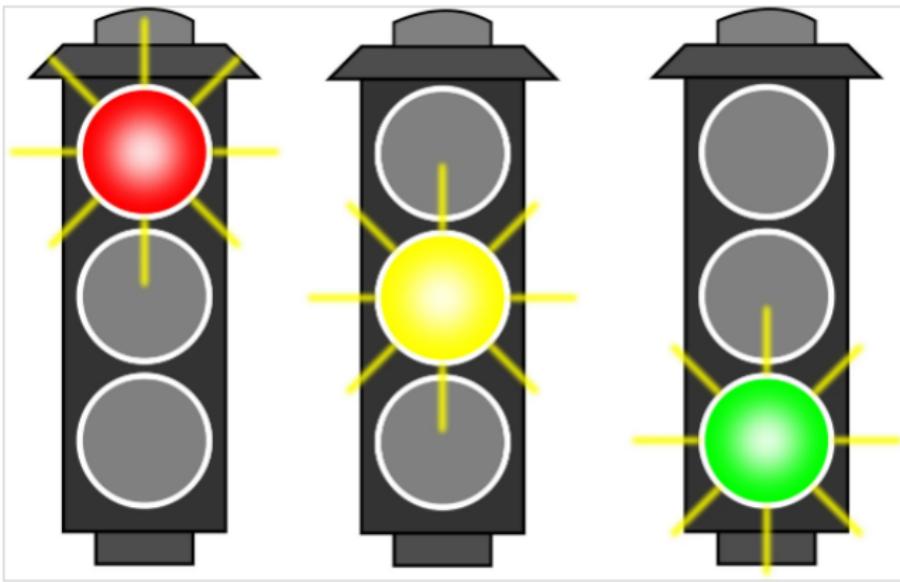


Objects





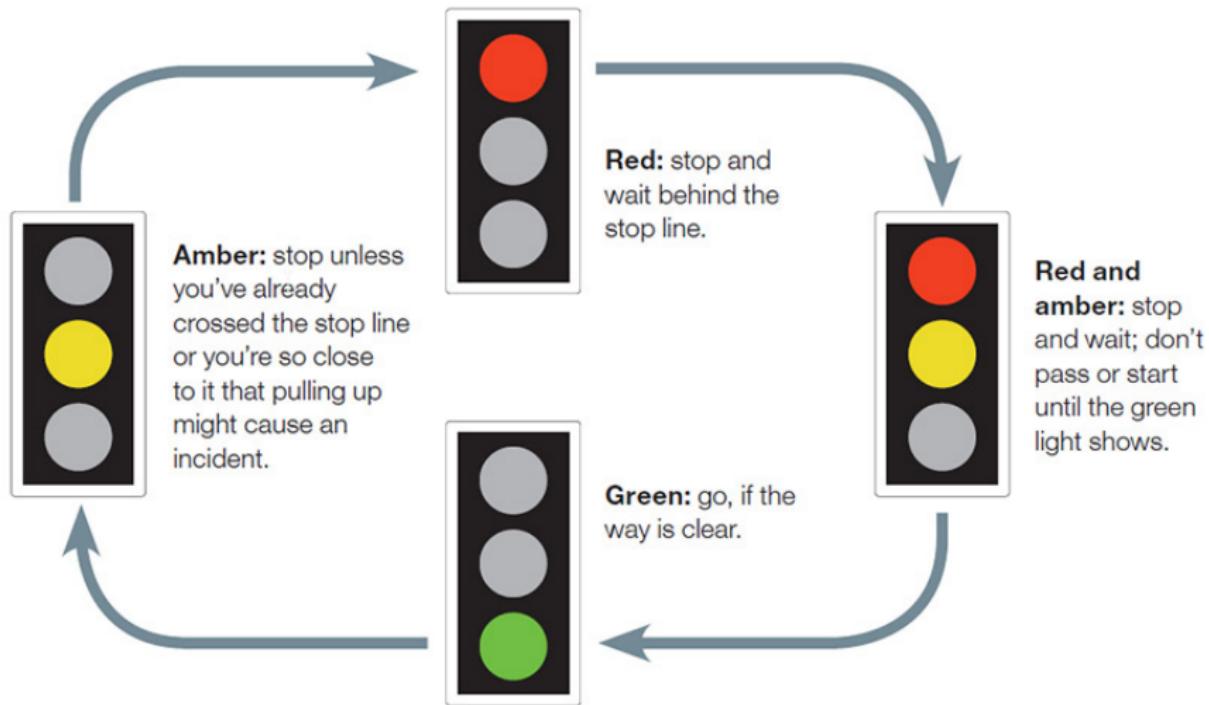
Traffic Light



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



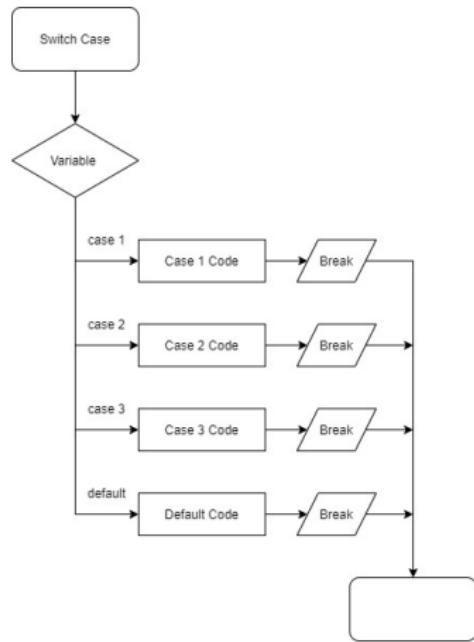
State Machine - Traffic Lights, British Style





SWITCH...CASE syntax - multiple IFs

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



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



Enumeration (enum)

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

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

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

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



Preprocessor commands

The C-compiler preprocessor executes the `#` commands before converting the program into code the Teensy can understand.

We have previously used the `#include` preprocessor command. Now, we will learn a few more:

```
1 #ifndef _BUTTON_H_    //if not defined, then execute the rest of code
2
3 #define _BUTTON_H_ //define the label
4
5 // Place your Header.h code here
6
7#endif // _BUTTON_H_
```

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



Soldering

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



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

So, we solder with solder!



Soldering

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





Solder Irons



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



Solder Tip Care



CLEANING YOUR TIPS

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

TINNING YOUR TIPS

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



OXIDIZED TIP



TINNED TIP

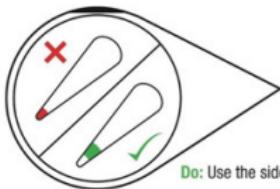


Properly Storing Tips

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



Good vs Bad Solder Joints



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

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



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



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



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



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



A

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



B

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



C

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



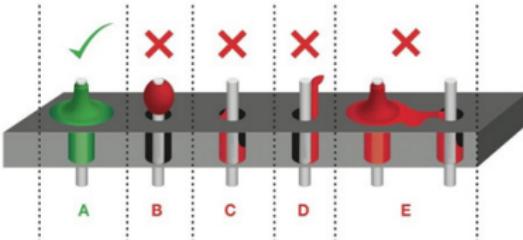
D

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



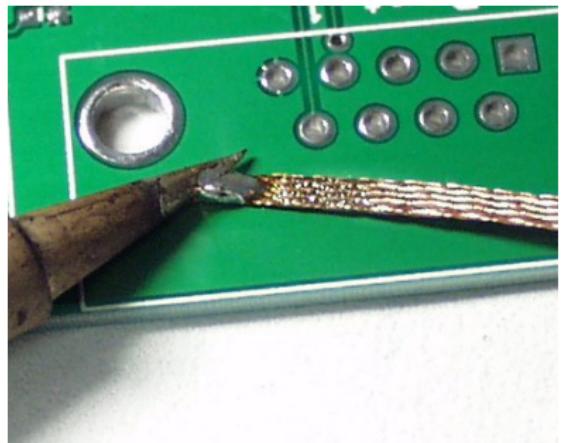
E

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

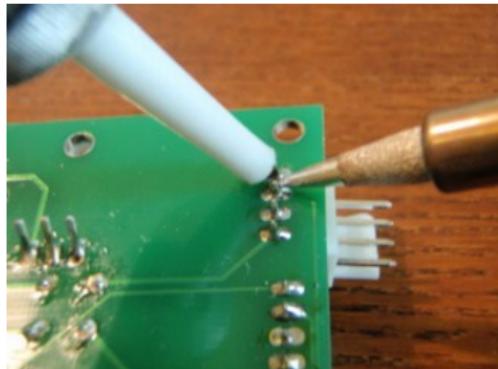




Desoldering



Solder Wick



Desoldering Pump



Generating Random Numbers

```
1  /*
2   * The random() function generates pseudo-random
3   * numbers.
4   *     random(min,max)
5   *     random(max)      //assumes min = 0
6   * returns a number between min and max-1
7   */
8
9 // print a random number from 0 to 299
10 randNumber = random(300);
11 Serial.printf("The number is = %i \n",randNumber);
12
13 // print a random number from 10 to 19
14 randNumber = random(10, 20);
15 Serial.printf("The number is = %i \n",randNumber);
```



Nested Statements



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

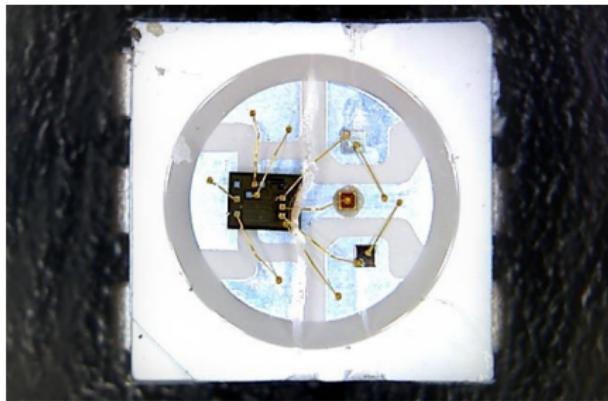


Nested For Loops

```
1 int tens;
2 int ones;
3
4 // Nested FOR Loops
5 for(tens=0;tens<10;tens++) {
6     for(ones=0;ones<10;ones++) {
7         Serial.printf("Combining %i(tens) and %i(ones) gives %i%i\n",tens,ones,tens,ones);
8     }
9 }
10
11 // An IF nested in Nested FOR Loops
12 for(tens=0;tens<10;tens++) {
13     for(ones=0;ones<10;ones++) {
14         if(tens != ones) {
15             Serial.printf("Combining %i(tens) and %i(ones) gives %i%i\n",tens,ones,tens,ones);
16         }
17     }
18 }
19
20 // Using break to break out of a loop
21 for(tens=0;tens<10;tens++) {
22     for(ones=0;ones<10;ones++) {
23         Serial.printf("Combining %i(tens) and %i(ones) gives %i%i\n",tens,ones,tens,ones);
24         if((tens+ones) = 10) {
25             break;
26         }
27     }
28 }
```



NeoPixels

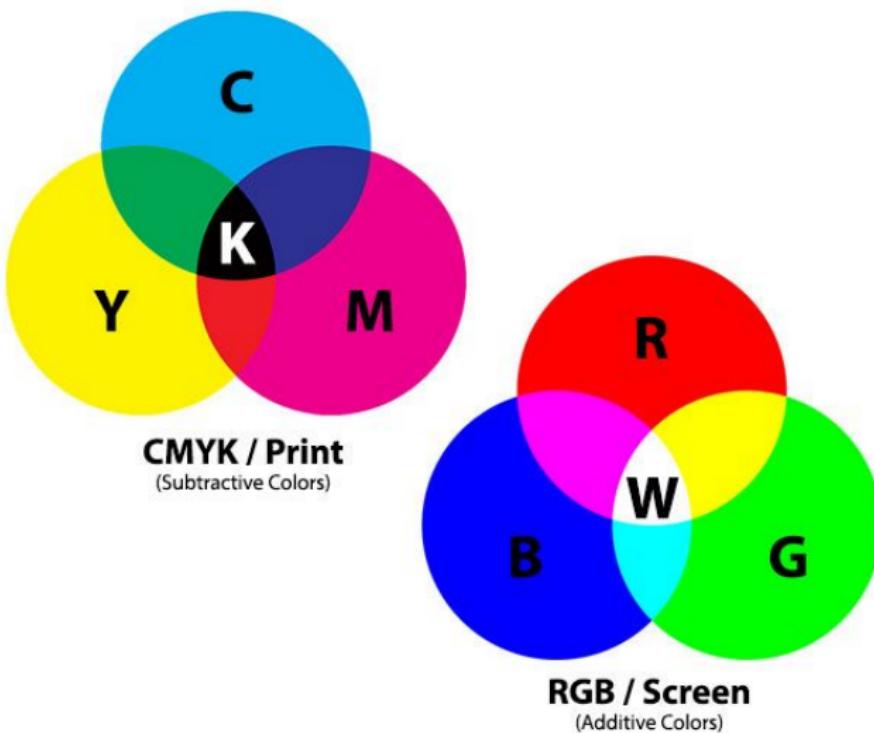


NeoPixels are:

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

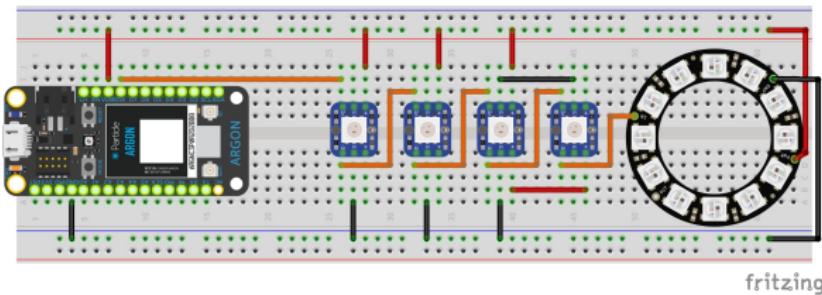


CYMK vs RGB Colors

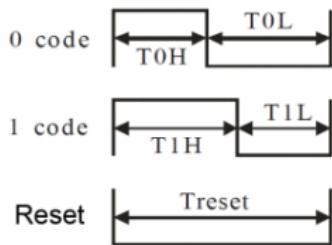




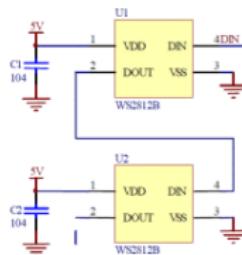
NeoPixel Programming



WS2812 Protocol



LED-Chain



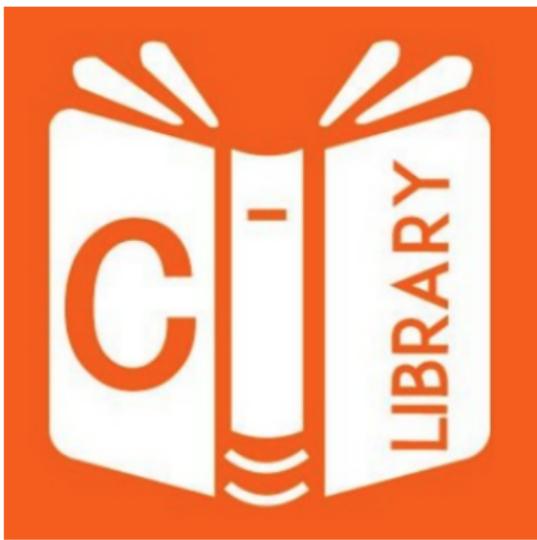


Using NeoPixel Class and Methods

```
1 #include <neopixel.h>
2
3 const int PIXELPIN = D8;      // Pin the NeoPixels are connected to
4 const int PIXELCOUNT = 16;    // Total number of NeoPixels
5
6 Adafruit_NeoPixel pixel(PIXELCOUNT, PIXELPIN, WS2812B); //declare object
7 /* Argument 1 = Number of pixels
8 * Argument 2 = GPIO pin number
9 * Argument 3 = Pixel type flags, add together:
10 * Use:
11 *   WS2811        // 400 KHz datastream (NeoPixel)
12 *   WS2812        // 800 KHz datastream (NeoPixel)
13 *   WS2812B       // 800 KHz datastream (NeoPixel)
14 *   WS2813        // 800 KHz datastream (NeoPixel)
15 *   WS2812B2      // 800 KHz datastream (NeoPixel)
16 *   SK6812RGBW    // 800 KHz datastream (NeoPixel RGBW)
17 */
18 void setup() {
19   pixel.begin();
20   pixel.show(); //initialize all off
21 }
22
23 void loop() {
24   pixel.setPixelColor(n, red, green, blue); // n is the pixel number being set
25   pixel.setPixelColor(n, color);           // red,green,blue = 0 - 255
26   pixel.setBrightness(bri);              // hex code 0x000000 - 0xFFFFFFFF
27   pixel.show();                         // 0 - 255
28   pixel.show();                         // nothing changes until show()
29   pixel.clear();                       // even clear() needs a show()
30 }
```



Installing Libraries

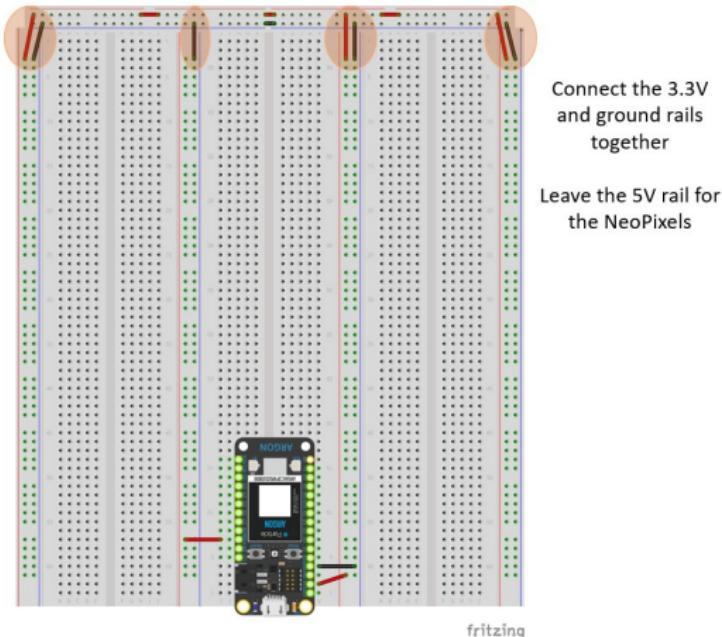


Using the Command Palette within VSCode:

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

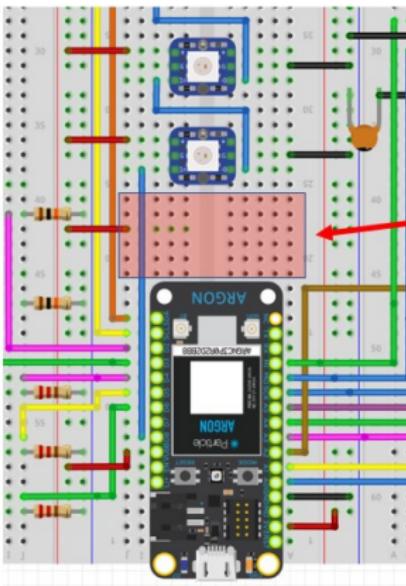


Move to Big Breadboard





Leave Space for Future Components



Leave Open
for Future
Components



Assignment: NeoPixels



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

① L04_01_neoPixel

- Using FOR loop and individual R/G/B, light up 16 pixels; small delay between.

② L04_02_colorHeader

- Implement a header file that contains the pixel colors.

③ L04_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 L04_04_pixelFill

- Light up 7 segments of different colors using the fill() method.



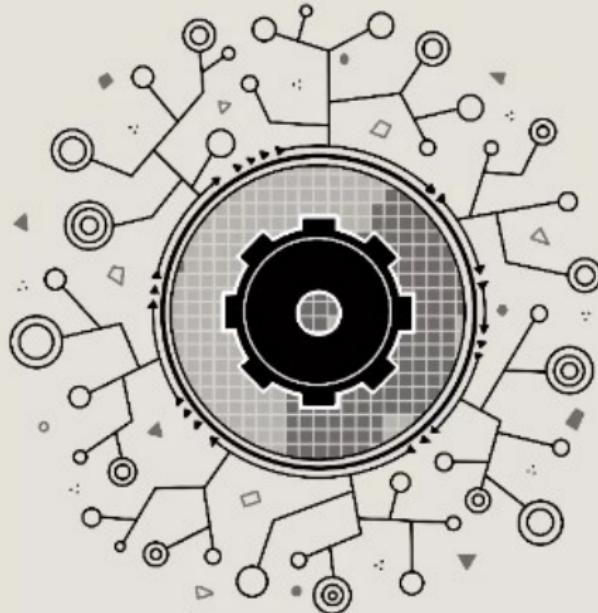
RandomSeed()

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

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



Algorithms



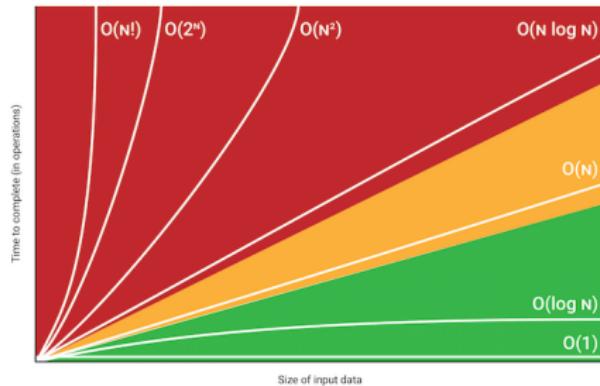
Algorithm

[*'al-gə-ri-thəm*]

A set of instructions for solving a problem or accomplishing a task.



Big O Notation



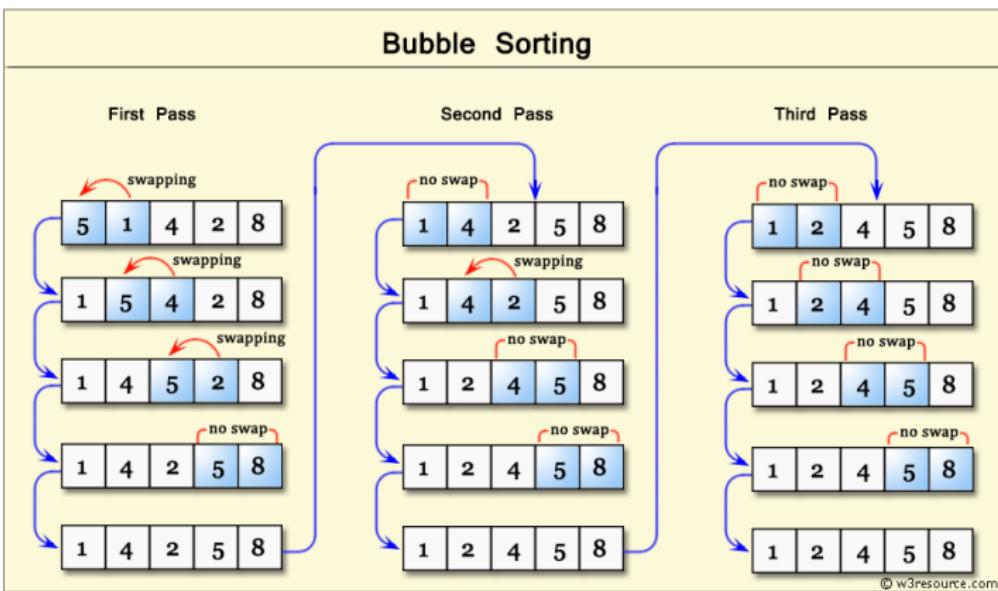
Big O notation is a way to describe the speed or complexity of a given algorithm.

- Big O notation tells the number of operations an algorithm will make.
- It gets its name from the literal "Big O" in front of the estimated number of operations.
- Big O establishes a worst-case run time



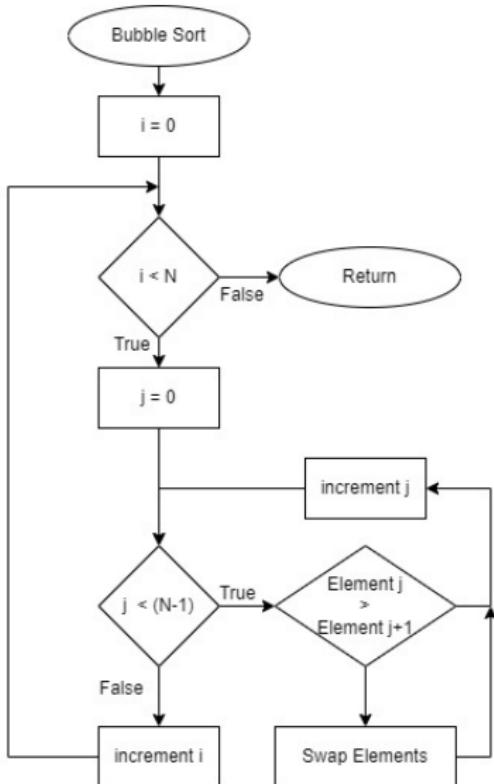
Bubble Sort

Bubble Sorting





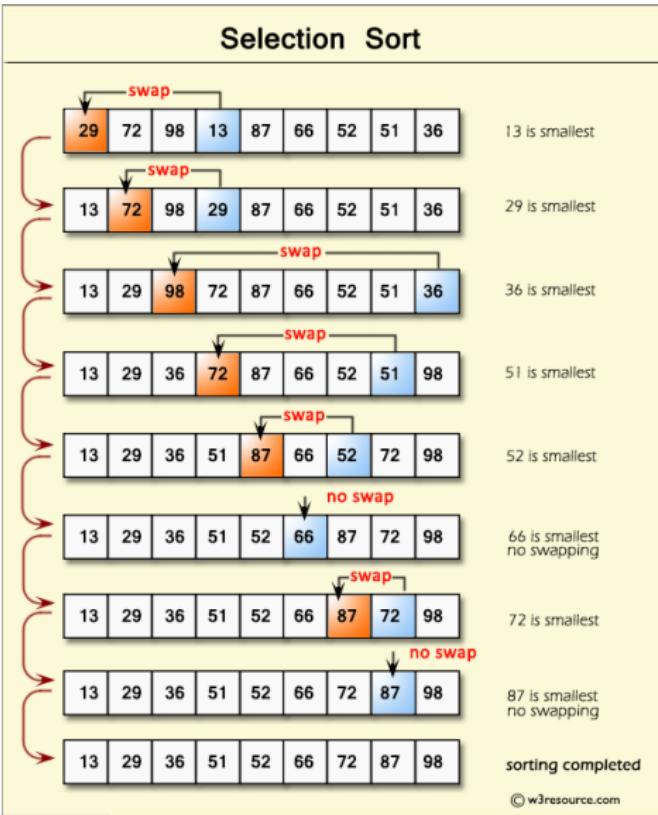
Bubble Sort Algorithm - $O(n^2)$



- Larger elements are swapped (one-by-one) with smaller ones (i.e., they float to the top)
- Takes $O(n^2)$ iterations as there is a $(N - 1)$ loop nested in a N loop.

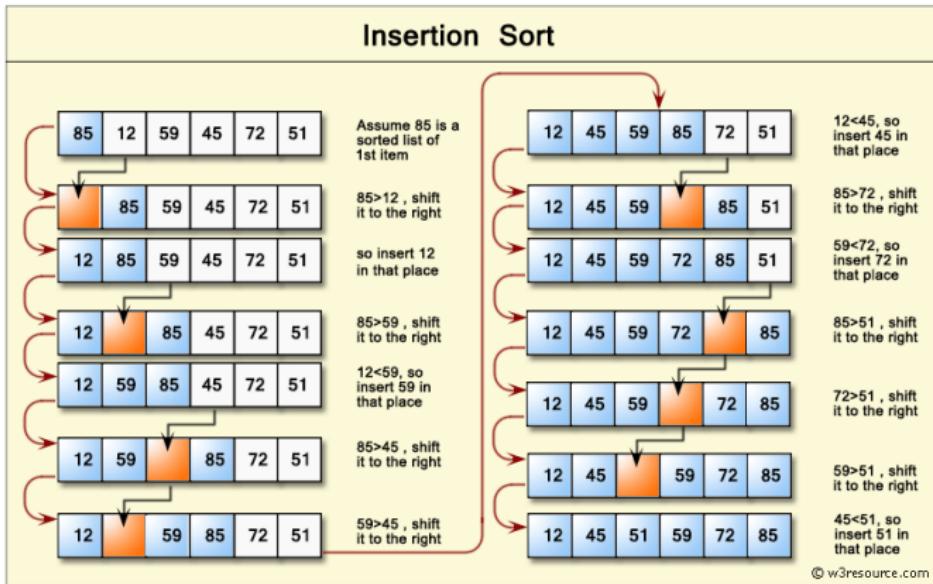


Selection Sort





Insertion Sort

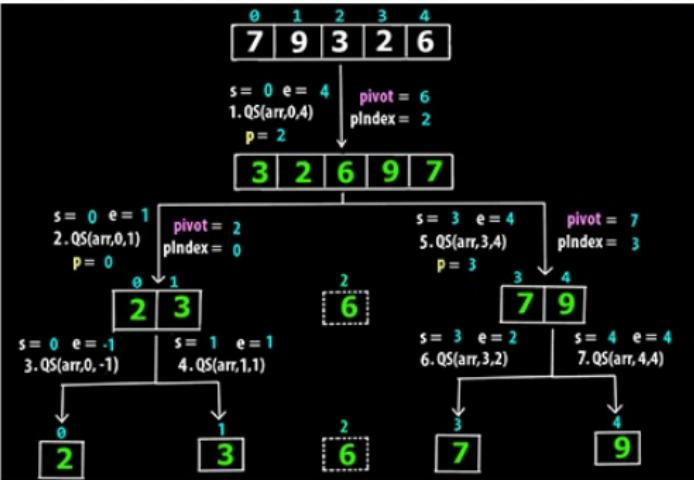




Quick Sort

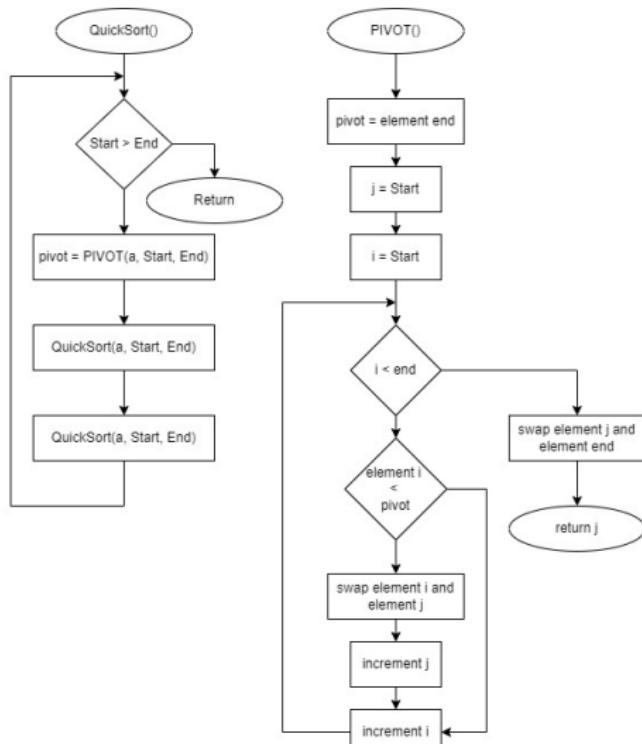
QUICK SORT ALGORITHM

```
QuickSort(arr[], s, e)
{
    if(s < e)
    {
        p = Partition(arr[],s,e)
        QuickSort(arr[], s, (p-1))
        QuickSort(arr[], (p+1), e)
    }
}
```





Quick Sort Algorithm - $O(n \log(n))$



- Any pivot point is selected (in this case, the last element)
- Smaller elements are put to the left of the pivot, larger to the right.
- Each side of the pivot is then sorted using the same method. This is called recursion.
- This continues until everything is sorted.
- In general, this takes $O(n \log(n))$ iterations



Quick Sort Algorithm - $O(n \log(n))$

```
1 int partition (int arr[], int low, int high) {
2     int pivot = arr[high];
3     int i = (low - 1);
4     int swap;
5
6     for (int j = low; j <= high- 1; j++)      {
7         if (arr[j] <= pivot)                  {
8             i++;
9             swap = arr[i];
10            arr[i] = arr[j];
11            arr[j] = swap;
12        }
13    }
14    swap(&arr[i + 1], &arr[high]);
15    return (i + 1);
16 }
17
18 void quickSort(int arr[], int low, int high) {
19     if (low < high)      {
20         int pi = partition(arr, low, high);
21
22         quickSort(arr, low, pi - 1);
23         quickSort(arr, pi + 1, high);
24     }
25 }
```



Assignment: L04_05_sorting



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

- Randomly fill an array of length equal to number of neopixels with colors from blue to red.
- Implement a function for the bubble sort that displays the change in the array after each iteration.
- Implement either the selection or insertion sort on a new random array
- (Optional) implement the quicksort

Module 5 - Encoders



IoT Humor

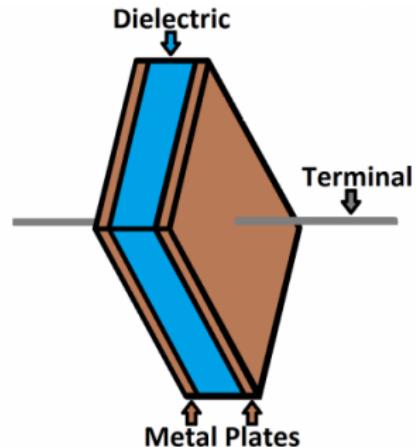




Capacitors

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

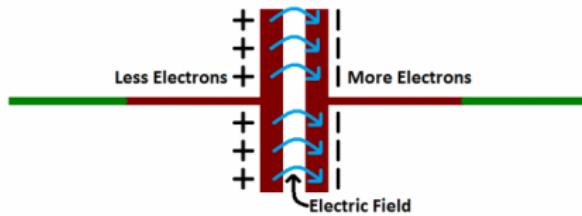
- The dielectric can be made out of all sorts of insulating materials; paper, glass, rubber, ceramic, plastic, or anything that will impede the flow of current.
- The plates are made of a conductive material; aluminum, tantalum, silver, or other metals.





Capacitors

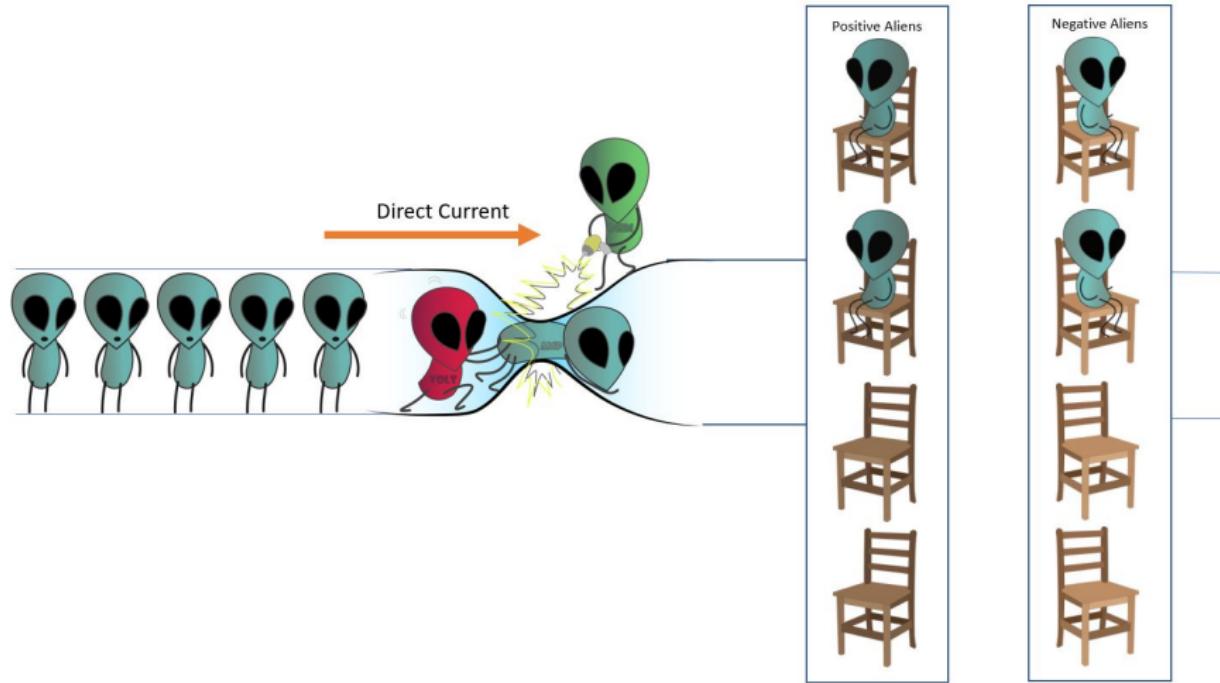
When current flows into a capacitor, the charges get "stuck" on the plates because they cannot get past the insulating dielectric. Electrons build up on one of the plates, and it becomes overall negatively charged. The large amount of negative charges pushes away like charges on the other plate, making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



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



Capacitors in Circuits

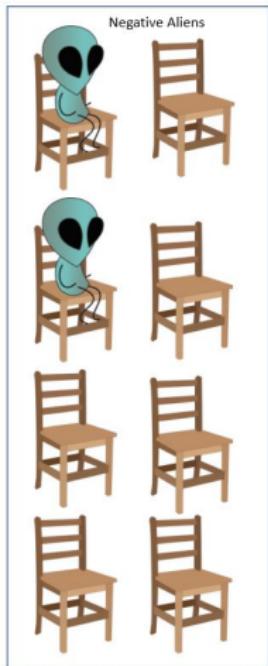
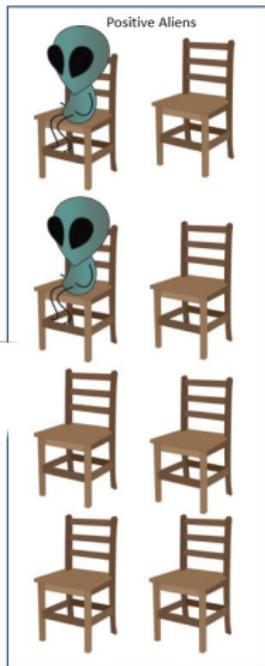
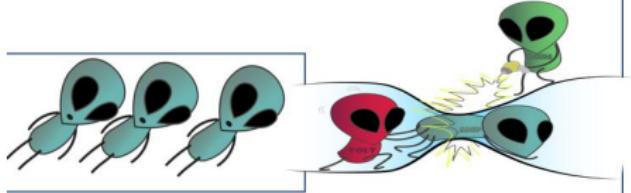




Capacitors in Circuits

Larger values of R and C

Direct Current





RC Time Constant

Capacitance is defined as:

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

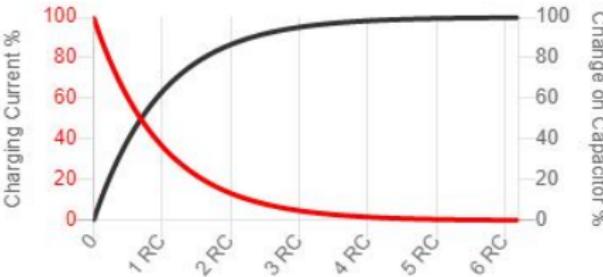
The current through a capacitor is:

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

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

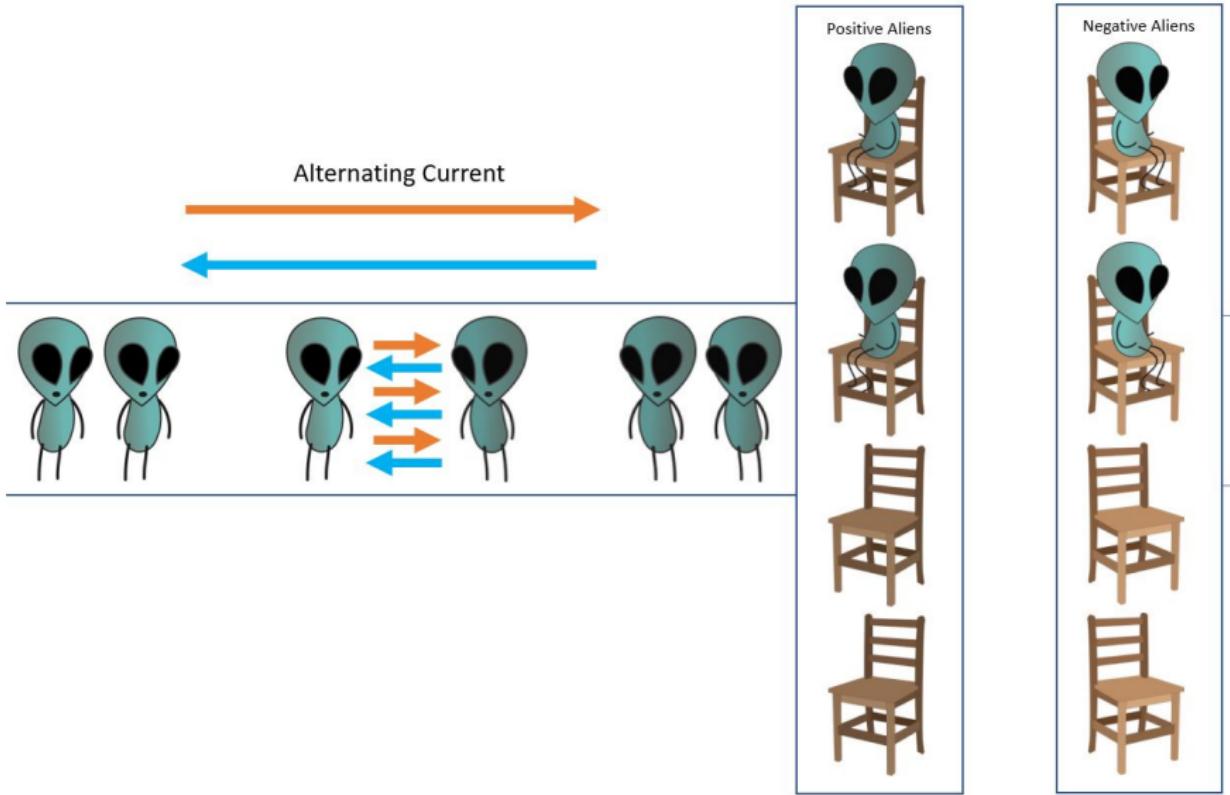
$$\tau = RC$$

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





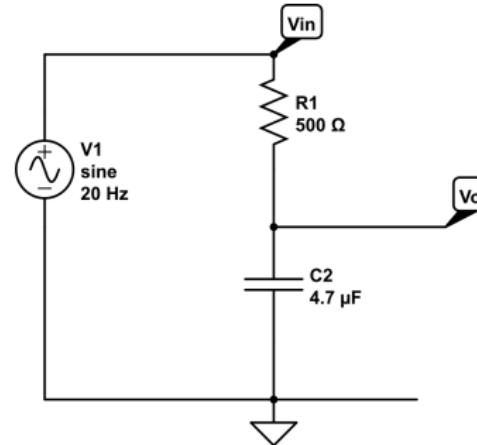
Alternating Current





Low Pass Filter - cutoff frequency f_c

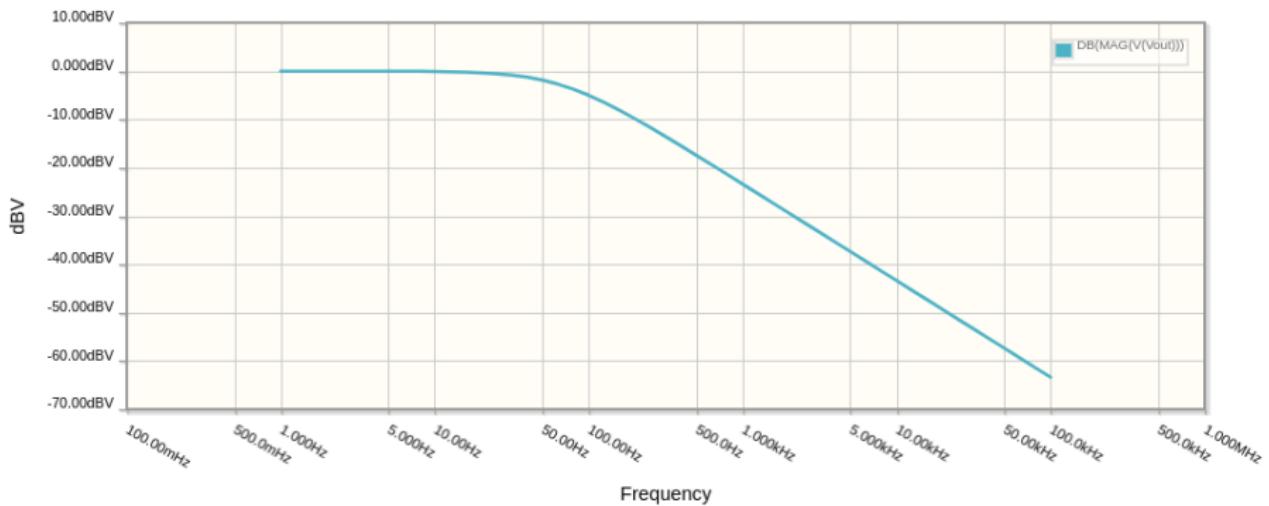
- At low frequencies, there is plenty of time for the capacitor to charge up to practically the same voltage as the input voltage.
- At high frequencies, the capacitor only has time to charge up a small amount before the input switches direction. The output goes up and down only a small fraction of the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



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



Low Pass Filter Response



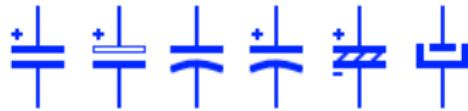
$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi(500)(4.7 \times 10^{-6})} = 67.5678 \text{ Hz}$$



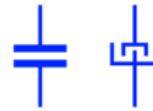
Capacitors - does it matter how they are placed

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

Polarized Electrolytic Capacitor

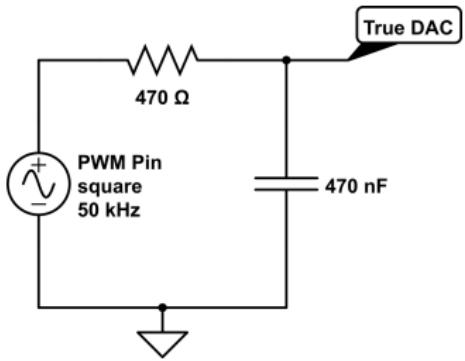


Generic Capacitor





True DAC on Argon



`analogWrite(pin, value, frequency)`

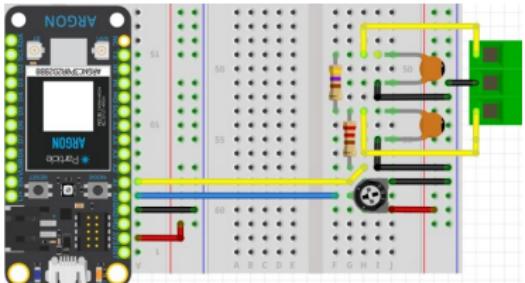
The Particle microcontrollers do not have a true DAC (Digital to Analog Converter). However, we can convert a PWM pin to a DAC signal using a low pass filter.

- The PWM signal oscillates at 500 Hz, but we can increase up to 50,000 Hz using a third parameter in `analogWrite()`.
- Using $R = 470\Omega$ and $C = 0.47\mu F$ will give a $f_c = 720\text{Hz}$.



Assignment: Low Pass Filters

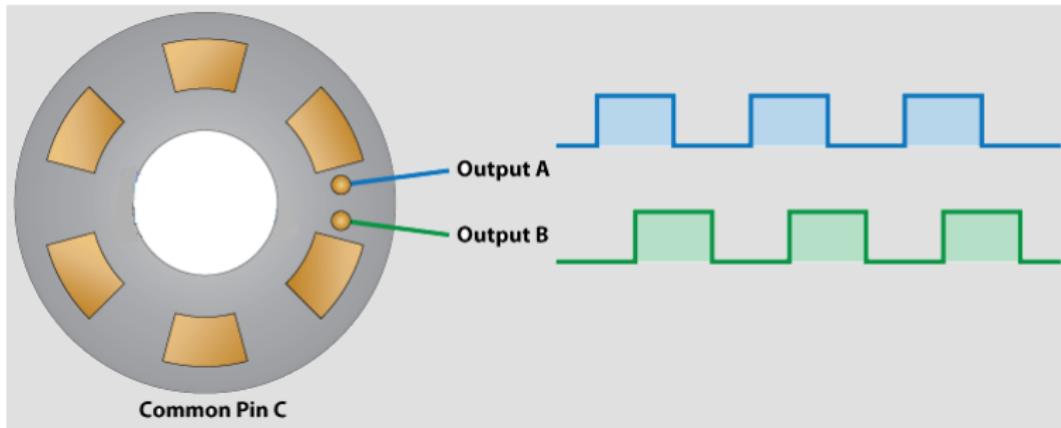
① L05_00_lowPass



- Notebook:
schematic
- Fritzing diagram
- Wire your circuit
- Write the code
- Output 0.825V to pin A5 (no wire will be connected to this output).
- Connect a potentiometer to A0. Use it to vary a frequency (ν) from 0 to 500 Hz.
- Create a sine wave of frequency ν :
 $\sin(2\pi\nu t)$ and output on the A1 pin. Set the PWM frequency to 50,000 Hz.
- Create a DAC to convert the PWM signal on A1 to analog using a low pass filter with $f_c \approx 700\text{Hz}$
- Create a low pass filter with $f_c \approx 67\text{Hz}$
- Use an oscilloscope to measure the A5, A1, post DAC, and post low-pass filter.
- Record in your notebook how the signals change when ν is varied with the potentiometer.

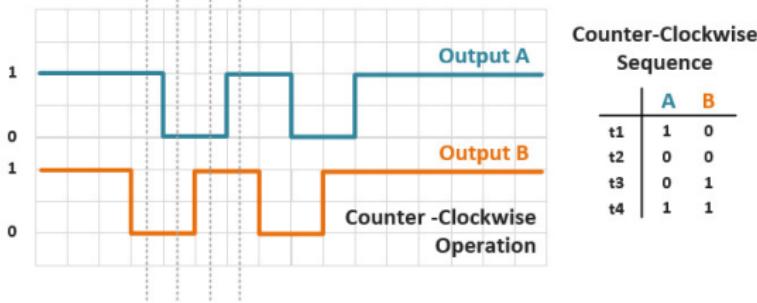
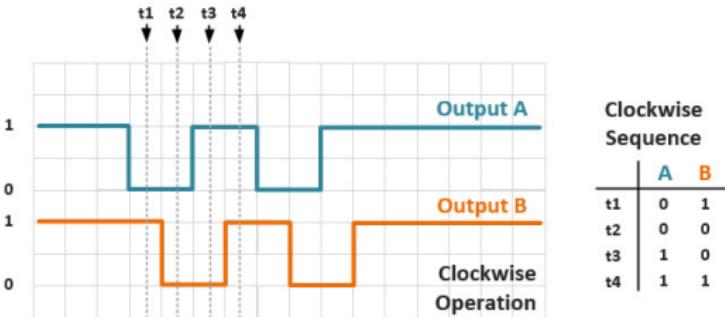


Encoders





Encoders



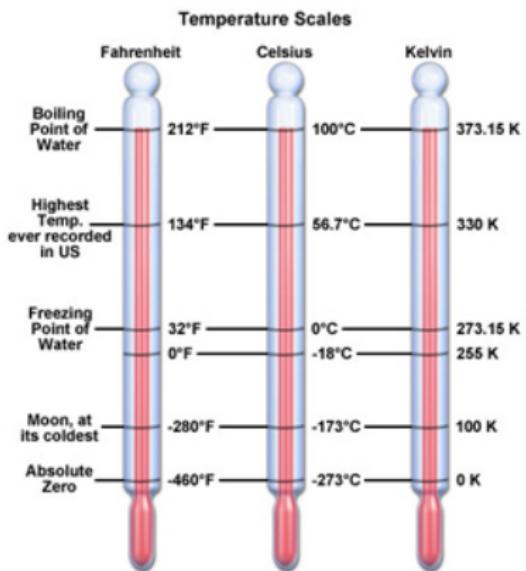


The Encoder Class

```
1 #include <Encoder.h>
2 Encoder myEnc(PINA, PINB);
3 // The "c" pin on the encoder is connected to GND
4
5 void setup() {
6 }
7
8 void loop() {
9     // read encoder position
10    position = myEnc.read();
11
12    // set encoder to a position
13    myEnc.write(maxPos);
14 }
```



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 L05_01_encoder

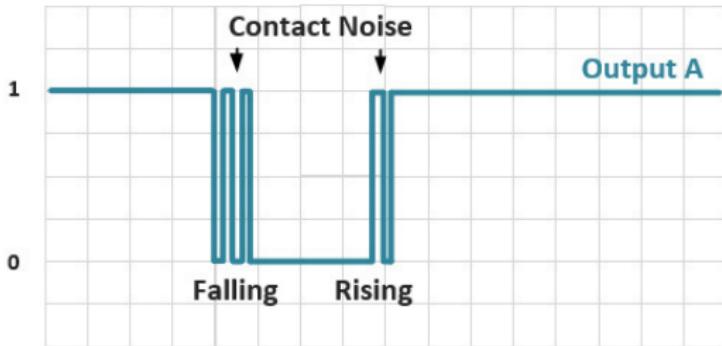
- Display the encoder position to the screen, but only when encoder is moved.
- What do you notice about the encoder position as you turn it slowly?

② L05_02_NeoPixel

- The encoder has 96 positions. Map the encoder input to 16 NeoPixels.
- Bound the input, so the mapped range is from 0 to 15.
- What is the difference in performance if you bind the input vs the mapped range?
- Use the encoder to light up the four NeoPixels and the ring.

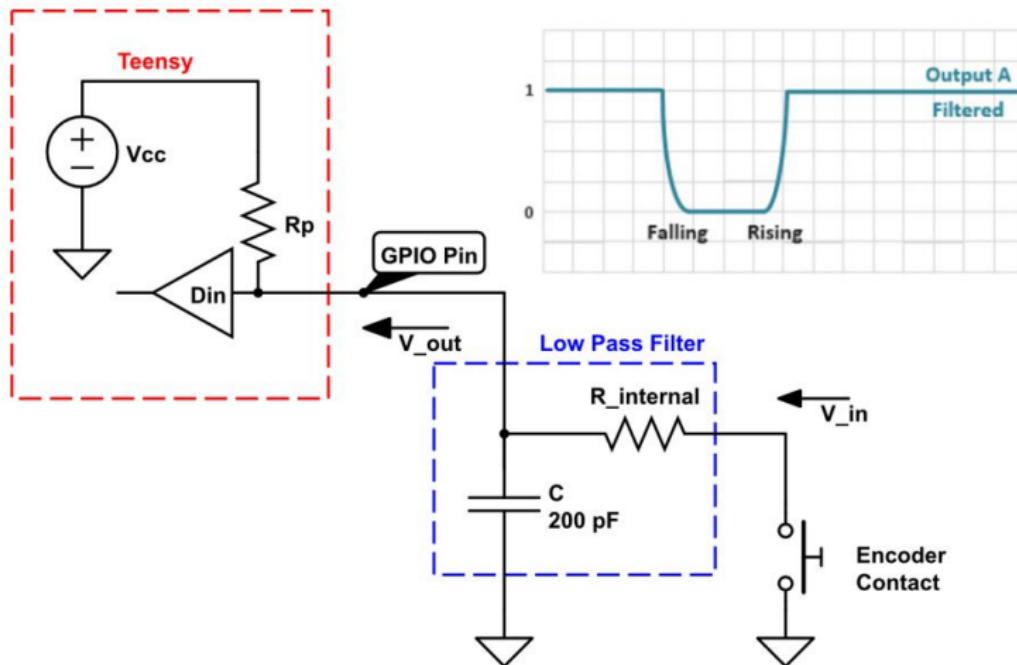


Encoder Jitter



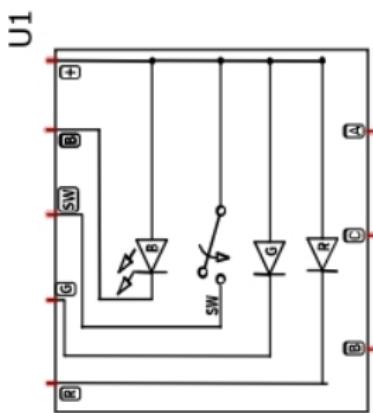


Encoder - Low Pass Filter





Encoder - LEDs and Button



fritzing

The encoder has a Red, Green, and Blue LED, as well as a Switch (technically a Button), that act like discrete components and are not associated with the Encoder.h library. Note, the LEDs are active low (the pins need to be set to ground to activate the light).



Assignment: Encoders



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

① L05_03_encoder_switch

- Connect the encoder switch and LEDs.
- Use the switch to turn on/off the NeoPixels.
- Set encoder LED to red for off and green for on.
- Disable turning the encoder when off.

② L05_04_rainbow1

- Use a button to cycle the NeoPixel ring colors through the colors of the rainbow; one color change each time button is pressed.

③ L05_05_rainbow2

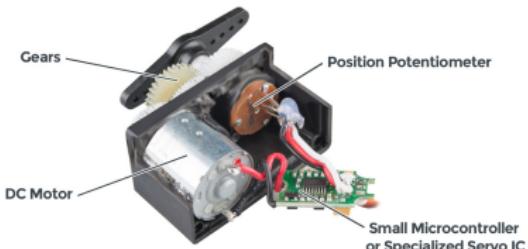
- Have it continuously cycle (i.e., while pressed colors change every one second).

Module 6 - 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

The Particle Argon has the Servo class built-in, so no library is needed.

① Header

- Servo myServo; - create object myServo of class Servo

② void setup()

- myServo.attach(pin) - attach the Servo object to a pin (this must be a PWM pin)

③ void loop()

- myServo.write(angle) - move servo to angle (in degrees)



Assignment: L06_Servo



① L06_01_Servo

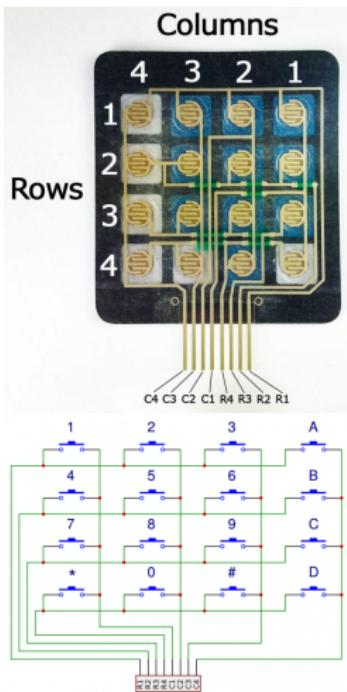
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

- Connect the servo and a button to the Teensy
- Create a HelloServo-type code that moves the servo to 180 degrees, waits, and then moves it to zero.
- Modify the code to have the servo oscillate between 0 and 180 degrees using a sine wave pattern.
- Have the button to start and stop the motion.
- Extra: Have the motion begin again at the point in the cycle where it stops.

Recall, in C++ (`math.h`) the math constants are `M_<name>`. For example, `M_PI` = π .



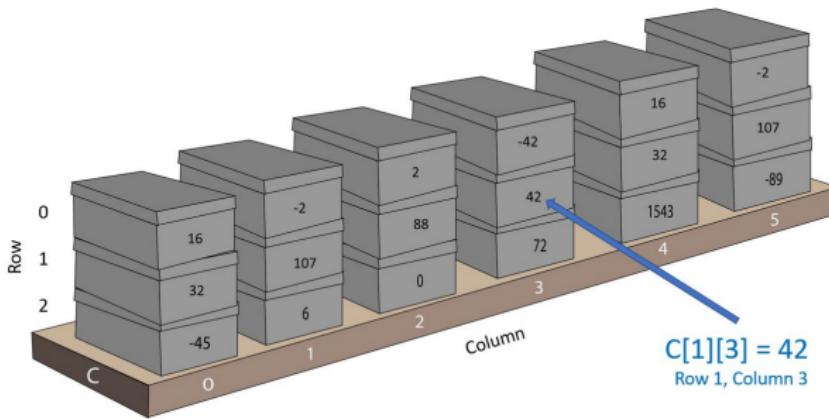
KeyPad



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



2-dimensional arrays



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



Char and Byte Datatype

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

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[STRUCTURE TEXT]	34	22	“	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQ/UART]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[PRINT]	39	27	*	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARriage RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SOFT DLE]	46	2E	=	78	4E	N	110	6E	n
15	F	[SOH/FN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRAN BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[SOFT DLE]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	{	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	\
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	-
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	-	127	7F	[DEL]

ASCII: American Standard Code For Information Interchange

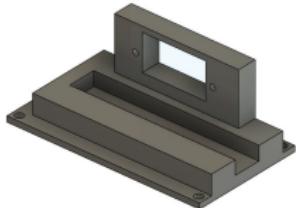


Keypad.h

```
1 #include <Keypad.h>
2
3 const byte ROWS = 4;
4 const byte COLS = 4;
5 char customKey;
6
7 char hexaKeys[ROWS][COLS] = {
8     {'1', '2', '3', 'A'},
9     {'4', '5', '6', 'B'},
10    {'7', '8', '9', 'C'},
11    {'*', '0', '#', 'D'}
12};
13
14 byte rowPins[ROWS] = {9, 8, 7, 6};      \\keypad leads 8,7,6,5
15 byte colPins[COLS] = {5, 4, 3, 2};      \\keypad leads 4,3,2,1
16
17 Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
18
19 void setup(){
20     Serial.begin(9600);
21 }
22
23 void loop(){
24     customKey = customKeypad.getKey();
25
26     if (customKey){
27         Serial.printf("Key Pressed: %c\n",customKey);
28         Serial.printf("Key Pressed (Hex Code) 0x%02X\n",customKey);
29     }
30 }
```



Assignment: L06_02_Lock



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

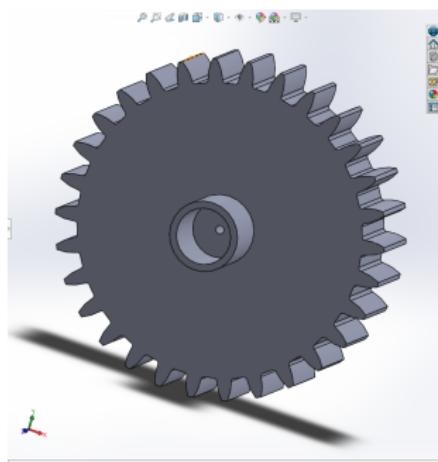
- ➊ Design and print a lockholder based on the class example.
 - Optional: design and print your own gear and lock slider
- ➋ Using the keypad, create a digital lock
 - Implement 4-digit digital "key" in an array.
 - Use the keypad to enter a code and store the entered code in an array
 - Create a bool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

^aRemember to use local variables in the function

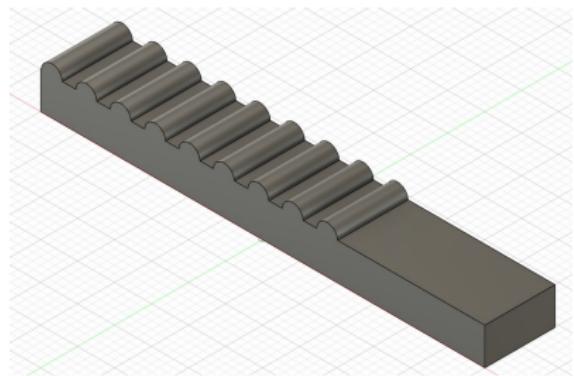


Optional Designs

GEAR



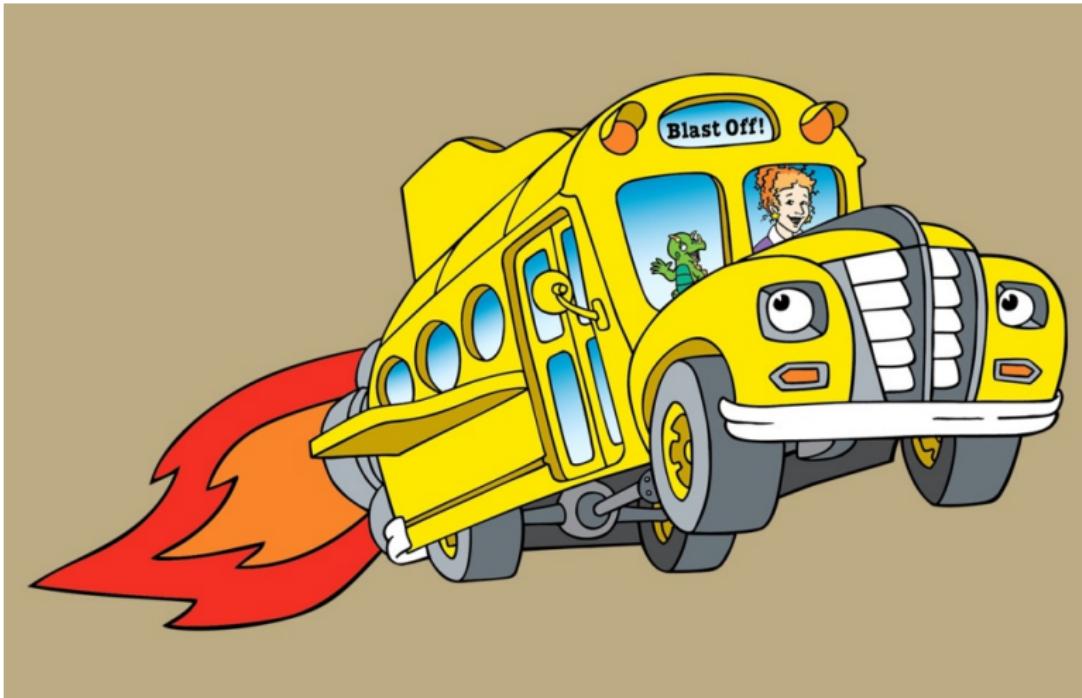
LOCK SLIDER



Module 7 - I^2C

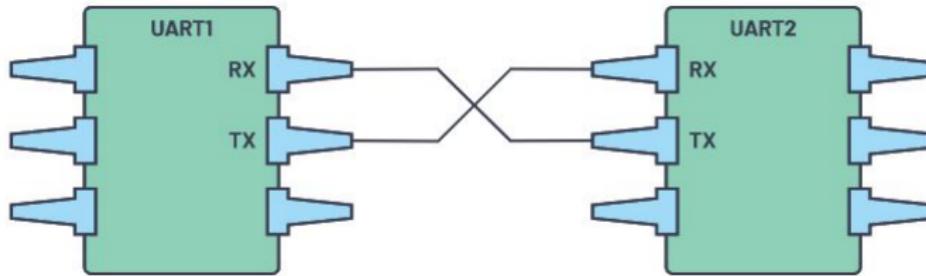


Buses and Interfaces





UART

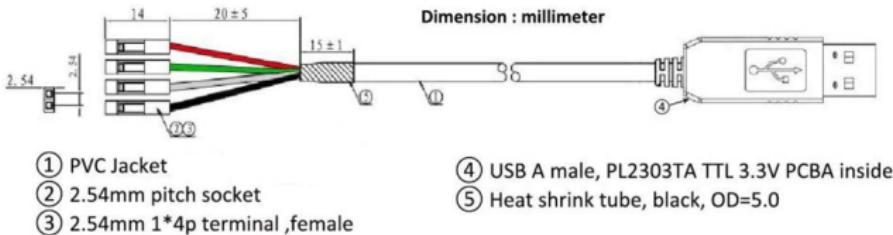


Universal Asynchronous Receiver/Transmitter

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



The USB Cable is a UART connection

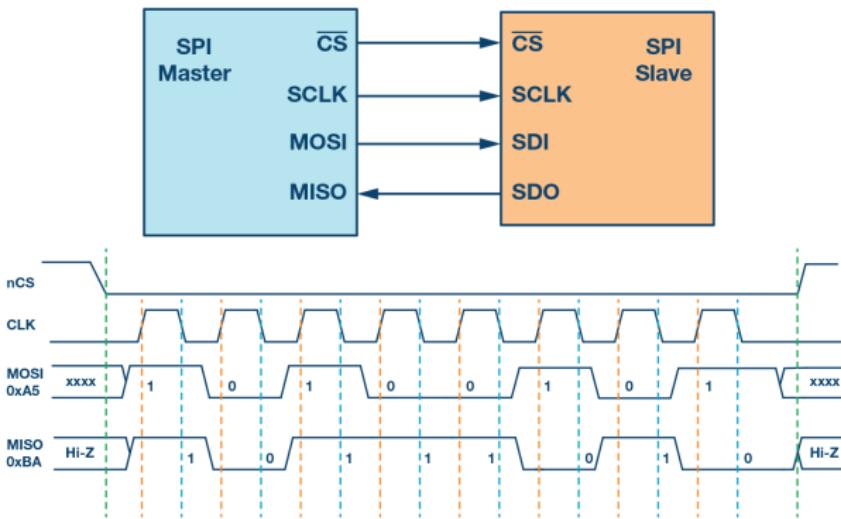


1*4P Female Socket	Name	Colour	Description
Pin 1	TXD	White	Transmit Asynchronous Data
Pin 2	RXD	Green	Receive Asynchronous Data
Pin 3	GND	Black	Device ground supply
Pin 4	VCC	Red	+5V





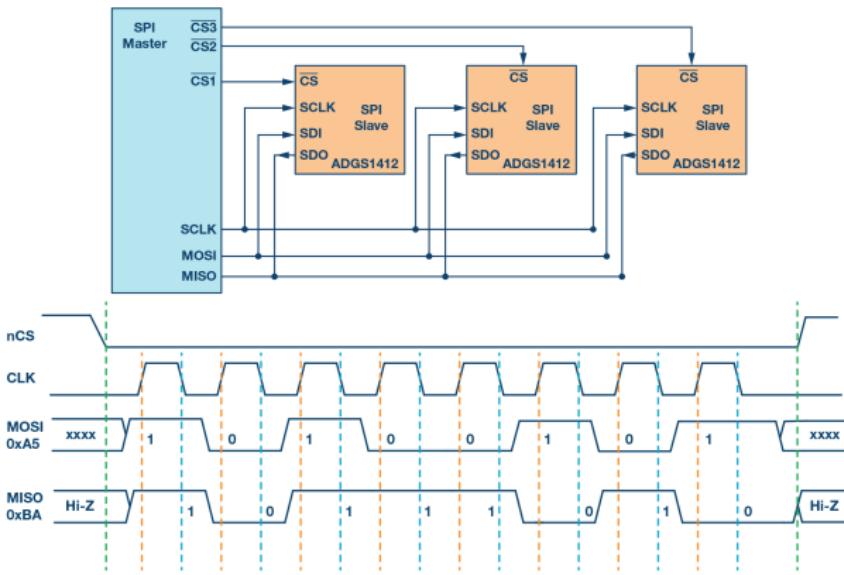
Serial Peripheral Interface



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



SPI - Chip Select

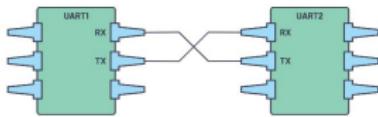


- SPI uses the \overline{CS} lines to select which peripheral is active.
- Having two SPI devices selected at the same time causes interference.
- In void `setup()`, always initialize all SPI devices as "off"
 - Note: \overline{CS} is active LOW ("off" is HIGH)

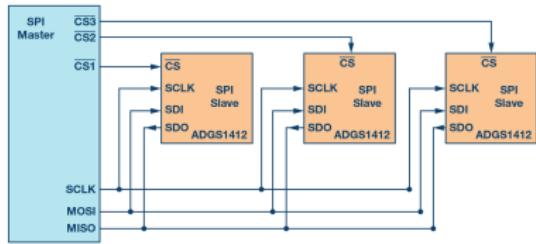


Serial UART vs SPI

UART

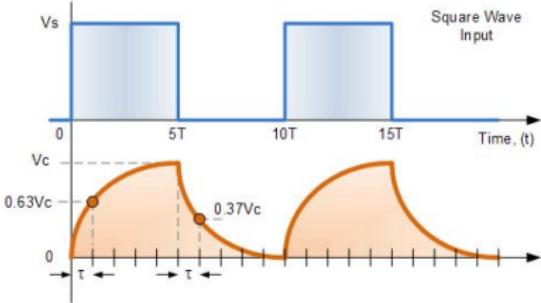
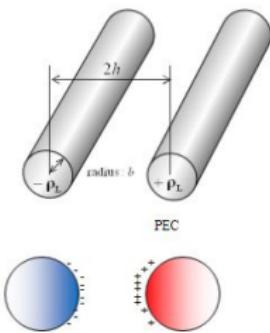


SPI





RC Time Constant



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



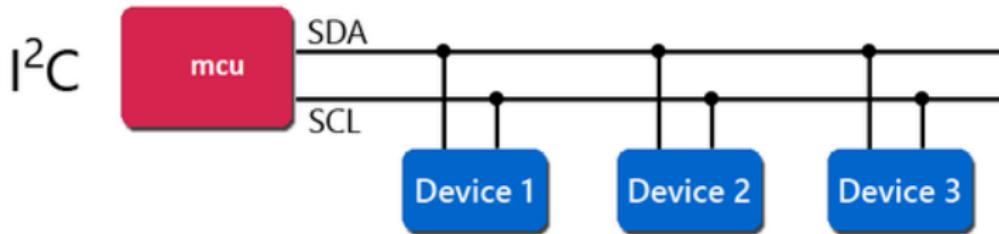
IoT Humor



Backing Up the Internet of Things



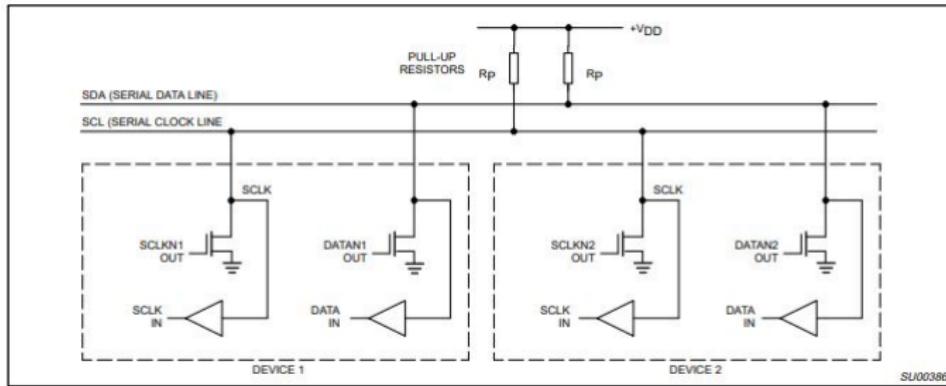
Inter-integrated Circuit (I^2C)





I^2C Pullup Resistors

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



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



I²C vs SPI

I²C v/s SPI

I ² C	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 I ² C
More Overhead (due to acknowledgement, start, stop)	Less Overhead
Noise sensitivity is high	Less noise sensitivity



L07_00_I2CScanner

Let's create an I2C scanner

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



Char Datatype - Revisited

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

ASCII control characters		ASCII printable characters		Extended ASCII characters	
00	NULL (Null character)	32	space	64	Ø
01	SOH (Start of Header)	33	!	65	À
02	STX (Start of Text)	34	"	66	Á
03	ETX (End of Text)	35	#	67	Â
04	EOT (End of Trans.)	36	\$	68	Ã
05	ENQ (Enquiry)	37	%	69	È
06	ACK (Acknowledgement)	38	&	70	É
07	BEL (Bell)	39	'	71	Í
08	BS (Backspace)	40	(72	Ó
09	HT (Horizontal Tab)	41)	73	Ù
10	LF (Line feed)	42	*	74	Ï
11	VT (Vertical Tab)	43	+	75	À
12	FF (Form feed)	44	-	76	Á
13	CR (Carriage return)	45	.	77	Â
14	SO (Shift Out)	46	,	78	Ã
15	SI (Shift In)	47	/	79	È
16	DLE (Data link escape)	48	0	80	Í
17	DC1 (Device control 1)	49	1	81	Ó
18	DC2 (Device control 2)	50	2	82	Ù
19	DC3 (Device control 3)	51	3	83	Ï
20	DC4 (Device control 4)	52	4	84	À
21	NAK (Negative acknowledgement)	53	5	85	Á
22	SYN (Synchronous idle)	54	6	86	Â
23	ETB (End of transmission block)	55	7	87	Ã
24	CAN (Cancel)	56	8	88	È
25	EM (End of medium)	57	9	89	Í
26	SVD (Software vendor)	58	;	90	Ó
27	ESC (Escape)	59	:	91	Ù
28	FS (File separator)	60	<	92	Ï
29	GS (Group separator)	61	=	93	À
30	RS (Record separator)	62	>	94	Á
31	US (Unit separator)	63	?	95	Â
127	DEL (Delete)		-		

ASCII 248	
Ø	alt + 248 (Degree symbol)
most consulted	
ñ	é ñ n with tilde (alt + 164)
█	black square (alt + 254)
²	superscript two, square (alt + 255)
°	degree symbol (alt + 251)
'	apostrophe, single quote (alt + 39)
µ	letter Mu, micro, microm (alt + 232)
©	copyright symbol (alt + 164)
®	registered trademark (alt + 165)
³	superscript three, cube (alt + 252)
á	á with acute accent (alt + 160)

```

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

```



A word about example code

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

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



Assignment: I²C



① Install Adafruit_SSD1306 library

- Run SSD_1306_128x64_i2c example, changing the I2C address to match your OLED.
- In your notebook document the commands to create an object, initialize the object, and the various commands to display text.

② L07_01_OLEDWrite

- Without cut/paste, using your notes and printf() display:
 - Hello World
 - Your Name using spanish honorific (señor, señora, señorita).
 - Your Birthday (e.g., 04/03/1968) using separate variables for month, day, and year.
- Rotate screen using setRotation(rot) method.
 - rot is an int from 0 to 3
- OPTIONAL: Make your own bitmap:
<https://diyusthad.com/image2cpp> and
<https://www.reduceimages.com>

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



Using the Adafruit_BME280 class

```
1 // Define BME280 object
2 Adafruit_BME280 bme; //this is for I2C device
3
4 // Initialize the BME280 in void setup()
5 status = bme.begin(hexAddress);
6 if (status == false) {
7     Serial.printf("BME280 at address 0x%02X failed to
8     start", hexAddress);
9 }
10
11 // Getting data from BME280
12 tempC = bme.readTemperature(); //deg C
13 pressPA = bme.readPressure(); //pascals
14 humidRH = bme.readHumidity(); // %RH
```



Assignment: I^2C



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

① L07_02_BME280

- Read BME280 data.
- Convert to tempF and inHg.
- Print to Serial.Monitor and the OLED display.

② L08_03_BME280_SDMicro

- Add in saving data to the μ SD card once per minute using millis() as the timestamp.
- Use your NeoPixels to give a visual indication of room conditions.

Module 8 - Internet



The Internet





IP Addresses

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

IPv4 address in dotted-decimal notation

172 . 16 . 254 . 1

↓ ↓ ↓ ↓

10101100.00010000.11111110.00000001

8 bits

32 bits (4 bytes)

An IPv6 address (in hexadecimal)

2001:0DB8:AC10:FE01:0000:0000:0000:0000

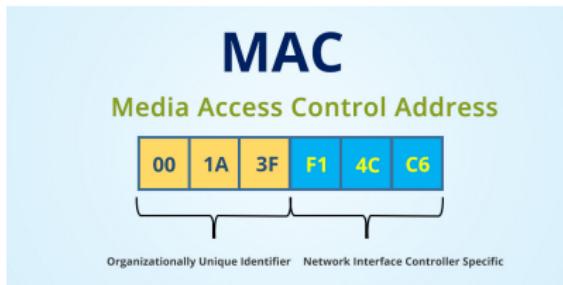
2001:0DB8:AC10:FE01:: Zeroes can be omitted

0010000000000001:0000110110111000:1010110000010000:1111111000000001:

2000000000000-200000000000-20000000000-200000000000



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.



Finding MAC Address

```
1 SYSTEM_MODE(SEMI_AUTOMATIC)
2 byte mac[6];
3
4 void setup() {
5
6     WiFi.on();
7     WiFi.connect();
8     while(WiFi.connecting()) {
9         Serial.printf(".");
10        delay(250);
11    }
12    Serial.printf("\nWaiting for IP Address\n");
13    delay(1000);
14    Serial.printf("Scan Argon for WiFi Information \n");
15    Serial.printf("ip address: %s \n", WiFi.localIP().toString().c_str());
16    WiFi.macAddress(mac);
17    Serial.printf("mac: %02X:%02X:%02X:%02X:%02X:%02X \n", mac[0], mac[1], mac[2], mac[3], mac
18 [4], mac[5]);
```

Serial monitor opened successfully:
.....
Waiting for IP Address
Scan Argon for WiFi Information
ip address: 10.0.0.25
mac: C8:2B:96:B5:30:88



Forcing Connection to Specific Network

In finding your IP address, you connected to DDCIOT (strongest signal). If there are multiple networks and we want to force connection to a specific network:

```
1 SYSTEM_MODE(MANUAL);
2
3 void setup() {
4     Serial.begin(9600);
5     waitFor(Serial.isConnected, 15000);
6
7     WiFi.on();
8     WiFi.setCredentials("IoTNetwork");
9     // If network requires a password
10    // setCredentials(const ""NetworkName", "Password");
11
12    WiFi.connect();
13    while(WiFi.connecting()) {
14        Serial.printf(".");
15    }
16    Serial.printf("\n\n");
17 }
```



Assignment: Wemo



- Schematic
- Fritzing diagram
- Wire your circuit
- Write the code

① L08_01_Wemo

- Using wemo.h, use the button to toggle Wemo on/off (one toggle per press)
- Implement a method to select different Wemo (encoder, second button, etc.)

② L08_02_Wemo_Timer

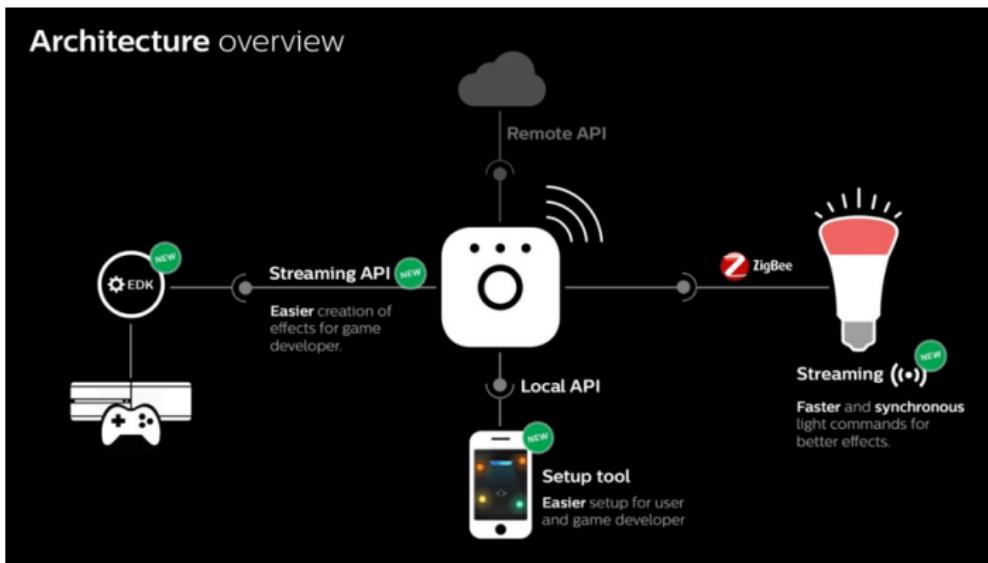
- Create timer that turns off a Wemo 10 secs after you push "off" button without using delay().

③ L08_03_Wemo_Object

- Copy wemo.h to wemoObj.h
- Modify it to be use Class and Methods.
- Modify your code to use a wemoObj object.



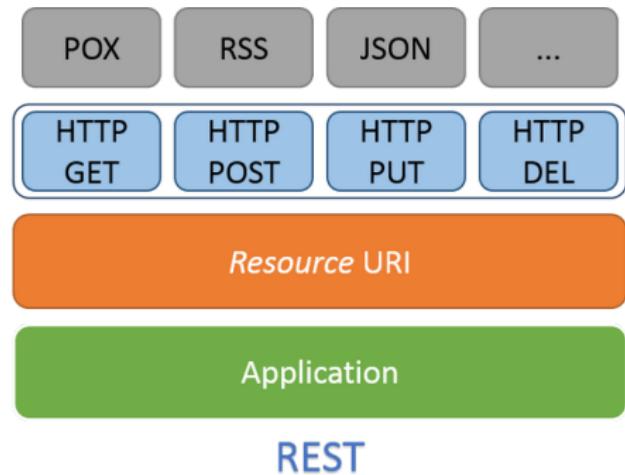
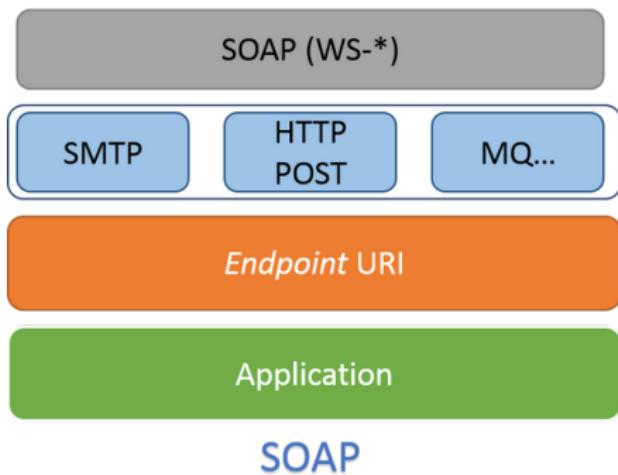
Phillips Hue API



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.

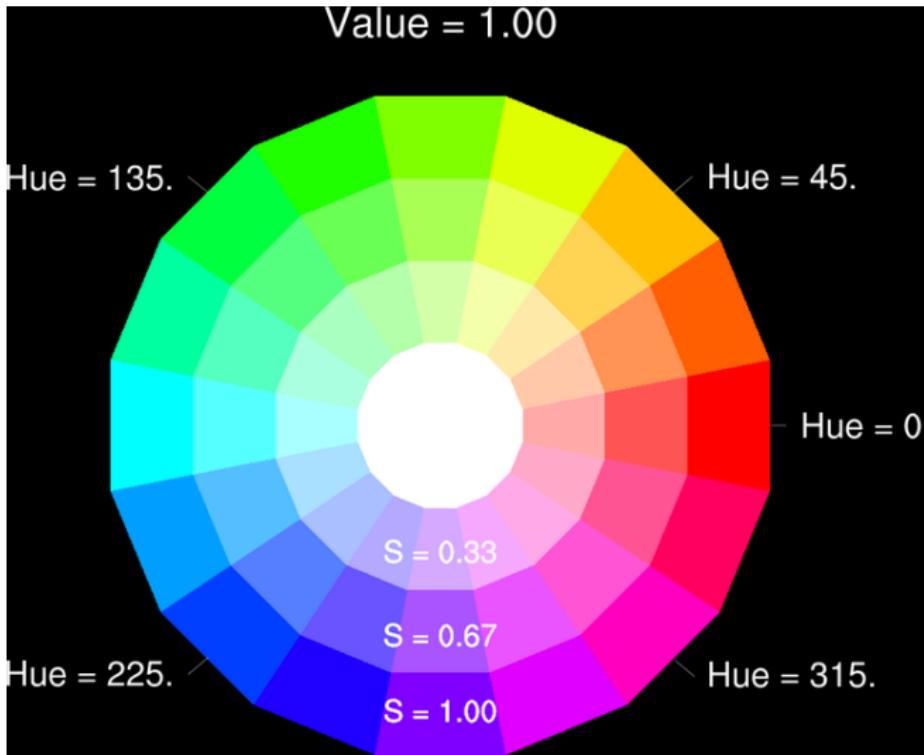


SOAP vs REST





HSV Colors





Assignment: L08_04_Hue



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

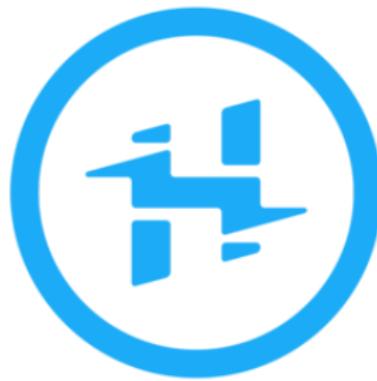
1 Using the hue.h library and L11_00_HueHeader as a template, create code that:

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

Midterm 1 - Smart Room Controller



IoT Portfolio

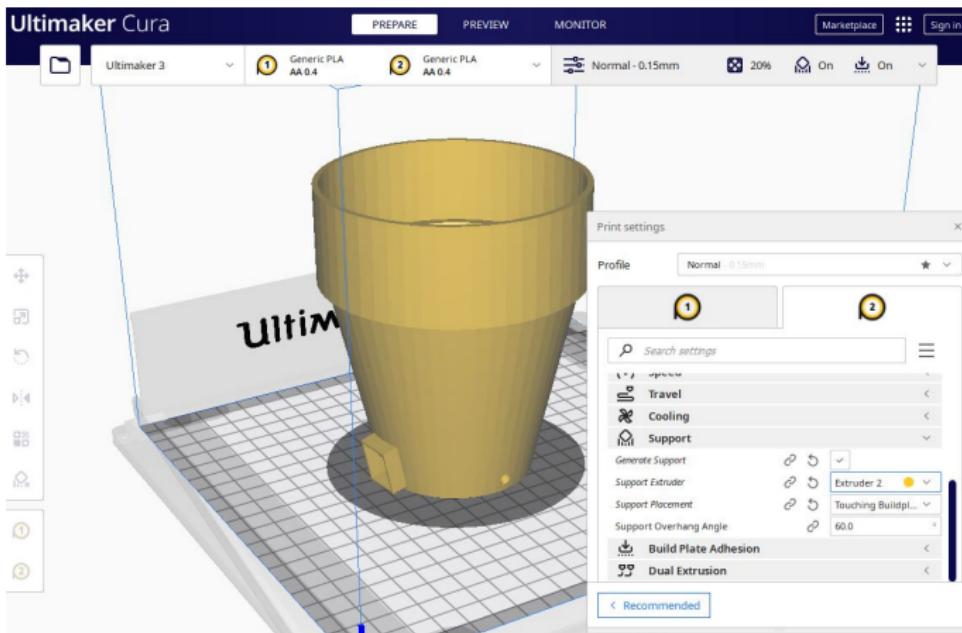


hackster.io

Create a [hackster.io](#) account.



Printing IoT Flowerpot



It is important to change Support Placement to Touching Buildplate.



Midterm Project - Smart Room Controller

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



Smart Room Controller - Minimum Requirements

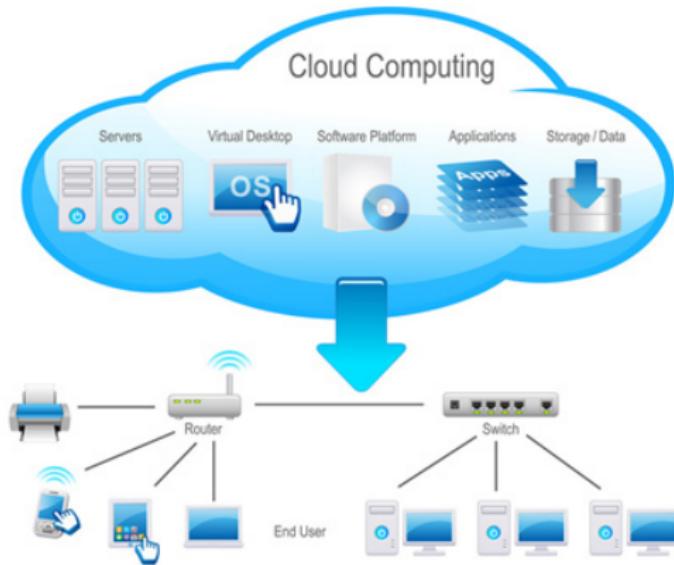
You are developing a Smart Room Controller prototype that you are going to pitch to a perspective investor.

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

Module 9 - The Cloud



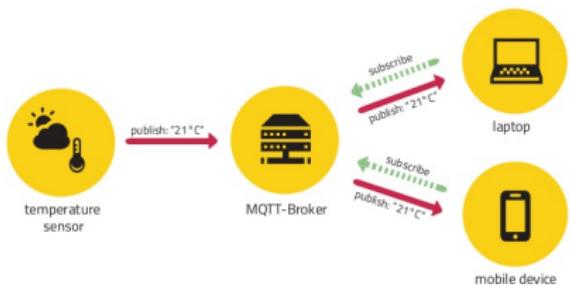
The Cloud





MQTT: MQ Telemetry Transport

Publish / Subscribe



MQTT Ports



MQTT + TCP



1883
Official IANA Port

MQTT + TLS



8883
Official IANA Port

MQTT + Websockets



80 / 443
Standard HTTP Ports



Adafruit.io



Let's create an Adafruit.io account.

Get Started

FREE
forever

30 data points per minute
30 days of data storage
Triggers every 15 minutes
5 feed limit

[Sign Up Now](#)

Power Up

\$10 or \$99
per month per year

60 data points per minute
60 days of data storage
Triggers every 5 seconds
Unlimited feeds

[Learn more about IO+>](#)

[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

```
1 #include <Adafruit_MQTT.h>
2
3 #include "Adafruit_MQTT/Adafruit_MQTT.h"
4 #include "Adafruit_MQTT/Adafruit_MQTT_SPARK.h"
5
6 /***** Global State (you don't need to change this!) *****/
7 TCPClient TheClient;
8
9 /***** Adafruit.io Setup *****/
10 #define AIO_SERVER      "io.adafruit.com"
11 #define AIO_SERVERPORT   1883                      // use 8883 for SSL
12 #define AIO_USERNAME     "<username>"
13 #define AIO_KEY          "<key>"
14
15 // Setup the MQTT client class with the WiFi client, MQTT server and login details.
16 Adafruit_MQTT_SPARK mqtt(&TheClient,AIO_SERVER,AIO_SERVERPORT,AIO_USERNAME,AIO_KEY);
17
18 /***** Feeds *****/
19 // Setup Feeds to publish or subscribe
20 // Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>
21 Adafruit_MQTT_Publish mqtt0bj1 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/
    FeedNameA");
22 Adafruit_MQTT_Subscribe mqtt0bj2 = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/
    FeedNameB");
23
24
25 /***** Declare Variables*****/
26 float value1;    //variable that will hold data to be published to adafruit.io feed
27 float value2;    //variable that will hold data received from adafruit.io feed
```

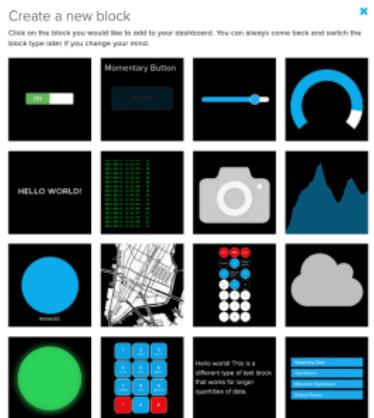


MQTT Publish and Subscribe

```
1 void setup() {
2   Serial.begin(9600);
3   waitFor(Serial.isConnected, 15000); //wait for Serial Monitor to startup
4
5   WiFi.connect(); //Connect to internet, but not Particle Cloud
6   while(WiFi.connecting()) {
7     Serial.printf(".");
8   }
9
10 mqtt.subscribe(&mqttObj2); // Setup MQTT subscription for FeedNameB feed.
11 }
12
13 void loop() {
14   // Publishing to a MQTT feed
15   if(mqtt.Update()) { //if mqtt object (Adafruit.io) is available to receive data
16     Serial.printf("Publishing %0.2f to Adafruit.io feed FeedNameB \n",value1);
17     mqttObj1.publish(value1);
18   }
19   // Two new functions that will be useful:
20   // atof() - ASCII to Float: converts an ASCII string to a floating point number
21   // atoi() - ASCII to Integer: converts an ASCII string to an integer
22
23   // Receive data from a subscription to an MQTT feed
24   Adafruit_MQTT_Subscribe *subscription;
25   while ((subscription = mqtt.readSubscription(100))) { //wait a moment for new feed data
26     if (subscription == &mqttObj2) { // assign new data to appropriate variable
27       value2 = atof((char *)mqttObj2.lastread); //value2 = data from MQTT subscription
28       Serial.printf("Received %0.2f from Adafruit.io feed FeedNameB \n",value2);
29     }
30   }
31 }
```



Assignment: L09_01_SubscribePublish



- ① Modify the starter code for your Adafruit.io
- ② Publish
 - Publish a random number to a feed once every 6 seconds (do not use a delay).
 - Create a line chart on your dashboard to display the random number.
- ③ 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 (D7) when pressed.
- ④ Experiment with other blocks
 - Replace the button with a slider.
 - Control the brightness of an LED.
 - Display data with other dashboard blocks.



Local and Static Variables

```
1 const int TEMPFREQ = 10000, MOISTFREQ = 30000, MOISTPIN = A3;
2 float tempC;
3 int moist;
4 void loop() {
5     tempC = getTemp(TEMPFREQ);
6     moist = getMoisture(MOISTPIN, MOISTFREQ);
7 }
8
9 float getTemp(int timeInterval) {
10    int currentTime;
11    static int lastTime = -999999;
12    static float data;
13
14    currentTime = millis();
15    if(currentTime - lastTime > timeInterval) {
16        lastTime = millis();
17        data = BME.readTemperature();
18    }
19    return data;
20 }
21
22 int getMoisture(int probePIN, int timeInterval) {
23    static int lastTime = -999999;
24    static int data;
25
26    if(millis() - lastTime > timeInterval) {
27        lastTime = millis();
28        data = analogRead(probePIN);
29    }
30    return data;
31 }
```



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.



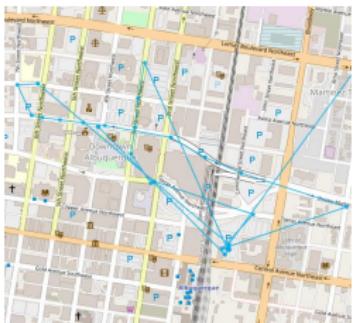
JSON Parser Generator

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

```
1 #include <JsonParserGeneratorRK.h>
2
3 void createEventPayLoad(float tempValue, float presValue, float humValue) {
4     JsonWriterStatic<256> jw;
5     {
6         JsonWriterAutoObject obj(&jw);
7
8         jw.insertKeyValue("Temperature", tempValue);
9         jw.insertKeyValue("Pressure", presValue);
10        jw.insertKeyValue("Humidity", humValue);
11    }
12    Particle.publish("env-vals", jw.getBuffer(), PRIVATE);
13 }
```



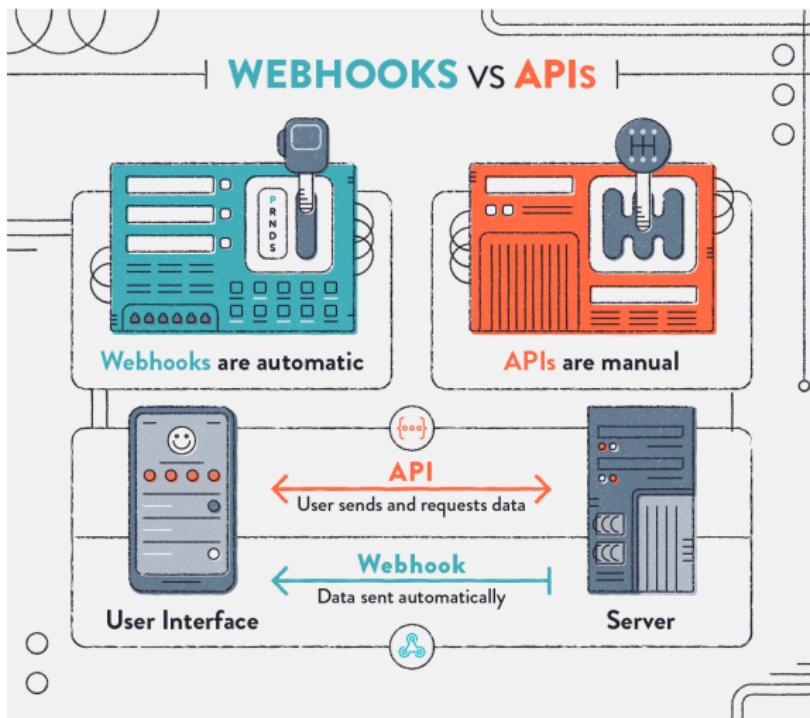
Assignment: L09_02_GPSPublish



- ① Every 10 seconds (without using delays), generate random GPS coordinates within Albuquerque.
- ② Publish to Adafruit.io using MQTT and JSON format. Use "lat" and "lon" as the names for the JSON data elements.
- ③ Using the Map block to create a dashboard that shows the GPS coordinates



Webhooks





Step 1 - OpenWeather

- Create an account at openweathermap.org
- Generate an API key at:
https://home.openweathermap.org/api_keys

You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.

Key	Name	Status	Actions	Create key
07f56344e3fe7a27974a52eb54783b71	Default	Active		<input type="text" value="API key name"/>
dacc7f9f41f4cca0a274cf925b97a356	IoTClass	Active		



Step 2 - Create Webhook

From `console.particle.io`:

The screenshot shows the Particle Integrations page. On the left is a sidebar with icons for Particle, Personal, and a search bar. The main area is titled "Integrations" and contains five cards, each representing a "Webhook" integration:

- Card 1: "Webhook" with icons for "temp", "any device", and "thingspeak.com".
- Card 2: "Webhook" with icons for "temp", "any device", and "thingspeak.com".
- Card 3: "Webhook" with icons for "temp", "any device", and "thingspeak.com".
- Card 4: "Webhook" with icons for "temp", "any device", and "thingspeak.com".
- Card 5: A dashed-line box containing a plus sign with the text "NEW INTEGRATION" below it.

The screenshot shows the "Sandbox" section of the Particle Integrations page, specifically the "New Integration" sub-page. The sidebar on the left includes icons for Particle, Personal, and a search bar. The main content area lists several integration options:

- Google Maps**: Geolocate Particle devices via visible Wi-Fi access points or Cellular towers.
- Azure IoT Hub**: Stream Particle device data into the Azure ecosystem.
- Google Cloud Platform**: Tie into an enterprise grade suite of cloud-based data storage and analysis tools.
- Webhook**: Push Particle device data to other web services in real-time.



Step 3 - Select Custom Template

The screenshot shows the Particle Webhook Builder interface. On the left is a sidebar with icons for Device, Project, and Help. The main area has a header "Sandbox" with a dropdown arrow. Below it, the navigation path is "Integrations > New Integration > Webhook". There are two tabs: "WEBHOOK BUILDER" and "CUSTOM TEMPLATE", with "CUSTOM TEMPLATE" underlined. A section titled "Particle webhook template reference" contains a code editor with the following JSON template:

```
1 [  
2   "event": "",  
3   "url": "",  
4   "requestType": "POST",  
5   "noDefaults": false,  
6   "rejectUnauthorized": true  
7 ]
```



Step 4 - Update Custom Template with API format

Adapted from <https://openweathermap.org/api/one-call-api> and <https://openweathermap.org/current>

```
1  {
2      "event": "GetWeatherData",
3      "responseTopic": "{{PARTICLE_DEVICE_ID}}/{{PARTICLE_EVENT_NAME}}",
4      "url": "https://api.openweathermap.org/data/2.5/onecall",
5      "requestType": "GET",
6      "noDefaults": true,
7      "rejectUnauthorized": true,
8      "responseTemplate": "{\"lat\":{{lat}},\"lon\":{{lon}},\"dt\":{{current.dt}},\"temp\":
9          \":{{current.temp}},\"uvi\":{{current.uvi}},\"clouds\":{{current.clouds}},\"ws\":{{\n10         current.wind_speed}},\"wd\":{{current.wind_deg}} }",
11      "query": {
12          "lat": "{{lat}}",
13          "lon": "{{lon}}",
14          "exclude": "minutely,hourly,daily,alerts",
15          "units": "metric",
16          "appid": "dacc7f9f41f4cca0a274cf925b97a356"
17      }
18 }
```

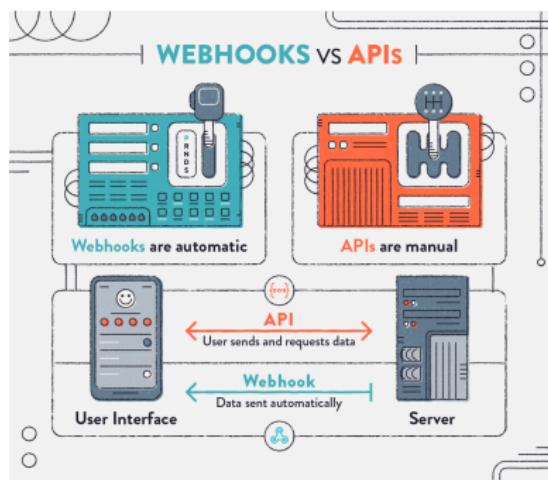


Step 5 - Particle Argon Code

```
1 const char *EVENT_NAME = "GetWeatherData";
2 unsigned int lastTime;
3 const float lat=35.0045, lon=-106.6465; //update to your favorite location
4
5 void setup() {
6     Serial.begin(9600);
7     waitFor(Serial.isConnected,15000);
8     String subscriptionName = String::format("%s/%s/", System.deviceID().c_str(),
9         EVENT_NAME);
10    Particle.subscribe(subscriptionName, subscriptionHandler, MY_DEVICES);
11    Serial.printf("Subscribing to %s\n", subscriptionName.c_str());
12 }
13
14 void loop() {
15     if((millis() - lastTime) > 60000) {
16         Serial.printf("\n\nTime = %i\n",millis());
17         Particle.publish(EVENT_NAME, "", PRIVATE);
18         Particle.publish(EVENT_NAME, String::format("{\"lat\":%0.5f,\"lon\":%0.5f}", lat,
19             lon), PRIVATE);
20         lastTime = millis();
21     }
22 }
23 void subscriptionHandler(const char *event, const char *data) {
24     JSONValue outerObj = JSONValue::parseCopy(data);
25     JSONObjectIterator iter(outerObj);
26     while(iter.next()) {
27         Serial.printf("key=%s value=%s\n", (const char *) iter.name(), (const char *)
28             iter.value().toString());
29     }
30 }
```



Assignment: L09_03_GetWeather

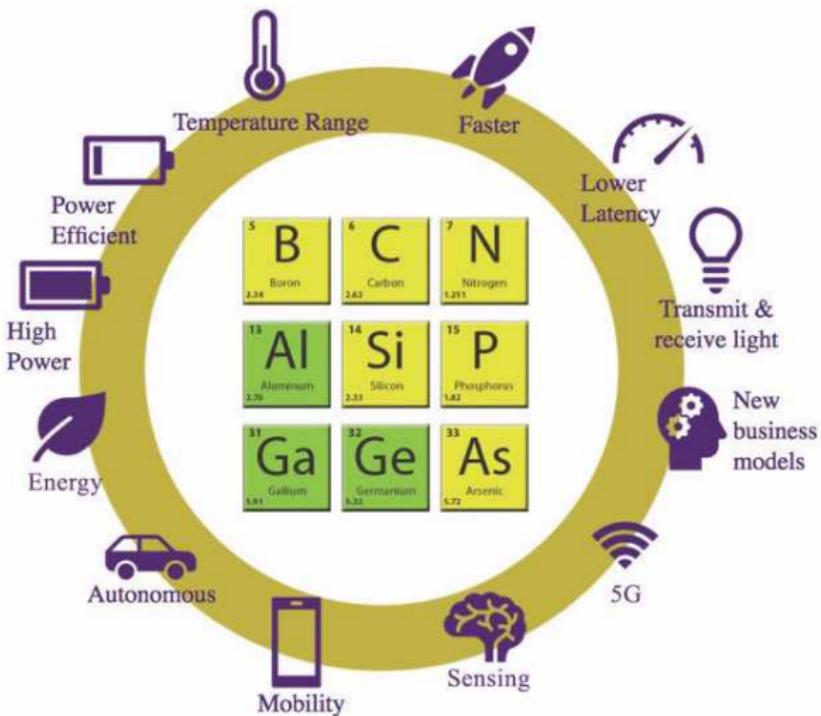


- ① Using the OpenWeatherMap webhook get the outside weather conditions.
- ② Display the OLED:
 - GPS location (hard coded)
 - Indoor conditions from BME280
 - Current outdoor conditions.

Module 10 - Semiconductors



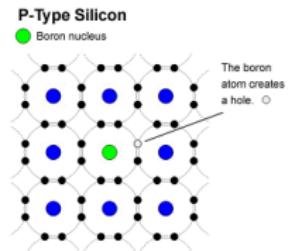
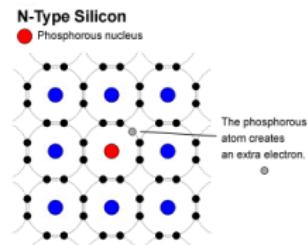
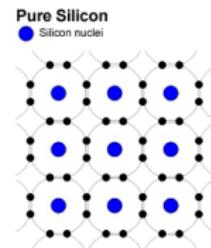
Semiconductors





Semiconductor

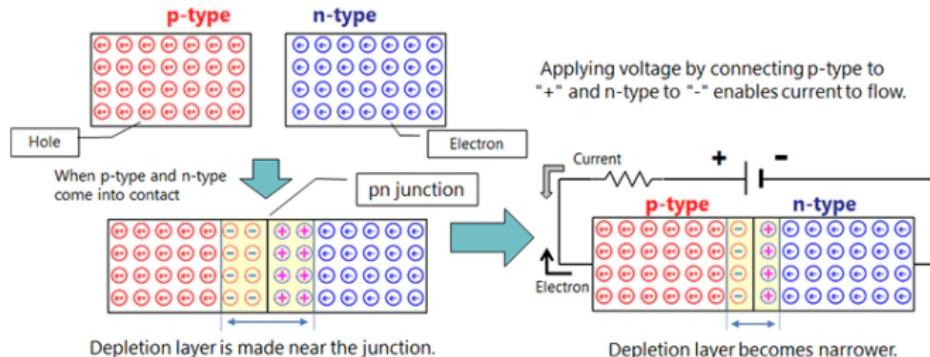
- A silicon atom has four electrons in its outer shell and bonds tightly with four surrounding silicon atoms creating a crystal matrix with eight electrons in the outer shells. The tight bonds make pure silicon non-conducting.
- Phosphorus has five electrons, and when combined, the fifth electron becomes 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

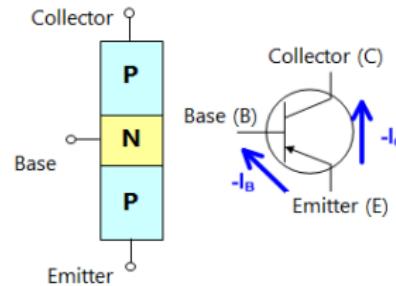
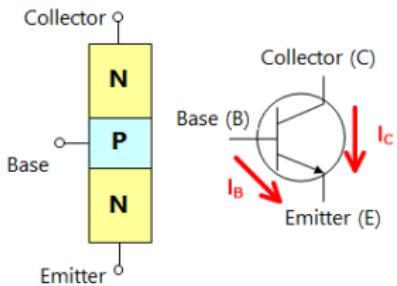
- When p-type and n-type semiconductors are bonded, holes and free electrons are attracted, combine, and disappear near the boundary. Since there are no carriers in this area, it is called a depletion layer and it is an insulator.
- A positive voltage applied to the p-type region causes electrons to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





Bipolar Junction Transistor

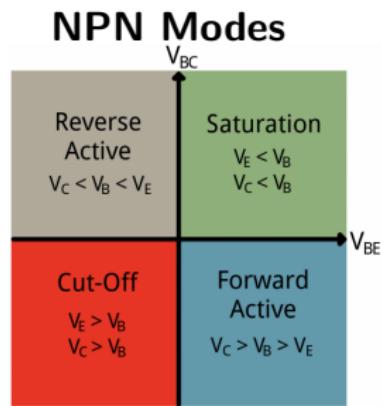
The transistor has three regions, namely base, emitter and collector. The emitter is a heavily doped terminal and emits electrons into the base. The base terminal is lightly doped and passes the emitter-injected electrons on to the collector. The collector terminal is intermediately doped and collects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





Bipolar Junction Transistor - Modes of Operation

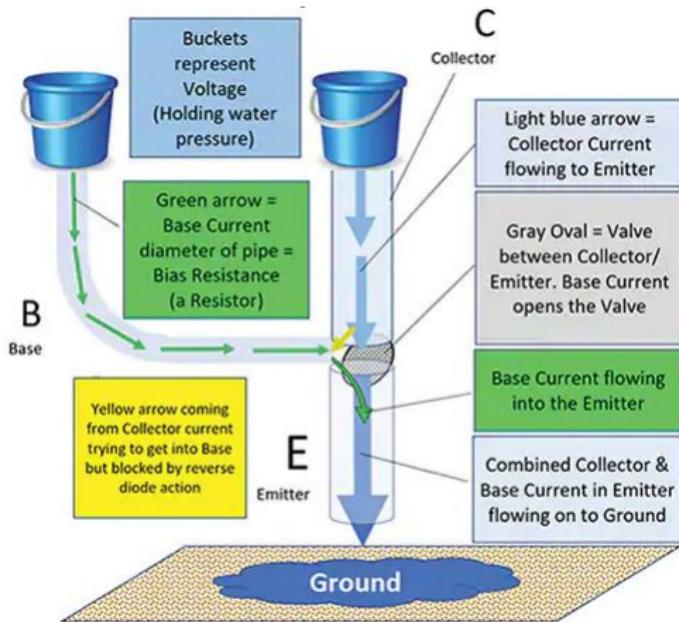
- **Saturation:** Current freely flows from collector to emitter. (ON Switch)
- **Cut-off:** No current flows from collector to emitter. (OFF Switch)
- **Active:** The current from collector to emitter is proportional to the current flowing into the base. (Amplifier)
- **Reverse-Active:** Like active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



Voltage relations	NPN Mode	PNP Mode
V _e < V _b < V _c	Active	Reverse
V _e < V _b > V _c	Saturation	Cutoff
V _e > V _b < V _c	Cutoff	Saturation
V _e > V _b > V _c	Reverse	Active



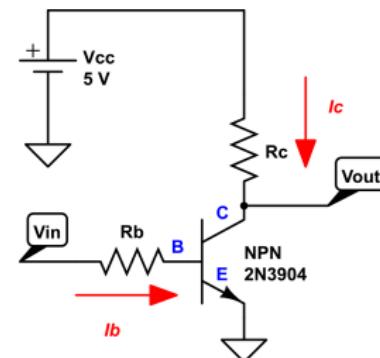
Water Analogy





Active Mode NPN Transistor Circuit

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



NOTE: V_{OUT} is an amplified but inverted signal of V_{IN} .

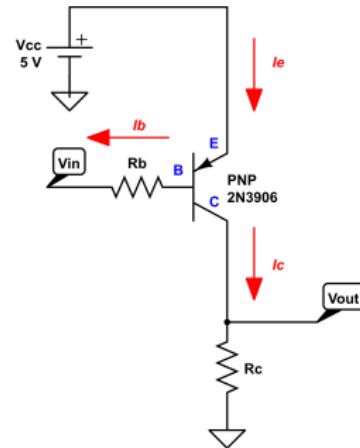
This simple circuit will step-up a 0 - 3.3V output from the microcontroller to 0 - 5.0V (inverted). The low impedance of the output will also provide sufficient current to drive a higher current device (e.g., a relay).



PNP Transistor

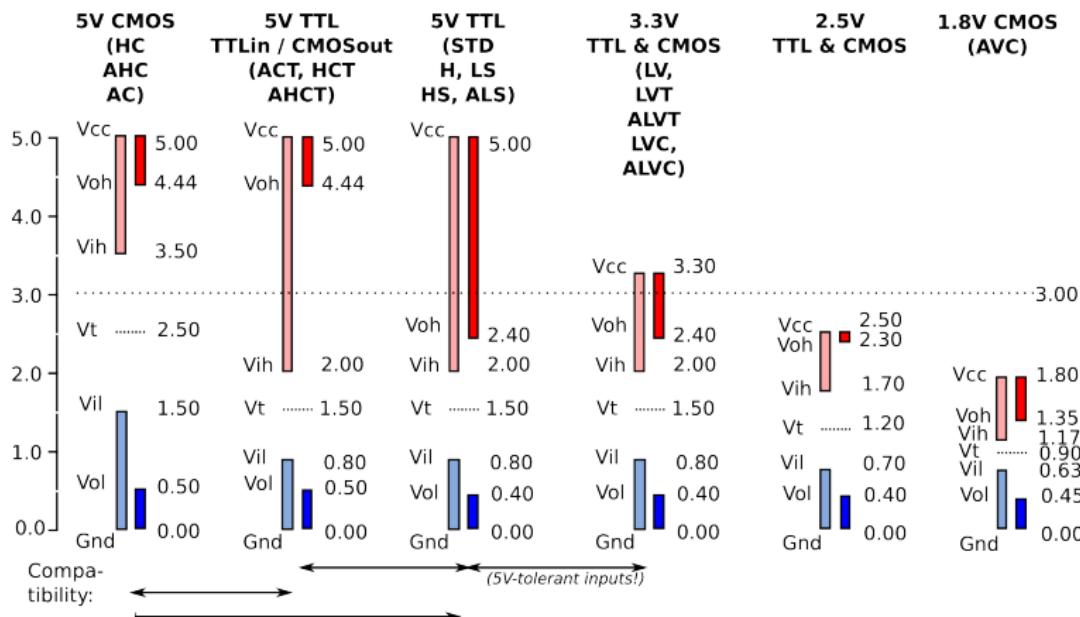
NPN Transistors are more common than PNP for a number of reasons:

- 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



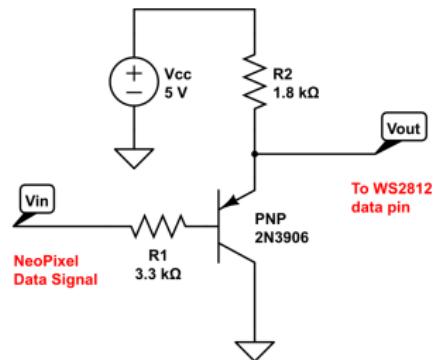
Data source: EETimes, A brief recap of popular logic standards (Mark Pearson, Maxim).

Or, what is wrong with the NeoPixels.



Emitter Follower - Saturation and Cutoff Mode

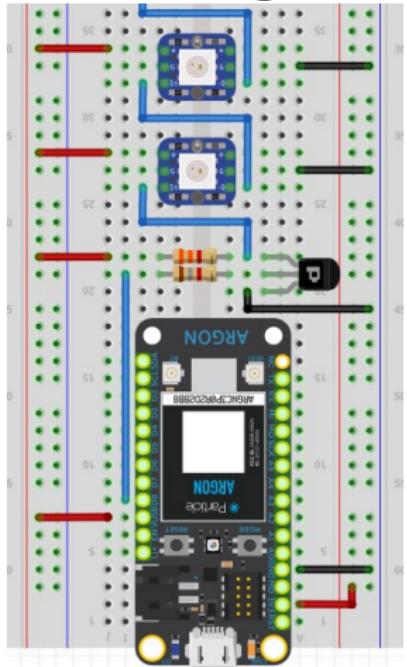
- NeoPixels are designed around 5V CMOS transistors.
 - $V_{IH} > 3.5V$
 - 3.3V Microcontroller
 - $V_{OH} = 3.3V$
- An Emitter Follower (i.e., a PNP transistor wired backwards) is a current amplifier, but will also produce a $V_{OUT} = 3.9V$.
- Alternatively, the first NeoPixel could be sacrificed by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



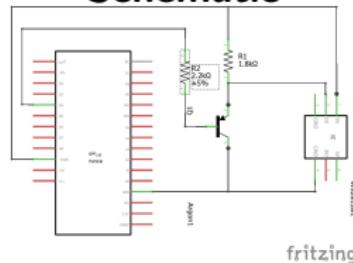


Emitter Follower Layout

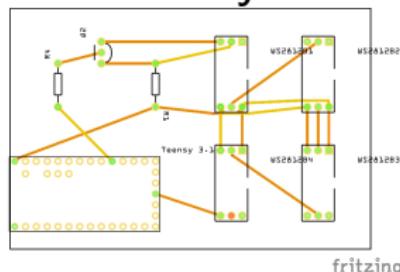
Fritzing



Schematic

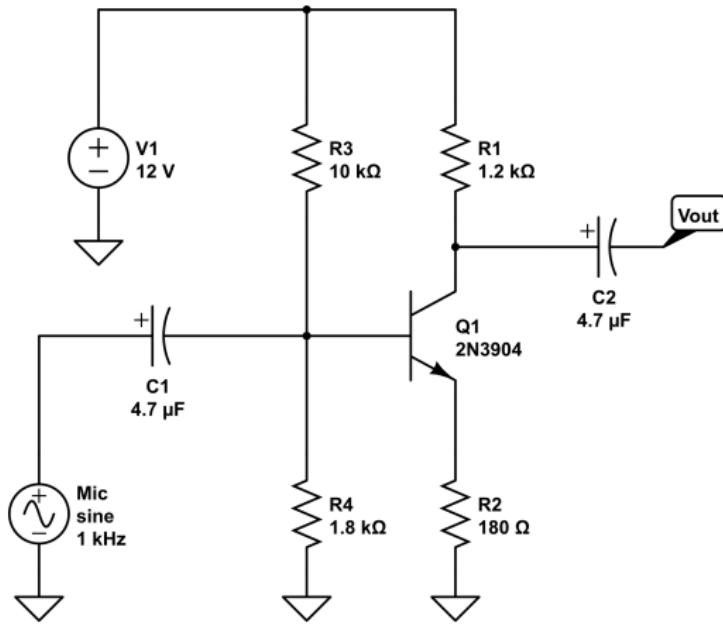


PCB Layout





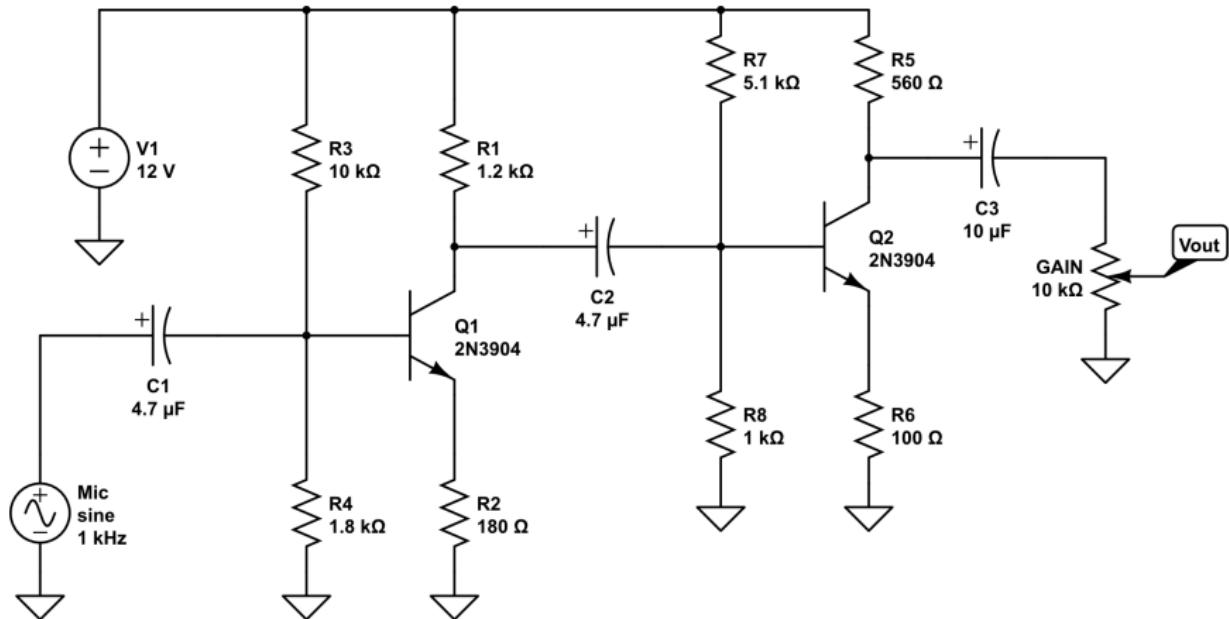
Typical NPN Pre-Amplifier Circuit



This is referred to as a Common Emitter amplifier as the Emitter ground is common to both the input and output voltage. The Common Emitter amplifies both voltage and current.

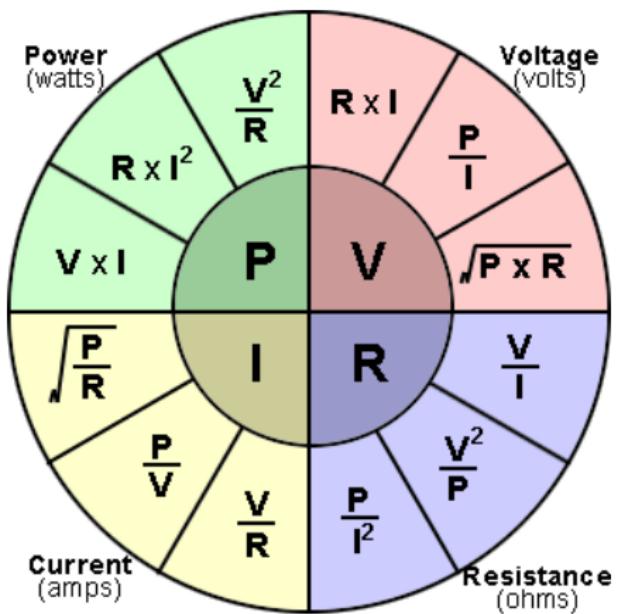


Two Stage Pre-Amplifier





Ohm's Law - Revisited





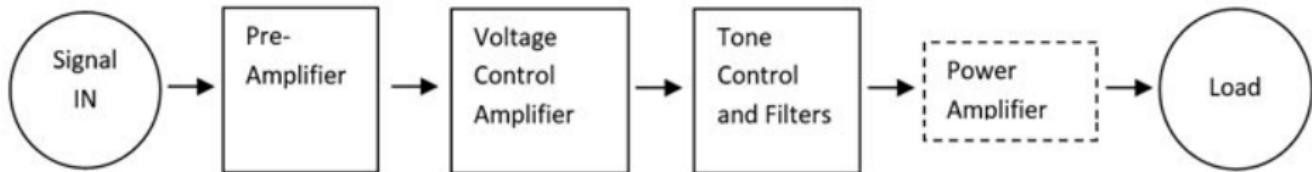
PreAmp vs PowerAmp

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.

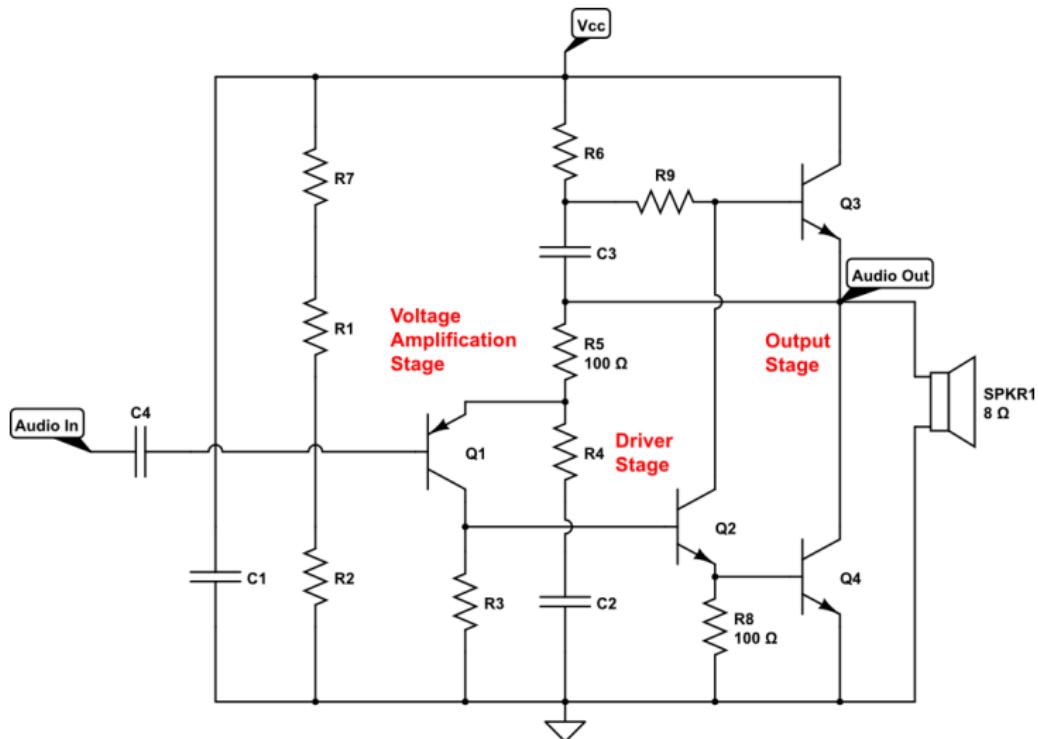
PowerAmp:

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





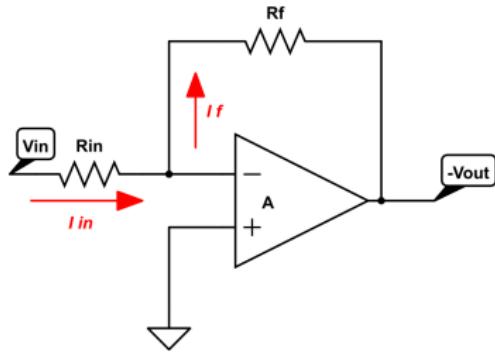
Power Amplifier



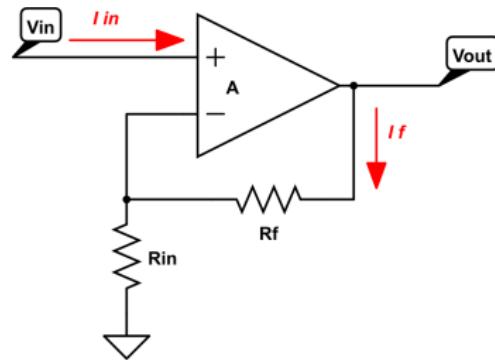


Op Amp Lesson

Inverting Op Amp



Non-inverting Op Amp



$$A = \frac{V_{out}}{V_{in}} = -\frac{R_f}{R_{in}}$$

$$A = \frac{V_{out}}{V_{in}} = 1 + \frac{R_f}{R_{in}}$$

Power the OpAmp with $V^+ = 12V$ and $V^- = -12V$ using TPS5430



Assignment: L10_Semiconductor

① L10_01_NeoPixel

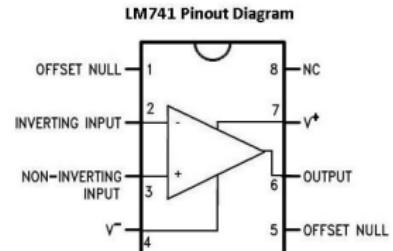
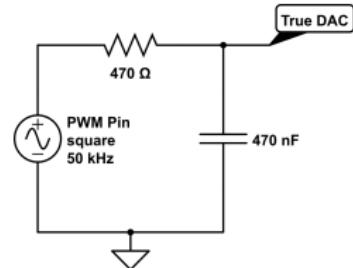
- Add a Emitter-Follower into your NeoPixel circuit to boost the pixel commands to 5V.

② L10_02_NPNAmp

- Using the DAC (the first resistor/capacitor) and code from L05_00_lowPass to create a sine wave.
- Amplify the signal using an NPN preamp.
- Measure the circuit at each node using the oscilloscope.

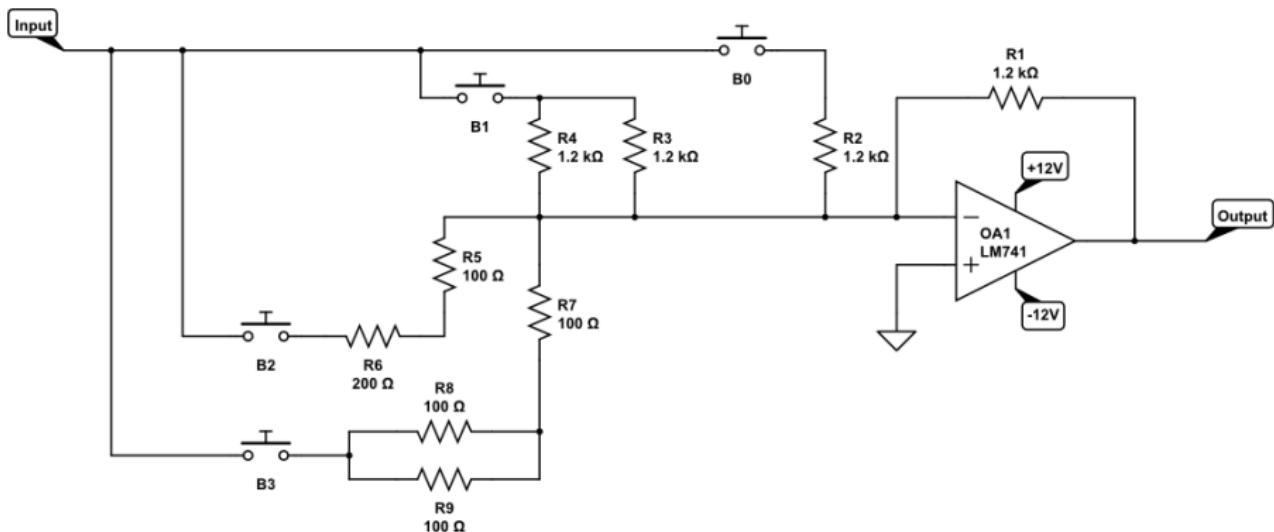
③ L10_03_OpAmp

- Replace the NPN preamp with an amplification circuit using the LM741 Op Amp. Use a potentiometer for R_{in} .





L10_04_MysteryCircuit

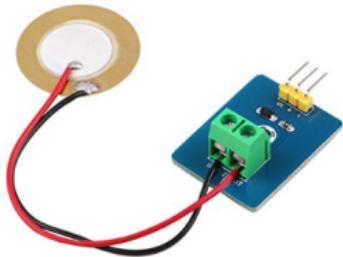


- Layout circuit in Fritzing
- Build and test circuit with input at Oscilloscope station
- Record output voltage for all combinations of buttons presses
- Figure out what it is doing and why it works.

Module 11 - Sensors



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 voltage (and thus the amount of pressure on the crystal) can be measured.



Assignment: Carnival Game



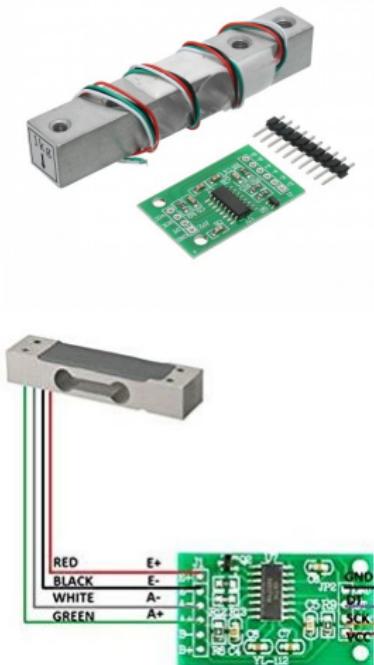
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

① L11_01_HighStriker

- Connect the piezo sensor and the NeoPixel tower to the Argon
- Each time the piezo is struck, find the maximum voltage generated.
- Light up the NeoPixel tower proportional to the force the piezo is struck with.



Load Cells

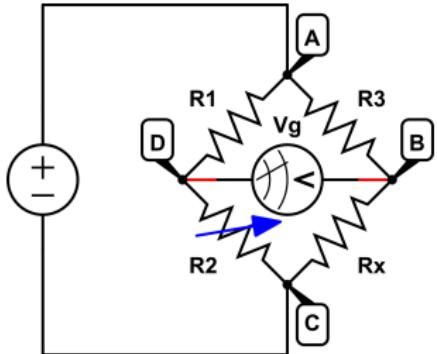


- ① A load cell is a force transducer. It converts a force such as tension, compression, pressure, or torque into an electrical signal that can be measured and standardized. As the force applied to the load cell increases, the electrical signal changes proportionally. The most common types of load cells used are hydraulic, pneumatic, and strain gauge.
- ② The 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).



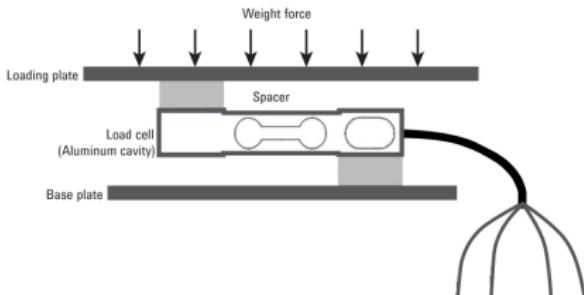


HX711A Library

```
1 // From the Command Palette install the HX711A library, that will give you HX711.h
2 #include "HX711.h"
3 HX711 myScale(DT,CLK);      // any two digital pins
4
5 const int CAL_FACTOR=1000; //changing value changes get_units units (lb, g, ton, etc.)
6 const int SAMPLES=10; //number of data points averaged when using get_units or get_value
7
8 float weight, rawData, calibration;
9 int offset;
10
11 void setup() {
12     myScale.set_scale();           // initialize loadcell
13     delay(5000);                // let the loadcell settle
14     myScale.tare();              // set the tare weight (or zero)
15     myScale.set_scale(CAL_FACTOR); //adjust when calibrating scale to desired units
16 }
17
18 void loop() {
19     // Using data from loadcell
20     weight = myScale.get_units(SAMPLES); // return weight in units set by set_scale();
21     delay(5000)                      // add a short wait between readings
22
23     // Other useful HX711 methods
24     rawData = myScale.get_value(SAMPLES); // returns raw loadcell reading minus offset
25     offset = myScale.get_offset();       // returns the offset set by tare();
26     calibration = myScale.get_scale();   // returns the cal_factor used by set_scale();
27 }
```



Assignment: L11_Sensor (Learn to Calibrate)



① L11_02_Scale

- Notebook: schematic
 - Fritzing diagram
 - Wire your circuit
 - Write the code
- Set initial CAL_FACTOR to 1000 and measure a known weight. (Note: one cup of water (in a paper cup) is approx. 244 g).
 - Adjust CAL_FACTOR until you get the expected measurement in grams.
 - Post data to Adafruit.io and/or ThingSpeak™
 - Optional: Send text via IFTTT.



More DataTypes - Strings, strings, and char[]

```
1 // A string (lowercase 's') is an array of characters
2 char lastName[7] = "Rashap";
3 char firstName[6] = {'B', 'R', 'I', 'A', 'N'};
4 char name[12];
5
6 //The "*" indicates a pointer, which we will learn about later
7 char *myName = "Brian";
8
9 // A String is a Class that holds a character array
10 String instructor = "BRIAN RASHAP";
11
12 void setup() {
13   Serial.begin();
14
15   Serial.printf("lastName = %s, %i\n", lastName, sizeof(lastName));
16   Serial.printf("firstName = %s, %i\n", firstName, sizeof(firstName));
17   Serial.printf("name = %s, %i\n", name, sizeof(name));
18   Serial.printf("myName = %s, %i\n", myName, sizeof(myName));
19
20   // We can not use %s for the variable instructor, why?
21 }
```

```
Serial monitor opened successfully:
lastName = Rashap, 7
firstName = BRIAN, 6
name = , 12
myName = Brian, 4
```



Too Much Time On My Hands

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

```
1 // Declare Global Variables in Header
2 String DateTime, TimeOnly;
3
4 void setup() {
5     Time.zone(-7);           // MST = -7, MDT = -6
6     Particle.syncTime();    // Sync time with Particle Cloud
7 }
8
9 void loop() {
10    DateTime = Time.timeStr();           //Current Date and Time from Particle Time class
11    TimeOnly = DateTime.substring(11,19); //Extract the Time from the DateTime String
12
13 // %s prints an array of char
14 // the .c_str() method converts a String to an array of char
15 Serial.printf("Date and time is %s\n",DateTime.c_str());
16 Serial.printf("Time is %s\n",TimeOnly.c_str());
17
18 delay(10000); //only loop every 10 seconds
19 }
```

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

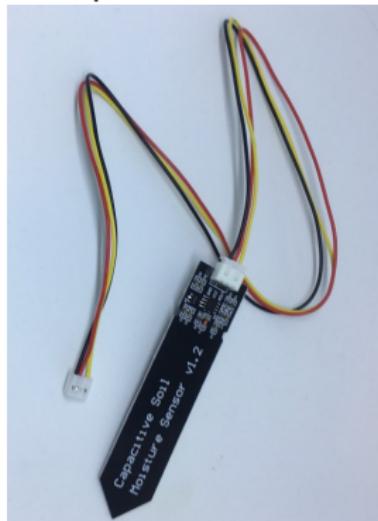


Soil Moisture Sensors

Resistive Sensor



Capacitive Sensor





Assignment: Moisture Probe



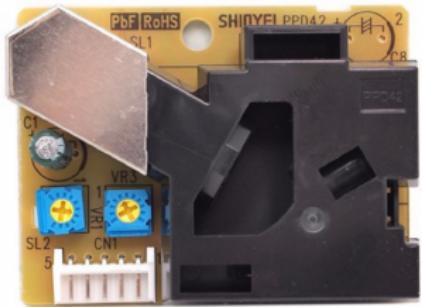
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

① L11_03_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.



Seeed Sensors



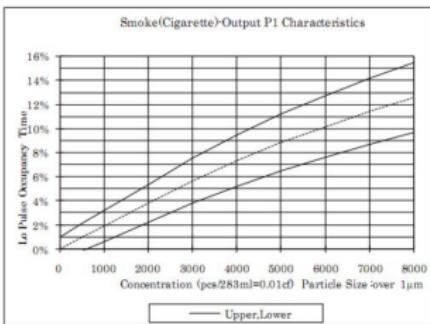
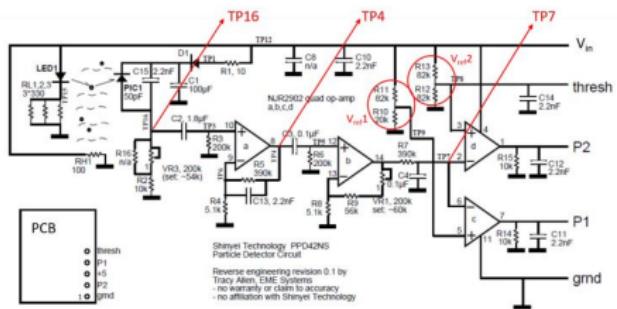
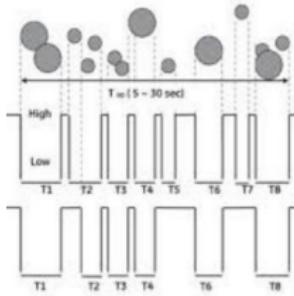
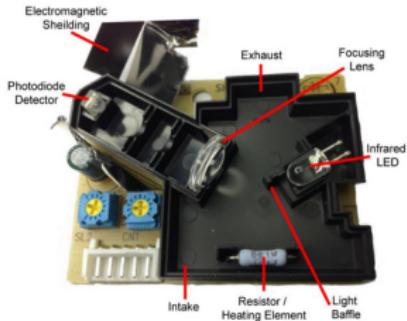
Seeed Grove - Dust Sensor



Seeed Grove - Air Quality
Sensor v1.3

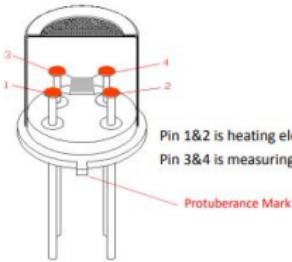


Shinyei PPD42NS low-cost dust sensor



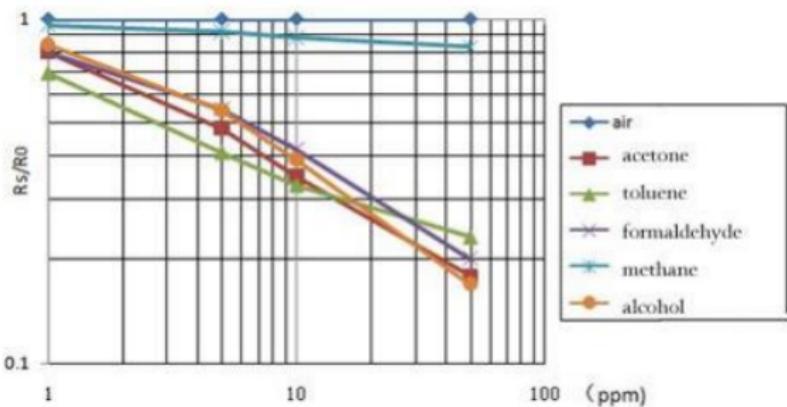


MP-503 Air Quality Sensor



Pin 1&2 is heating electrode,
Pin 3&4 is measuring electrode.

Protuberance Mark





Seeed Assignment



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

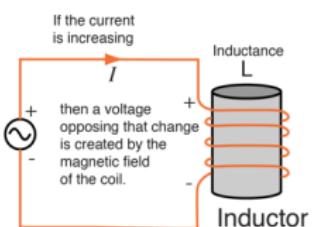
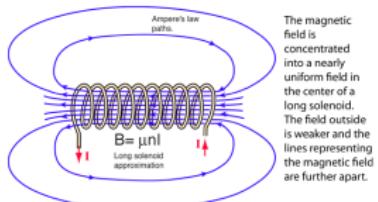
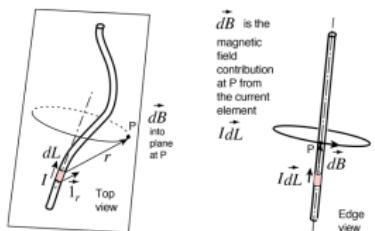
① L11_04_SeedSensors

- 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.
- Display air quality and particulate concentration to an Adafruit.io dashboard.

Midterm 2 - House Plant Watering System



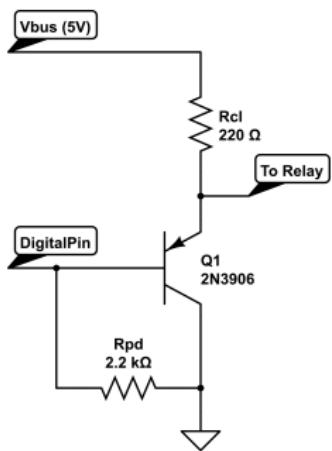
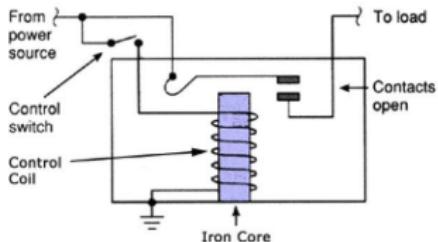
Inductors



- Current flowing in a wire produces a magnetic field (B) around the wire (from Ampere's Law).
- Wire wrapped into a coil produces a magnetic field that resembles a bar magnet through the center of the coil.
- Also, in a coil, this magnetic field produces an effect known as Inductance (L) that opposes changes in electric current.



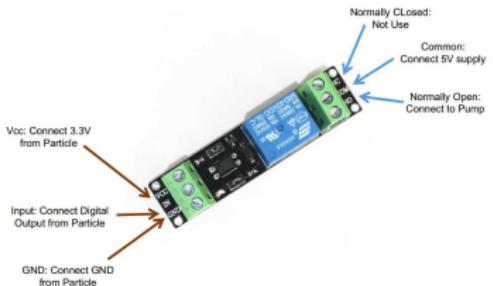
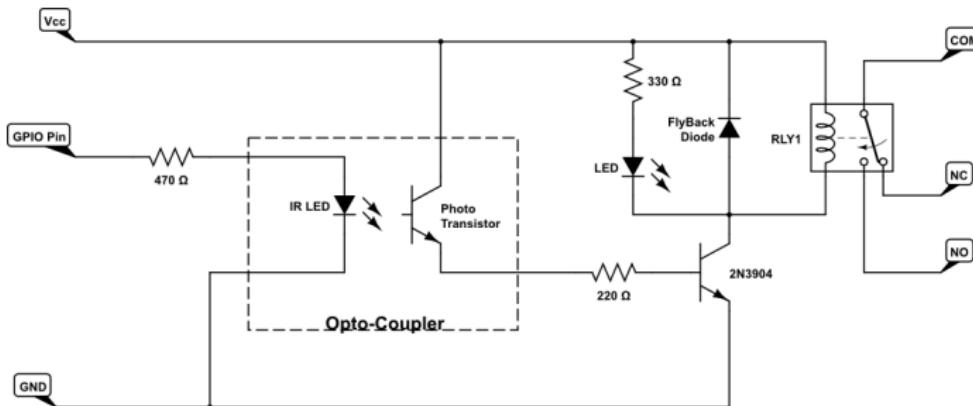
Relays



- 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. To provide sufficient current, use a current amplifying emitter follower to draw current directly from the USB connection (V_{BUS}).



Optocoupled Relay

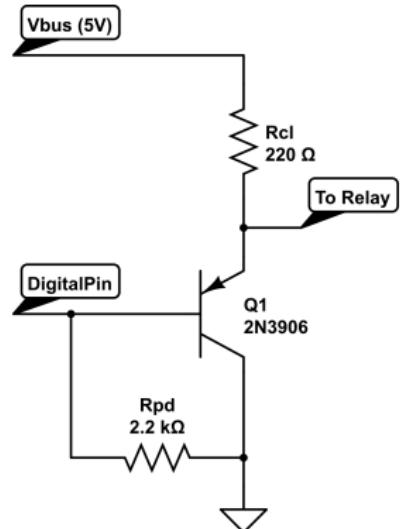


- Optocoupler isolates the relay load (which could be up to 240V) from the microcontroller electronics.



Smart Houseplant Watering System

Design



2N3906 Emitter Follower

① Components:

- 2N3906 Emitter Follower and Relay
- BME280 and SEEED sensors
- OLED Display

② Publish soil moisture and room environmental data to a new dashboard.

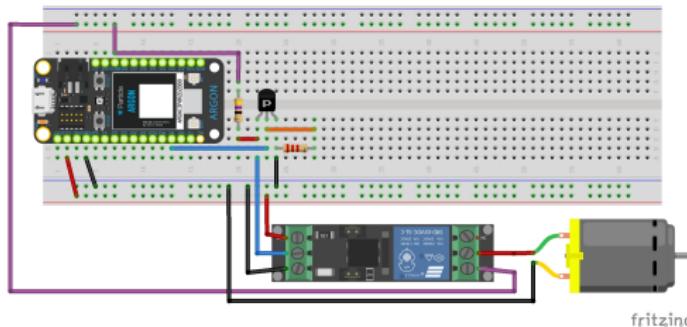
③ Automatically water your plant when the soil is too dry.

- Only turn on the pump for a very short period of time ($\frac{1}{2}$ sec).

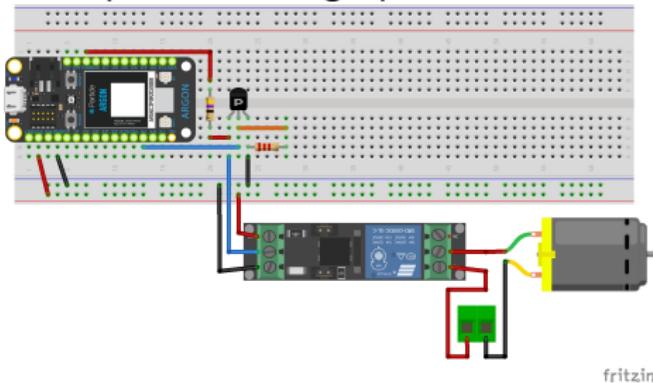
④ Integrate a button into your dashboard that manually waters the plant.



Relay and Pump Fritzing Diagram



If V_{BUS} doesn't provide enough power, add external supply.





Midterm 2 Continued

① Integrate the entire system

- Create a user friendly Adafruit.io dashboard
- Buy and/or create a structure to hold the plant, pump, and sensors.
- Integration of SMS/email messaging using Zapier (following slides).
- Video demo your Plant Watering System

② Add to your portfolio

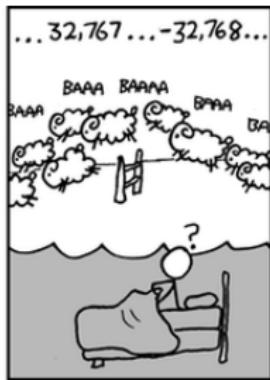
- Add the project to your Hackster.io feed.
- Create your own Github repository with a copy of your final project folder.

③ Class presentation - hackster, video demo, dashboard

Module 12 - Memory: Bit, Bites, and Memory



Counting Sheep





Negative Numbers

Question: How are negative numbers represented in binary?

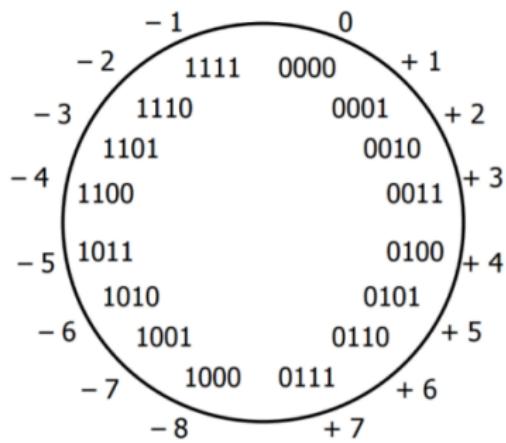
Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short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

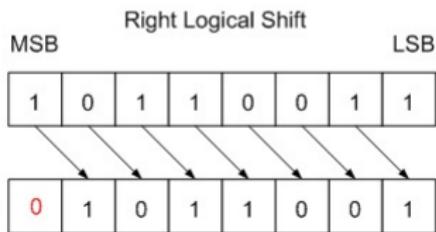
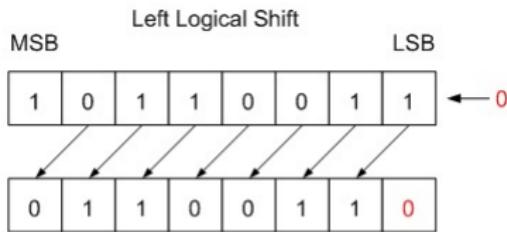
The following table lists the Bitwise operators supported by C. Assume variable 'A' holds 60 and variable 'B' holds 13, then –

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.	$(\sim A) = \sim(60)$, i.e., 1100 0011
<<	Binary Left Shift Operator. The left operand's 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 operand's value is moved right by the number of bits specified by the right operand.	$A >> 2 = 15$ i.e., 0000 1111

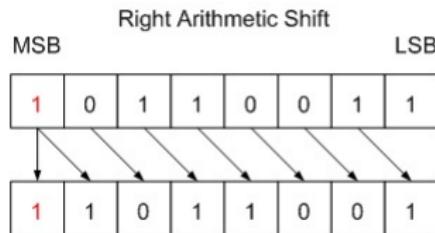
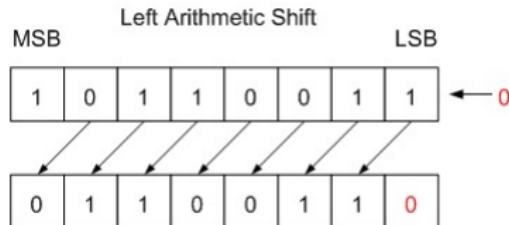


Bit Shifting

Logical Bit Shift



Arithmetic Bit Shift

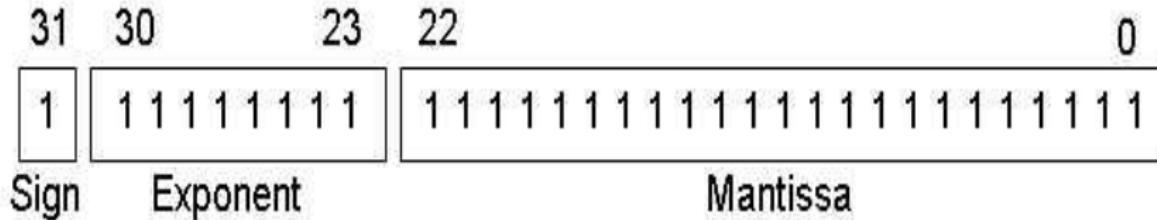


Whether the logical or arithmetic right shift is used depends on the datatype of the variable (unsigned or signed).



BONUS: But what about Floating Point

IEEE-754 Floating Point



Example: In scientific notation: $-36382.36 = -3.638236 \times 10^4$
 With binary exponential equals $-1 \times 1.1103014945983887 \times 2^{15}$

- Sign: Negative = 1
- Exponent: 15 = 10001110 (with an exponent bias of 10000000)
- Mantissa: 11103014945983887 = 000011100001111001011100

Floating point representation is: 11000111000011100001111001011100



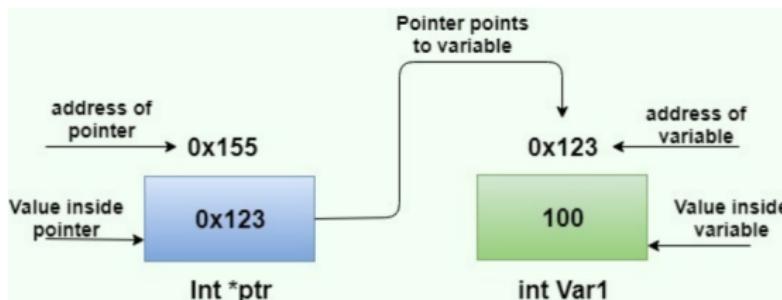
Electrically Erasable Programmable Read Only Memory

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.

```
1 len = EEPROM.length(); //available EEPROM bytes
2 // Argons have 4096 bytes of emulated EEPROM.
3 // Addresses 0x0000 through 0xFFFF
4
5 addr = 0x00AE;      //addr between 0 and len-1
6
7 val = 0x45;
8 EEPROM.write(addr, val);
9
10 value = EEPROM.read(addr);
```



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

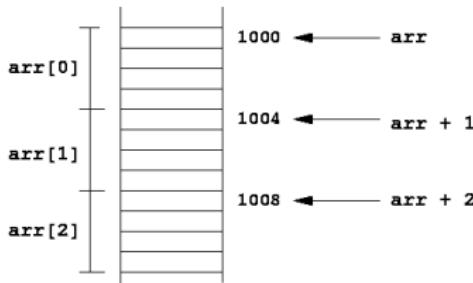
```
1 int data = 13;
2 int data2;
3 int *ptr;
4
5 void setup() {
6   Serial.begin(9600);
7   delay(1000);
8   ptr = &data;           //point ptr to the memory location of data
9   data2 = *ptr;          //set data2 to value of data (13)
10
11 // Print the Value and Address of the Variables
12 Serial.printf("Variable      Value      Address \n");
13 Serial.printf("  data        %i        0x%X  \n",data, &data);
14 Serial.printf("  ptr         0x%X        0x%X  \n",ptr, &ptr);
15 Serial.printf("  data2       %i        0x%X  \n",data2,&data2);
16 }
```

Serial monitor opened successfully:

Variable	Value	Address
data	13	0x2003E380
ptr	0x2003E380	0x2003E3F4
data2	13	0x2003E3F0



Pointers and Arrays



```
1 int arr[] = {100, 200, 300};  
2  
3 void loop() {  
4     // Compiler converts below to *(arr + 2).  
5     Serial.printf("%i \n", arr[2]);  
6  
7     // So below also works.  
8     Serial.printf("%i \n", *(arr + 2));  
9 }
```

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



Finding Average of an Array

```
1 // This function finds the average of an array.  
2 // The array is passed to it as a pointer.  
3  
4 float getAverage(int *array ,int size) {  
5     int j;  
6     float total=0;  
7     for(j=0;j<size;j++) {  
8         total += array[j];  
9     }  
10    return total/size;  
11 }
```



Finding Average of Arrays in Action

```
1 int xArray[4], yArray[256];
2 int *pointerX, *pointerY;
3 float average;
4 int i, sizeX, sizeY;
5
6 void setup() {
7     pointerX=&xArray[0];
8     sizeX=sizeof(xArray)/4;
9     pointerY=&yArray[0];
10    sizeY=sizeof(yArray)/4;
11    for(i=0;i<sizeX;i++) {
12        xArray[i] = random(0,255);
13    }
14    for(i=0;i<sizeY;i++) {
15        yArray[i] = random(256,512);
16    }
17    average = getAverage(pointerX, sizeX);
18    average = getAverage(pointerY, sizeY);
19 }
```

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



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.

```
1 void setup() {
2     x = 4;
3     y = 2;
4 }
5
6 void loop() {
7     swap(&x, &y);
8     Serial.printf("%i%i\n", x, y);
9     delay(1000);
10}
11
12 void swap(int *x, int *y) {
13     int temp;
14
15     temp = *x;
16     *x = *y;
17     *y = temp;
18 }
```



memcpy(), (char *), and strtol()

```
1 int color;
2 byte data[] = {0x23,0x42,0x41,0x34,0x32,0x35,0x44,0x39,0x35};
3 byte buf[6];
4
5
6 /* memcpy() - copy from specific memory locations to new locations
7 *   memcpy(to, from, size);
8 *       to -> pointer to starting address of where to copy to
9 *       from -> pointer to starting address of where to copy from
10 *      size -> number of bbytes to copy
11 */
12
13 memcpy(buf, &data[1],6);      copy bytes 1 through 6 and place in buf
14
15 /* (char *) - typecasting a data type to a char-type pointer */
16
17 Serial.printf("Converting the data array to ascii symbols returns %s,\n", (char *)data);
18
19
20 /* strtol() - string to long - similar to atoi()
21 *   strtol(charString, end, base)
22 *       charString -> string that contains number to be converted
23 *       end -> character to end conversion on (set to NULL)
24 *       base -> base of integer (16 for hex)
25 */
26
27 color = strtol((char *)buf,NULL,16); // convert string to int (hex)
```



EXAMPLE: Adafruit MQTT Subscribe - Color Picker

Pick a Color



#fa7802

July 14th 2021, 11:23:46AM

Color Data

Date/Time	User	Action	Color
2021/07/14 10:35AM	Default	ColorSend	#1d31e5
2021/07/14 10:36AM	Default	ColorSend	#e51d3f
2021/07/14 11:19AM	Default	ColorSend	#3fe51d
2021/07/14 11:19AM	Default	ColorSend	#3a1de5
2021/07/14 11:20AM	Default	ColorSend	#b826fb
2021/07/14 11:22AM	Default	ColorSend	#f3fb26
2021/07/14 11:23AM	Default	ColorSend	#fa7802

```
1 int color;
2 byte buf[6];
3
4 Adafruit_MQTT_Subscribe *subscription;
5 while ((subscription = mqtt.readSubscription(1000))) {
6     if (subscription == &mqttColor) {
7         Serial.printf("Received from Adafruit: %s \n", (char *)mqttColor.lastread);
8         memcpy(buf, &mqttColor.lastread[1], 6);           //strip off the '#'
9         Serial.printf("Buffer: %s \n", (char *)buf);
10        color = strtol((char *)buf, NULL, 16);           // convert string to int (hex)
11        Serial.printf("Buffer: 0x%02X \n", color);
12    }
13 }
```



L12_Memory Assignments



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

① L12_01_ColorPicker

- Create a feed and dashboard on Adafruit.io using the Color Picker block.
- Subscribe to your Color Picker feed and convert the lastRead() to a integer (hex)
- Light up your NeoPixel ring the received color.
- Using pointers create a function to convert the hex color into individual R,G,B components using Bit Shifting and AND.
- From void loop, store the components of the color as bytes in the Argon's EEPROM.

② L12_02_RetrieveShow

- Retrieve the color from EEPROM memory.
- Convert to a hex color code (for example 0xABCDFF)
- Display the color on the NeoPixel ring using setPixelColor(n,hexColor)



Useful Properties: Identity Element

An identity element is a special type of element of a set with respect to a binary operation on that set, which leaves any element of the set unchanged when combined with it.

Addition:

$$x + 0 = x \quad (1)$$

Multiplication:

$$x * 1 = x \quad (2)$$

Bitwise AND:

$$x \& 1 = x \quad (3)$$

Bitwise OR:

$$x | 0 = x \quad (4)$$



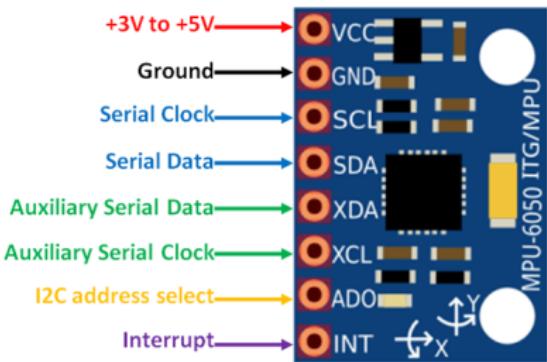
Added Bonus: Array of Functions via Pointers

```
1 //Array of pointers to each of the functions
2 int (* funky[4])(int x, int y) = {add,sub,mult,divi};
3
4 int a,b,answer,i;
5
6 void setup() {
7     Serial.begin(9600);
8 }
9
10 void loop() {
11     a = random(0,100);
12     b = random(0,100);
13     for(i=0;i<4;i++) {
14         answer = funky[i](a,b);
15         Serial.printf("For function %i: a = %i and b = %i equals %i \n",i,a,b,answer);
16         delay(250);
17     }
18     Serial.printf("\n\n\n");
19     delay(3000);
20 }
21
22 // The Functions
23 int add(int x,int y) {return x+y;}
24
25 int sub(int x,int y) {return x-y;}
26
27 int mult(int x,int y) {return x*y;}
28
29 int divi(int x,int y) {return x/y;}
```

Module 13 - Motion



MPU6050 Accelerometer

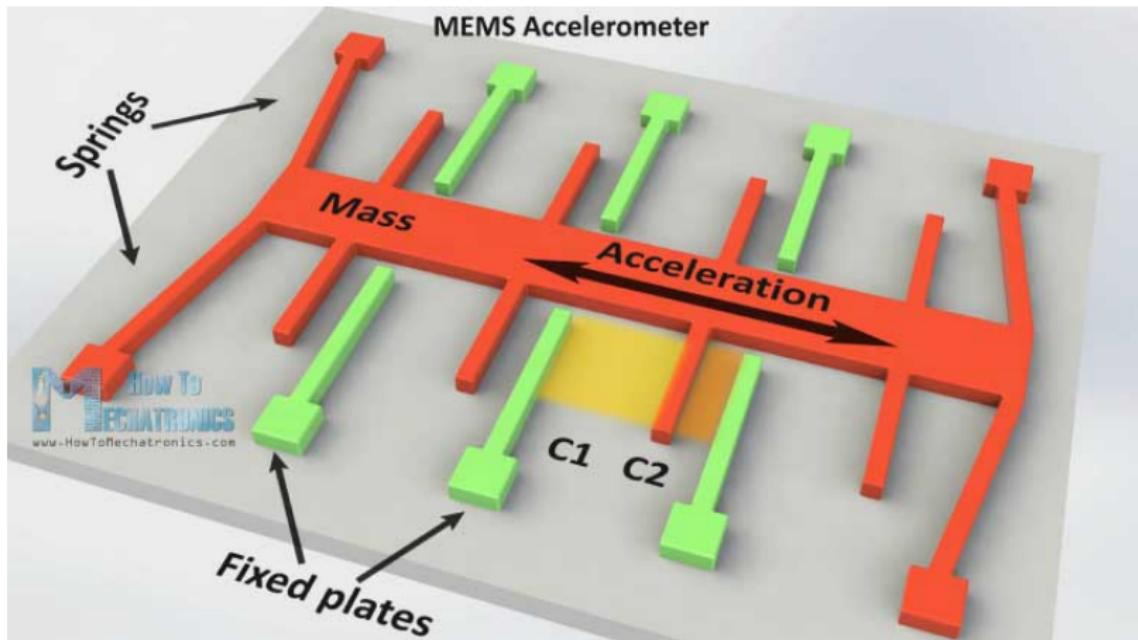


- Data Output - 16 Bit
- Gyros range: $\pm 250 \ 500 \ 1000 \ 2000 \ ^\circ/\text{s}$
- Accel range: $\pm 2 \ \pm 4 \ \pm 8 \ \pm 16 \text{g}$

- The interrupt pin notifies the MPU about available data. To reduce power consumption, the processor can go into sleep mode and the interrupt can be used to wake up the processor.
- 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





DIP Switches and Register Maps



Remember: serial usb setup-done

Table 18: Memory map

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
hum_lsb	0xFE				hum_lsb<7:0>					0x00
hum_msb	0xFD				hum_msb<7:0>					0x80
temp_xlsb	0xFC		temp_xlsb<7:4>			0	0	0	0	0x00
temp_lsb	0xFB				temp_lsb<7:0>					0x00
temp_msb	0xFA				temp_msb<7:0>					0x80
press_xlsb	0xF9		press_xlsb<7:4>			0	0	0	0	0x00
press_lsb	0xF8				press_lsb<7:0>					0x00
press_msb	0xF7				press_msb<7:0>					0x80
config	0xF5	t_sb[2:0]			filter[2:0]			spi3w_en[0]		0x00
ctrl_meas	0xF4	osrs_t[2:0]			osrs_p[2:0]		mode[1:0]			0x00
status	0xF3				measuring[0]		im_update[0]			0x00
ctrl_hum	0xF2						osrs_h[2:0]			0x00
calib26.calib41	0xE1..0xF0				calibration data					individual
reset	0xE0				reset[7:0]					0x00
id	0xD0				chip_id[7:0]					0x60
calib00..calib25	0x88..0xA1				calibration data					individual

Registers:	Reserved registers	Calibration data	Control registers	Data registers	Status registers	Chip ID	Reset
Type:	do not change	read only	read / write	read only	read only	read only	write only



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 7.5×10^{18} grains of sand on Earth. A long long integer and floating point numbers are larger than this.



Initializing MPU6050

```
1 // Initialize the MPU in the void setup()
2 void setup() {
3     // Begin I2C communications
4     Wire.begin();
5
6     // Begin transmission to MPU-6050
7     Wire.beginTransmission(MPU_ADDR);
8
9     // Select and write to PWR_MGMT1 register
10    Wire.write(0x6B);
11    Wire.write(0x00); // wakes up MPU-6050
12
13    // End transmission and close connection
14    Wire.endTransmission(true);
15 }
```



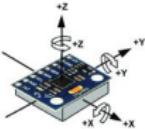
Reading Acceleration Data from the MPU-6050

```
1 // Declare variables
2 byte accel_x_h, accel_x_l;      //variables to store the individual bytes
3 int16_t accel_x;                //variable to store the x-acceleration
4
5 // Set the "pointer" to the 0x3B memory location of the MPU and wait for data
6 Wire.beginTransmission(MPU_ADDR);
7 Wire.write(0x3B); // starting with register 0x3B
8 Wire.endTransmission(false); // keep active.
9
10 // Request and then read 2 bytes
11 // Syntax:
12 //     Wire.requestFrom(I2C_addr, quantity, stop);
13 //     Wire.read(); //repeat this for each byte to be read
14
15 Wire.requestFrom(MPU_ADDR, 2, true);
16 accel_x_h = Wire.read(); // x accel MSB
17 accel_x_l = Wire.read(); // x accel LSB
18
19 accel_x = accel_x_h << 8 | accel_x_l;      // what happens if declared int instead?
20 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: L13_Motion



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

① L13_01_MP6050

- Read values from the memory addresses associated with X, Y, and Z acceleration.
- Convert the returned acceleration values to standard gravity units (e.g. when flat on the table, $a_z = -1G$).

② L13_02_AutoRotate

- Display date and time on an OLED display.
- Use accel values to auto-rotate the OLED.

③ Extra

- Modify L16_01_MP6050 to be able to modify range/sensitivity with a button or encoder.



SOH CAH TOA

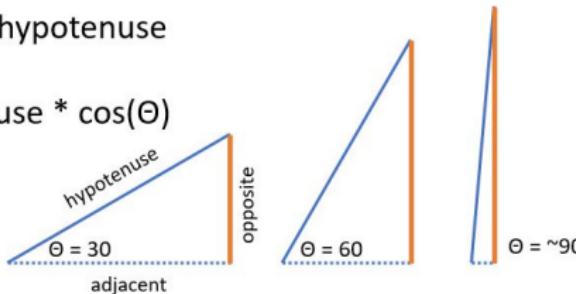
- $\sin = \text{opposite over hypotenuse}$
- $\cos = \text{adjacent over hypotenuse}$
- $\tan = \text{opposite over adjacent}$

$$\cos(\Theta) = \text{adjacent} / \text{hypotenuse}$$

or

$$\text{Adjacent} = \text{hypotenuse} * \cos(\Theta)$$

$$\Theta = 0$$



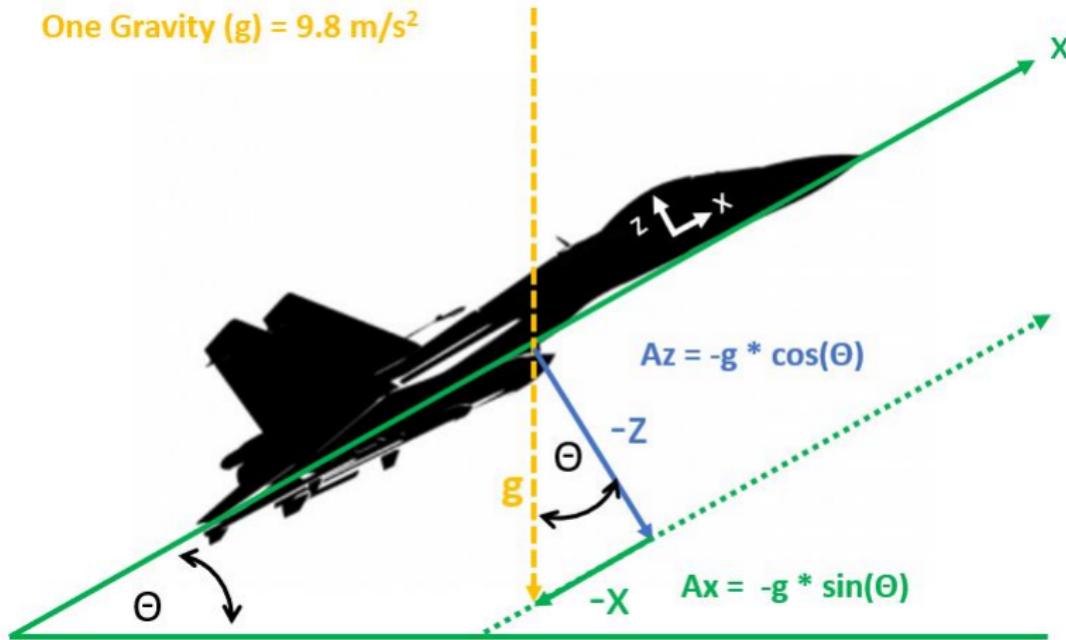
$$\sin(\Theta) = \text{opposite} / \text{hypotenuse}$$

or

$$\text{opposite} = \text{hypotenuse} * \sin(\Theta)$$

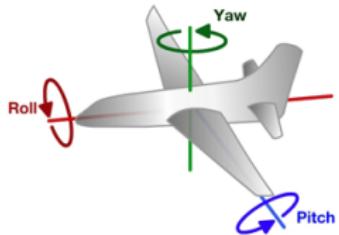


Gravity and Orientation





Assignment: L13_Motion



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

① L13_03_Airplane

- Calculate pitch $\theta = -\arcsin(a_x)$.
- Calculate roll $\phi = \arctan2(a_y, a_z)$.

② L13_04_Shock

- Store a_{tot} in an array every 10ms for 5s.
- Find and print the max value from the array.
- Repeat.

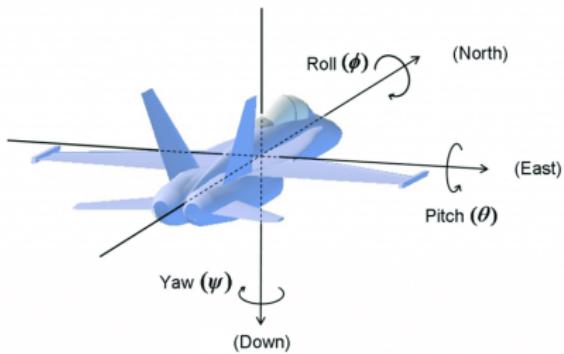
③ Extra

- Improve L16_03_Airplane with equations
from next slide

Recall, that trigonometric functions return radians which needs to be converted to degrees. See the Unit Circle slide in L02_HelloLED.



Pitch and Roll Improved



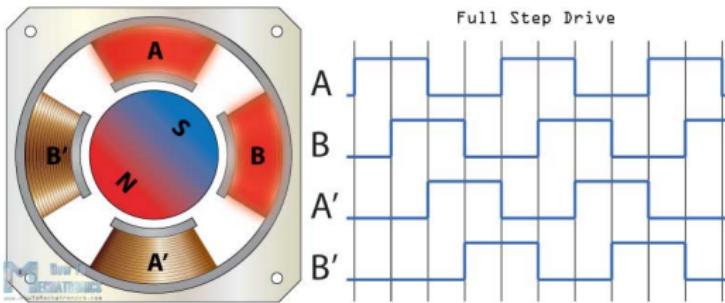
$$\text{Pitch}(\Theta) = \arctan\left(\frac{a_x}{\sqrt{a_y^2+a_z^2}}\right)$$

$$\text{Roll } (\Phi) = \arctan\left(\frac{a_y}{\sqrt{a_x^2+a_z^2}}\right)$$

$$\text{Yaw } (\Psi) = \arctan\left(\frac{\sqrt{a_x^2+a_y^2}}{a_z}\right)$$



Stepper Motors



28BYJ Stepper Motor

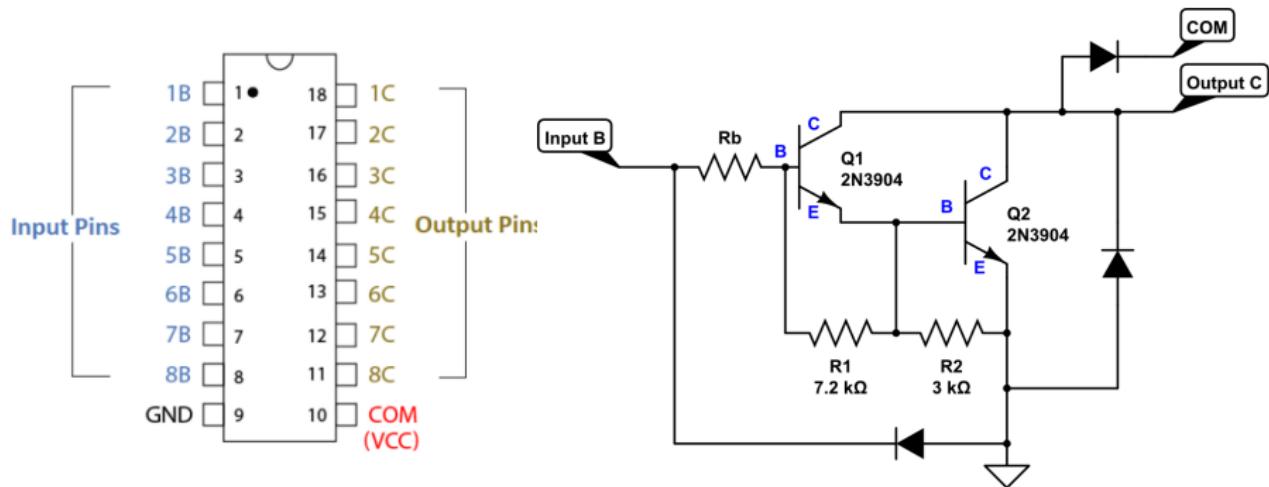
- 2048 steps per revolution
 - 32 steps per rotor revolution
 - Gear ratio 1:64
- Capable of 10-15 RPM (at 5V)
- ULN2003 Darlington Array driver

Stepper.h

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



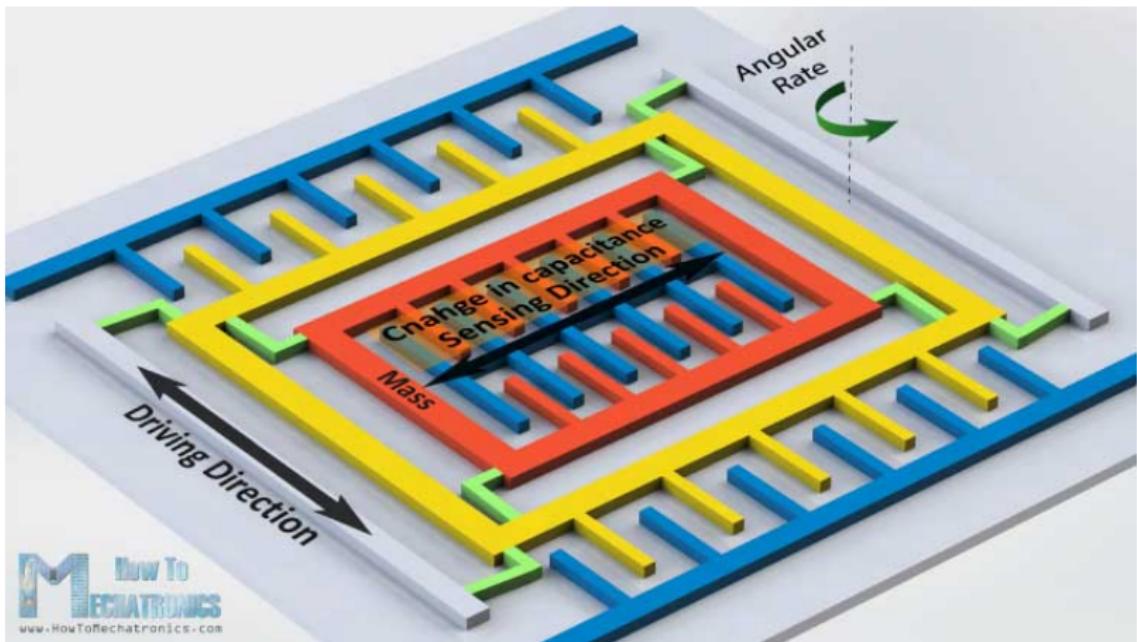
ULN2003 Darlington Array



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



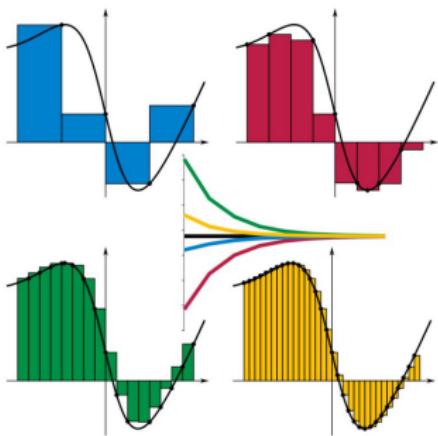
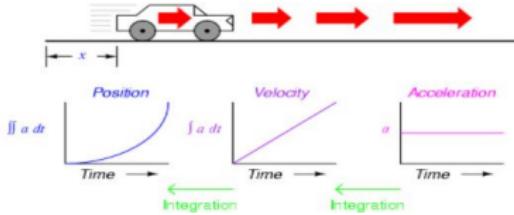
Gyroscopes



The gyroscope measures angular velocity (degrees per second).



Acceleration, Velocity, and Position



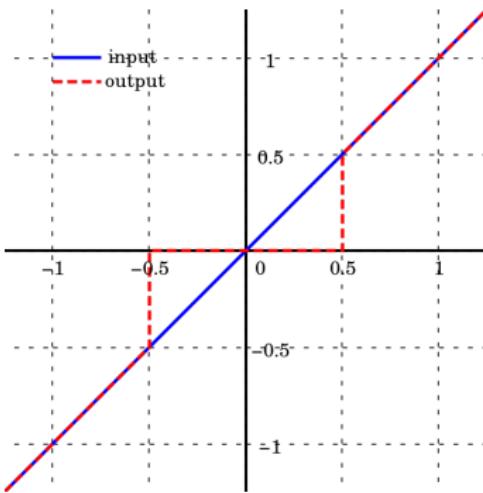
The Riemann Sum method can be used to "integrate" acceleration to velocity and velocity to position.

- Gyroscope output is angular velocity: ω ($\frac{\text{degrees}}{\text{second}}$)
- To get change in angular position ($\Delta\theta$), multiple each angular velocity by the time step: $\Delta\theta = \omega * \Delta t$
- Use the Riemann Sum to get the resulting angular position

$$\theta = \sum_{t=0}^{t=T} (\omega * \Delta t)$$



Deadband



A deadband is a band of input values that in a control system create an output that is zero.



Assignment: L13_Motion



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

① L3_05_Stepper

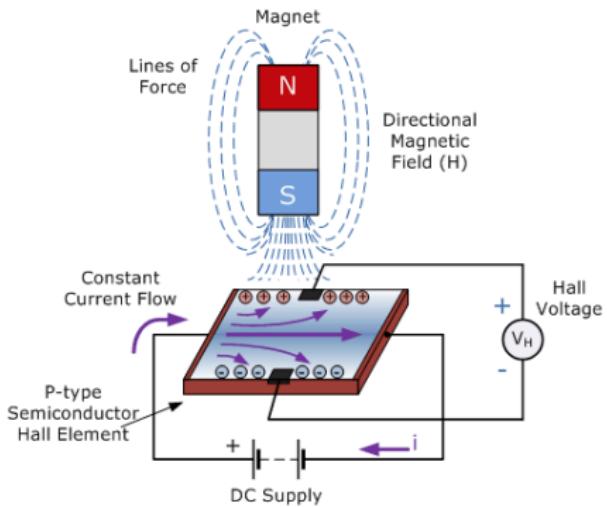
- Make a gear/pointer to show motor position (cardboard, laser wood, 3D print)
- Wire the stepper motor noting the order: IN1, IN3, IN2, IN4.
- Move the motor 2 rotations clockwise, pause, 1 rotation counter-clockwise, repeat.

② L13_06_DriveByWire

- Connect the MPU-6050 to your system.
- Obtain the z-axis rotation from the appropriate register on the MPU-6050. Convert to angular rotation ($^{\circ}$ per sec).
- Calculate the angular position ($^{\circ}$) of the gyroscope.
- Have the stepper motor track movement in the gyroscope.



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.



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.

```
1 pinMode(pin, INPUT); \\ can also use INPUT_PULLUP or INPUT_PULLDOWN  
2 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 value goes from low to high,
- FALLING for when the pin value goes from high to low.



Software Timers

There are also software timer interrupts. The Argon can manage up to 10 timers simultaneously.

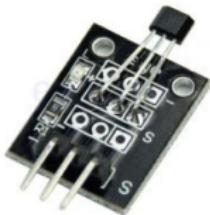
```
1 Timer timer(1000, printEverySecond);
2
3 void setup() {
4     Serial.begin(9600);
5     timer.start();
6 }
7
8 void printEverySecond() {
9     static int count = 0;
10    Serial.printf("count=%i, time = %u ms \n", count, millis());
11    count++;
12 }
```

Note:

- The timer callback is similar to an interrupt - it shouldn't block.
- Multiple timers are serviced sequentially when several timers trigger simultaneously, thus requiring special consideration when writing callback functions.



Assignment: L13_Motion



① L13_07_Alarm

- Connect Hall Effect Sensor, button, and Neopixel to simulate an alarm system.
- Use the button to enable / disable alarm.
- When armed:
 - Neopixel is GREEN when magnet detected.
 - Blinking RED when magnet not detected.

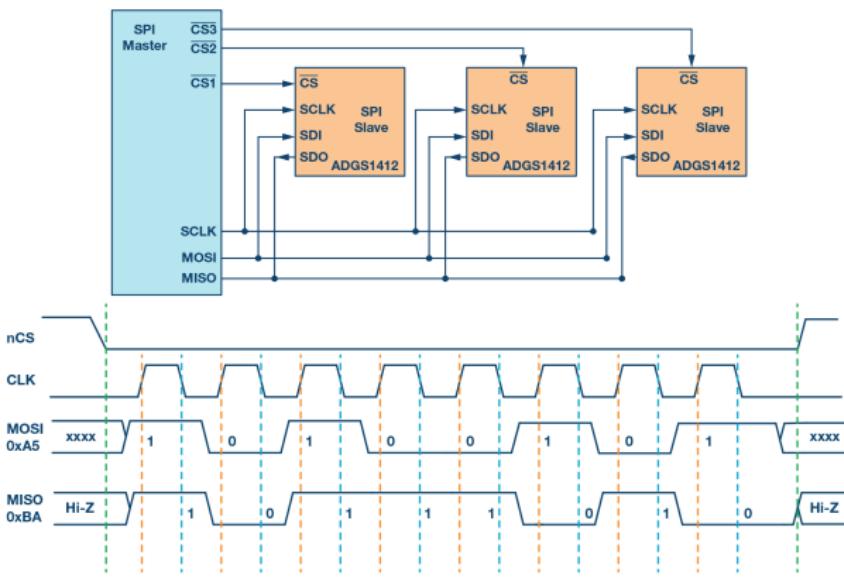
② L13_08_RPM

- Notebook:
 - schematic
 - Fritzing diagram
 - Wire your circuit
 - Write the code
- Place magnet on shaft of lathe.
 - Create an interrupt function that returns the time per rotation using the Hall Effect Sensor.
 - Convert this time to rotations per minute.
 - Display on Adafruit.io databoard.
 - EXTRA:
 - Create a speedometer using a servo motor.
(The Servo class natively available).
 - Measure RPM on other equipment at FUSE.

Module 14 - DataXfer



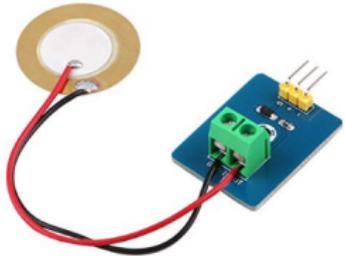
SPI Revisited



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



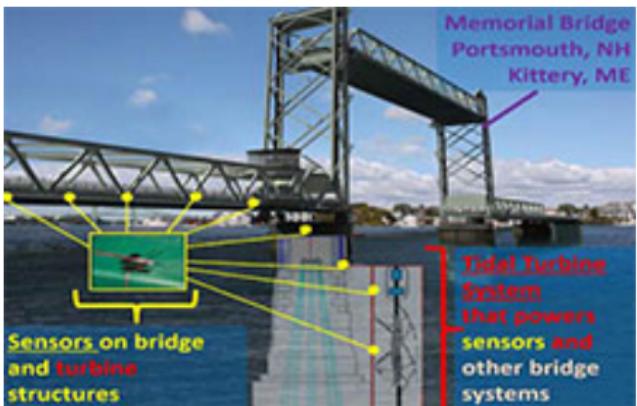
Piezoelectric Elements - Revisited



- 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.



Structural Engineering Sensors





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 does not deliver performance, reliability and scalability afforded by modern file systems (such as NTFS or exFAT).

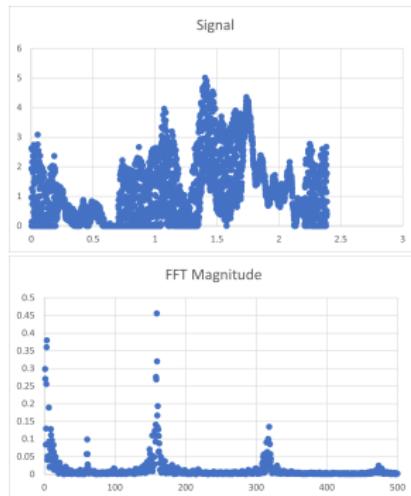
Note: FAT file name follows a 8.3 format (e.g., ABCDEFGH.txt). We will be appending two digits, so our base name can be up to 6 characters (e.g. FILE_BASE_NAME = "mydata" → mydata42.csv).



Galaxy S9+ Vibration Analysis

Using the FFT Tutorial in Class Slides

	A	B	C	D	E
1	TimeStamp	Signal	Frequency	FFT Magnitude	Complex FFT
2	0.00061	2.62	0.4209321	2.000009766	4096.02
3	0.00119	1.93	0.8418642	0.836477508	-552.77950380473+1621.47055713326i
4	0.00177	0.96	1.2627963	0.298364661	-414.982558995766-448.522671528173i
5	0.00235	0.13	1.6837284	0.270681692	353.443826952842-427.069259698417i
6	0.00293	0	2.1046606	0.083873335	-72.9068609990069+155.532672030055i
7	0.003509	0	2.5255927	0.129856545	252.456289478309+83.6253876318606i
8	0.004089	0	2.9465248	0.255636434	-516.88842919695+83.210943362584i
9	0.004669	0	3.3674569	0.360085774	423.28074046682-603.882664071708i
10	0.005249	0.14	3.788389	0.379410535	88.80273300538+771.941714466294i
11	0.005829	1.26	4.2093211	0.04066392	-41.3210095359081-72.3054897410451i
12	0.006409	2.16	4.6302532	0.051383144	94.6397062012862-46.0135075977235i
13	0.006988	2.57	5.0511853	0.084507321	25.9352596801745-171.116717569232i
14	0.007568	1.89	5.4721175	0.041174283	36.0724144460193-76.2199128809511i
15	0.008148	0.73	5.8930496	0.04976769	-73.9953870941929-70.094447984849i



The Galaxy S9 vibrates at 159.11 Hz



Assignment: L14_DataXfer



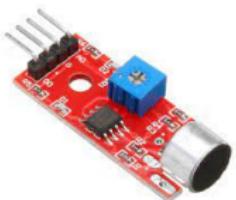
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

① L14_01_Vibration

- Connect the piezo sensor, a button, and a μ SD module to your Argon.
- Each time button is pressed, execute a loop 4096 times:
 - Every 500μ sec, collect piezoelectric data (without using a delay).
 - Save the piezo 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 the lathe, cell phone 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.).



Assignment: L14_DataXfer EXTRA



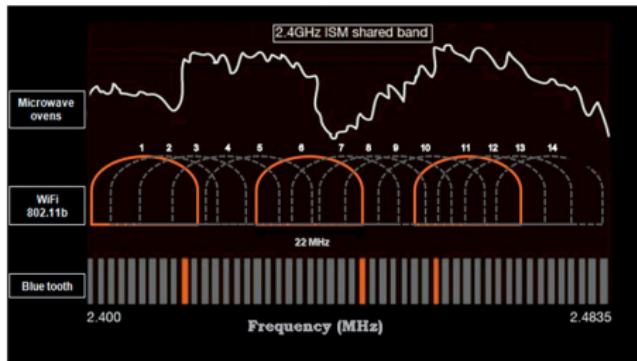
- Notebook: schematic
- Fritzing diagram
- Wire your circuit
- Write the code

① L14_01_Vibration

- Borrow a microphone.
- Install in place of the piezo sensor.
- Use Physics Toolbox Sensor Suite - Tone Generator to create a tone of a specific frequency.
- Record/save with the Argon, and create an FFT



Bluetooth



- The Bluetooth protocol operates at 2.4GHz in the same unlicensed ISM frequency band where RF protocols like ZigBee and WiFi also exist.
- Bluetooth networks (commonly referred to as piconets) use a master/slave model to control when and where devices can send data. In this model, a single master device can be connected to up to seven different slave devices. Any slave device in the piconet can only be connected to a single master.



Bluetooth - Generic Access Profile

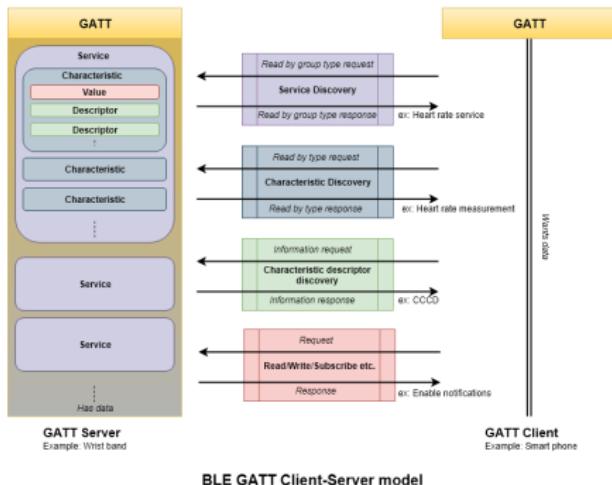


- The Generic Access Profile (GAP) controls connections and advertising in Bluetooth. GAP is what makes your device visible to the outside world, and determines how two devices can (or can't) interact with each other.
- GAP defines various roles for devices, but the two key concepts to keep in mind are Central devices and Peripheral devices.
 - Peripheral devices are small, low power, resource constrained devices that can connect to a much more powerful central device. Peripheral devices are things like a heart rate monitor, a BLE enabled proximity tag, etc.
 - Central devices are usually the mobile phone or tablet that you connect to with far more processing power and memory.



Bluetooth - Generic Attribute Profile (GATT)

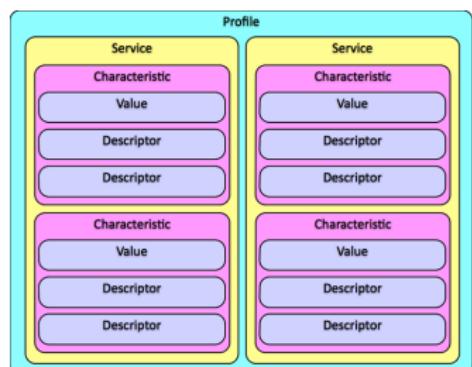
- Generic Attribute Profile defines the way that two BLE devices transfer data back and forth using concepts called Services and Characteristics. It makes use of a generic data protocol called the Attribute Protocol (ATT), which is used to store Services, Characteristics and related data in a simple lookup table using 16-bit IDs for each entry in the table.





Bluetooth - Services and Profiles

- A Profile is a pre-defined collection of Services. The Heart Rate Profile, for example, combines the Heart Rate Service and the Device Information Service.
- Services break data up into logic entities, and contain specific chunks of data called characteristics. A service can have one or more characteristics, and each service distinguishes itself from other services with a unique numeric ID called a UUID, which can be either 16-bit (official BLE Services) or 128-bit (custom services).
- A Characteristic contains a single data point or an array of related data. For example: X/Y/Z values of an accelerometer.





ASCII Reminder

- ASCII characters (the symbols that we are use to reading) can be represented by a single byte (uint8_t).

ASCII TABLE

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

ASCII: American Standard Code For Information Interchange



Argon BLE - UART Service

```
1 // These UUIDs were defined by Nordic Semiconductor and are now the defacto standard for
2 // UART-like services over BLE. Many apps support the UUIDs now, like the Adafruit
3 // Bluefruit app.
4 const BleUuid serviceUuid("6E400001-B5A3-F393-E0A9-E50E24DCCA9E");
5 const BleUuid rxUuid("6E400002-B5A3-F393-E0A9-E50E24DCCA9E");
6 const BleUuid txUuid("6E400003-B5A3-F393-E0A9-E50E24DCCA9E");
7
8 BleCharacteristic txCharacteristic("tx", BleCharacteristicProperty::NOTIFY, txUuid,
9     serviceUuid);
10 BleCharacteristic rxCharacteristic("rx", BleCharacteristicProperty::WRITE_WO_RSP, rxUuid,
11     serviceUuid, onDataReceived, NULL);
12 BleAdvertisingData data;
13
14 //onDataReceived is used to receive data from Bluefruit Connect App
15 void onDataReceived(const uint8_t* data, size_t len, const BlePeerDevice& peer, void*
16     context) {
17     uint8_t i;
18
19     Serial.printf("Received data from: %02X:%02X:%02X:%02X:%02X:%02X \n", peer.address()
20         [0], peer.address()[1], peer.address()[2], peer.address()[3], peer.address()[4], peer
21         .address()[5]);
22     Serial.printf("Bytes: ");
23     for (i = 0; i < len; i++) {
24         Serial.printf("%02X ", data[i]);
25     }
26     Serial.printf("\n");
27     Serial.printf("Message: %s\n", (char *)data);
28 }
```



Argon BLE - UART Transmit Example

```
1 const int UART_TX_BUF_SIZE = 20;
2 uint8_t txBuf[UART_TX_BUF_SIZE];
3 uint8_t i;
4
5 SYSTEM_MODE(SEMI_AUTOMATIC); //Using BLE and not Wifi
6
7 void setup() {
8     Serial.begin();
9     waitFor(Serial.isConnected, 15000);
10
11    BLE.on();
12    BLE.addCharacteristic(txCharacteristic);
13    BLE.addCharacteristic(rxCharacteristic);
14    data.appendServiceUUID(serviceUuid);
15    BLE.advertise(&data);
16
17    Serial.printf("Argon BLE Address: %s\n", BLE.address().toString().c_str());
18 }
19
20 void loop() {
21     for(i=0;i<UART_TX_BUF_SIZE-1;i++) {
22         txBuf[i] = random(0x40,0x5B); //Capital ASCII characters plus @
23     }
24     txBuf[UART_TX_BUF_SIZE-1] = 0x0A;
25     txCharacteristic.setValue(txBuf, UART_TX_BUF_SIZE);
26     for(i=0;i<UART_TX_BUF_SIZE;i++) {
27         Serial.printf("%c",txBuf[i]);
28     }
29     delay(5000);
30 }
```



Formatted Print to a Buffer

```
1 // sprintf does a formatted print to a buffer of type char[  
2  
3 const int BUFSIZE = 50;  
4 byte buf[BUFSIZE];  
5 int people;  
6 float dogs,avg;  
7  
8 void setup() {  
9     Serial.begin(9600);  
10    people = 5;  
11    dogs = 13;  
12 }  
13  
14 void loop() {  
15     avg = dogs / people;  
16     sprintf((char *)buf,"The %i people have on average %0.2f dogs \n",people, avg);  
17     Serial.printf("Buf contains the string: %s", (char *)buf);  
18 }
```

The buffer can then be used to as the data payload between devices, for example, using BlueTooth.

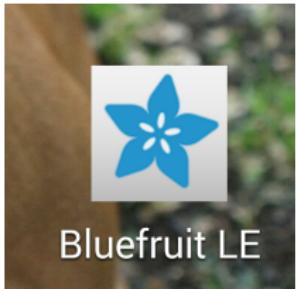
Note: Windows treats a `\n` as a LF-CR (0x0D0A), if a LF is needed (e.g., for BLE) then it needs to be inserted manually:

```
1 buf [BUFSIZE - 1] = 0x0A;
```



Assignment: L14_BlueTooth

L14_03_BlueTooth



Bluefruit LE

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

- Load Bluefruit Connect on your smart device.
Using the code on the proceeding slides,
establish BLE UART communications
- Attach the encoder and NeoPixel ring to your
Argon.
- Repeat the NeoPixel ring assignment
(L06_02_NeoPixel) on your Argon (with only
12 pixels).
- When it changes, send the the encoder or
NeoPixel position via BLE to Bluefruit
Connect.
- Reset the encoder and NeoPixel ring to the
appropriate state when values 0-11 are
received from Bluefruit Connect.



Assignment: L14_BlueTooth - Colors

L14_03_BlueTooth (Continued)

● Plotter function in Bluefruit Connect:

- ① Every time the encoder moves, generate and change the pixels to a random color (R,G,B format, not Hex).
- ② Plot the pixel number and three RGB components on the Bluefruit Plotter.

● Controller -> Color Picker screen on Bluefruit Connect:

- ① Send a color to the Argon.
- ② Identify in your code if ColorPicker string or general UART string is received.
- ③ If ColorPicker, then using bitwise left shift and OR to convert string to hex color similar to L15_02_RetrieveShow
- ④ Change your Neopixel color to match Color Picker

The screenshot shows the Bluefruit Connect app interface. At the top, there's a navigation bar with a back arrow and the text "Plotter". Below this is a list of instructions:

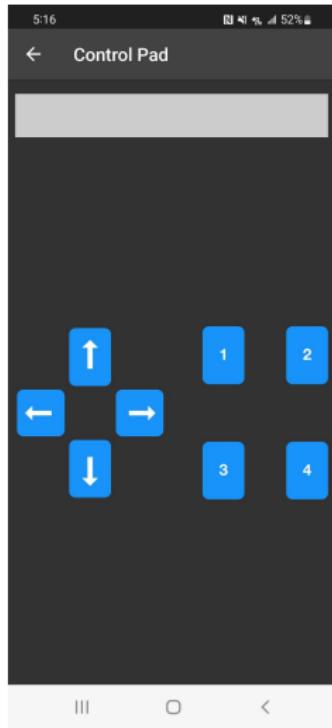
- The 'Plotter' utility can be used to plot incoming numeric data in a chart, without having to create a custom plotter code or application. It behaves similarly to the Serial Plotter in recent versions of the Arduino IDE.
- To plot one or more data streams to the plotter, send your numeric data in CSV format with one of the following separators:
 - ',' - Comma (0x2C)
 - ' ' - Space (0x20)
 - ';' - Semicolon (0x3B)
 - Horizontal Tab (0x09), '\t' in code
- Each unique set of data samples must be terminated by a LINE FEED character (0x0A), which is usually represented as '\n' in code.
- Only numeric data should be sent over the BLE UART connection(s).

ColorPicker String
5 bytes plus CR

[!] [C] [byte red] [byte green] [byte blue] [CRC]



Assignment: L14_BlueTooth - Colors (EXTRA)

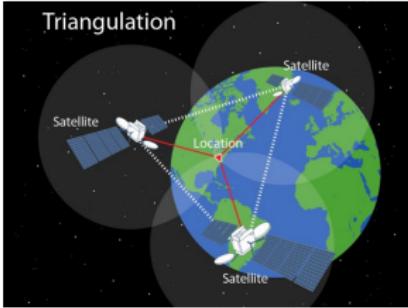


L14_03_BlueTooth (EXTRA)

- Experiment sending signals from the Bluefruit Connect Control Pad
- Use the Up/Down arrows to change the neopixel brightness
- Use the Left/Right arrows to cycle through a rainbow
- Code in a different neopixel effect for each of the number keys



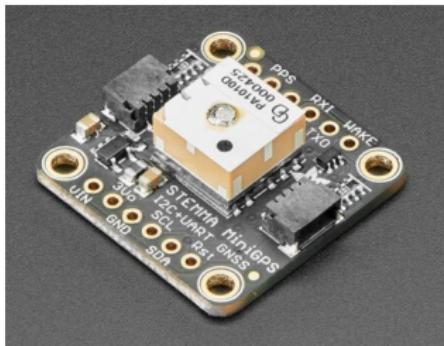
Global Positioning System



- 29 satellites (24 active plus 5 reserve) at an altitude of 12550 miles, circling the earth twice per day.
- Envisioned by Aerospace Corporation 1963
- First satellite 1973
- Open to commercial use 1985
- GPS receiver measures time it takes signal (at speed of light) to get from satellite to receiver.
- Uses triangulation from at least 4 satellites to obtain latitude, longitude, altitude, and time.
- GNSS is an international system of satellites that includes GPS and others



Global Positioning System



- Miniature GPS module
- Houses a complete GPS/GNSS solution
- Both I2C and UART interfaces
- STEMMA interface for easy prototyping



Adafruit_GPS Library

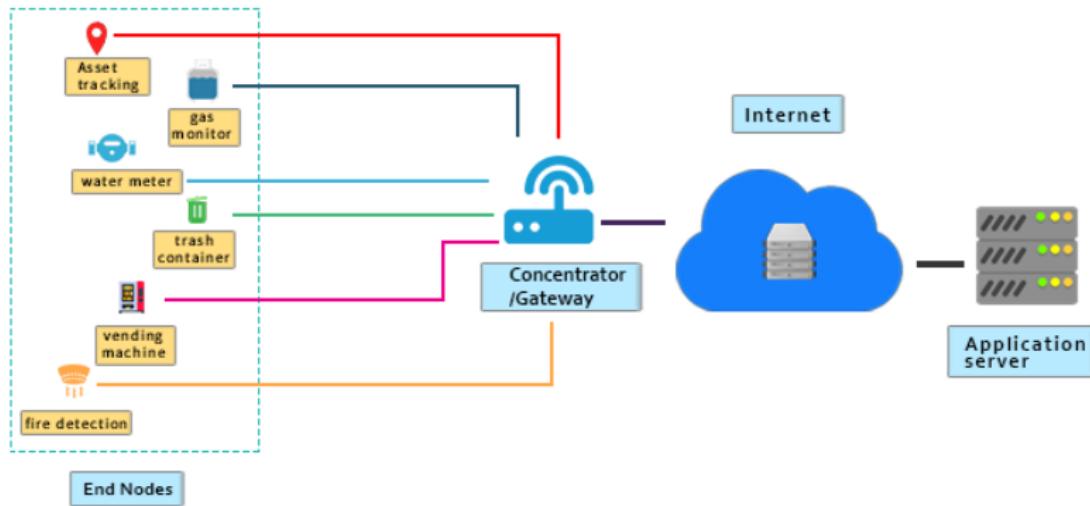
```
#GPGGA,202410.000,4042.6000,N,07400.4858,W,1,4,3.14,276.7,M,-34.2,M,,*63
#GPRMC,202410.000,A,4042.6000,N,07400.4858,W,0.08,161.23,160412,,,A*70
#GPGGA,202411.000,4042.5999,N,07400.4854,W,1,3,17.31,275.8,M,-34.2,M,,*5D
#GPRMC,202411.000,A,4042.5999,N,07400.4854,W,0.14,161.23,160412,,,A*7A
```

```
Time: 20:24:11.0
Date: 16/4/2012
Fix: 1 quality: 1
Location: 4042.5998N, 7400.4853W
Speed (knots): 0.14
Angle: 161.23
Altitude: 275.80
Satellites: 3
```

- Call `gps.read()` at the beginning of `void loop()`
- Then immediately call `GPS.parse(GPS.lastNMEA())` to make the data available
- When you need it, you can then access `GPS.latitude`, `GPS.longitude`, `GPS.speed`, etc.



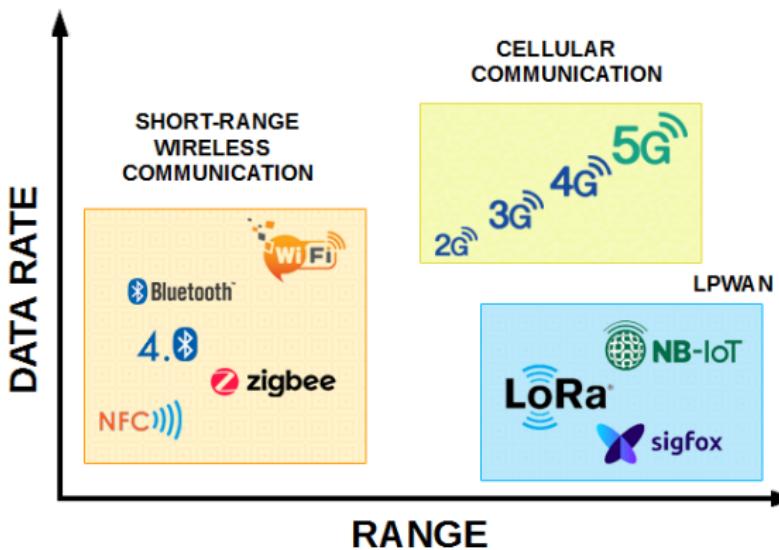
LoRa



LoRa is a long range, low power, inexpensive technology for Internet of Things



LoRa Range vs Data Rate



LoRa uses license-free sub-gigahertz radio frequency ISM bands in the deployed region such as 868 MHz in Europe and 915MHz in North America.



LoRa Features

LoRa has many desirable features:

- It has very wide coverage range about 5 km in urban areas and 15 km in suburban areas
- Battery lifetime up to 15 years
- One LoRa gateway takes care of thousands of nodes.
- Easy to deploy and low cost.
- Enhanced secure data transmission by embedded end-to-end AES128 encryption



RXLR896 LoRa Module

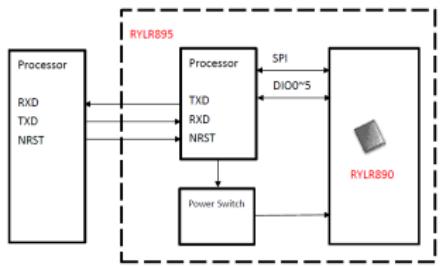
Features:

- Semtech SX1276 Engine
- Excellent blocking immunity
- Low receive current
- High sensitivity
- Control easily by AT commands
- 127 dB Dynamic Range RSSI
- Designed with integrated antenna
- AES128 Data encryption





AT Commands

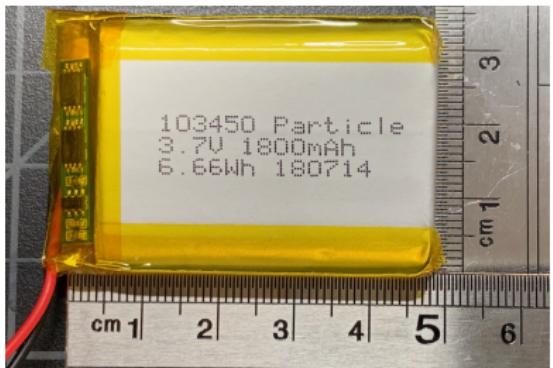


AT commands are commands which are used to control the modems where AT stands for Attention. These commands were derived from Hayes commands which were used by the Hayes smart modems. Every wireless modem requires an AT command to interact with a computer machine.

Communication between the Argon and RYLR896 takes place over UART (Serial1) using the same "Serial" commands that were used in Lesson 3 and Lesson 4.



Batteries - Going Mobile



- ① All Particle platforms have JST-PH pins for a Lithium Polymer (LiPo) battery
 - Always check wiring polarity
- ② Battery can be charged via USB port or V_{bus} .
- ③ Battery power is available on LiPo+ or 3.3V pins



L17_03_LoRaGPS

Features:



- ① Using the L14_00_GPS code, obtain GPS coordinates
- ② Modify L14_04_LoRaGPS:
 - Integrate the LoRa, GPS, and OLED
 - Get your own RADIOADDRESS from instructors
 - Using IoT_Timer, turn on the D7 LED for 5 seconds when LoRa data received.
 - Display FUSE Sound and Particulates to OLED
 - Send live GPS coordinates back to LoRa base-station
 - Find the distance you can go N/S/E/W and get a LoRa signal back to FUSE.
 - Class Trip: repeat at ABQ Biopark

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

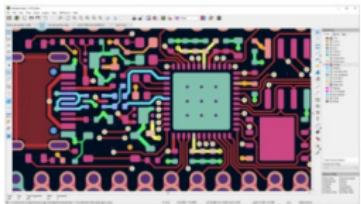
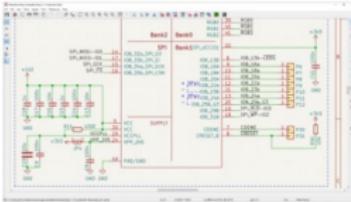
Module 15 - In the Wild



KiCad: Breadboard are good, but PCBs are better

Schematic Capture

KiCad's Schematic Editor supports everything from the most basic schematic to a complex hierarchical design with hundreds of sheets. Create your own custom symbols or use some of the thousands found in the official KiCad library. Verify your design with integrated SPICE simulator and electrical rules checker.



PCB Layout

KiCad's PCB Editor is approachable enough to make your first PCB design easy, and powerful enough for complex modern designs. A powerful interactive router and improved visualization and selection tools make layout tasks easier than ever.

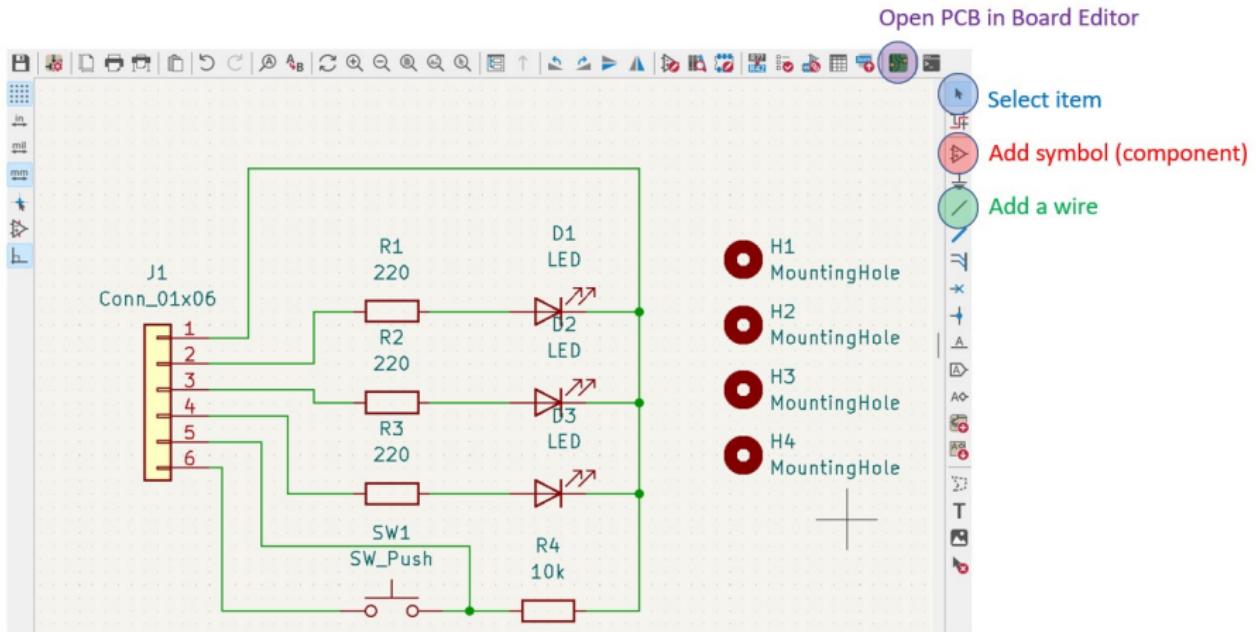
3D Viewer

KiCad's 3D Viewer allows easy inspection of your PCB to check mechanical fit and to preview your finished product. A built-in raytracer with customizable lighting can create realistic images to show off your work.





Start with schematics





Rich library of components

Components or Modules

The screenshot shows the Altium Designer interface with three windows open:

- Top Window:** Shows the "Choose Symbol [17844 items loaded]" dialog with the search term "2N3904". It lists several entries under the "Transistor, NPN" category, including "2N3904" and "MJE3904".
- Middle Window:** Shows the "Choose Symbol [17844 items loaded]" dialog with the search term "imu". It lists various IMU components, including "MPU_6000", "MPU_6500", "MPU_9100", "MPU_9200", "MCU_Microchip_SAMA", "MCU_Infineon", "8000", "8000A", "MCU_Module", "QFN80_NANO", "ResenSensIC-MI", "ResenSensIC-CMI", "ResenSensIC-CDI", "ResenSensIC-CMDSI", "ResenSensIC-CMDSI-L", "ResenSensIC-CMDSI-L", and "Cassini80".
- Bottom Window:** Shows the "Default [Sensor_Motion_InvenSense_QFN-24_Accelero_P0.5mm]" PCB preview window. It displays a schematic symbol for an InvenSense QFN-24 Accelerometer, showing pins for SDA, SCL, ADO, INT, PSYNC, CCLKIN, AUX_SDA, AUX_SCL, CSOUT, REGOUT, and HPOUT. Below the schematic is a 3D model of the component package.

A red arrow points from the text "PCB Preview" to the bottom window.



Select a footprint for each component

The screenshot shows the KiCad software interface for selecting component footprints. The main window displays a schematic diagram with components R1, D1, and D2. A callout points from the text "Right Click" to the component D1. A red arrow points from the word "Library" to the "Footprint Library Browser" window at the bottom left.

Symbol Properties window (Top Left):

Name	Value	Show	H Align	V Align	Italic	Bold	Text Size
Reference	D1	<input checked="" type="checkbox"/>	Center	Center	<input type="checkbox"/>	<input type="checkbox"/>	1.27 mm
Value	LED	<input checked="" type="checkbox"/>	Center	Center	<input type="checkbox"/>	<input type="checkbox"/>	1.27 mm
Footprint	LED_THT/LED_D3.0mm	<input checked="" type="checkbox"/>	Center	Center	<input type="checkbox"/>	<input type="checkbox"/>	1.27 mm

Footprint Library Browser window (Bottom Left):

- File View Help
- Search icons
- Grid: 1.2700 mm (0.0500 in)
- Zoom Auto
- Filter: Connector_PinHeader_1.00mm, Connector_PinHeader_1.27mm, Connector_PinHeader_2.00mm, Connector_PinHeader_2.34mm, Connector_PinHeader_2.50mm, Connector_PinHeader_3.27mm, Connector_PinHeader_3.50mm, Connector_PinHeader_4.00mm, Connector_PinHeader_5.00mm, Connector_PinHeader_6.35mm, Connector_PinHeader_8.00mm, Connector_PinHeader_10.00mm, Connector_PinHeader_12.70mm, Connector_PinHeader_15.24mm, Connector_PinHeader_20.00mm, Connector_PinHeader_25.40mm, Connector_RJ45, Connector_Samtec, Connector_Samtec_HLE_SMD, Connector_Samtec_HLE_THT, Connector_SATA_SAS, Connector_Stock, Connector_TE-Connectivity, Connector_USB, Connector_Wago, Connector_Wuerth, Converter_ACDC, Converter_DCDC, Crystal, Diode_SMD, Diode_THT, Display, Display_2Segment, ...
- Pads: 2
- Vias: 0
- Track Segments: 0
- Nets: 0
- Unrouteable: 0
- Z 15.28 X 5.0800 Y -2.5400 dx 5.0800 dy -2.5400 dist 5.6796
- grid X 1.2700 Y 1.2700 mm

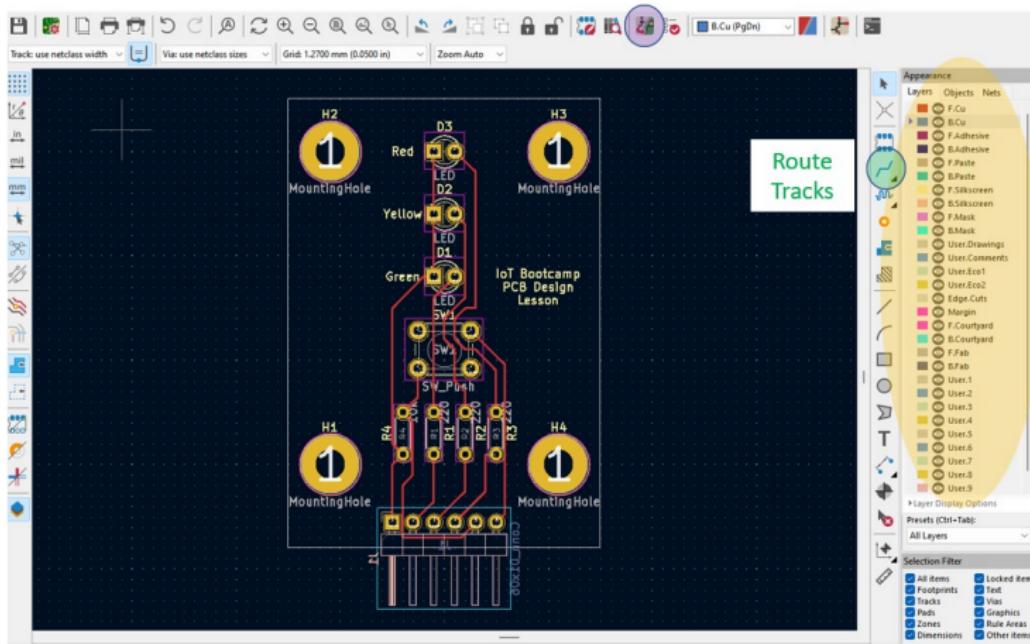
Schematic Diagram (Bottom Right):

- Component D1 is labeled "LED_THT/LED_D3.0mm".
- Component D1 has two pins labeled "D1" and "D2".
- Pin "D1" is connected to a green wire.
- Pin "D2" is connected to a red wire.
- Pin "D2" is also connected to a resistor component labeled "R1" (value 220).
- Pin "D2" is also connected to a diode component labeled "D2".
- Pin "D1" is also connected to a diode component labeled "D2".
- A reference designator "REF**" is shown above the component.
- The component is highlighted with a yellow box containing the numbers "1" and "2".
- The component is labeled "ED_D3.0mr".



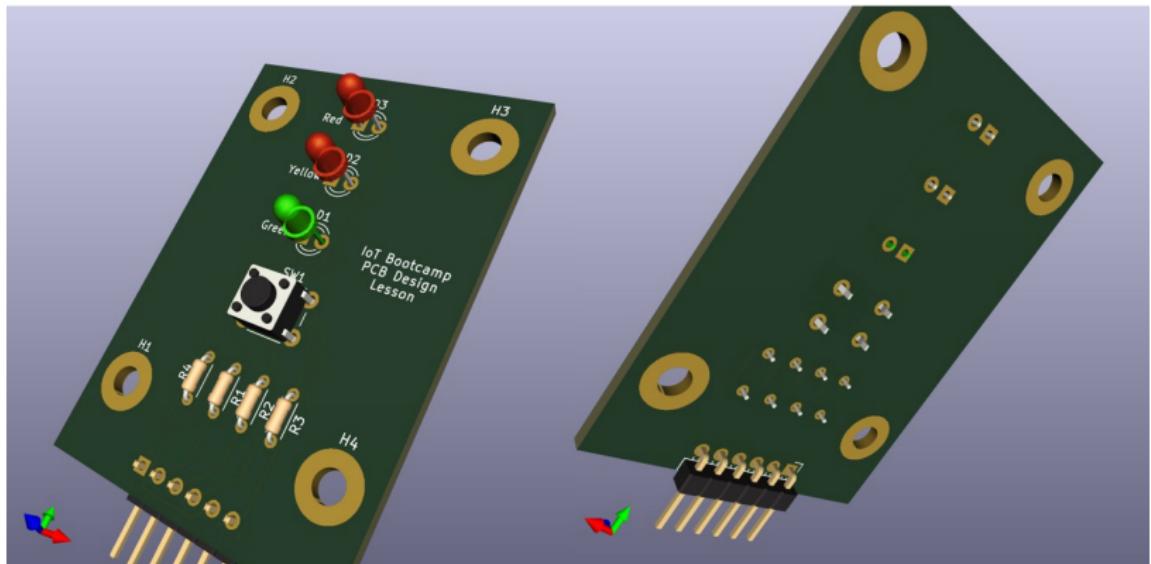
Generate PCB, Route Wires

Update PCB with changes made to schematic





3D View





EN and RST pins

The Argon has two pins that are extremely useful in real world situations:



① EN pin

- The EN pin is not a power pin, per se, but it controls the 3V3 power.
 - Device enable pin is internally pulled-up. To disable the device (force the device into a deep power-down state), connect this pin to GND.
 - This pin is essentially an on/off pin.

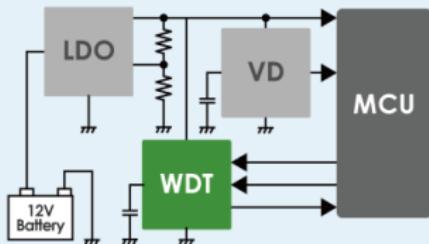
② RST pin

- Active-low system reset input. This pin is internally pulled-up.



Watchdog Timer

e.g. Circuits peripheral to the MCU and WDT
(in an automotive environment)



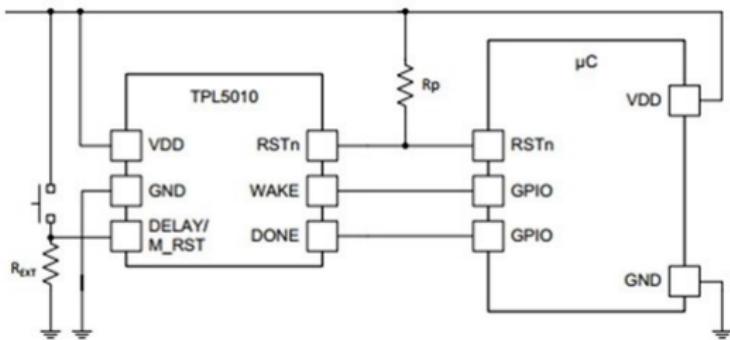
The WDT takes the role of "watchdog" and watches over MCU operation at all times.



The watchdog timer communicates with the MCU at a set interval. If the MCU does not output a signal, outputs too many signals or outputs signals that differ from a predetermined pattern, the timer determines that the MCU is malfunctioning and sends a reset signal to the MCU.



Hardware Watchdog Timers - TPL5010



The timeout frequency is set by resistor R_{EXT} using a formula from the data sheet.

Timeout Interval	Calculated Resistance
1 minute	22 Ω
5 minutes	43 Ω
30 minutes	92 Ω
1 hour	125 Ω
2 hours	170 Ω



Application Watchdog

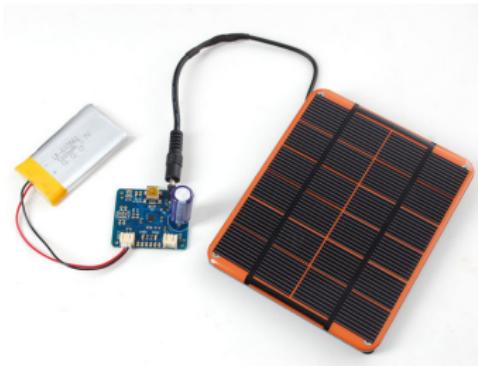
The Argon also has a watchdog timer that can be implemented in code. It is not as robust as the hardware timer, but better than nothing.

```
1 // Prototype
2 // ApplicationWatchdog(unsigned timeout_ms, std::function<void(void)> fn, unsigned
3 // stack_size=DEFAULT_STACK_SIZE);
4
5 // Global variable to hold the watchdog object pointer
6 ApplicationWatchdog *wd;
7
8 void watchdogHandler() {
9     // Do as little as possible in this function, preferably just a reset
10    System.reset(RESET_NO_WAIT);
11 }
12
13 void setup() {
14     // Start watchdog. Reset the system after 60 seconds if the application is unresponsive
15     // The stack_size default is 512, but this is too small. Use at least 1536.
16     wd = new ApplicationWatchdog(60000, watchdogHandler, 1536);
17 }
18
19 void loop() {
20     while (some_long_process_within_loop) {
21         ApplicationWatchdog::checkin(); // resets the AWDT count
22     }
23 } // AWDT count reset automatically after loop() ends
```



Solar Charging

Should be easy, but isn't.





Cloud Flash

During the deployment phase, it isn't convenient to have to hook up a USB cable to push updates. The Particle ecosystem allows for sending code over-the-air (OTA).

A screenshot of a terminal window. The top bar is dark blue with the text '>cloud' in white. Below it is a light blue bar with the text 'Particle: Cloud Compile' on the left and 'recently used' on the right. The main area is dark grey with two entries: 'Particle: Cloud Flash' on the left and 'other commands' on the right. The text is white.

- Cloud Compile - compile your program and download the binary
- Cloud Flash - compile and flash it to the selected device OTA

The OTA operations require:

- The device is connected to the Particle Cloud (breathing cyan)
- The computer is into the same account that claimed the device.
- The DeviceID is set to the device name.

Capstone



Capstone Projects

Intent:

- Based on a direct observation or need expressed by a 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:

- Practical application of smart home, manufacturing, community environment, or immersive entertainment.
- Code MUST follow the IoT Style Guide
- Needs to include a Cloud Dashboard component and concepts from Module 15 - In the Wild
- Project will include a video, presentation, GitHub, and Hackster.io.

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

Module 16 - CPL Grab Bag



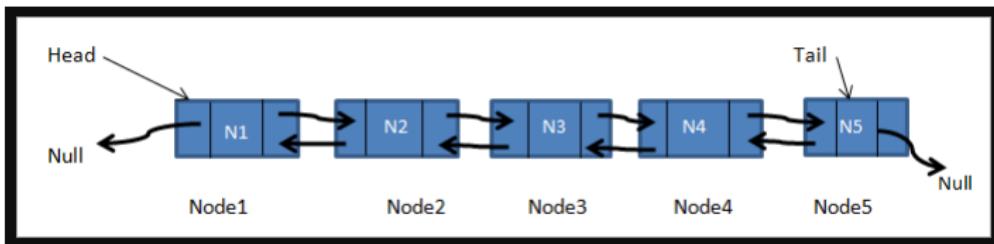
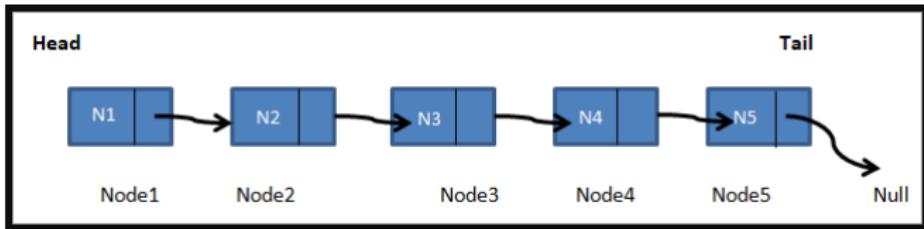
Struct datatype and Member Operators

struct enables the programmer to create a variable that structures a selected set of data.

```
struct name  
struct Employee {  
    char name[10];  
    int idNumber;  
    float salary;  
}  
  
Employee instructor;      } Declare individual variable of data type Employee  
Employee IoTEngineers[10]; } Declare an array of data type Employee  
  
void setup {  
    instructor.name = "Brian";  
    instructor.idNumber = "42";  
    instructor.salary = 212.47;  
    IoTEngineers[1].name = "Sally";  
}
```



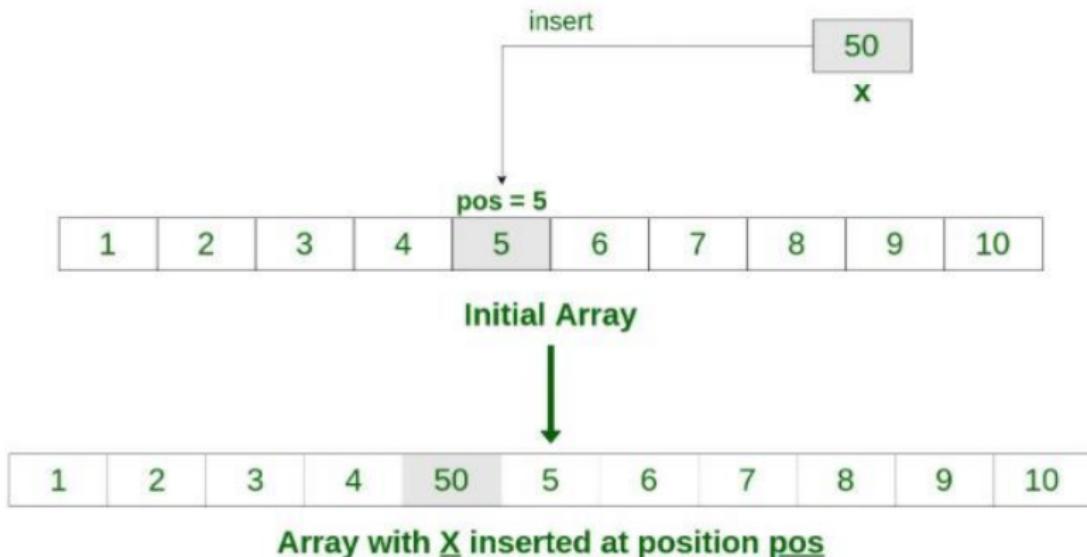
Linked Lists and Doubly Linked Lists



```
1 struct node {  
2     struct node *prev;  
3     int data;  
4     struct node *next;  
5 };
```

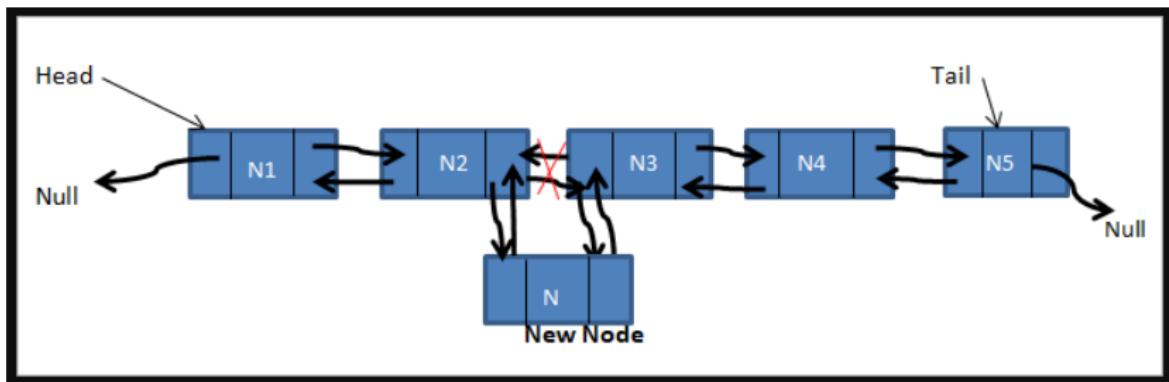


Inserting a "cell" into an Array



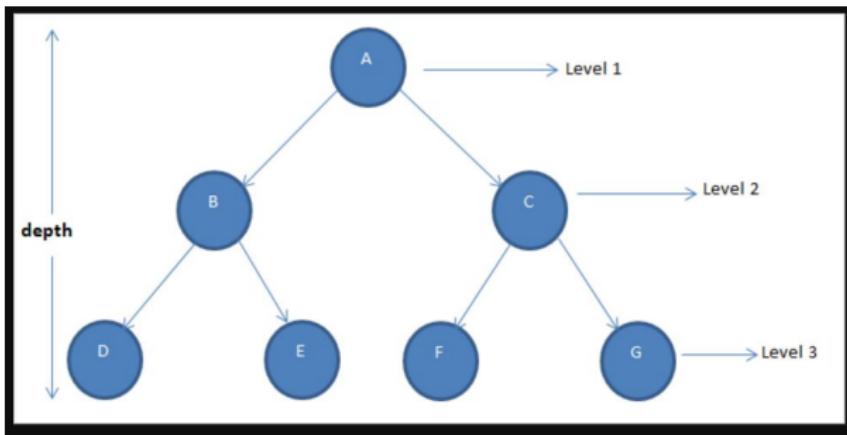


Inserting a "cell" into a Linked List





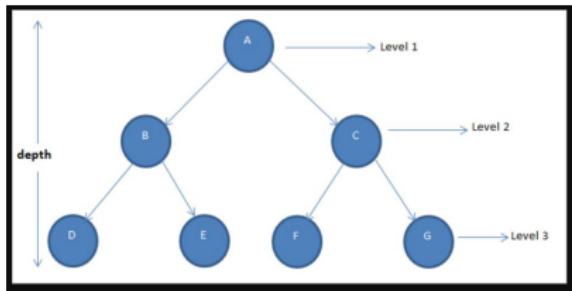
Binary Trees



```
1 struct bintree_node{  
2     bintree_node *left;  
3     bintree_node *right;  
4     int data;  
5 };
```



Binary Trees



Uses of Binary Trees

- Binary Search
- Hash Trees
- Heaps
- Huffman Coding
- Syntax Tree

```
1 struct bintree_node{  
2     bintree_node *left;  
3     bintree_node *right;  
4     int data;  
5 };
```

Non-Embedded C++



HelloWorld in C++

```
1 // include the standard input-output library
2 // most include statements need the .h, but iostream is an exception
3 #include <iostream>
4
5 // namespace is used to declare regions with the global space
6 // std enable the standard console (monitor) and input (keyboard)
7 using namespace std;
8
9 char myName[20];
10
11 //main is the first function executed in your cpp code
12 int main()
13 {
14     cout << "Hello World!!!\n";      // cout = output to console
15     cout << "What is your name: ";
16     cin >> myName;                // cin = input from console
17     cout << "Hello " << myName << ", I hope you are having a nice day.\n";
18
19     return 0; // denotes successfully executed
20 }
```

Instructions for installing and writing C++ code in VSCode:
<https://code.visualstudio.com/docs/languages/cpp>



The Big Question: What about main()

```
1 #include <Arduino.h>
2
3 extern "C" int main(void)
4 {
5 #ifdef USING_MAKEFILE
6
7     // To use Teensy 3.0 without Arduino, simply put your code here.
8     // For example:
9
10    pinMode(13, OUTPUT);
11    while (1) {
12        digitalWriteFast(13, HIGH);
13        delay(500);
14        digitalWriteFast(13, LOW);
15        delay(500);
16    }
17
18
19#else
20    // Arduino's main() function just calls setup() and loop()....
21    setup();
22    while (1) {
23        loop();
24        yield();
25    }
26#endif
27 }
```



Hello World - What a microcontroller sees

HelloWorld.ino

```
1 void setup() {
2     Serial.begin(9600);
3     Serial.printf("Hello World! \n");
4 }
5
6 void loop() {}
```

Hex Code and Assembly Language

```
1 HelloWorld.bin:      file format binary
2 Disassembly of section .data:
3 00000000 <.data>:
4      101c: bd10      pop {r4, pc}
5      101e: 4402      add r2, r0
6      1020: 4603      mov r3, r0
7      1022: 4293      cmp r3, r2
8      1024: d002      beq.n 0x102c
9      1026: f803 1b01  strb.w r1, [r3], #1
10     102a: e7fa      b.n 0x1022
11     102c: 4770      bx lr
12     102e: 0000      movs r0, r0
13     1030: b538      push {r3, r4, r5, lr}
```

Useful BASH commands



Redirect from stdout (standard output)

The > and >> signs are used for redirecting the output of a program to something other than stdout (standard output, which is the terminal by default).

- The >> appends to a file or creates the file if it doesn't exist.
- The > overwrites the file if it exists or creates it if it doesn't exist.

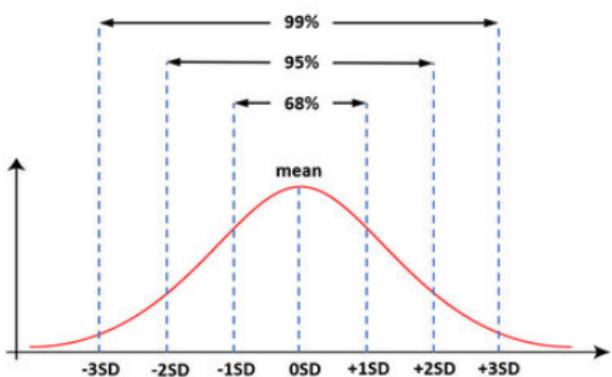
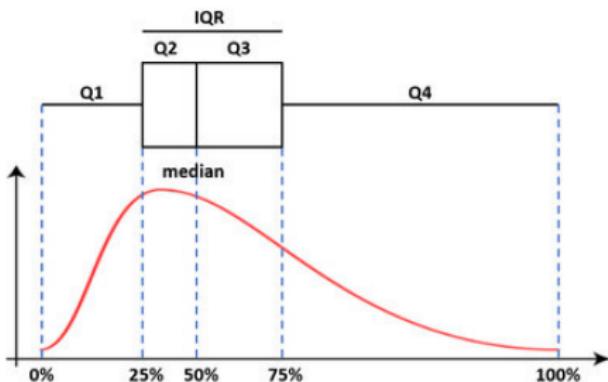
Examples:

```
1 # Create a file called "allmyfiles.txt" and fill with the directory listing
2
3 ls > allmyfiles.txt
4
5
6 # Adds "End of directory listing" to the end of "allmyfiles.txt"
7
8 echo "End of directory listing" >> allmyfiles.txt
9
10 # Create a zero-byte file with the name "newfile.txt"
11
12 > newfile.txt
13
14 # Redirect Particle Serial Monitor output to the file "filename.csv"
15
16 Particle serial monitor --follow >> filename.csv
```

Statistics



Central Tendency - Mean vs Median

A**B**



Statistics

- Average or Mean: The sum of a list of numbers, divided by the number of elements in the list.
- Median: "Middle value" of a list.
- Root-Mean Square (RMS): the square-root of the mean of the squares of the elements in the list.
- Standard Deviation: the RMS of the set of deviations between each element of the set and the mean of the set.
- Quartile: There are three quartiles
 - The lower quartile is a number (not necessarily a number in the list) such that at least $1/4$ of the numbers in the list are no larger than it, and at least $3/4$ of the numbers in the list are no smaller than it.
 - The second quartile is the median.
 - The upper quartile is a number such that at least $3/4$ of the entries in the list are no larger than it, and at least $1/4$ of the numbers in the list are no smaller than it
- Inter-quartile range (IQR): the upper quartile minus the lower quartile.



Assignment: L04_Functions

EVENTS VITALS HEALTH CHECK

NAME	DATA	DEVICE	PUBLISHED AT
particle/device/updat...	false	Lalonde	10/3/20 at 1:09:40 pm
spark/device/diagnos...	{"device": "network", "s...	Lalonde	10/3/20 at 1:09:40 pm
spark/device/app-hash	651A9E3A5D0A02A4115...	Lalonde	10/3/20 at 1:09:39 pm
env-vals	{"PhotoDiode": 487, "LED": 47}	Lalonde	10/3/20 at 1:09:38 pm
LED Output	47	Lalonde	10/3/20 at 1:09:38 pm
PhotoDiode	487	Lalonde	10/3/20 at 1:09:38 pm

env-vals
Published by e00fce68e7addcdcfabbb57d on 10/3/20 at 1:09:38 pm

PRETTY RAW

```
{  
  "PhotoDiode": 487,  
  "LED": 47  
}
```

① L04_01_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.