SEICHE 2023
Intermediate Arduino Programming for IoT

Instructor: Paul Frommeyer

www.paulfrommeyer.com

Corporate Sponsor: DXC Technology

REMINDER

YOU <u>MUST</u> 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

Lesson 12: 2May23 – API Keys and Gradution

- Class Exercise: Getting and displaying weather information
- Trimester Review and Graduation

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 <u>everything else</u>!

FLASH DRIVE

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

IntermediateProgramming-Software

ESP8266FS-0.5.0.zip

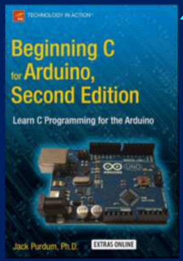
IntermediateProgramming-Lessons

- 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 12 – API Keys and Graduation

Part A

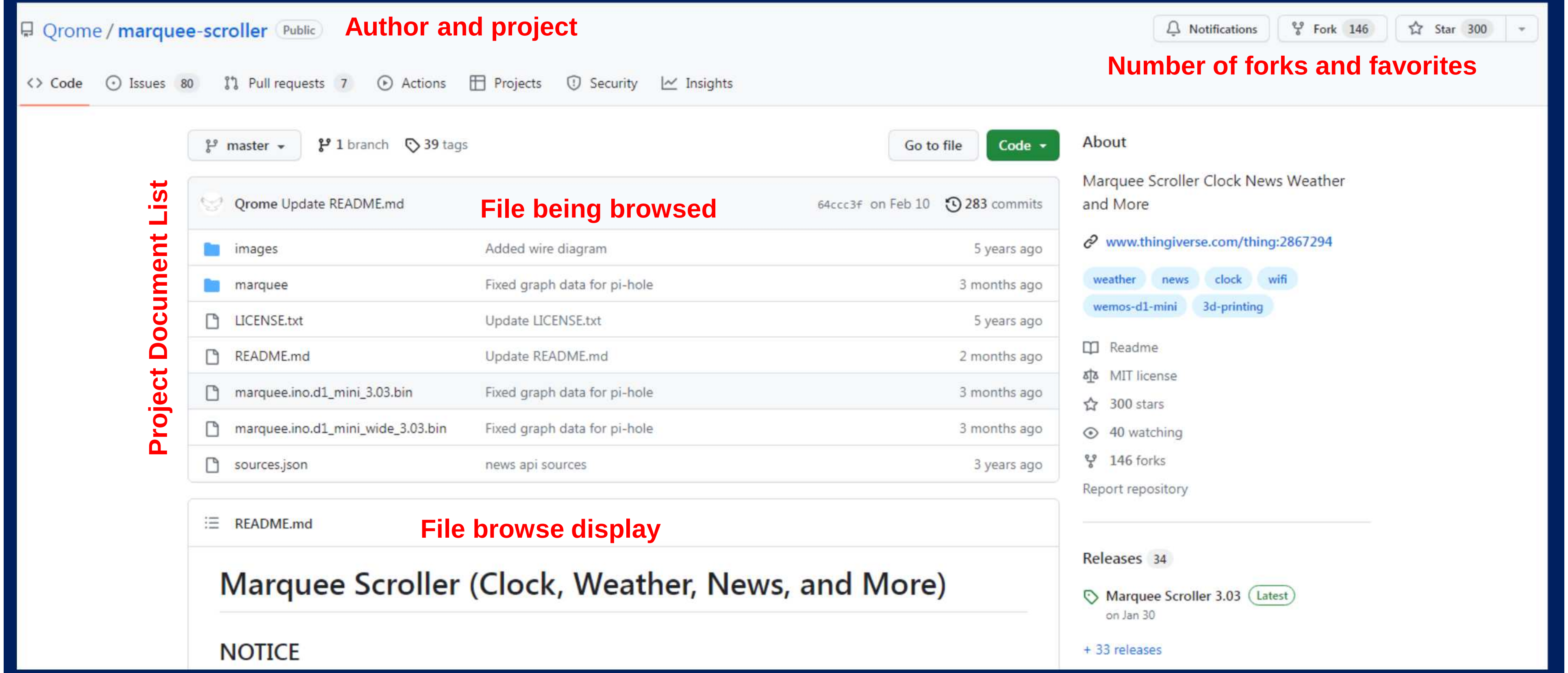
- Review: Marquee Scroller
- Review: Uploading binary files with ESPtool
- Obtaining API keys walkthrough
  - Openweathermap
  - News API

Part B

- Trimester Review
  - Introduction to Arduino
  - Basic Arduino Programming
  - Intermediate Arduino Programming
- Paul’s Github Page
- Graduation
  - Certificates
  - Recognition Awards

CLASS EXERCISE – Marquee Scroller

- Marquee Scroller is software written by Github member "Qrome". It is available at https://github.com/qrome/marquee-scroller
- Review of main project homepage



Qrome Update README.md — 64ccc3f on Feb 10 — 283 commits

images	Added wire diagram	5 years ago
marquee	Fixed graph data for pi-hole	3 months ago
LICENSE.txt	Update LICENSE.txt	5 years ago
README.md	Update README.md	2 months ago
marquee.ino.d1_mini_3.03.bin	Fixed graph data for pi-hole	3 months ago
marquee.ino.d1_mini_wide_3.03.bin	Fixed graph data for pi-hole	3 months ago
sources.json	news api sources	3 years ago

Marquee Scroller (Clock, Weather, News, and More)

NOTICE

About

Marquee Scroller Clock News Weather and More

www.thingiverse.com/thing:2867294

weather news clock wifi wemos-d1-mini 3d-printing

Readme
MIT license
300 stars
40 watching
146 forks
Report repository

Releases 34

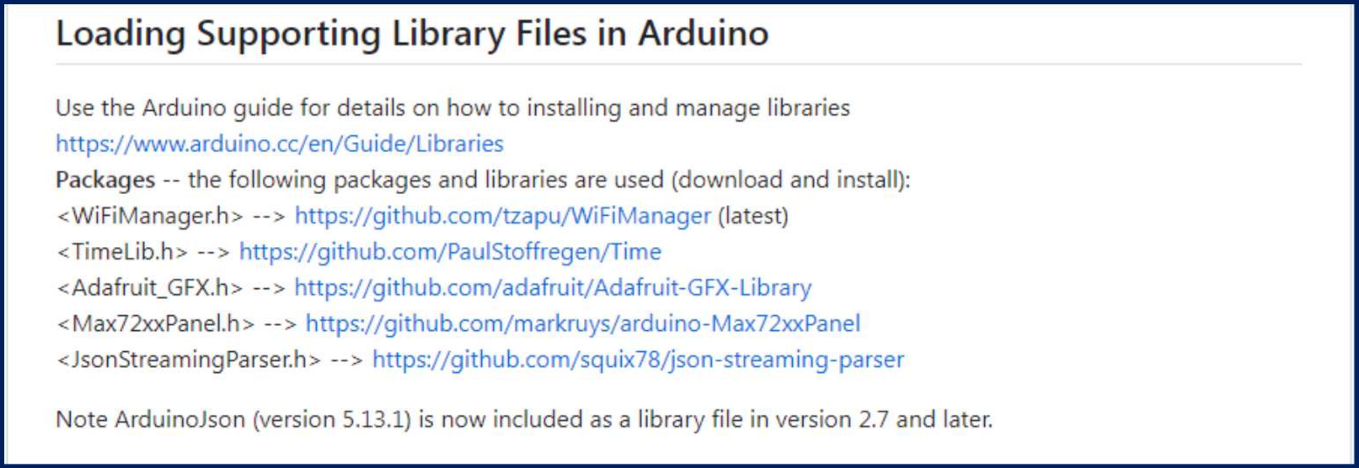
Marquee Scroller 3.03 Latest on Jan 30

+ 33 releases

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.

However, for those of you who were not here last week, you can cheat and just load the binary directly using ESPtool!

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

Recap: Flashing Software Binaries

- The Arduino IDE takes Processing/C++ source code and compiles it into a single binary file which is then transmitted to the microcontroller where it is stored in flash memory
- Uploading of “raw” binary executable files to a microcontroller is called “flashing”
- It is possible to create precompiled binaries and flash them without having to compile from source code or use the Arduino IDE
- This is very, very similar to installing an app on a phone (and is identical to initially installing the *operating system* onto a phone, which is done at the factory)
- We are going to learn how to do this ourselves!

Recap: Flashing ESP8266 Software on Linux

- Exit the Arduino IDE and/or ensure it is NOT running

- Assure you have copied the following file in today's Lesson folder to your hard drive (use the Dolphin GUI file explorer; if you copied the whole Lesson 5 folder, you're good to go, just make sure you can find the file)
  `SKETCH12A-MarqueeScroller-d1_mini`

- Open a Konsole window, and enter the following command:
  ```
  which esptool
  ```

- If your Linux system reports a path for the tool, you can skip to the installation step. You should have already installed the tool as part of Lesson5 last semester, so you should be good to go without any additional work needed.

- If however you do not have esptool installed, enter the following command into the Konsole window (**enter your account password when/if prompted**)
  ```
  sudo apt install esptool
  ```

- One you have esptool installed, enter the following commands (you'll need to interpolate the path for the directory where you placed the binary; if in doubt, just copy the binary to your Documents folder:
  ```
  cd [directory where the Marquee Scroller binary file is]
  esptool --port /dev/ttyUSB0 write_flash 0x0 ./SKETCH12A-
  MarqueeScroller-d1_mini
  ```

- Your ESP8266 display should restart and begin running the image you just uploaded!

Recap: Flashing on Windows with Python3

- If you have a Windows workstation, you'll need to install Python3 and then install esptool.py. **This should have all been completed as part of Lesson5 last semester, so you should be good to go and can skip directly to the last step.**

- Assure you have copied the following file in today's Lesson folder to your hard drive (use the Windows file explorer; if you copied the whole Lesson 12 folder, you're good to go, just make sure you can find the file)
  `SKETCH12A-MarqueeScroller-d1_mini`

- For Windows 10/11, run the `python-3.10.4-amd64` installer from the Software folder (copy it from your flash drive FIRST!) in the Basic Arduino Programming folder(s) *from last semester*

- Once Python is installed, open a CMD (DOS CLI) window

- Enter the following commands into the DOS CLI prompt (you can skip the REM comments):
  ```
  pip install esptool
  ```

- Enter the following
  ```
  REM Change to the directory you copied the Marquee Scroller binary to
  cd %userprofile%\Documents\folder_with_the_binary
  REM Connect your board and find the port it's on with Device Mgr
  REM Then flash the binary first
  esptool.py --port COM?? write_flash 0x0 ./SKETCH12A-MarqueeScroller-d1_mini
  ```

Recap: Marquee Scroller Setup

- **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**

Recap: 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.

Recap: 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?

Trimester Review

- It’s been a long but hopefully worthwhile year. Here’s a walk down memory lane recalling the skills and knowledge you have hopefully gained from these classes

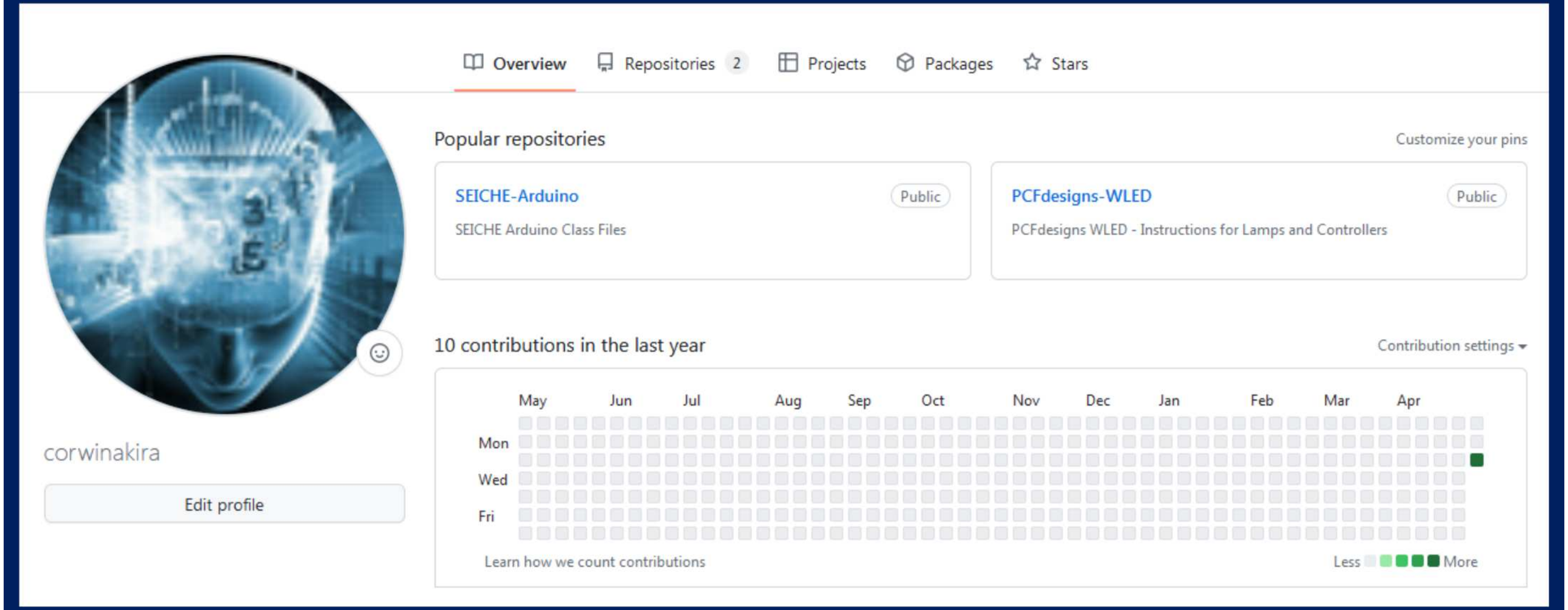
- Introduction to Arduino
  Basic understanding of DC electricity, understanding and use of basic electronic components, use of a digital multimeter, connection of different components to a microcontroller, connection of smart/programmable LED strands to a microcontroller, installation and use of WLED software

- Basic Arduino Programming
  Use of the Arduino IDE, Understanding of basic C/C++ programming concepts, including basic data structures, control flow statements, conditionals, and arithmetic expressions, using libraries, loading precompiled binaries, the binary and hexadecimal number systems, reading I2C sensors, generating sound with a microcontroller, use of fonts with an LED matrix, Network Time Protocol, using fonts and text effects with an LED matrix display

- Intermediate Arduino Programming
  Memory organization, addressing and pointers, the C preprocessor, creating and using your own fonts, using Visual Studio Code, filesystems, wifi operation and use with a microcontroller, web technology and HTML, creating and using web servers, gathering and using form data, web scraping, JSON and REST API's, GitHub, Internet services API's and keys

Paul’s Github Page

- I have a Github page! Who knew?!
  https://github.com/corwinakira
- Since Github can be used for organizing and storing documents of all types, I am in the process of updating my Github site with WLED instructional and SEICHE class content



corwinakira

Popular repositories

SEICHE-Arduino Public
SEICHE Arduino Class Files

PCFdesigns-WLED Public
PCFdesigns WLED - Instructions for Lamps and Controllers

10 contributions in the last year

Learn how we count contributions

Final Wrap-Up

- Completion Certificates – You’ve all earned them!
- Recognition Awards – Those who demonstrated excellence

Formal End of Lesson 12

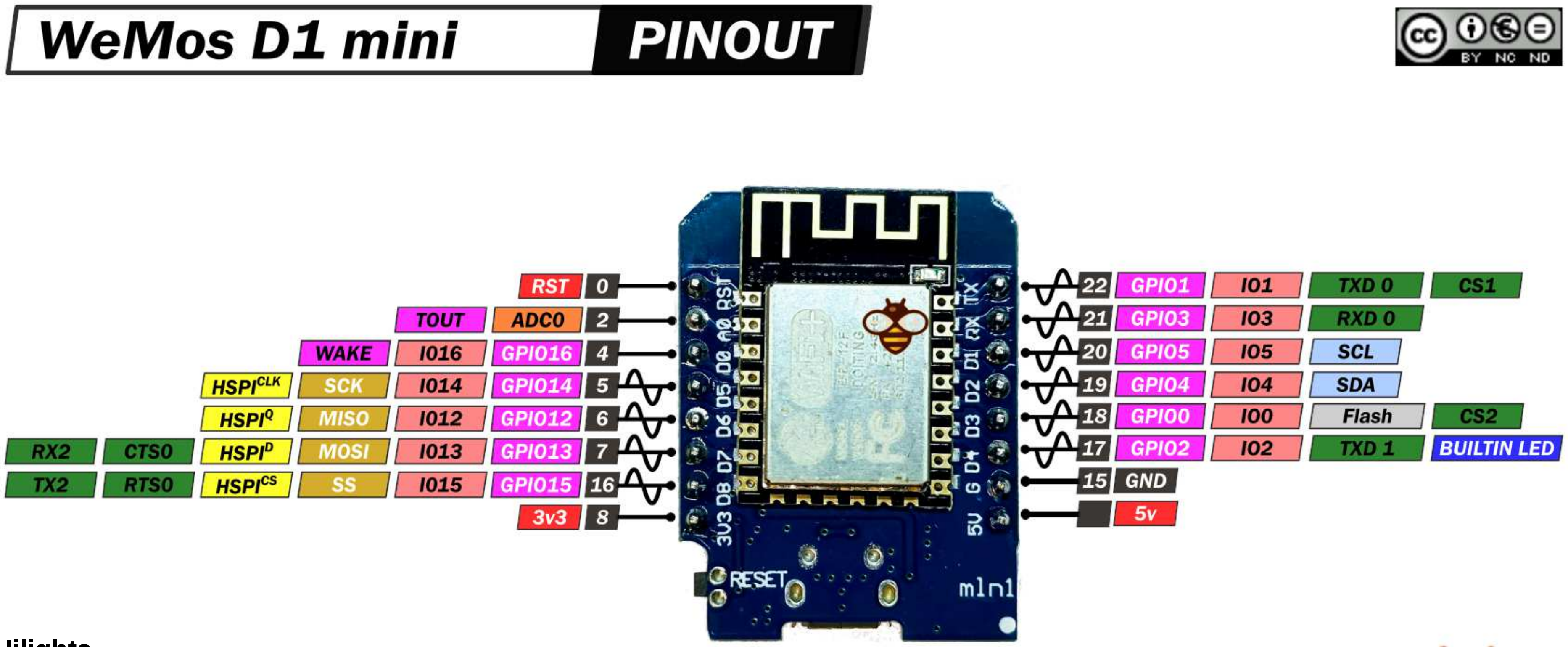
AND THAT'S THE END OF
SEICHE ARDUINO AND ELECTRONICS

If I don't see y'all again, have a good life!

- You can e-mail me at paulfrommeyer@gmail.com or paul@paulfrommeyer.com or paul@palas.com
- My website is **http://www.palas.com/~paul**
- And you can find my slow-updating Github page at

  https://github.com/corwinakira

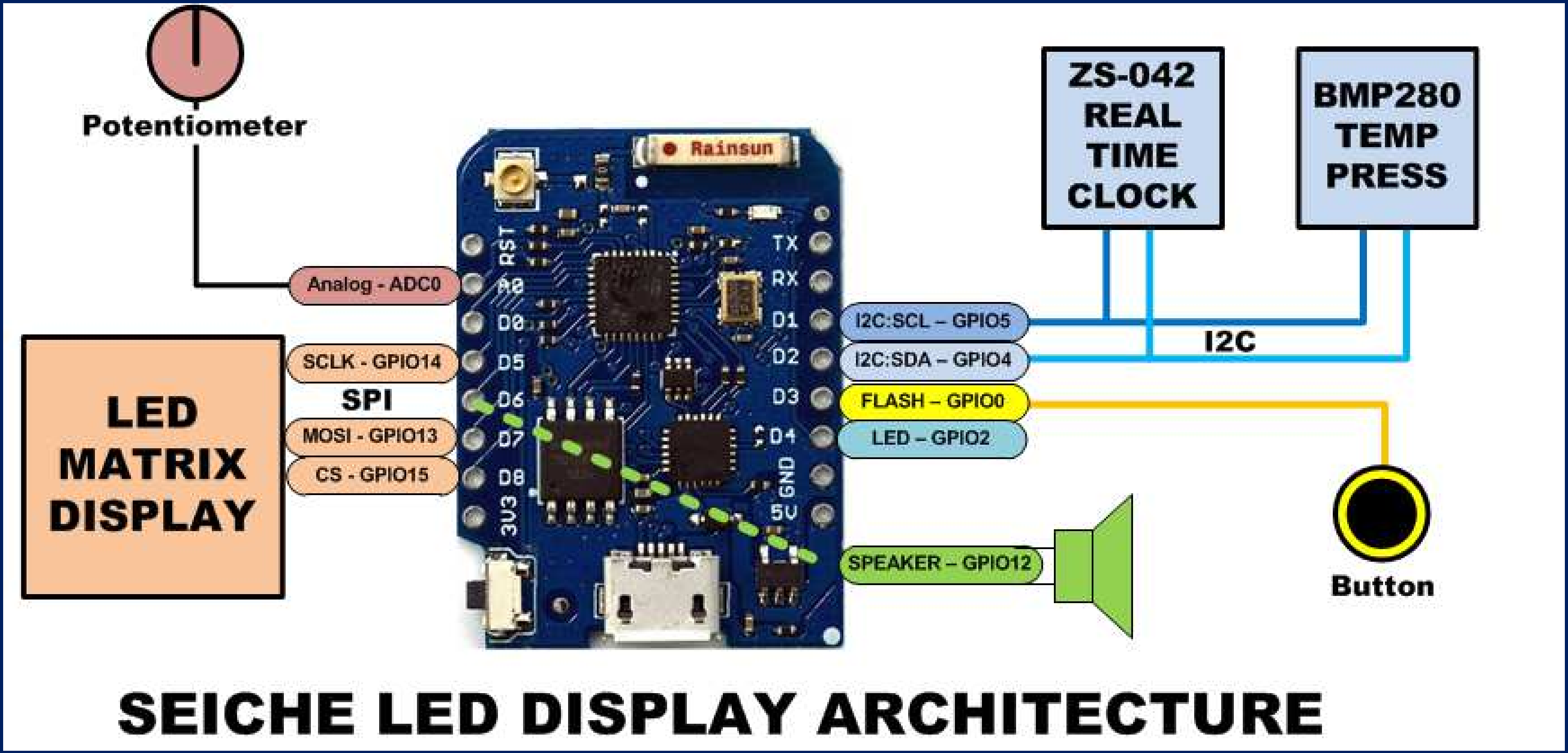
Our Microcontroller: The WeMos D1 Mini



Hilights

- 11 digital IO: all are interrupt and pwm capable (except D0/GPIO16)
- 1 analog input (3.2V max input): A0/ADC0
- Micro USB <u>or</u> 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