SEICHE 2023
Intermediate Arduino Programming for IoT

Instructor: Paul Frommeyer

www.paulfrommeyer.com

Corporate Sponsor: DXC Technology

REMINDER

YOU MUST HAVE A *WORKING* LED MATRIX DISPLAY TO TAKE THIS CLASS!

If displays are lost or damaged, replacements are $70 with 1-week lead time for replacement

Lesson Plan Overview

Lesson 1: 7Feb23 – Review, Addressing, and Pointers

- Overview of lesson plan
- Class Exercise: Validation of sketch uploading
- Class Exercise: Validation of binary create and upload
- More on the Preprocessor
- Memory Organization and Variables
- Review of addressing and pointers
- Using pointers to variables
- Declaring pointers in function calls
- Call-by-value Function Calls
- Call-by-reference Function Calls

Lesson 2: 14Feb23 – More Gory Details

- Class Exercise: Call-by-reference variables
- Introduction to Complex Data Structures
- Using complex data structures
- Storage declarations and Arduino

Lesson 3: 21Feb23 – Fonts Redux

- MD_Parola library font usage review
- The MD_Parola font format
- Designing your own fonts
- Declaring your own fonts
- Using your own fonts with MD_Parola

Lesson 4: 28Feb23 – Intro to Visual Studio Code

- Introduction to VSC
- https://code.visualstudio.com/docs/introvideos/overview
- VSCode Installation
- VSCode Integration with Arduino and ESP8266

Lesson 5: 7Mar23 – Filesystems

- Introduction to mass storage filesystems
- The SD FAT, FAT16, and FAT32 filesystems
- The exFAT filesystem
- Introduction to SPIFFS
- Class Exercise: Using SPIFFS

Lesson 6: 21Mar23 – WiFi

- Review of WiFi technology (but no gory details)
- Class Exercise: Review of WiFi *access* with WiFi Manager
- Web server and client HTTP architecture
- Introduction to HTML
- Creating a basic ESP8266 web server

Lesson 7: 28Mar23 – Web Technology

- More on HTML tags
- HTTP and Mime Types
- Using SPIFFS
- Class Exercise: Display A Web Page

Lesson 8: 4Apr23 – Doing More With Web Servers

- HTML Form Data
- Asynchronous Web Server Installation
- Class Exercise: Displaying Sensor Data With Async Web Server
- Class Exercise: Retrieving Form Data with Async Web Server

Lesson 9: 11Apr23 – Wifi Manager, API's, Web Scraping

- Class Exercise: Displaying form data with MD_Parola
- APIs
- Web Scraping

Lesson 10: 18Apr23 – JSON and REST API's

- Introduction to JSON
- Using JSON with Arduino
- Using REST with Arduino

Lesson 11: 25Apr23 – Internet Access

- Git and Github Quick Tour
- Class Exercise – Marquee Scroller Software
- Obtaining weather information from the Internet
- Class Exercise: Obtaining weather and news API keys
- Class Exercise: Getting and displaying weather information

Lesson 12: 2May23 – IoT Messaging and MQTT

- Introduction to IoT network messaging and MQTT
- Subscribing to an MQTT service
- Publishing to an MQTT service
- Class Exercise: Using MQTT and WiFi

Flash Drive Contents Reminder

- When I update parts of your flash drives, you can either *recopy the entire section*, or *just copy the updated part*.
- However, **you will need to explicitly copy any new files so that they are locally accessible on your laptops**
- Copying just *lesson* files will– wait for it!– *copy only the lesson files.*

Make certain you have copied everything in RED!

You should already have previously copied everything else!

FLASH DRIVE

- Archive–BasicProgrammingWith Arduino-Fall2022
- Archive-IntroductionToArduino-Spring2022
- IntermediateProgramming-Software
- IntermediateProgramming-Lessons
- IntermediateProgramming-Documentation

IntermediateProgramming-Software

ESP8266FS-0.5.0.zip

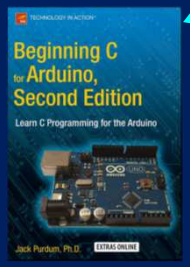
IntermediateProgramming-Lessons

- IntermediateProgramming-Lesson1
- IntermediateProgramming-Lesson2
- IntermediateProgramming-Lesson3
- IntermediateProgramming-Lesson11

IntermediateProgramming-Documentation

- ArduinoReference
  - **Beginning C for Arduino, 2nd Edition**
- BoardsGuides
- D1 Mini Reference
- ElectronicsReference
- IoT Reference
- LED Matrix Display Architecture
- HackSpace Magazine

Can't do reading homework without the files!



Lesson 10 – Github and Internet APIs

Part A

- Git Overview
- Github Overview
- Class Exercise – Marquee Scroller
  - Library Dependencies
  - Adding ZIP libraries directly to a sketch

Part B

- Obtaining weather information from the Internet
  - Open weathermap – openweathermap.org
- Obtaining news information from the Internet
  - Open news API - newsapi.org
- Class Exercise – Obtaining API Keys
  - Openweathermap
  - Newsapi.org
- Class Exercise – Entering API keys into displays

Version Control and Git

- In software engineering, version control (also known as revision control, source control, or source code management) is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information.
- Version control is a component of software configuration management.
- Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". For example, an initial set of files is "revision 1". When the first change is made, the resulting set is "revision 2", and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and, with some types of files, merged.
- The need for a logical way to organize and control revisions has existed for almost as long as writing has existed, but revision control became much more important, and complicated, when the era of computing began.
- In software development, distributed version control (also known as distributed revision control) is a form of version control in which the complete codebase, including its full history, is mirrored on every developer's computer.
- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers collaboratively developing source code during software development.
- Git's goals include speed, data integrity, and support for distributed, non-linear workflows.
- Git was originally authored by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development.
- As with most other distributed version control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server.
- Git's main web site is at https://git-scm.com

Source: Wikipedia and Git (https://git-scm.com/)

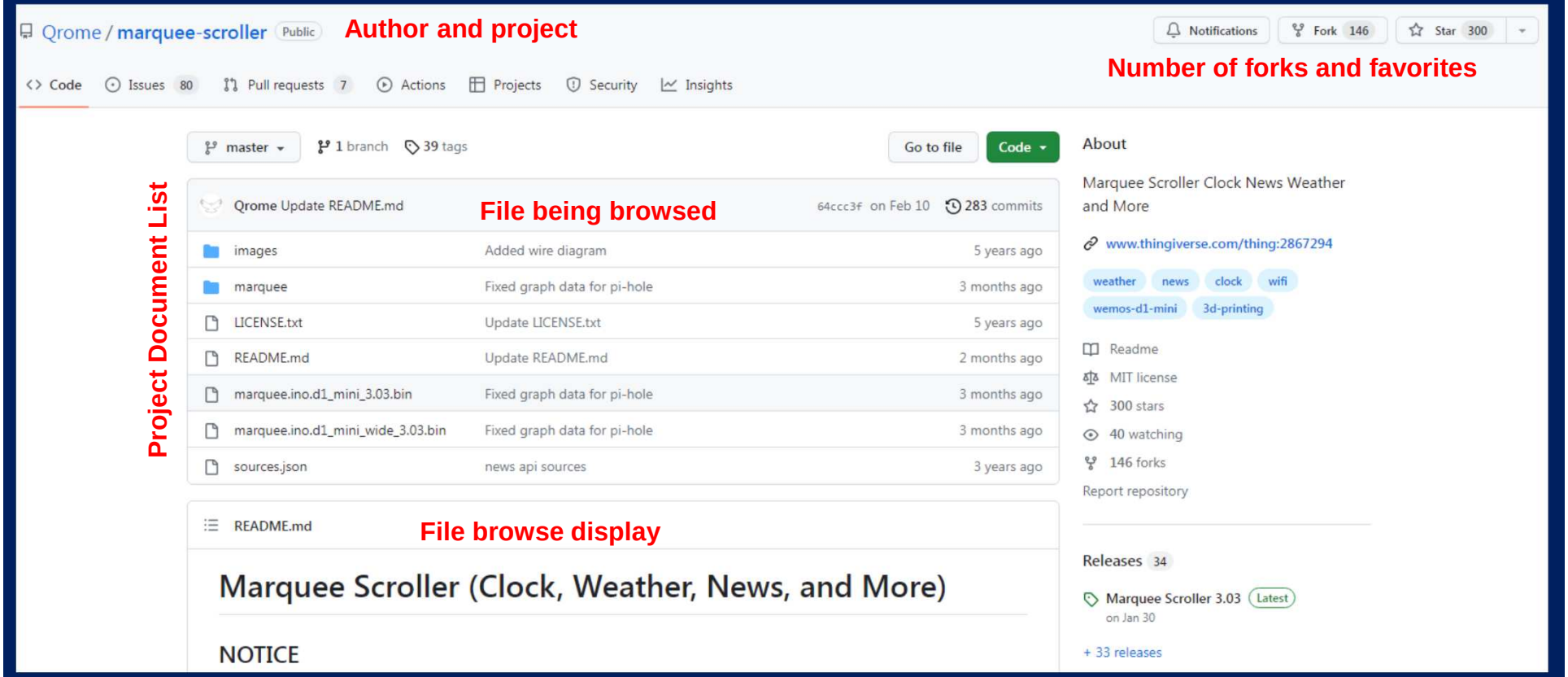
Github

- GitHub, Inc. (/ˈgɪthʌb/[a]) is an Internet hosting service for software development and version control using Git.
- It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.
- It is commonly used to host open source software development projects. As of January 2023, GitHub reported having over 100 million developers and more than 372 million repositories,including at least 28 million public repositories. It is the largest source code host as of November 2021.
- it has been a subsidiary of Microsoft since 2018. Thus, VS Code has native Github integration.
- Projects on GitHub.com can be accessed and managed using the standard Git command-line interface; all standard Git commands work with it.
- Anyone can browse and download public repositories but only registered users can contribute content to repositories. With a registered user account, users are able to have discussions, manage repositories, submit contributions to others' repositories, and review changes to code.
- The main purpose of GitHub.com is to facilitate the version control and issue tracking aspects of software development. Labels, milestones, responsibility assignment, and a search engine are available for issue tracking. For version control, Git (and by extension GitHub.com) allows pull requests to propose changes to the source code.
- Users with the ability to review the proposed changes can see a diff of the requested changes and approve them. In Git terminology, this action is called "committing" and one instance of it is a "commit." A history of all commits is kept and can be viewed at a later time.

Source: Wikipedia

CLASS EXERCISE – Marquee Scroller

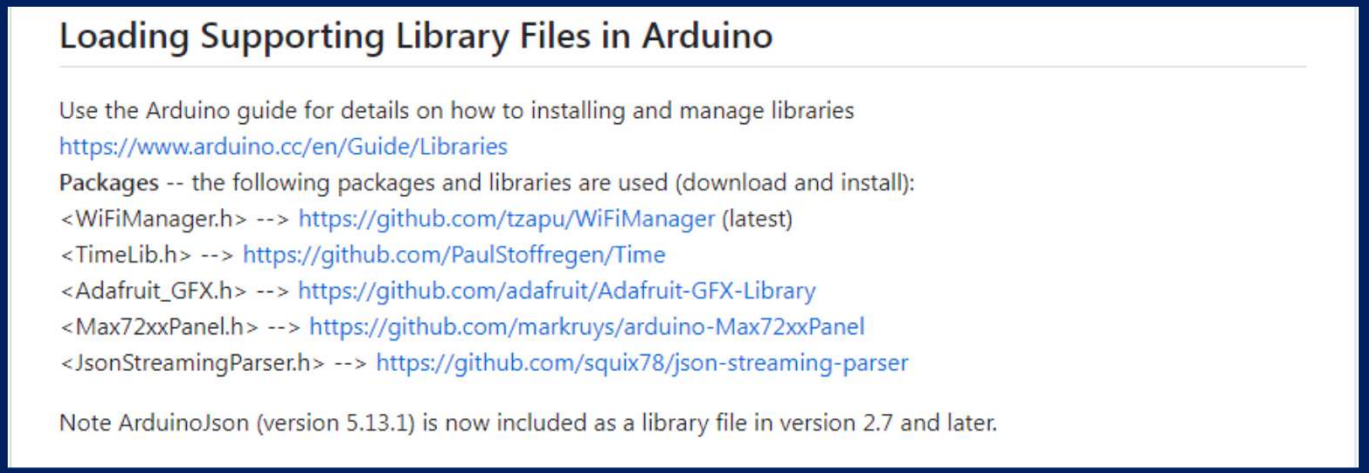
- Marquee Scroller is software written by Github member “Qrome”. It is available at https://github.com/qrome/marquee-scroller
- Let’s take a look at the main project homepage



Source: https://github.com/qrome/marquee-scroller

Marquee Scroller (cont.)

As we might expect, Marquee Scroller requires a set of libraries. These are documented in the README.md on Github



Loading Supporting Library Files in Arduino

Use the Arduino guide for details on how to installing and manage libraries
https://www.arduino.cc/en/Guide/Libraries
Packages -- the following packages and libraries are used (download and install):
<WiFiManager.h> --> https://github.com/tzapu/WiFiManager (latest)
<TimeLib.h> --> https://github.com/PaulStoffregen/Time
<Adafruit_GFX.h> --> https://github.com/adafruit/Adafruit-GFX-Library
<Max72xxPanel.h> --> https://github.com/markruys/arduino-Max72xxPanel
<JsonStreamingParser.h> --> https://github.com/squix78/json-streaming-parser

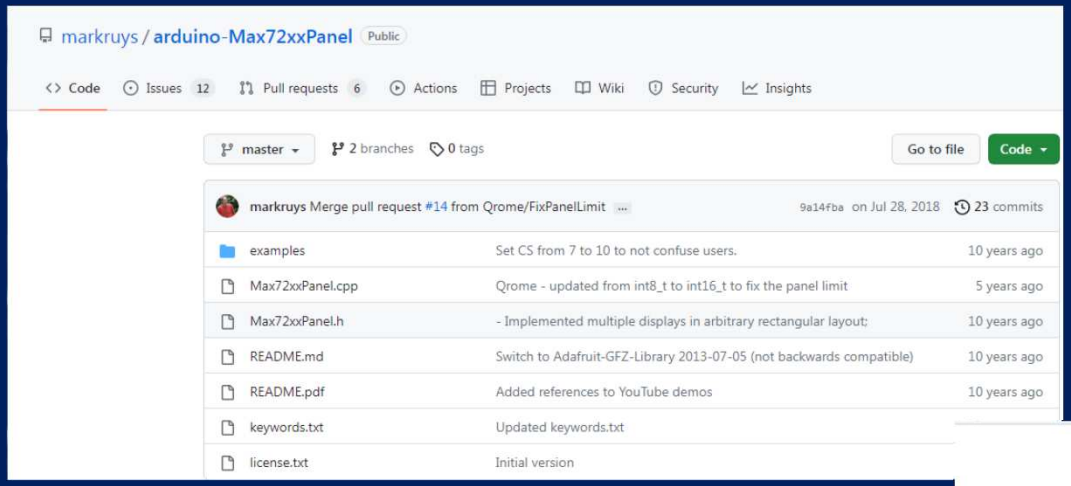
Note ArduinoJson (version 5.13.1) is now included as a library file in version 2.7 and later.

- **We have already loaded ArduinoJson in our previous class**
- **We should also already have loaded Adafruit_GFX in our Spring 2022 semester**
- **So we will need to load these remaining libraries, using these methods:**
  - **WifiManager – IDE**
  - **TimeLib – IDE**
  - **Max72XXPanel – Sketch ZIP file (to be discussed)**
  - **JsonStreamingParser – Sketch ZIP file (to be discussed)**

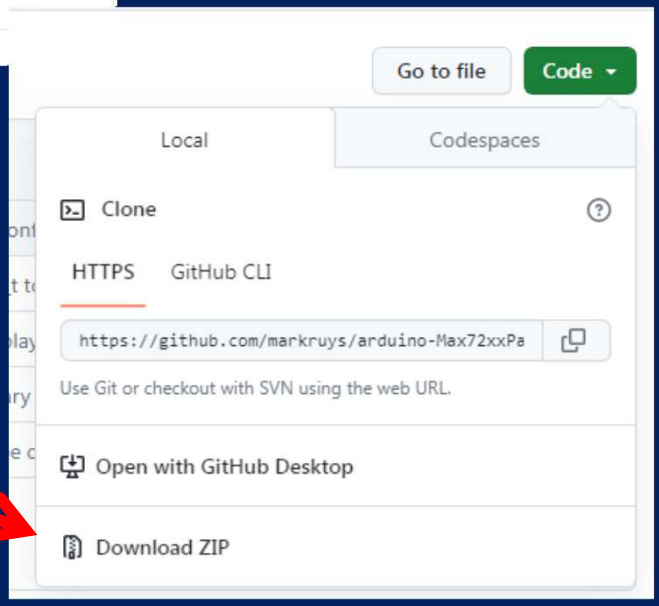
Source: https://randomnerdtutorials.com/decoding-and-encoding-json-with-arduino-or-esp8266/

Github – Obtaining Libraries

- Let's take a moment and look at the Github project page for the Max72xxPanel library.
- The Marquee Scroller readme indicated we need to load the Max72xx Panel library, but looking at the home page, where IS it???

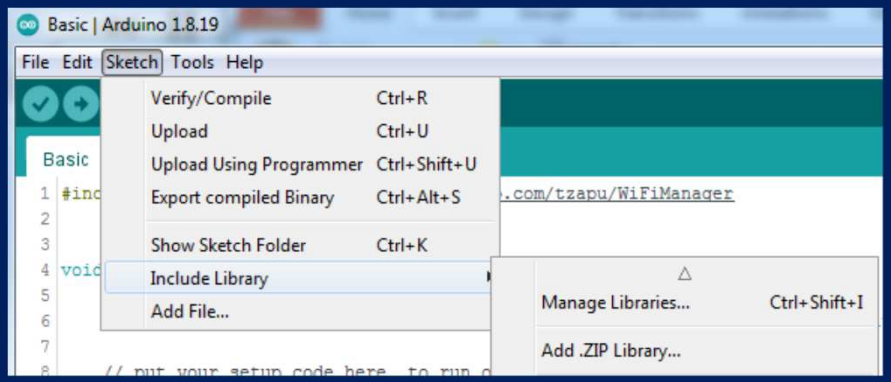


- The answer is, "you're looking at it." ☺ It's the *entire* project!
- What we need to do is download *the whole project* as a ZIP archive. And there is an easy way to do that!
- However, I've already downloaded the necessary ZIP files for you, so no need to download them now. Just be aware of how to do it.

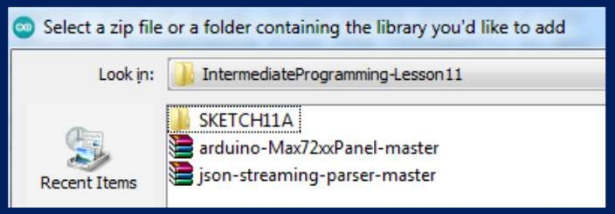


Adding ZIP Archives to a sketch

- Open SKETCH11A – **DO NOT UPLOAD IT**
- In the Sketch menu, select “Include Library-> Add .ZIP Library”



- Navigate to your Lesson11 folder and add **arduino-Max72xxPanel-master**
- Repeat the process for adding a ZIP library, and this time add json-streaming-parser-master



Marquee Scroller (cont.)

- **FIRST, let's VALIDATE the sketch (don't upload it yet.) You may see some orange *error messages* in your IDE log window, but that's OK. The Marquee Scroller code is not fully compliant with modern Arduino C++ syntax, but it will work OK.**
- **If you received "missing library" errors, you'll need to sort them. Engage Your Instructor for help if needed.**
- **If your validation completed successfully, you should see a "Done Compiling" status and a "Code In Flash" message in the IDE.**
- **And NOW you can upload your sketch; it will compile again, of course.**

Source: https://randomnerdtutorials.com/decoding-and-encoding-json-with-arduino-or-esp8266/

Marquee Scroller (cont.)

- The Marquee Scroller uses code called “WiFi Manager” that is a little different that the one we’ve used before
- The MS works like the WLED software we used in Spring 2022; if it DOES FIND a preconfigured WiFi network, it will join that network and print the IP address received on the display.
- If it DOES NOT find a preconfigured WiFi SSID, then it will go into “self-AP” mode and offer it’s own SSID, which you will need a Wifi capable client and browser to connect to, just like with WLED.
- At CFC, all of our displays should come up in AP mode the first time. As with WLED, it may be tricky figuring out which display is which because they will all be named the same. Best advice is to arrange to take turns, and I’ll go first to demonstrate.
- Once you have connected to your display, it should take you to a captive portal page (again functionally like WLED) and present you with a login/configuration screen
- The default credentials for Marquee Scroller are username “admin” and password “password”.
- It will also want a bunch of other stuff, and we will get to that next

Obtaining Internet Information

- We have talked previously about various services offering API's via the Internet that can be access via Arduino and other microcontrollers
- Two of the most popular are weather and news information
- A very popular weather API is openweathermap.org. While they went commercial a few years back, they still offer free API keys for personal and educational use. There are other services, but the Marquee Scroller code is designed to use openweathermap, thus we will too.
- Likewise a popular news headline site is News API, newsapi.org, which is what the Marquee Scroller code is configured to use.

Obtaining API Keys

- Nearly all Internet remote API services require users to register to obtain *authentication keys* that identify the user and their access level to the service.
- To obtain an openweathermap API key, you will need to browse to https://home.openweathermap.org/users/sign_up
- You will need to be 16 years of age or older to sign up. If you are not, you'll need to have a parent or guardian do this for you– or get a key from a hacker friend who *is* over 16. ☺
- I'll do a walkthrough of this process in class
- Once you have an API Key, you can enter it into the configuration page for your display running Marquee Scroller software
- The process is similar for News API, and will do a walkthrough of that as well.
- And that's it! Once you have entered your API keys, you should start receiving news and weather information on your displays. Just like having you own mini Times Square jumbotron, right?

Formal End of Lesson 9

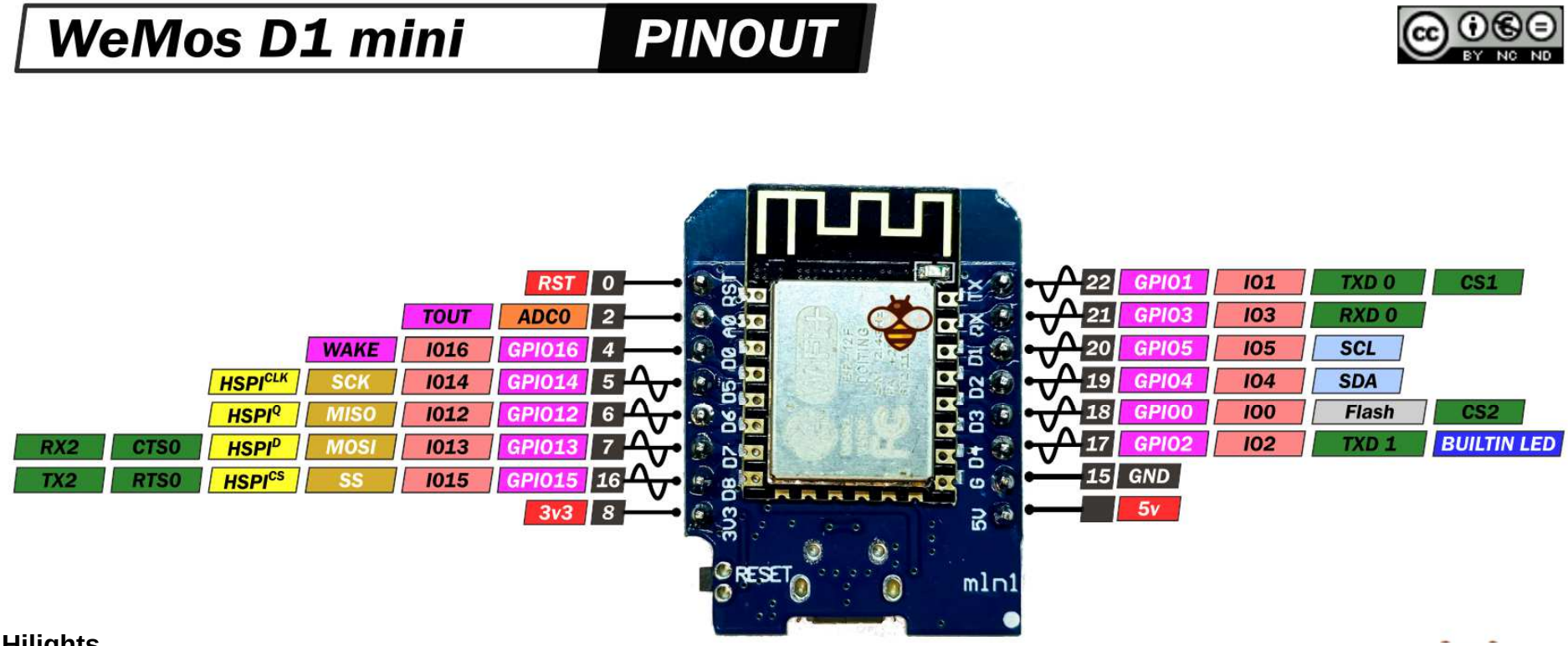
HOMEWORK

- **No Homework This Week**

In next week's exciting episode

- Scraping information from an Internet web page
- Obtaining weather information from the Internet
- Class Exercise: Getting and displaying weather information

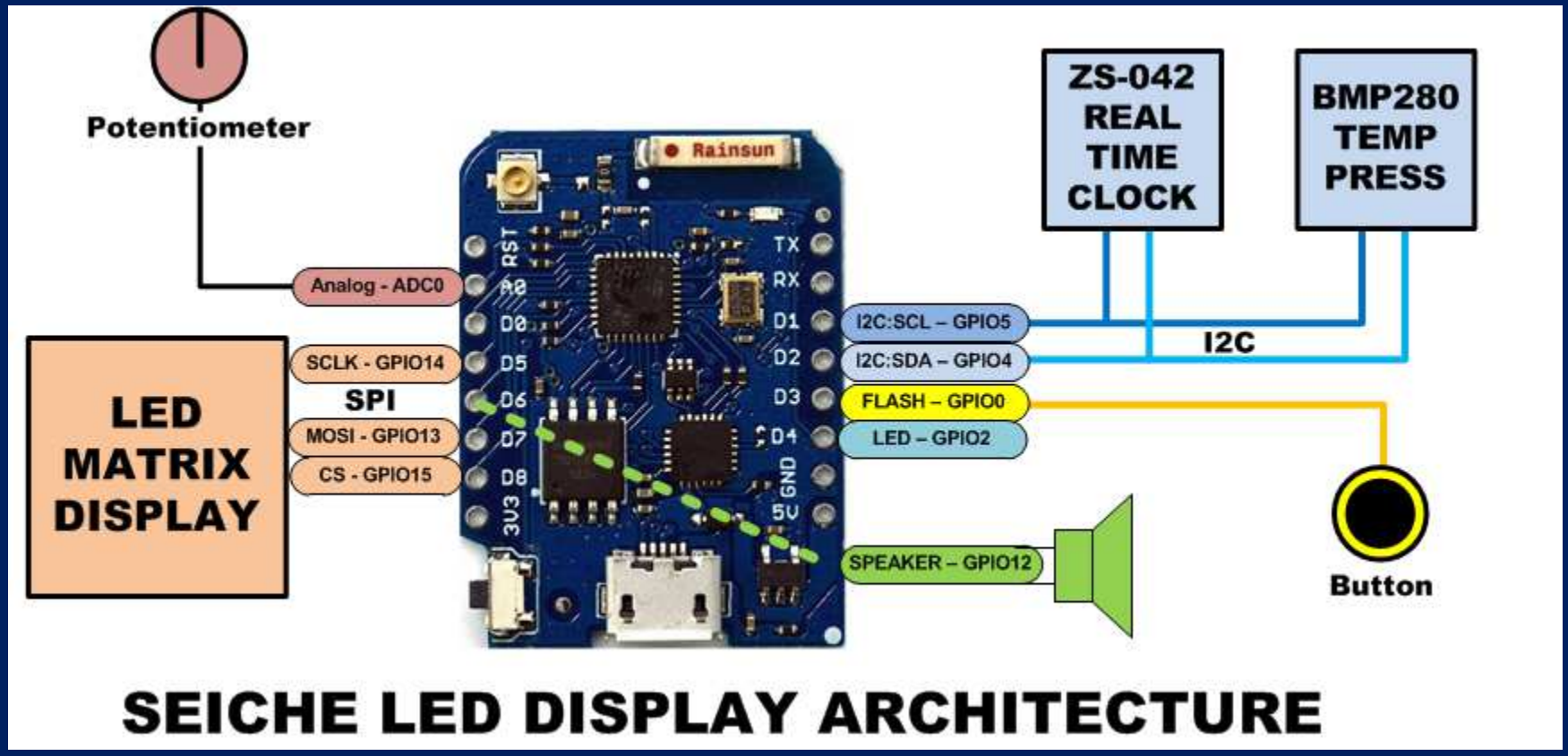
Our Microcontroller: The WeMos D1 Mini



Hilights

- **11 digital IO: all are interrupt and pwm capable (except D0/GPIO16)**
- **1 analog input (3.2V max input): A0/ADC0**
- **Micro USB or Type-C USB Port (clones usually have micro USB)**
- **Two SPI interfaces (one is used for on-board flash memory), one I2C interface, two serial ports**
- **Built-in WiFi (client or standalone access point modes) and Bluetooth**
- **Compatible with MicroPython, Arduino, NodeMCU**
- **Uses the CH340 USB-to-serial driver (installation usually needed on Windows)**
- **Extremely low cost (approx $3.00 US on Amazon; one of the two least expensive components in your kits)**

SEICHE LED Display Architecture



- Red MAX7219 8x32 LED matrix display (SPI)
- ZS-042 real-time clock module (I2C)
- BMP280 Temperature and Pressure Sensor (I2C)
- Piezoelectric speaker (PWM)
- 10KΩ Potentiometer (Analog-to-Digital Converter)
- Button (Pullup and interrupt)

`sprintf()` Function Syntax

- `sprintf()`'s formal syntax is:
  `sprintf(`char ***`buffer`**, const char ***`format`**, *variable list*`)`
- **`buffer`** is an array of type char (the parameter when passed can also be a pointer [address] for such an array) which will contain the formatted output of the function
- **`format`** is an unmodifiable (constant) array of char containing the format descriptors for all variables subsequently passed to the function, which is to say, it's the *format string*.
- The variable list are just the comma separated variables that are to be formatted
- All variables passed in a single call to `sprintf()` are combined into a single output string
- `sprintf()` automagically puts a terminating NULL (\0) at the end of the ASCII output

`sprintf()` Format Strings

- `sprintf()` format strings contain conversion specifiers that specify how each individual variable is to be converted
- `sprintf()` conversion specifiers have the following general syntax (all begin with a percent-sign):

  %[flags][minimum field width][.][precision][length][conversion character]

  - % - special token that indicates the start of a conversion specifier
  - Flags – these modify the behavior of the specification
  - Minumum field width – as it says on the tin; this the minimum number of characters to be converted
  - . – The period is a separator between field width and precision
  - Precision – means one of the following depending on the variable type and conversion specifier
    - The maximum number of characters to be generated from a string
    - The number of digits after the decimal point for type float conversions (e, E, or f)
    - The zero-filled minimum number of digits for an integer
  - Conversion character – A single character which determines the output type of the conversion specifier; a single character that specifies the type of output format for the corresponding data or variable

`sprintf()` Conversion Specifiers

Below are the general conversion specifiers and what they do.

Specifier	What it does
d, i	int - integer; signed decimal notation
o	int – unsigned octal (no leading zero)
x, X	int – unsigned hexadecimal, no leading 0x
u	int – unsigned decimal
c	int – single character, after conversion to unsigned char
s	char * - characters from string are printed until \0 (NULL) or *precision* is reached
f	double – decimal notation of form [-]mmm.ddd where number of decimals is specified by precision; precision of zero (0) suppresses the decimals altogether
e, E	double - exp notation; default precision of 6, 0 suppresses
g, G	double – Use %f for <10^4 or %e for >10^4
p	void * - print output as a pointer, platform dependent
n	Number of characters generated so far; goes into output
%	No conversion, put a % percent sign in the output

`sprintf()` Conversion Specifiers

Below are the flags and what they do.

Flag	What it does
-	Left justification
+	Always print number with a sign
spc (space)	Prefix a space if first character is not a sign
0 (zero)	Zero fill left for numeric conversions
#	Alternate output form depending on conversion character o – first digit will be zero x or X – 0x or 0X (respectively) prefixed to non-zero results e, E, f, g and G – Output will always have a decimal point g and G – trailing zeroes will never be removed