IoT Product Design and Coding Bootcamp

Brian Rashap, Ph.D.

07-JUN-2021

Copyright ©2021 CNM. All Rights Reserved.

IoT Fun

Introduction

Copyright ©2021 CNM. All Rights Reserved.

Brian Rashap, Ph.D.

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

Copyright ©2021 CNM. All Rights Reserved.

Cecilia Castillo, Assistant

- Software Engineer with over 25 years experience in government and commercial applications development, teaching, and mentoring.
- Side gig teaching robotics and technology to homeschool students.
- Enjoy rock climbing, bicycling, reading, binging crime shows.

Introductions

INTRODUCTIONS

Copyright ©2021 CNM. All Rights Reserved.

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!

Copyright ©2021 CNM. All Rights Reserved.

Grading

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

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

More information will be provided later today on how assignments are turned in.

Copyright ©2021 CNM. All Rights Reserved.

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

Copyright ©2021 CNM. All Rights Reserved.

Evolution of Industry

Copyright ©2021 CNM. All Rights Reserved.

Components of Industry 4.0

Copyright ©2021 CNM. All Rights Reserved.

IoT and Data Science

Copyright ©2021 CNM. All Rights Reserved.

IoT 2025

Copyright ©2021 CNM. All Rights Reserved.

Smart Facilities

Healthcare 2025

Copyright ©2021 CNM. All Rights Reserved.

Smart World

Copyright ©2021 CNM. All Rights Reserved.

And Out of This World

Copyright ©2021 CNM. All Rights Reserved.

IoT Growth

How ubiquitous is the Internet of Things?

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

Copyright ©2021 CNM. All Rights Reserved.

Let's Begin Our Journey

Copyright ©2021 CNM. All Rights Reserved.

Computer Languages

Mother Tongues

Tracing the roots of computer languages through the ages

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

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

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

Key

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

Survival of the Fittest

Reasons a language endures, with examples of some classic tongues

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

Copyright ©2021 CNM. All Rights Reserved.

Computer Languages

Copyright ©2021 CNM. All Rights Reserved.

Why C++

Copyright ©2021 CNM. All Rights Reserved.

CLI vs GUI

Copyright ©2021 CNM. All Rights Reserved.

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).

Copyright ©2021 CNM. All Rights Reserved.

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* >

Copyright ©2021 CNM. All Rights Reserved.

IoT Fun

Smart Room Controller

Our First Microcontroller

Smart Room Controller

Teensy 3.2

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

Arduino IDE for Teensy

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

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

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

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

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

Other Software

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

GitHub - Part 1

Copyright ©2021 CNM. All Rights Reserved.

Git and GitHub: Your Version Control Friends

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.

Why use a version control system?

Before Version Control Systems.

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

With Git.

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

Git vs. GitHub

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

→ git commit

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

→ git push

Commit vs. Push

Commit

- File saved to **LOCAL** repository. File is given a timestamp and a unique commit number that is the file version.

Push

- File is saved to **REMOTE** repository.

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 PAC

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

To create a PAC:

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 PAC rather than your password to access your account.

GitHub Authentication: Switching from password to PAC

If you have been connecting to GitHub from the command line using your password, you will need to switch to using your PAC.

On Windows 10:

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

On Mac OSx:

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

The next time you attempt a git command on the command line, you will be prompted to enter your username and password. Use your PAC in place of your password.

Using Git: Command line or GUI?

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

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

The command line

Windows

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

Mac or Linux

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

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

GITHUB Simplified

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

// From the repository directory, get updates
  git pull

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

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

GITHUB First Clone

ddc-iot-classroom-2

Accept the assignment — L01_HelloWorld

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

Accept this assignment

You're ready to go!

You accepted the assignment, **L01_HelloWorld**.

Your assignment repository has been created:

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

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

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

L01_HelloWorld

Copyright ©2021 CNM. All Rights Reserved.

Teensy on Breadboard

Horizontal Rows

Vertical Columns

Copyright ©2021 CNM. All Rights Reserved.

Basic Structure of Arduino Sketch

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Class Assignments

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

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

// Single Line Comments
```

Hello World in some of the 603+ Coding Languages

Fortran

```
C     Hello World in Fortran

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

C (K&R)

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

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

Python 2

```
# Hello world in Python 2

print "Hello World"
```

Assembler (Intel)

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

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

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

Powershell

```
# Hello World in Microsoft Powershell

'Hello World!'
```

LabVIEW

LaTeX

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

Unix Shell

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

echo Hello World
```

Lisp-Emacs

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

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

BASIC

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

C++

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

#include <iostream.h>

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

Perl

```
# Hello world in perl

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

Python 3

```
# Hello world in Python 3

print("Hello World")
```

Pascal

```
{Hello world in Pascal}

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

MATLAB

```
% Hello World in MATLAB.

disp('Hello World');
```

HTML

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

Postscript

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

Assignment L01_01_HelloWorld

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

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

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

L02_HelloLED

Copyright ©2021 CNM. All Rights Reserved.

Rubber Ducking - An Odd but Brilliant Tool

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

Introduction to Electrical Circuits

Energy

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

Copyright ©2021 CNM. All Rights Reserved.

Ohm's Law

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

Resistor Color Bands

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

Measuring Voltage, Current, and Resistance

Copyright ©2021 CNM. All Rights Reserved.

Power from the Teensy 3.2

The Teensy 3.2 has three pins related to power:

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

Copyright ©2021 CNM. All Rights Reserved.

One Resistor

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

Copyright ©2021 CNM. All Rights Reserved.

Kirchhoff's First Law

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

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

i1 + i2 = i3 + i4

Copyright ©2021 CNM. All Rights Reserved.

Kirchhoff's Second Law

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

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

Copyright ©2021 CNM. All Rights Reserved.

Kirchhoff's Second Law

Copyright ©2021 CNM. All Rights Reserved.

Resistors in Series and Parallel

Copyright ©2021 CNM. All Rights Reserved.

Resistors in Series and Parallel

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

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

Copyright ©2021 CNM. All Rights Reserved.

Install Fritzing - download from Teams

Smart House Plant Watering System

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L02_00_Resistors

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

Diodes

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

Copyright ©2021 CNM. All Rights Reserved.

Light Emitting Diodes

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

Copyright ©2021 CNM. All Rights Reserved.

Current Limiting Resistors

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment L02_01_helloLED

- Using Pin 5 as an output and the appropriate current limiting resistor, blink the LED once per second.
- Measure the voltage at both leads of the LED and record the voltage in your notebook.
- Change the resistor to $1k\Omega$ and then $10k\Omega$. What happens to the brightness? Measure the voltage and current in each case. Record in your notebook.

REMEMBER: Lab notebook, Fritzing, breadboard, then code

Constants and Variables

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

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

The use of Constants and Variables has several advantages:

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

The first two Data Types that we will be using:

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

Copyright ©2021 CNM. All Rights Reserved.

Operators

There are a number of operators that act on variables:

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

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

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

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

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

Constants and Variables Example

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

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

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

Loops

FOR Loop syntax

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

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

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. $j = j + 2$

Copyright ©2021 CNM. All Rights Reserved.

WHILE loop syntax

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


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

For vs While Loops

For vs While Loop

Comparison Chart

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

DifferenceBetween.net

Copyright ©2021 CNM. All Rights Reserved.

Assignment L02_04_helloLEDtri

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

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

While or Do While

Copyright ©2021 CNM. All Rights Reserved.

Number Systems

Decimal

92_{10}

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

Binary

01011100_2

Digits: 0,1

Hexadecimal

$5C_{16}$

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

Octal

34_8

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

Copyright ©2021 CNM. All Rights Reserved.

Bits, Nibbles, Bytes, and Words

Copyright ©2021 CNM. All Rights Reserved.

Data Types: Numbers

Data Type	8-bit AVR systems (Arduino Uno)			32-bit ARM systems (Teensy 3.2)		
	bytes	range (signed)	range (unsigned)	bytes	range (signed)	range (unsigned)
char	1	-128 to 127	0 to 255	1	-128 to 127	0 to 255
short	2	+/- 32,767	0 to 65,353	2	+/- 32,767	0 to 65,353
int	2	+/- 32,767	0 to 65,353	4	+/- 2,147,483,648	0 - 4,294,967,295
long	4	+/- 2,147,483,648	0 - 4,294,967,295	4	+/- 2,147,483,648	0 - 4,294,967,295
long long	8	+/- 9,223,372,036,854,770,000	0 to 18,446,744,073,709,551,615	8	+/- 9,223,372,036,854,770,000	0 to 18,446,744,073,709,551,615
float	4	3.4E +/- 38 (7 digits)	n/a	4	3.4E +/- 38 (7 digits)	n/a
double	4	3.4E +/- 38 (7 digits)	n/a	8	1.7E +/- 308 (15 digits)	n/a
long double	8	1.7E +/- 308 (15 digits)	n/a	8	1.7E +/- 308 (15 digits)	n/a
Unambiguous						
uint8_t	1	n/a	0 to 255	1	n/a	0 to 255
int8_t	1	-128 to 127	n/a	1	-128 to 127	n/a
uint16_t	2	n/a	0 to 65,353	2	n/a	0 to 65,353
int16_t	2	+/- 32,767	n/a	2	+/- 32,767	n/a
uint32_t	4	n/a	0 - 4,294,967,295	4	n/a	0 - 4,294,967,295
int32_t	4	+/- 2,147,483,648	n/a	4	+/- 2,147,483,648	n/a

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

Copyright ©2021 CNM. All Rights Reserved.

Math Warning and Type Casting

Be cognizant of the data type when performing math operations.

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

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

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

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

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

Header Files

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Basic Structure of Arduino Sketch Revisited

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

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

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

Sine Waves

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

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

Copyright ©2021 CNM. All Rights Reserved.

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.

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

Why is adding the decimal after 1000 important?

L03_Buttons

Copyright ©2021 CNM. All Rights Reserved.

IoT Fun

Copyright ©2021 CNM. All Rights Reserved.

Displaying to the Screen: The Serial Monitor

```
void setup() {

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

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

Print Statements

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

Format Specifiers Statements

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

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

//Output: Print an integer 42 and a float 3.1415
```

Copyright ©2021 CNM. All Rights Reserved.

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();

Copyright ©2021 CNM. All Rights Reserved.

One Pin - Many Functions

Software Programmable: Input or Output and Digital or Analog.

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

Switches

Lamp Off

Lamp On

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

A Button - SPST

SPST

double break

Copyright ©2021 CNM. All Rights Reserved.

Floating Inputs

Copyright ©2021 CNM. All Rights Reserved.

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?

Copyright ©2021 CNM. All Rights Reserved.

Our First Button and Pull Down Resistors

fritzing

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)

Copyright ©2021 CNM. All Rights Reserved.

Model Based Troubleshooting

Copyright ©2021 CNM. All Rights Reserved.

Assignment: Buttons

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

IF-ELSE Statements

IF-ELSE Statements

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

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

Copyright ©2021 CNM. All Rights Reserved.

Data Types: Boolean and Boolean Logic

Boolean datatype (bool) holds either a TRUE or FALSE

Boolean Logic Operations (condition statements)

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

Boolean Logic In Action

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

// ! (NOT) assignment function
x = !x;        // toggle x (if x = 1, then set x = 0, and visa-versa)

// ! (NOT) comparison
if (!x) {      // if x is false
  // do something
}

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

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

Copyright ©2021 CNM. All Rights Reserved.

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

Copyright ©2021 CNM. All Rights Reserved.

Resistors in Series and Parallel

Open Circuit Behavior

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

Behavior Under Load

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

Copyright ©2021 CNM. All Rights Reserved.

Voltage Dividing

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

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

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

Assignment L03_05_AnalogInput

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

Copyright ©2021 CNM. All Rights Reserved.

Anatomy of a Function

Anatomy of a function in C / C++

Datatype of data returned,
any C datatype (int, float, bool, etc.)

"void" if nothing is to be returned

Function Name

Parameters passed to
function, any C datatype

```
int myMultiplyFunction(int x, int y) {

    int result;

    Result = x * y;
    Return result;
}
```

Return statement,
datatype matches declaration

Curly braces required

Copyright ©2021 CNM. All Rights Reserved.

Types of Variables

```
int x;            // x is a global variable available in the entire program
void setup() {
  x = 1;
}
void loop() {
  x = addx();
}
int addx() {
   int y = 0;          // y is a local variable, resets every function call
   static int z = 0;   // z is a static local variable, maintains its value
   y = y + x;
   z = z + x;
   Serial.printf("x = %i, y = %i, and z = %i \n",x,y,z);
   return z;
}
```

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 Revisited

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

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

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment L03_05_AnalogInput Revisited

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

L04_oneButton

Copyright ©2021 CNM. All Rights Reserved.

IoT Humor

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

Copyright ©2021 CNM. All Rights Reserved.

Objects

Copyright ©2021 CNM. All Rights Reserved.

Traffic Light

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

Copyright ©2021 CNM. All Rights Reserved.

State Machine - Traffic Lights, British Style

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

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.

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

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

Copyright ©2021 CNM. All Rights Reserved.

Preprocessor commands

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

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

  #define _BUTTON_H_ //define the label

  // Place your Header.h code here

#endif // _BUTTON_H_
```

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

OneButton Library

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

Basic Structure of Arduino Sketch - Revisited

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

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

const int GREENLED = 5;         // declare global constants
bool greenState;                 // declare global variables

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

// loop() is used for code that runs continuously
void loop() {
  button1.tick();
  digitalWrite(GREENLED,greenState);
}

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

Copyright ©2021 CNM. All Rights Reserved.

OneButton Declarations

```
#include <OneButton.h>
OneButton button1(pin, activeLOW, pullUP);
void setup() {
  button1.attachClick(click1);
  button1.attachDoubleClick(doubleClick1);
  button1.attachLongPressStart(longPressStart1);
  button1.attachLongPressStop(longPressStop1);
  button1.attachDuringLongPress(longPress1);
  button1.setClickTicks(250);
  button1.setPressTicks(2000);
}
```

OneButton parameters (not variables):

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

Copyright ©2021 CNM. All Rights Reserved.

Using OneButton

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: OneButton

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

1. L04_01_oneButton
   - Use OneButton libary and button on Pin 23.
   - Click() - toggle bool variable buttonState.
   - doubleClick() - toggle bool variable blinker.
2. L04_02_oneButtonLED - Without changing the Click() and doubleClick() functions:
   - Single Click: toggle LED on/off using buttonState variable.
   - Double Click: when LED is on, toggle between solid and rapidly blinking (50ms) using blinker variable.
   - What happens if blinking is made slower (try > 1 second)?

Copyright ©2021 CNM. All Rights Reserved.

Avoiding Delays

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Arrays

- Syntax: datatype var[ ] = {element 1, element 2, element 3};
- Example: int c[ ] = {-45, 6, 8, 72, 1543, -89}

Copyright ©2021 CNM. All Rights Reserved.

Using Arrays

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: OneButton

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

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

Copyright ©2021 CNM. All Rights Reserved.

String datatype and Serial Read

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

```
String inputString;
int value;

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

void loop() {
  inputString = "";                                  // clear the variable inputString
  while(inputString=="") {                           // wait for input from Serial Monitor
    inputString = Serial.readStringUntil('\n'); // get input when enter is pushed
  }
  value = inputString.toInt();                       // convert String to integer
  Serial.printf("The number you entered is %i \n",value);
}
```

Assignment: Serial Read

L04_04_timer - Create a Stop Watch and Countdown Timer

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

L05_NeoPixels

Copyright ©2021 CNM. All Rights Reserved.

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!

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

Solder Irons

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

Solder Tip Care

CLEANING YOUR TIPS

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

TINNING YOUR TIPS

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

OXIDIZED TIP TINNED TIP

Properly Storing Tips

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

Good vs Bad Solder Joints

Don't: Use the very tip of the iron.
Do: Use the side of the tip of the iron, "The Sweet Spot."

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

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

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

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

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

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Desoldering

Solder Wick

Desoldering Pump

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

CYMK vs RGB Colors

CMYK / Print
(Subtractive Colors)

RGB / Screen
(Additive Colors)

NeoPixel Programming

Using NeoPixel Class and Methods

```
#include <Adafruit_NeoPixel.h>

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

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

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

Where do global header files go?

Generating Random Numbers

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: NeoPixels

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

1. L05_01_neoPixel
   - Using FOR loop and individual R/G/B, light up 16 pixels; small delay between.
2. L05_02_colorHeader
   - Implement a header file that contains the pixel colors.
3. L05_03_neoStrip, use setPixelColor() to implement functions:
   - Send a pixel of a random color down and back on the strip.
   - Light the strip up as a rainbow.
   - Send a pair of Maize and Blue lights down the strip.
4. L05_04_pixelFill
   - Light up 7 segments of different colors using the fill() method.

L06_Encoders

Copyright ©2021 CNM. All Rights Reserved.

IoT Humor

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

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.

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

Copyright ©2021 CNM. All Rights Reserved.

Low Pass Filter Response

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

Copyright ©2021 CNM. All Rights Reserved.

Capacitors - does it matter how they are placed

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

Serial Plotter

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: Low Pass Filters

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

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

Encoders

Encoders

Clockwise Sequence

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

Counter-Clockwise Sequence

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

Copyright ©2021 CNM. All Rights Reserved.

The Encoder Class

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

void setup() {
}

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

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

Copyright ©2021 CNM. All Rights Reserved.

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);

Copyright ©2021 CNM. All Rights Reserved.

Assignment: Encoders

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

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

Copyright ©2021 CNM. All Rights Reserved.

Encoder Jitter

Copyright ©2021 CNM. All Rights Reserved.

Encoder - Low Pass Filter

Copyright ©2021 CNM. All Rights Reserved.

Assignment: Encoders

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

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

L07_SPI

Copyright ©2021 CNM. All Rights Reserved.

Buses and Interfaces

Copyright ©2021 CNM. All Rights Reserved.

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)

Copyright ©2021 CNM. All Rights Reserved.

The USB Cable is a UART connection

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

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

Copyright ©2021 CNM. All Rights Reserved.

Serial Peripheral Interface

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

Copyright ©2021 CNM. All Rights Reserved.

SPI - Chip Select

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

Copyright ©2021 CNM. All Rights Reserved.

Serial UART vs SPI

UART

SPI

Copyright ©2021 CNM. All Rights Reserved.

RC Time Constant

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

Copyright ©2021 CNM. All Rights Reserved.

μSD Card Module

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L07_01_dataLogger

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

1. The starter code details the pin assignments for SPI. Use these for your schematic and Fritzing.
2. Modify the starter code to:
   - Read two inputs:
     1. The value of the encoder (unbounded)
     2. A potentiometer connected to Pin 22.
   - Write to μSD Card a timestamp (in millis()) and the two input values every 5 seconds.

L08_Ethernet

The Internet

IP Addresses

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

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

Ethernet Port

Copyright ©2021 CNM. All Rights Reserved.

Assignment: Wemo

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

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

1. L08_00_EthernetTest
   - Create your own mac.h MAC Address.
2. L08_01_Wemo
   - From the wemo.h library determine how the functions are called.
   - Create code to turn on/off multiple Wemo outlets in the classroom.
3. L08_02_Wemo_Timer
   - Create timer that turns off a Wemo 10 secs after you push "off" button without using delay().
4. L08_03_Wemo_Object
   - Modify the wemo.h library to be a Class and Methods.
   - Modify your wemo code to create and use a wemo object.

L09_Servo

Copyright ©2021 CNM. All Rights Reserved.

Servo Motors

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

Servo library

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L09_Servo

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

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.

2-dimensional arrays

- Declare Array: int c[3][6] = {{-45,6,0,72,1543,-98},{32,107,88,92,32,107},{16,-2,2,-42,16,-2}};
- Set a Cell: c[1][3] = 42;
- Access a Cell: x = c[1][3]; → x = 42

Copyright ©2021 CNM. All Rights Reserved.

Char and Byte Datatype

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

ASCII TABLE

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

ASCII: American Standard Code For Information Interchange

Keypad.h

```
#include <Keypad.h>

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

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

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

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L09_02_Lock

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

1. Design and print a lockholder based on the class example.
   - Optional: design and print your own gear and lock slider
2. Using the keypad, create a digital lock
   - Implement 4-digit code to lock and unlock.
   - Store code in an array
   - Create a bool function that compares code to a key
   - 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.

Copyright ©2021 CNM. All Rights Reserved.

Optional Designs

1. Create a sketch
2. Go to Tools in the top menu

image.png

3. Select Add-ins

4. Select the first SpurGear

image.png

5. Then enter your measurements noting that it tells you what to use for Module

GEAR

LOCK SLIDER

L10_I^2C

Copyright ©2021 CNM. All Rights Reserved.

IoT Humor

Backing Up the Internet of Things

Inter-integrated Circuit (I^2C)

I^2C vs SPI

I2C v/s SPI

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

L10_00_I2CScanner

Let's create an I2C scanner

1. 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
2. Create a Fritzing diagram for adding the BME280 and the OLED to your Teensy.
3. Wire up your schematic and determine the I2C addresses of each device.

Char Datatype - Revisited

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

ASCII control characters

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

ASCII printable characters

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

Extended ASCII characters

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

ASCII 248

°

alt + 248
(Degree symbol)

most consulted

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: I^2C

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

1. L10_01_OLEDWrite
   - Install Adafruit_SSD1306 library.
   - Review and run SSD_1306_128x64_i2c example.
     - Change the I2C address in the example to match your OLED.
   - Create variables for your birthday. Using the syntax from testdrawstyles() print using printf():
     - Hello World
     - Your Name using spanish honorific (señor, señora, señorita).
     - Your Birthday (04/03/1968)
   - Experiment with rotating the screen using the setRotation(rot) method.
     - rot is an int from 0 to 3

Copyright ©2021 CNM. All Rights Reserved.

Using the Adafrunit_BME280 class

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: I^2C

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

1. L10_02_BME280
   - Read BME280 data.
   - Convert to tempF and inHg.
   - Print to Serial.Monitor and the OLED display.
2. L10_03_BME280_SDMicro
   - Add in saving data to the μSD card once per minute using millis() as the timestamp.
   - Use your NeoPixels to give a visiual indication of room conditions.
3. L10_04_RTC (Optional)
   - Add the 12.5 pF crystal to the Teensy.
   - Display the time on the OLED.
   - Add a true timestamp to the saved data.

Real Time Clock

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

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

L11_Hue

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

SOAP vs REST

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L11_01_Hue

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

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

GitHub - Part 2

The repository - aka the repo

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

Personal Repositories

The readme, license, .gitignore files

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

Copyright ©2021 CNM. All Rights Reserved.

Practice: Create a repo in GitHub

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

Practice: Clone the repo to your local machine

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

README.md: some common markdowns

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Practice: customizing your README.md file

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

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

Practice: Committing and pushing changes

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

Practice: View changes

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

The .gitignore file

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

Examples:

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

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

Practice: Creating and using a .gitignore file

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

Copyright ©2021 CNM. All Rights Reserved.

What to do if you accidentally COMMIT a sensitive file

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

What to do if you accidentally PUSH a sensitive file

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

FIRST CHANGE THE PASSWORD OR INFORMATION IMMEDIATELY.

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

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

Midterm 1 - Smart Room Controller

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, BME208, uSD card, Servo motor, etc.)
5. Device has at least two modes
   - Manual - user controls lights and outlets
   - Automatic - external source triggers response of lights and/or outlet(s)
6. Include something that you designed and made at FUSE (case, button cover, encoder dial, etc.)
7. Optional: use a component that we haven't learned in class

Particle Argon

Copyright ©2021 CNM. All Rights Reserved.

Our Second Microcontroller

Copyright ©2021 CNM. All Rights Reserved.

Coding expectations for the rest of the course

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

Copyright ©2021 CNM. All Rights Reserved.

Previous Capstones

Previous Capstone Playlist

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

Particle Argon

Main processor:

Nordic Semiconductor nRF52840 SoC

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

Argon Wi-Fi network coprocessor:

Espressif ESP32-D0WD 2.4 GHz Wi-Fi coprocessor

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

Argon general specifications:

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

Copyright ©2021 CNM. All Rights Reserved.

Why Particle

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

Copyright ©2021 CNM. All Rights Reserved.

Particle: Edge to Cloud

Copyright ©2021 CNM. All Rights Reserved.

Particle: Prototyping to Production

Copyright ©2021 CNM. All Rights Reserved.

Particle Argon Pin Layout

v1.0

Copyright ©2021 CNM. All Rights Reserved.

Please Do Not Skip Ahead During Setup

Copyright ©2021 CNM. All Rights Reserved.

Particle Software - ParticleCLI and Visual Studio Code

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

Copyright ©2021 CNM. All Rights Reserved.

Particle Setup

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

```
particle login
```

5. Ensure you have the latest Particle CLI.

```
particle update-cli
```

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

Copyright ©2021 CNM. All Rights Reserved.

Updating your Argon to latest Device OS

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

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

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

```
particle serial identify

Your device id is e00fce681ffffffffc08949b
Your system firmware version is 1.5.2
```

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

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

Claim Your Device

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

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

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

```
particle usb setup-done
```

3. You have successfully set up your Argon!

Copyright ©2021 CNM. All Rights Reserved.

Useful Particle CLI Commands

1. Enter DFU mode from the CLI.

```
particle usb dfu
```

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

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

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

```
particle list
```

4. Search for available libraries.

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

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

Copyright ©2021 CNM. All Rights Reserved.

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

Copyright ©2021 CNM. All Rights Reserved.

Directory Structure - VERY IMPORTANT

EXPLORER

OPEN EDITORS 1 UNSAVED
- PM25_Test.ino src 3

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

PM25_Test.ino

src > PM25_Test.ino > ...

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Command Palette - Ctrl-Shift-P

L12_HelloParticle

Copyright ©2021 CNM. All Rights Reserved

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: L12_HelloParticle

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

1. L12_01_HelloParticle
   - Blink the onboard LED (Pin D7).
2. L12_02_HelloNightLight
   - The anode of the photodiode is connected to Pin A0. *Note, unlike an LED, the cathode (short pin) of the photodiode is connected to 3.3V.*
   - The LED anode to Pin D4.
   - Using analogRead/digitalWrite, turn on the LED when the photodiode is dark.
   - Using analogWrite, turn on the LED slowly as the room darkens.

Analog Resolution

Assignment L12_03_Resolutions

- Using a known voltage value (i.e., 3.3V), determine the resolution (in bits) of analogRead() on the Argon.
- analogWrite the value 63 and measure the resulting voltage with your voltmeter. Use this to determine the resolution (in bits) of analogWrite(). Hint: the max value will produce 3.3V.

Particle Publish

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

EVENTS VITALS HEALTH CHECK

Search for events ADVANCED

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

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

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

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

JSON

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

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

JSON Parser Generator

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

```
#include <JsonParserGeneratorRK.h>

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

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

Installing Libraries

Using the Command Palette within VSCode:

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

Assignment: L12_HelloParticle

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

Analog Resolution

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

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

SYSTEM_MODE

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

- AUTOMATIC,
- SEMI_AUTOMATIC,
- MANUAL.

```

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

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

L13_Semiconductors

Copyright ©2021 CNM. All Rights Reserved.

Semiconductors

Copyright ©2021 CNM. All Rights Reserved.

But First... True DAC on Argon

$analogWrite(pin, value, frequency)$

The Particle microcontrollers do not have a true DAC (Digital to Analog Converter) like pin A14 on the Teensy. However, we can convert a PWM pin to a DAC signal using a low pass filter.

- In lesson L06_Encoders we learned about Low Pass Filters.
- The PWM signal oscillates at 500 Hz, but we can increase up to 5,000 Hz using a third parameter in analogWrite().
- Using $R = 470\Omega$ and $C = 2.2\mu F$ will give a $f_c = 153Hz$.

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L13_00_DAC

Using the full size breadboard:

- Create a PWM output of a sine wave at $75Hz$ using a PWM frequency of $5kHz$.
- Add in a low pass filter of $150Hz < f_c < 750Hz$ to create a true DAC output.
- Add in a second low pass filter with an $f_c < 50Hz$.
- Add a second analog pin with a constant output voltage at about 0.825 V
- On the oscilloscope, measure the PWM, DAC, and final filter signals. Record in your lab notebook.

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

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).

NPN Modes

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

Copyright ©2021 CNM. All Rights Reserved.

Water Analogy

Copyright ©2021 CNM. All Rights Reserved.

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

NPN Pre-Amplifier Circuit

Two Stage Pre-Amplifier

Ohm's Law - Revisited

Copyright ©2021 CNM. All Rights Reserved.

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

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

Non-inverting Op Amp

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: Amplifiers

Using your breadboard from L13_00_DAC:

1. L13_01_NeoPixel
   - Add a Emitter-Follower into your NeoPixel circuit to boost the pixel commands to 5V.
2. L13_02_NPNAmp
   - Amplify your True DAC signal using an NPN preamp.
   - Measure your circuit at each node using the oscilliscope [a].
3. L13_03_OpAmp
   - Replace the NPN preamp with an amplification circuit using the LM741 Op Amp.

[a] If you do not have an oscilloscope, you can use your Teensy with the code from L06_01_lowPass.

L14_PlantWatering

More DataTypes - Strings, strings, and char[]

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

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

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

Too Much Time On My Hands

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

```
// Declare Global Variables in Header
String DateTime, TimeOnly;

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

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

  //Print using formatted print
  //%s prints an array of char
  //the .c_str() method converts a String to an array of char
  Serial.printf("Date and time is %s\n",DateTime.c_str());
  Serial.printf("Time is %s\n",TimeOnly.c_str());

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

Copyright ©2021 CNM. All Rights Reserved.

Soil Moisture Sensors

Resistive Sensor

Capacitive Sensor

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L14_SoilMoisture

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

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

The Cloud

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

Copyright ©2021 CNM. All Rights Reserved.

MQTT: MQ Telemetry Transport

Copyright ©2021 CNM. All Rights Reserved.

Adafruit.io

Let's create an Adafruit.io account.

MQTT Elements explained

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

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

MQTT Elements in VSCode

```
#include <Adafruit_MQTT.h>

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

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

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

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

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

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

MQTT Publish and Subscribe

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


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


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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L14_03_SubscribePublish

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

Copyright ©2021 CNM. All Rights Reserved.

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.

Midterm 2 - Plant Watering System

Assignment: L14_04_PlantWater

2N3906 Emitter Follower

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

Copyright ©2021 CNM. All Rights Reserved.

Relay and Pump Fritzing Diagram

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

Seeed Sensors

Seeed Grove - Dust Sensor

Seeed Grove - Air Quality Sensor v1.3

L14_04_PlantWater (continued)

1. Integrate both Seeed sensors into your Plant Water project.
   - Look up the Seeed sensors online to see how they work.
   - Do not blindly copy the examples. Only use the code you need.
   - By looking at the .cpp code, determine how to get a quantitative value for air quality, in addition to the qualitative level.
2. Integrate the entire system
   - Create a user friendly Adafruit.io dashboard
   - Buy and/or create a structure to hold the plant, pump, and sensors.
   - Optional: integration SMS messaging using Zapier (following slides).
   - Be prepared to demo your Plant Watering System for your classmates
3. 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.

Using Zapier

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

Optional Assignment: L14_04_PlantWater

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

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

L15_BEI - Binary, EEPROM, and Interrupts

Counting Sheep

Negative Numbers

Question: How are negative numbers represented in binary?

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

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

2's Compliment

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

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

The negative plus the positive equals zero.

Bitwise Operations

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

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

Bit Shifting

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: $-36382.36 = -3.638236 * 10^4$
With binary exponential equals $-1 * 1.1103014945983887 * 2^{15}$

- Sign: Negative = 1
- Exponent: 15 = 10001110 (with an exponent bias of 10000000)
- Mantissa: 1110301494598387 = 00001110000111100101110 0

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.

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

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

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

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

Interrupts

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

```
pinMode(pin, INPUT); \\ can also use INPUT_PULLUP or INPUT_PULLDOWN
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.

```
Timer timer(1000,printEverySecond);

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

void printEverySecond() {
    static int count = 0;
    Serial.printf("count=%i, time = %u ms \n", count,millis());
    count++;
}
```

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.

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.

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

// print a random number hex color value
randNumber = random(0x0000,0xFFFFFF);
Serial.printf("My color is 0x%06X \n",randNumber);
```

L15_BEI Assignments

1. L15_01_ConvertStore
   - Convert random hex color into R, G, and B components using Bit Shifting and Bitwise AND.
     - First sketch out how this might be done in your lab notebook. Show the instructor.
     - Then, and only then, write the code.
   - Store the components of the color as bytes in the Argon's EEPROM.
2. L15_02_RetrieveShow
   - Retrieve the color from EEPROM memory.
   - Convert to a Hex color code (for example 0xABCDEF)
   - Display the color in NeoPixel using setPixelColor(i,hexColor)
3. L15_03_Interrupt Create an interrupt Stop Button
   - Add a button to your Argon.
   - Using delays, blink your NeoPixel GREEN (once per second).
   - Using an interrupt, when the button is pressed, have the NeoPixel turn to rapidly flashing RED.

L16_Motion

Copyright ©2021 CNM. All Rights Reserved.

MPU6050 Accelerometer

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

- 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.

Copyright ©2021 CNM. All Rights Reserved.

Accelerometers

Gravity and Orientation

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

Copyright ©2021 CNM. All Rights Reserved.

REMINDER - Data Types: Numbers

Data Type	8-bit AVR systems (Arduino Uno)			32-bit ARM systems (Teensy 3.2)		
	bytes	range (signed)	range (unsigned)	bytes	range (signed)	range (unsigned)
char	1	-128 to 127	0 to 255	1	-128 to 127	0 to 255
short	2	+/- 32,767	0 to 65,353	2	+/- 32,767	0 to 65,353
int	2	+/- 32,767	0 to 65,353	4	+/- 2,147,483,648	0 - 4,294,967,295
long	4	+/- 2,147,483,648	0 - 4,294,967,295	4	+/- 2,147,483,648	0 - 4,294,967,295
long long	8	+/- 9,223,372,036,854,770,000	0 to 18,446,744,073,709,551,615	8	+/- 9,223,372,036,854,770,000	0 to 18,446,744,073,709,551,615
float	4	3.4E +/- 38 (7 digits)	n/a	4	3.4E +/- 38 (7 digits)	n/a
double	4	3.4E +/- 38 (7 digits)	n/a	8	1.7E +/- 308 (15 digits)	n/a
long double	8	1.7E +/- 308 (15 digits)	n/a	8	1.7E +/- 308 (15 digits)	n/a
Unambiguous						
uint8_t	1	n/a	0 to 255	1	n/a	0 to 255
int8_t	1	-128 to 127	n/a	1	-128 to 127	n/a
uint16_t	2	n/a	0 to 65,353	2	n/a	0 to 65,353
int16_t	2	+/- 32,767	n/a	2	+/- 32,767	n/a
uint32_t	4	n/a	0 - 4,294,967,295	4	n/a	0 - 4,294,967,295
int32_t	4	+/- 2,147,483,648	n/a	4	+/- 2,147,483,648	n/a

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

Copyright ©2021 CNM. All Rights Reserved.

Initializing MPU6050

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

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

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

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

Reading Acceleration Data from the MPU-6050

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

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

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L16_Motion

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

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

Copyright ©2021 CNM. All Rights Reserved.

Pitch and Roll Improved

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

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)

Copyright ©2021 CNM. All Rights Reserved.

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

Angular Rate
Cnahge in capacitance
Sensing Direction
Mass
Driving Direction
How To MECHATRONICS
www.HowToMechatronics.com

Deadband

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L16_Motion

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

1. L16_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. L16_06_DriveByWire
   - Connect the MPU-6050 to your system.
   - Obtain the z-axis rotation from the appropriate register on the MPU-6050.
   - Map the gyro movement to change in angular position (°per sec)
   - Have the stepper motor track movement in the gyroscope.

Copyright ©2021 CNM. All Rights Reserved.

Acceleration, Velocity, and Position

The reimann sum method can be used to "integrate" acceleration to velocity and velocity to position.

Copyright ©2021 CNM. All Rights Reserved.

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.

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L16_Motion

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

1. L16_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. L16_08_RPM
   - Place magnet on shaft of lathe.
   - Using Hall Sensor, measure time per rotation.
   - Convert to rotations per minute.
   - Display on Adafruit.io databoard.
   - EXTRA:
     - Create a speedometer using a servo motor. (The Servo class natively available).
     - Measure RPM on other equipment at FUSE.

Copyright ©2021 CNM. All Rights Reserved.

Piezoelectric Elements

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

Copyright ©2021 CNM. All Rights Reserved.

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).

Copyright ©2021 CNM. All Rights Reserved.

Galaxy S9+ Vibration Analysis

Using the FTT 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	-73.9953870941929-70.0944447984849i

The Galaxy S9 vibrates at 159.11 Hz

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L16_Motion

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

1. L16_09_Vibration
   - Connect the piezo sensor, a button, and a μSD module to your Argon.
   - Each time button is pressed, execute a loop 4096 times:
     - Every 500μsec, 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.).

Copyright ©2021 CNM. All Rights Reserved.

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

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

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

float weight, rawData, calibration;
int offset;

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L16_Motion (Learn to Calibrate)

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

1. L16_10_Scale
   - Set initial CAL_FACTOR to 1000 and measure a known weight. (Note: one cup of water (in 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.

Copyright ©2021 CNM. All Rights Reserved.

IFTTT

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

Copyright ©2021 CNM. All Rights Reserved.

ThingSpeak Dashboard and Matlab Analysis

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

ThingSpeak.h

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

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

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

void loop() {

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

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

  ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

}
```

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

L17_Bluetooth

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)

BLE GATT Client-Server model

- 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.
- GATT comes into play once a dedicated connection is established between two devices, meaning that you have already gone through the advertising process.

Copyright ©2021 CNM. All Rights Reserved.

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	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

ASCII: American Standard Code For Information Interchange

Argon BLE - UART Service

```
// These UUIDs were defined by Nordic Semiconductor and are now the defacto standard for
// UART-like services over BLE. Many apps support the UUIDs now, like the Adafruit Bluefruit app.
const BleUuid serviceUuid("6E400001-B5A3-F393-E0A9-E50E24DCCA9E");
const BleUuid rxUuid("6E400002-B5A3-F393-E0A9-E50E24DCCA9E");
const BleUuid txUuid("6E400003-B5A3-F393-E0A9-E50E24DCCA9E");

BleCharacteristic txCharacteristic("tx", BleCharacteristicProperty::NOTIFY, txUuid, serviceUuid);
BleCharacteristic rxCharacteristic("rx", BleCharacteristicProperty::WRITE_WO_RSP, rxUuid, serviceUuid, onDataReceived, NULL);
BleAdvertisingData data;

//onDataReceived is used to recieve data from Bluefruit Connect App
void onDataReceived(const uint8_t* data, size_t len, const BlePeerDevice& peer, void* context) {
    uint8_t i;
    Serial.printf("Received data from: %02X:%02X:%02X:%02X:%02X:%02X \n", peer.address()[0], peer.address()[1],peer.address()[2], peer.address()[3], peer.address()[4], peer.address()[5]);
    for (i = 0; i < len; i++) {
        Serial.printf("%c",data[i]);
    }
    Serial.printf("\n");
}
```

Argon BLE - UART Transmit Example

```
const int UART_TX_BUF_SIZE = 20;
uint8_t txBuf[UART_TX_BUF_SIZE];
uint8_t i;

void setup() {
    Serial.begin();

    BLE.on();
    Serial.printf("Argon BLE Address: %s\n",BLE.address().toString().c_str());

    BLE.addCharacteristic(txCharacteristic);
    BLE.addCharacteristic(rxCharacteristic);
    data.appendServiceUUID(serviceUuid);
    BLE.advertise(&data);
}

void loop() {
    for(i=0;i<UART_TX_BUF_SIZE-1;i++) {
        txBuf[i] = random(0x40,0x5B);        //Captial ASCII characters plus @
    }
    txBuf[UART_TX_BUF_SIZE-1]=0x0D;        //Append a carriage return to end of buffer
    txCharacteristic.setValue(txBuf, UART_TX_BUF_SIZE);
    for(i=0;i<UART_TX_BUF_SIZE;i++) {
        Serial.printf("%c",txBuf[i]);
    }
    Serial.printf("\n");
    delay(5000);
}
```

Copyright ©2021 CNM. All Rights Reserved.

Assignment: L17_BlueTooth

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

L17_01_BlueTooth

- Load Bluefruit Connect on your smart device.
- 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 bluetooth to Bluefruit Connect.
- Reset the encoder and NeoPixel ring to the appropriate state when values 0-11 are received from Bluefruit Connect.

Finishing Up

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 ($*$).

Copyright ©2021 CNM. All Rights Reserved.

Pointers

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

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

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

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

Pointers and Arrays

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Finding Average of an Array

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

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

Copyright ©2021 CNM. All Rights Reserved.

Finding Average of Arrays in Action

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

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

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

Array of Functions via Pointers

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

int a,b,answer,i;

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

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

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Returning Multiple Values from a Function

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

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Struct datatype and Member Operators

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

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

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved.

Linked Lists and Doubly Linked Lists

Singly Linked List

Doubly Linked List

Copyright ©2021 CNM. All Rights Reserved.

Inserting a "cell" into an Array

Copyright ©2021 CNM. All Rights Reserved.

Inserting a "cell" into a Linked List

GitHub - Part 3

Collaborating with Github

Using Git inside an IDE - Visual Studio Code

You can run Git commands from the Visual Studio Code terminal. Advanced IDEs (Interactive Development Environments) usually integrate with a VCS like Git.

1. Open a project folder in VS Code.
2. Open a terminal window in VS Code.
3. Choose your preferred Git command tool.
   - Windows Command
   - Windows Powershell
   - Git Bash → like Linux
4. Now type your Git and file/directory commands in the terminal.

This is easier than Powershell because you are already in your project folder so VS Code can show your Git status in the IDE.

Copyright ©2021 CNM. All Rights Reserved.

Typical GitHub workflow and branches

- Master or Main is the good code. This is generally the code that is also available on a production system.
- A production system is a live system that people are actually using.
- You don't want to push unfinished, untested code to your master or main branch.

Typical workflow:

Feature → development → staging/testing/release → master (main)/production

`https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow`

Practice: Branching and merging code

1. Create a test repo in GitHub and clone it to your local system.
2. Create a new VS Code project in your new repo. Write some code and push to your new repo.
3. Create a branch called "dev"
   - git branch dev → create the new branch
   - git checkout dev → switch to the new branch
   - git branch → confirm the branch you are now in.
4. Make changes to your test file.
5. commit and push your changes.
6. git checkout main → to switch back to the main branch.
7. git branch → to see what branch you are in.
8. git merge dev → to merge your dev branch changes into main.
9. git push → to push the merged changes to main
10. Look at your commit history to see how the merge was handled.

Review: Git commands

git clone → to copy a complete repo to your local machine.
git log → shows commit history for a repo.
git show <commit number> → shows details for a specified commit.
git diff <file name> → shows differences between file versions.
git status → shows what is staged for commit so you know what will be committed *BEFORE* you commit it.
git branch → to create a new branch or see what branch you are currently in.
git checkout <file name> → to rollback files to the current Git version.
git checkout <branch name> or **git switch** <branch name> → to change branches.
git pull → to get the latest code from a repo.
git add → to stage files to be committed.
git commit → to apply timestamp and version to files in a local repo.
git push → to upload commits to a remote repo.

Copyright ©2021 CNM. All Rights Reserved.

Collaboration in GitHub

- Collaboration allows multiple people (GitHub accounts) to work on the same project.
- For your capstones, you will need to add classmates as collaborators to your repo.
- Applying rules to the GitHub workflow helps avoid merge conflicts and problem code.
  - Communicate with other team members about changes you are making.
  - Segregate assignments if possible to avoid changing the same code at the same time.
  - **Always do a git pull from a branch** before merging to avoid having your branch ahead of a higher level branch.
  - Use a separate branch for development to avoid breaking the master or main branch of your teammates' code.
  - Thoroughly test your code before committing and pushing to GitHub.
  - Have a teammate do a code review before merging to a higher level branch.

Collaboration workflow

1. Share a project repo with all team members in GitHub.
2. Clone the shared repo to your local machine.
3. If you do not have a development or working branch, create one.
4. Switch to the new branch in the repo.
5. Do your development and testing.
6. Commit and push your branch to GitHub.
7. Notify the repo owner that your code is ready for review and merging to the next higher level branch.

Note: design a workflow that works for your team. Avoid working on the master or main branch. Reserve the master or main branch for final production-level, presentation-worthy code.

Practice: Collaborating

1. Add your teammate as a collaborator to your repo.
   - Settings > Manage Access > Invite Collaborator
2. Approve the invitation to collaborate in your email.
3. Confirm that you can now see your teammate's repo.
4. Clone your teammate's repo into your IoT folder.
5. Create a new branch for your development.
   - git branch yourname-dev
   - git checkout yourname-dev
6. Make a change to a file on your dev branch.
7. Commit and push your change.
8. Have your teammate confirm that they can see your new branch and change in their repo.

Pull Requests

A pull request is a safety feature in GitHub that enforces a code review before merging code from a downstream branch into an upstream branch.

A pull request is initiated in the GitHub GUI.

To enforce pull requests before merge

1. Go to Settings > Branches > Add Rule.
2. Choose "Require pull request reviews before merging."
3. Click "Create" to save the rule.

Note: some collaborator features may only be available with a GitHub Pro account.

Copyright ©2021 CNM. All Rights Reserved.

Practice: Approving and merging pull requests

1. Push your changes to your branch in GitHub.
2. Create a pull request for your branch in GitHub.
3. Have your teammate review and approve your pull request.
4. Have your teammate merge your pull request into an upstream branch in the repo.
5. View the result in GitHub.

Copyright ©2021 CNM. All Rights Reserved.

Handling merge conflicts

In spite of your best efforts, you will eventually encounter a merge conflict. This occurs when Git cannot figure out how to merge two files because the changes are too disparate.

1. Open the conflicted file in a text editor or VS Code or whatever IDE you are using.
2. Notice the merge conflict markers in the file.
   - <<<<<<< HEAD text in HEAD version ======= text in your version >>>>>>> your-branch
3. Select the text you want to keep and
4. Remove the text you want to delete.
5. Delete the conflict markers.
6. Save the adjusted file.
7. Commit and push to GitHub.
8. Go to GitHub and confirm that your branch now has the corrected file.

Additional Git resources

- **Installing git:** https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- **git cheat sheet:** https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf
- **GitHub documentation:** https://docs.github.com/en/github
- **Git workflow:** https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
- **What to do if it goes all wrong:** https://dangitgit.com/
- **Resolving merge conflicts:** https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/resolving-a-merge-conflict-using-the-command-line

Capstone

Copyright ©2021 CNM. All Rights Reserved.

Capstone Projects

Intent:

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

Guidelines:

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

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

Copyright ©2021 CNM. All Rights Reserved.

The Big Question: What about main()

```
#include <Arduino.h>

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

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

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


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

Copyright ©2021 CNM. All Rights Reserved.

Hello World - What a microcontroller sees

HelloWorld.ino

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

void loop() {}
```

Hex Code and Assembly Language

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

BONUS - CMOS

Copyright ©2021 CNM. All Rights Reserved

AND, OR, and NOT

Symbol

Truth Table

B	A	Q
0	0	0
0	1	0
1	0	0
1	1	1

Symbol

Truth Table

B	A	Q
0	0	0
0	1	1
1	0	1
1	1	1

Symbol

Truth Table

A	Q
0	1
1	0

Copyright ©2021 CNM. All Rights Reserved

NAND

NOR

Logic Gates using only NOR Gates
NOR Gate Symbol
A
B
Q = $\overline{A+B}$
NOT Gate
(Inverter)
A
$\overline{A}$
OR Gate
A
B
$\overline{A+B}$
A+B
Buffer
A
$\overline{A}$
A
AND Gate
A
B
$\overline{A}$
$\overline{B}$
A.B
Exclusive-OR
A
B
$\overline{A+B}$
Q
$\overline{A}$
$\overline{B}$
A.B
NAND Gate
A
B
$\overline{A}$
$\overline{B}$
A.B
$\overline{A.B}$
Exclusive-NOR
A
B
$\overline{A+B}$
$\overline{A}.B$
$A.\overline{B}$
Q

Copyright ©2021 CNM. All Rights Reserved

CMOS Inverter

WORK
IN PROGRESS

Copyright ©2021 CNM. All Rights Reserved

Thevenin Equivalent Circuits

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

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

Equivalent Circuit

Useful relationships

Copyright ©2021 CNM. All Rights Reserved

ThingSpeak Via Webhooks

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

1. Particle Argon
2. BME280

Copyright ©2021 CNM. All Rights Reserved

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.

Copyright ©2021 CNM. All Rights Reserved

JSON Parser Generator

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

```
#include <JsonParserGeneratorRK.h>

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

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

Copyright ©2021 CNM. All Rights Reserved

ThingSpeak Dashboard

Copyright ©2021 CNM. All Rights Reserved

Step 1 - Create ThingSpeak Channel

ThingSpeak™ | Channels | Apps | Support | Commercial Use | How to Buy | Account | Sign Out

My Channels

New Channel

Search by tag

Name	Created	Updated
FUSEMakerspace Private Public Settings Sharing API Keys Data Import / Export	2020-01-09	2020-03-27 16:04
Home IoT Plant Watering Private Public Settings Sharing API Keys Data Import / Export	2020-04-16	2020-04-16 19:56
Dew Point Measurement Private Public Settings Sharing API Keys Data Import / Export	2020-04-16	2020-04-19 13:38
Home Weather Station Private Public Settings Sharing API Keys Data Import / Export	2020-04-17	2020-04-17 18:52

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

Upgrade

Need to send more data faster?

Need to use ThingSpeak for a commercial project?

Upgrade

Copyright ©2021 CNM. All Rights Reserved

Step 2 - Get API Key

ThingSpeak™ | Channels | Apps | Support | Commercial Use | How to Buy | Account | Sign Out

Dew Point Measurement

Channel ID: **1039626**
Author: mwa0000017234878
Access: Private

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Write API Key

Key: BK1I6D1UTGPQ5B5H

Generate New Write API Key

Read API Keys

Key: SQG42O05TWWWF26R

Note:

Save Note | Delete API Key

Add New Read API Key

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=BK1I6D1UTGPQ5B
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/1039626/feeds.json?a
```

Read a Channel Field

```
GET https://api.thingspeak.com/channels/1039626/fields/1.jso
```

Read Channel Status Updates

```
GET https://api.thingspeak.com/channels/1039626/status.json?
```

Learn More

Copyright ©2021 CNM. All Rights Reserved

Step 3 - Create Webhook

From `console.particle.io`:

Copyright ©2021 CNM. All Rights Reserved

Step 4 - Add JSON Data

Particle

Personal

Integrations > Edit Integration

WEBHOOK BUILDER CUSTOM TEMPLATE

Read the Particle webhook guide

Event Name
env-vals

URL
https://api.thingspeak.com/update

Request Type
POST

Request Format
Web Form

Device
Herbert

Advanced Settings

For information on dynamic data that can be sent in any of the fields below, please visit our docs.

FORM FIELDS

Default Custom

api_key	xxxxxxxxxxxxxxxxxx
field1	{{{Moisture}}}
field2	{{{Temperature}}}
field3	{{{Pressure}}}
field4	{{{Humidity}}}
field5	{{{Plant Watered}}}

Copyright ©2021 CNM. All Rights Reserved

Step 5 - Particle Cloud Events

Copyright ©2021 CNM. All Rights Reserved

Step 6 - Create Channel

Home IoT Plant Watering

Channel ID: **1039355**
Author: mwa0000017234878
Access: Public

Plant Watering in Home IoT Classroom

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import

Channel Settings

Percentage complete	50%
Channel ID	1039355
Name	Home IoT Plant Watering
Description	Plant Watering in Home IoT Classroom
Field 1	Moisture ☑
Field 2	Temperature ☑
Field 3	Pressure ☑
Field 4	Humidity ☑
Field 5	Watered ☑
Field 6	☐
Field 7	☐
Field 8	☐
Metadata	JSON

Help

Channels
eight field
status dat
visualize i

Chanr

- Per
cha
cha
- Cha
- Des
- Fiel
cha
- Met
- Tag
- Lin
Thi
- Sho
- Vid
info

Copyright ©2021 CNM. All Rights Reserved

Step 7 - Create Dashboard

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

Copyright ©2021 CNM. All Rights Reserved

NCD.io ControlEverywhere

Copyright ©2021 CNM. All Rights Reserved

NCDio TMG3993 Proximity/Color Sensor

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

Copyright ©2021 CNM. All Rights Reserved

TMG3993 Initialization

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

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

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

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

Copyright ©2021 CNM. All Rights Reserved

NCDio ACD121C MQ9 CO Sensor

12-Bit Analog to Digital Conversion

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

Copyright ©2021 CNM. All Rights Reserved

NCDio ACD121C MQ131 Ozone Sensor

12-Bit Analog to Digital Conversion

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

Copyright ©2021 CNM. All Rights Reserved

NCDio PECMAC Current Sensor

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

GitHub

Copyright ©2021 CNM. All Rights Reserved.

Git and GitHub: Your Version Control Friends

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.

Why use a version control system?

Before Version Control Systems.

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

With Git.

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

Git vs. GitHub

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

→ git commit

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

→ git push

Commit vs. Push

Commit

- File saved to **LOCAL** repository. File is given a timestamp and a unique commit number that is the file version.

Push

- File is saved to **REMOTE** repository.

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

Installing Git on Windows

There are two good ways to install Git on Windows.

1. The official build is available at `https://git-scm.com/download/win`.
2. You can also use the GitHub GUI tool at `https://windows.github.com`. During the installation of this tool you can choose to install command-line tools as well (i.e. Git's command-line interface).

Installing Git on Mac

There are three ways to install Git on Mac.

1. A Git installer is available at `https://git-scm.com/download/mac`.
2. You can download the GitHub GUI tool at `https://mac.github.com`. During the installation of this tool, you can choose to install command-line tools as well (i.e. Git's command-line interface).
3. You can install the Xcode Command-Line Tools. On Mavericks (10.9) or above, you can do this simply by trying to run Git from the terminal the very first time. If you don't have it installed already, it will prompt you to install it.

Installing Git on Linux

You can use the package management system that comes with your Linux distribution.

On Debian based distributions like Ubuntu, use apt-get:

```
brian:~$ sudo apt-get install git-all
```

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 PAC

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

To create a PAC:

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 PAC rather than your password to access your account.

GitHub Authentication: Switching from password to PAC

If you have been connecting to GitHub from the command line using your password, you will need to switch to using your PAC.

On Windows 10:

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

On Mac OSx:

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

The next time you attempt a git command on the command line, you will be prompted to enter your username and password. Use your PAC in place of your password.

GitHub Authentication: Using SSH and a Personal SSH Key

Another option for connecting to your GitHub account via command line is by authenticating with an SSH key.

Below, you can find instructions for doing so. After clicking the below link, locate the instructions specific to your operating system.

`https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh`

Using Git: Command line or GUI?

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

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

The command line

Windows

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

Mac or Linux

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

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

The repository - aka the repo

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

The readme, license, .gitignore files

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

Practice: Create a repo in GitHub

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

Practice: Clone the repo to your local machine

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

Practice: Committing and pushing changes

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

README.md: some common markdowns

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

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

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

Practice: customizing your README.md file

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

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

Practice: View changes

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

The .gitignore file

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

Examples:

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

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

Practice: Creating and using a .gitignore file

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

What to do if you accidentally COMMIT a sensitive file

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

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

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

What to do if you accidentally PUSH a sensitive file

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

FIRST CHANGE THE PASSWORD OR INFORMATION IMMEDIATELY.

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

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

Typical GitHub workflow and branches

- Master or Main is the good code. This is generally the code that is also available on a production system.
- A production system is a live system that people are actually using.
- You don't want to push unfinished, untested code to your master or main branch.

Typical workflow:

Feature → development → staging/testing/release → master (main)/production

`https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow`

Copyright ©2021 CNM. All Rights Reserved.

Practice: Branching and merging code

1. Create a branch called "dev"
   - git branch dev → create the new branch
   - git checkout dev → switch to the new branch
   - git branch → confirm the branch you are now in.
2. Make changes to your test file.
3. commit and push your changes.
4. git checkout main → to switch back to the main branch.
5. git branch → to see what branch you are in.
6. git merge dev → to merge your dev branch changes into main.
7. git push → to push the merged changes to main
8. Look at your commit history to see how the merge was handled.

Copyright ©2021 CNM. All Rights Reserved.

Using Git inside an IDE - Visual Studio Code

You can run Git commands from the Visual Studio Code terminal. Advanced IDEs (Interactive Development Environments) usually integrate with a VCS like Git.

1. Open a project folder in VS Code.
2. Open a terminal window in VS Code.
3. Choose your preferred Git command tool.
   - Windows Command
   - Windows Powershell
   - Git Bash → like Linux
4. Now type your Git and file/directory commands in the terminal.

This is easier than Powershell because you are already in your project folder so VS Code can show your Git status in the IDE.

Review: Git commands

git clone → to copy a complete repo to your local machine.
git log → shows commit history for a repo.
git show <commit number> → shows details for a specified commit.
git diff <file name> → shows differences between file versions.
git status → shows what is staged for commit so you know what will be committed *BEFORE* you commit it.
git branch → to create a new branch or see what branch you are currently in.
git checkout <file name> → to rollback files to the current Git version.
git checkout <branch name> or **git switch** <branch name> → to change branches.
git pull → to get the latest code from a repo.
git add → to stage files to be committed.
git commit → to apply timestamp and version to files in a local repo.
git push → to upload commits to a remote repo.

Collaboration in GitHub

- Collaboration allows multiple people (GitHub accounts) to work on the same project.
- For your capstones, you will need to add classmates as collaborators to your repo.
- Applying rules to the GitHub workflow helps avoid merge conflicts and problem code.
  - Communicate with other team members about changes you are making.
  - Segregate assignments if possible to avoid changing the same code at the same time.
  - **Always do a git pull from a branch** before merging to avoid having your branch ahead of a higher level branch.
  - Use a separate branch for development to avoid breaking the master or main branch of your teammates' code.
  - Thoroughly test your code before committing and pushing to GitHub.
  - Have a teammate do a code review before merging to a higher level branch.

Collaboration workflow

1. Share a project repo with all team members in GitHub.
2. Clone the shared repo to your local machine.
3. If you do not have a development or working branch, create one.
4. Switch to the new branch in the repo.
5. Do your development and testing.
6. Commit and push your branch to GitHub.
7. Notify the repo owner that your code is ready for review and merging to the next higher level branch.

Note: design a workflow that works for your team. Avoid working on the master or main branch. Reserve the master or main branch for final production-level, presentation-worthy code.

Practice: Collaborating

1. Add your teammate as a collaborator to your repo.
   - Settings > Manage Access > Invite Collaborator
2. Approve the invitation to collaborate in your email.
3. Confirm that you can now see your teammate's repo.
4. Clone your teammate's repo into your IoT folder.
5. Create a new branch for your development.
   - git branch yourname-dev
   - git checkout yourname-dev
6. Make a change to a file on your dev branch.
7. Commit and push your change.
8. Have your teammate confirm that they can see your new branch and change in their repo.

Pull Requests

A pull request is a safety feature in GitHub that enforces a code review before merging code from a downstream branch into an upstream branch.

A pull request is initiated in the GitHub GUI.

To enforce pull requests before merge

1. Go to Settings > Branches > Add Rule.
2. Choose "Require pull request reviews before merging."
3. Click "Create" to save the rule.

Note: some collaborator features may only be available with a GitHub Pro account.

Practice: Approving and merging pull requests

1. Push your changes to your branch in GitHub.
2. Create a pull request for your branch in GitHub.
3. Have your teammate review and approve your pull request.
4. Have your teammate merge your pull request into an upstream branch in the repo.
5. View the result in GitHub.

Handling merge conflicts

In spite of your best efforts, you will eventually encounter a merge conflict. This occurs when Git cannot figure out how to merge two files because the changes are too disparate.

1. Open the conflicted file in a text editor or VS Code or whatever IDE you are using.
2. Notice the merge conflict markers in the file.
   - <<<<<<< HEAD text in HEAD version ======= text in your version >>>>>>> your-branch
3. Select the text you want to keep and
4. Remove the text you want to delete.
5. Delete the conflict markers.
6. Save the adjusted file.
7. Commit and push to GitHub.
8. Go to GitHub and confirm that your branch now has the corrected file.

Additional Git resources

- **Installing git:** https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- **git cheat sheet:** https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf
- **GitHub documentation:** https://docs.github.com/en/github
- **Git workflow:** https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
- **What to do if it goes all wrong:** https://dangitgit.com/
- **Resolving merge conflicts:** https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/resolving-a-merge-conflict-using-the-command-line