SEICHE 2023 Intermediate Arduino Programming for IoT

Instructor: Paul Frommeyer

www.paulfrommeyer.com

Corporate Sponsor: DXC Technology

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 – Internet Access

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

Lesson 11: 18Apr23 – 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

Lesson 12:2May23 – 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

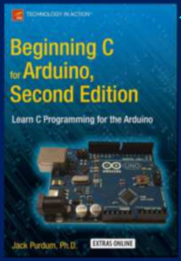
IntermediateProgramming-Lessons

- IntermediateProgramming-Lesson1
- IntermediateProgramming-Lesson2
- IntermediateProgramming-Lesson3
- IntermediateProgramming-Lesson9

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!


Beginning C for Arduino, Second Edition
Learn C Programming for the Arduino
Jack Purdum, Ph.D.

Lesson 9 – Accessing The Internet

Part A

- Class Exercise – HTML form display with MD_Parola
- Application Programming Interfaces
- Types of Web Site API's

Part B

- Page Scraping Video
  https://youtu.be/kWdfCCwhQTE

Class Exercise 9A: MD_Parola Form Data

- Go ahead and open SKETCH9A in the IDE, *but do not upload it!!*
- Validate the WiFi credentials!
- Once you are satisfied the WiFi credentials are correct, go ahead and upload the sketch
- The IP address obtained from the WiFi should scroll across the display
- Surf to that IP address, you should see a form with a single entry
- Go ahead and enter text into the form, it should scroll across your display
- Note: I have only tested this with the text SEICHE. There are probably failure conditions!!

Sketch 9A Explanation

- The code for the Smart Notice Board is written in the C++ programming language and uses various libraries to control the ESP8266 microcontroller and the dot matrix LED display. These libraries include the WiFi Manager library, the MD_Parola library, the MD_MAX72xx library, and the SPI library.
- Next, the code sets up the parameters for the dot matrix LED display using the HARDWARE_TYPE and MAX_DEVICES constants. The CS_PIN constant is also set up to define the chip select pin for the display. We don't need to specify the other SPI pins as MD_MAX72XX defaults to the hardware SPI pins.
- After that, we set up the WiFi module using the WiFiManager library and started a server on port 80. The curMessage and newMessage arrays are used to store the current and new messages, respectively. The newMessageAvailable flag is used to indicate if a new message is available for display.

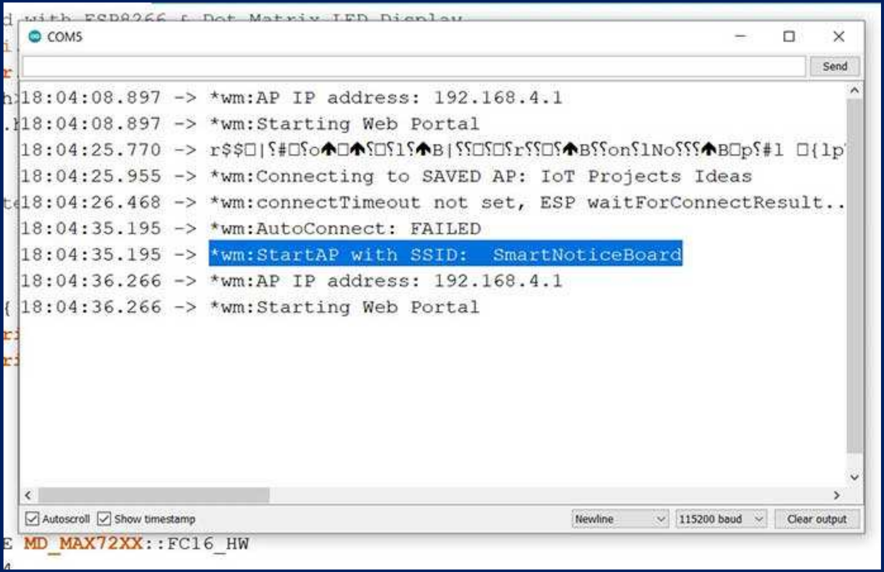
```
// WiFi Server object and parameters
WiFiServer server(80);

// Scrolling parameters
uint8_t frameDelay = 25;  // default frame delay value
textEffect_t  scrollEffect = PA_SCROLL_LEFT;

// Global message buffers shared by Wifi and Scrolling functions
#define BUF_SIZE  512
char curMessage[BUF_SIZE];
char newMessage[BUF_SIZE];
bool newMessageAvailable = false;
```

Source: Random Nerd Tutorials

Sketch 9A Explanation (cont.)

- The code then defines the WebResponse and WebPage constants, which are used to create the web page that allows the user to send messages to the Smart Notice Board.
- The SendData() function is used to send the message to the Smart Notice Board.
- Finally, the setup() function is used to initialize the dot matrix LED display, WiFi manager, and the ESP8266 microcontroller. It also prints the ESP8266 Local IP Address after a successful WiFi connection.
- And the loop() function is used to continuously check for new messages and update the display as required.
- Something like this should display in SerialMonitor once the sketch is running
  - Notice how the first time through, Wifi Manager cannot find a saved AP so it starts up in standalone mode using SSID <u>SmartNoticeBoard.</u>
  - This will be a repeat of last year at first startup as all displays will have the same SSID and IP address!

Source: Random Nerd Tutorials



```
18:04:08.897 -> *wm:AP IP address: 192.168.4.1
18:04:08.897 -> *wm:Starting Web Portal
18:04:25.770 -> r$$□|ʕ#□ʕo🡅□🡅ʕ□ʕ1ʕ🡅B|ʕʕ□ʕ□ʕrʕʕ□ʕ🡅Bʕʕonʕ1Noʕʕʕ🡅B□pʕ#1 □{1p
18:04:25.955 -> *wm:Connecting to SAVED AP: IoT Projects Ideas
18:04:26.468 -> *wm:connectTimeout not set, ESP waitForConnectResult..
18:04:35.195 -> *wm:AutoConnect: FAILED
18:04:35.195 -> *wm:StartAP with SSID:  SmartNoticeBoard
18:04:36.266 -> *wm:AP IP address: 192.168.4.1
18:04:36.266 -> *wm:Starting Web Portal
```

```
MD_MAX72XX::FC16_HW
```

Application Programming Interface

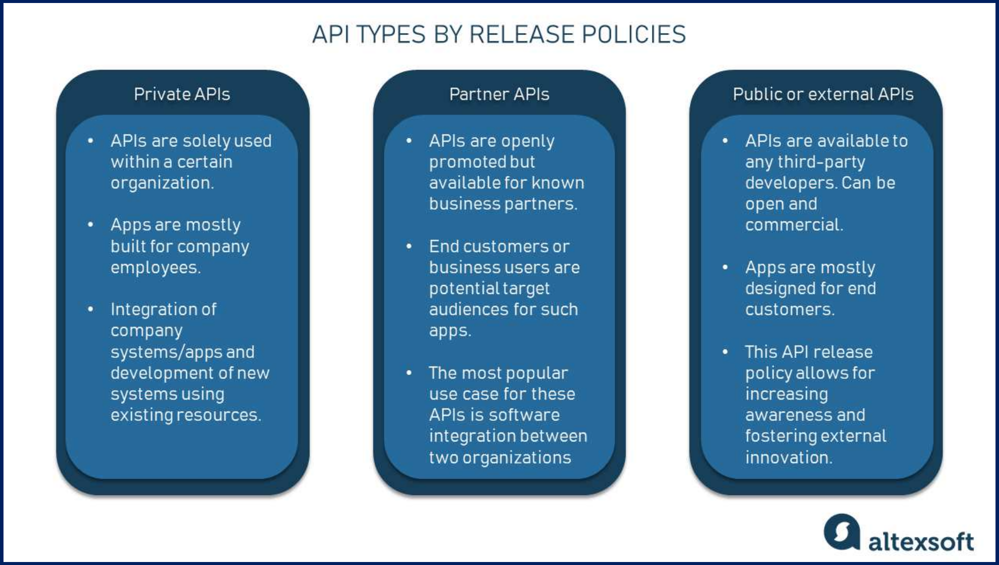
- An API is an Application Programming Interface, a set of definitions and protocols for building and integrating application software.
- APIs are mechanisms that enable two software components to communicate with each other using a *common, agreed-upon* set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone "talks" to this system via APIs and shows you daily weather updates on your phone.
- API stands for Application Programming Interface. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.
- API architecture is usually explained in terms of client and server. The application sending the request is called the client, and the application sending the response is called the server. So in the weather example, the bureau's weather database is the server, and the mobile app is the client.

Source: Amazon

Types of APIs

APIs are classified both according to their architecture and scope of use. We have already explored the main types of API architectures so let’s take a look at the scope of use.

- **Private APIs – These are internal to an enterprise and only used for connecting systems and data within the business.**
- **Public APIs –These are open to the public and may be used by anyone. There may or not be some authorization and cost associated with these types of APIs.**
- **Partner APIs – These are only accessible by authorized external developers to aid business-to-business partnerships.**
- **Composite APIs – These combine two or more different APIs to address complex system requirements or behaviors.**



API TYPES BY RELEASE POLICIES

Private APIs
- APIs are solely used within a certain organization.
- Apps are mostly built for company employees.
- Integration of company systems/apps and development of new systems using existing resources.

Partner APIs
- APIs are openly promoted but available for known business partners.
- End customers or business users are potential target audiences for such apps.
- The most popular use case for these APIs is software integration between two organizations

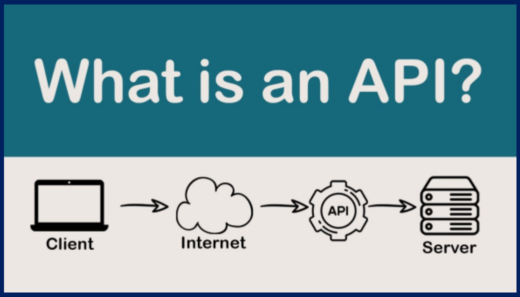
Public or external APIs
- APIs are available to any third-party developers. Can be open and commercial.
- Apps are mostly designed for end customers.
- This API release policy allows for increasing awareness and fostering external innovation.

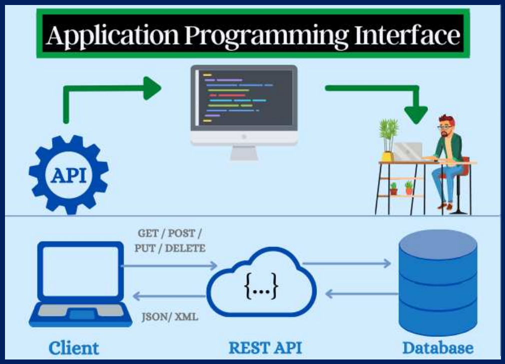
altexsoft

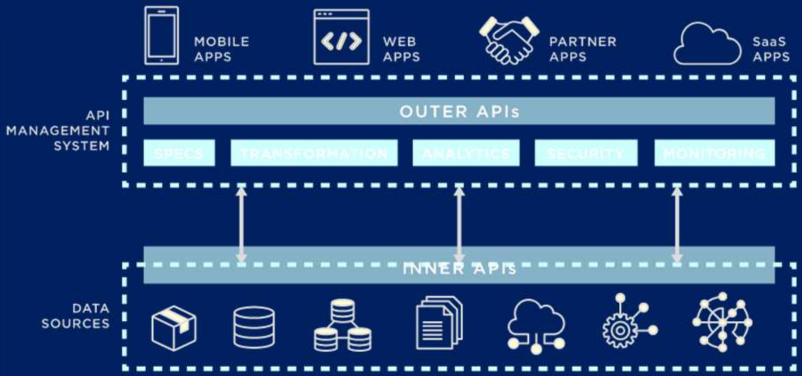
Types of Web API’s

- SOAP APIs – These APIs use Simple Object Access Protocol. Client and server exchange messages using XML. This is a less flexible API that was more popular in the past.
- RPC APIs – These APIs are called Remote Procedure Calls. The client completes a function (or procedure) on the server, and the server sends the output back to the client.
- Websocket APIs – Websocket API is another modern web API development that uses JSON objects to pass data. A WebSocket API supports two-way communication between client apps and the server. The server can send callback messages to connected clients, making it more efficient than REST API.
- REST APIs – These are the most popular and flexible APIs found on the web today. The client sends requests to the server as data. The server uses this client input to start internal functions and returns output data back to the client. Let’s look at REST APIs in more detail below.

Source: Amazon







The REST API

- REST stands for Representational State Transfer. REST defines a set of functions like GET, PUT, DELETE, etc. that clients can use to access server data. Clients and servers exchange data using HTTP.
- The main feature of REST API is statelessness. Statelessness means that servers do not save client data between requests. Client requests to the server are similar to URLs you type in your browser to visit a website. The response from the server is plain data, without the typical graphical rendering of a web page.
- A Web API or Web Service API is an application processing interface between a web server and web browser. All web services are APIs but not all APIs are web services. REST API is a special type of Web API that uses the standard architectural style explained above.
- The different terms around APIs, like Java API or service APIs, exist because historically, APIs were created before the world wide web. Modern web APIs are REST APIs and the terms can be used interchangeably.

The REST API (cont.)

- REST APIs offer four main benefits:

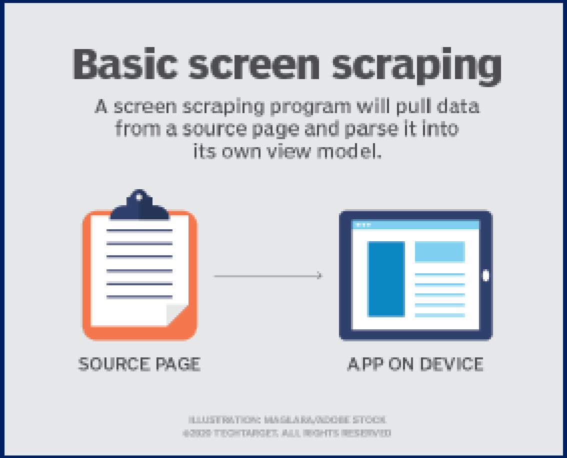
1. Integration – APIs are used to integrate new applications with existing software systems. This increases development speed because each functionality doesn't have to be written from scratch. You can use APIs to leverage existing code.
2. Innovation – Entire industries can change with the arrival of a new app. Businesses need to respond quickly and support the rapid deployment of innovative services. They can do this by making changes at the API level without having to re-write the whole code.
3. Expansion – APIs present a unique opportunity for businesses to meet their clients' needs across different platforms. For example, maps API allows map information integration via websites, Android,iOS, etc. Any business can give similar access to their internal databases by using free or paid APIs.
4. Ease of maintenance – The API acts as a gateway between two systems. Each system is obliged to make internal changes so that the API is not impacted. This way, any future code changes by one party do not impact the other party.

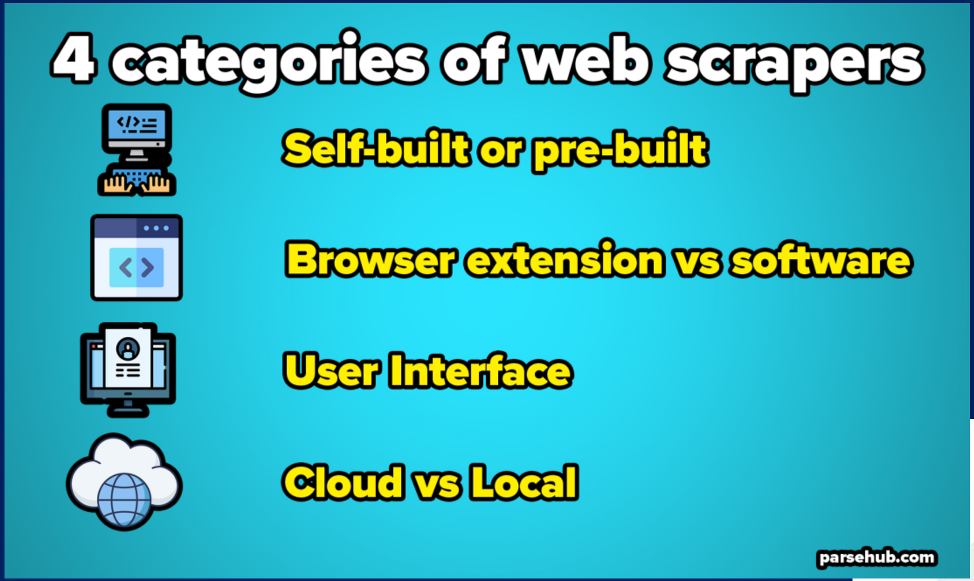
Web or Page Scraping

- Web scraping refers to the extraction of data from a website. This information is collected and then exported into a format that is more useful for the user. Be it a spreadsheet or an API. [parsehub.com]
- Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites.[1] Web scraping software may directly access the World Wide Web using the Hypertext Transfer Protocol or a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis [Wikipedia]
- Scraping a web page involves fetching it and extracting from it. Fetching is the downloading of a page (which a browser does when a user views a page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, extraction can take place. The content of a page may be parsed, searched and reformatted, and its data copied into a spreadsheet or loaded into a database. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be finding and copying names and telephone numbers, companies and their URLs, or e-mail addresses to a list (contact scraping). [Wikipedia]

Web or Page Scraping (cont.)

- The most common techniques used for Web Scraping are
  - Human copy-and-paste.
  - Text pattern matching.
  - HTTP programming.
  - HTML parsing.
  - DOM parsing.
  - Vertical aggregation.
  - Semantic annotation recognizing.
  - Computer vision web-page analysis.


Basic screen scraping
A screen scraping program will pull data from a source page and parse it into its own view model.
SOURCE PAGE
APP ON DEVICE
ILLUSTRATION: MASLAKA/ADOBE STOCK
©2020 TECHTARGET. ALL RIGHTS RESERVED



Formal End of Lesson 9

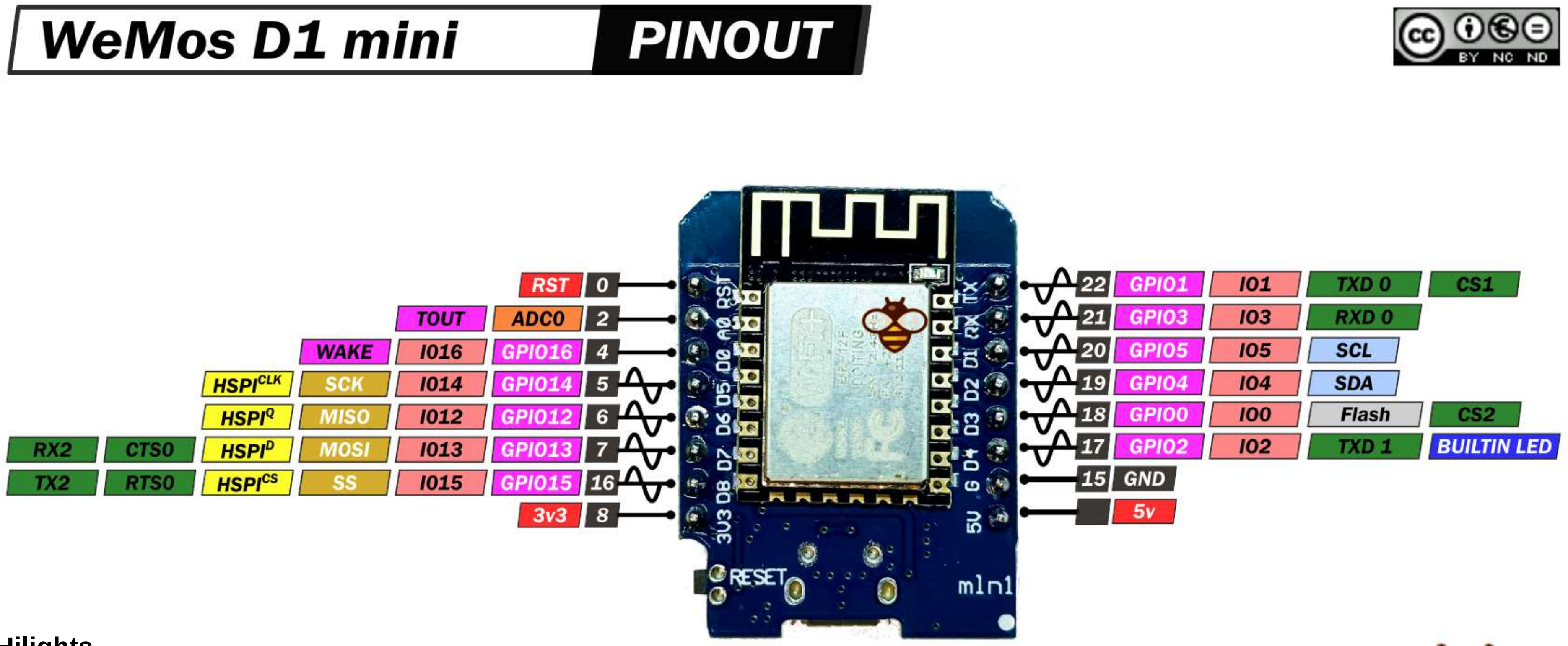
HOMEWORK

- **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

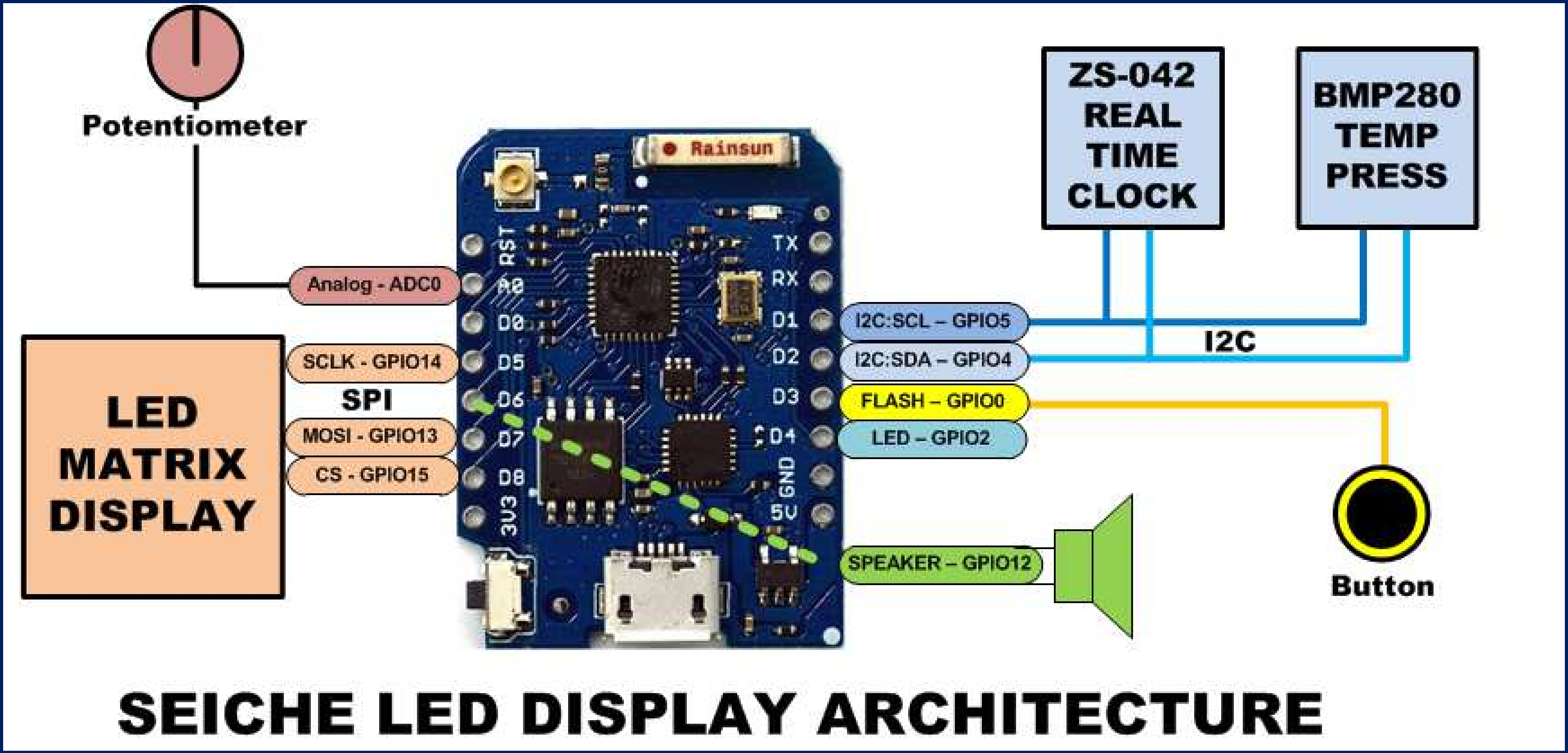
Our Microcontroller: The WeMos D1 Mini



Hilights

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