SEICHE 2022
Basic Arduino Programming

Instructor: Paul Frommeyer

www.paulfrommeyer.com

Corporate Sponsor: DXC Technology

Lesson Plan Overview

Lesson 1 – Intro and Setup
[may require 2 classes]

- Introduction to class format
- Overview of lesson plan
- Presentation format (monitor, camera, screen, whiteboard)
- Review of microcontrollers and types of boards
- SEICHE LED display architecture
  - ESP8266 pinout
  - High level architecture

Lesson 2 – Laptop operation review – Windows and Linux

- Inventory of USB drives
- Installation of Arduino IDE software
- Installation of CH340/ESP8266 serial port drivers (Windows only)
- Control panel/settings location
- Home directories and folder hierarchy
- Arduino file locations
- Search functions
- (Windows) Device Manager
- (Linux) Konsole
- Copying flash drive contents [critical]
- Open questions and issues
- **IDE essentials**
- Starting the Arduino IDE
- Basic Arduino sketch (program) structure
- Loading example sketches
- Loading and configuring new boards
- Connecting boards
- Identifying the microcontroller serial port
  - Linux
  - Windows

Lesson 3 – Libraries, Sketch structure, Serial Monitor, Variables, Binary Number System Pt1

- Libraries
- Sketch structure (A note on brace formatting)
- The serial port monitor
- Printing to the serial port monitor
- Variables and the assignment operator
- Binary number system Pt. 1.

Lesson 4 – Expressions, Conditionals, Blocks and Functions

- Arithmetic Expressions and Operators
- Incrementing and Decrementing Variables
- Truth Values in C++
- The If-Then Statement
- Code Blocks
- Functions

Lesson 5 – Binary Images, Arrays, Characters, Strings, Loops

- Loading Binary Images
- Arrays
- Characters and Character Codes
- Strings
- Conditional Loops Part 1

Lesson 6 – Loops (cont.), LED Matrix Displays, Nested Loops Advanced Functions, Binary Numbers Part 1

- For-Next Loops
- SPI Peripherals
- Using a MAX7219 LED Matrix Display
- Lighting and clearing individual pixels
- Advanced Functions
- Nested Loops

Lesson 7 – The Binary Number System (may take 2 lessons)

- Numerals vs numbers
- Review: the base 10 system and digit place values
- New: the base 2 system and digit place values
- Bits and bytes and nybbles
- Binary addition and subtraction

Lesson 8 – Producing Sound

- Formatting printed output in Serial Monitor
- Shifting and exponents
- Displaying text on the LED matrix display
- Review of sound wave theory
- Analog vs Pulse Width Modulation
- Producing sound tones with an Arduino microcontroller

Lesson 9 – Reading pins

- Reading buttons
- Millis() and debouncing buttons
- Reading analog values from a potentiometer

Lesson 10 – The I2C Bus and Peripherals

- I2C Bus Operation
- Initializing the I2C bus
- Accessing an I2C temperature sensor
- Displaying text on the LED matrix
- Default fonts

Lesson 7 – The Binary Number System

Part A – Homework Exercises

- Take your homework sketch from last week and modify it to print the contents of an integer array of 99 elements, starting from the first element to the last, then from the last to the first. Use while instead of if-then. You'll execute the sketch next week.
- Modify Sketch6B to use one or more for loops to light all the pixels at the edge of the first LED square

Part B – The Binary Number System

- Classroom Exercise – Count the dots!
- Classroom Exercise – Chips ahoy!
- Numerals vs numbers
- New: the base 2 system and digit place values
- Bits and bytes and nybbles
- Binary addition and subtraction

Homework Review 1

- Take your homework sketch from last week and modify it to print the contents of an integer array of 99 elements, starting from the first element to the last, then from the last to the first. Use while instead of if-then. You'll execute the sketch next week.
- Who was able to come up with a solution to this?
- Notice that you were *not* required to *initialize* the array (fill it with data)
- Discussion of different approaches to solving the problem
- What did you see when printing out the values stored in the array?

Homework Review 2

- Modify Sketch6B to use one or more for loops to light all the pixels at the edge of the first LED square

- Who was able to come up with a solution to this?
- Notice that you were *not* required to light *only* the pixels at the first edge
- Discussion of different approaches to solving the problem
- Did anyone try to light all the pixels on all the 8x8 square modules? What happened?

Let there be light?

- Think about a light switch; how many positions or “states” can it have?
- Based on the number of states of a single switch, how many different arrangements of lighting can we make with:
  - 2 switches?
  - 3 switches?
  - 4 switches?

Hold this thought.

Classroom Review – Roman Numerals

- The Arabic numeral system that we use was not the first used in Western civilization (such as it is). In addition to Babylonian Cuneiform and Hebrew logographics, the Roman system was the most widely used in the ancient world during the days of the Roman Empire.
- Romans did not use or understand the *zero*. That was a Hindu invention that found its way to the mid east via trade routes.
- Thus, Roman numerals use explicit digits for the first through the eighth cardinal numbers, and a "subtraction arrangement" for the ninth, with a new explicit symbol for the tenth and significant decimal "mileposts" like fifty or one-hundred
- Here the Roman numerals through twenty:

I, II, III, IV, V, VII, VIII, IX, X, XI, XII, XIII, XIV, XV, XVI, XVII, XVIII, XIX, XX, … XL, L [fifty]

Classroom Exercise #1!

- Using the cards that you have just been given, represent the following *Roman numerals* by displaying the corresponding number of dots, **flipping over any cards that you do not need**, *assuring that the cards aren't covering each other*, and arranging them from largest on the left to smallest on the right. (Each student should represent just one numeral with their cards, starting widdershins from the instructor)

V

VII

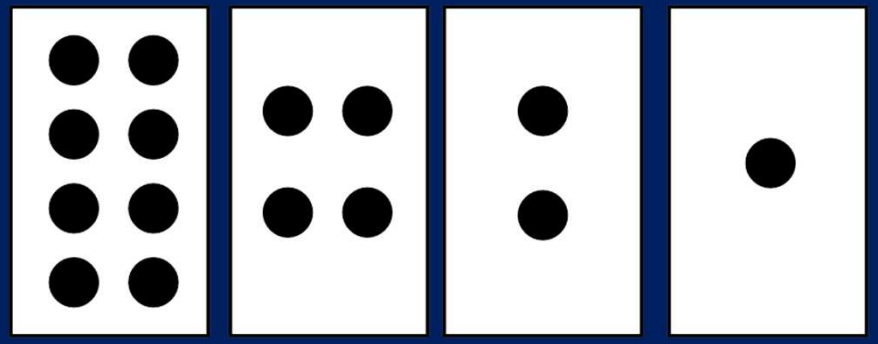
IX

XI

XII

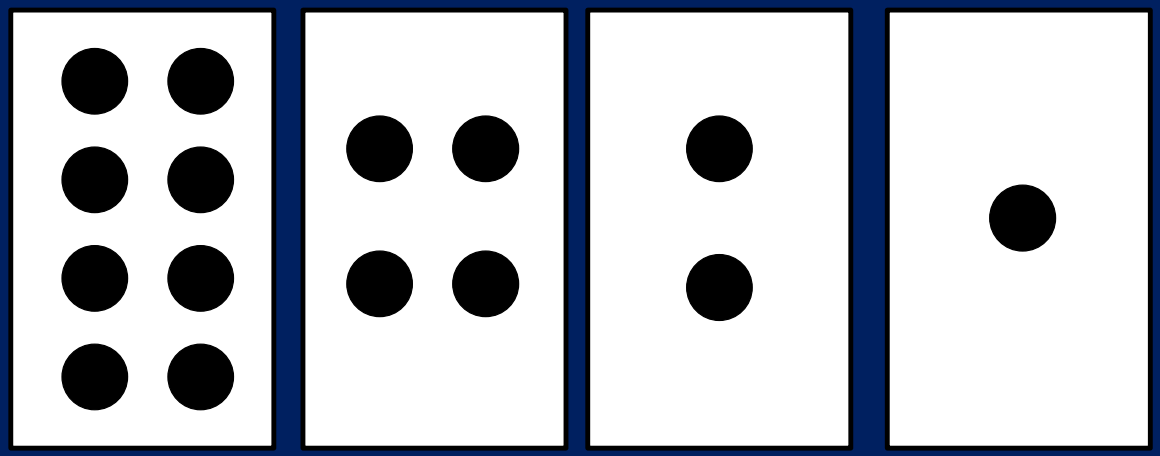
XIII

XV



Classroom Exercise #2!

- Now, instead let's use poker chips to *represent* the four cards. The great state of Texas (hook 'em Horns!) will represent a card being turned face-up, and the initials (PLF) will indicate a card being turned face down (down with your instructor?).
- It's very important to *use the chips with the same numerical order as the cards*, from greatest on the left to least on the right.



Represent: 7, 9, 3, 4, 8, 12, 14

Map vs Territory

- Title translation: "**This is not a pipe.**"
- Why on Earth did the artist René Magritte choose *that* title for his work?
- What is he trying to help us understand?

Map vs Territory (cont.)

3

This is not a number

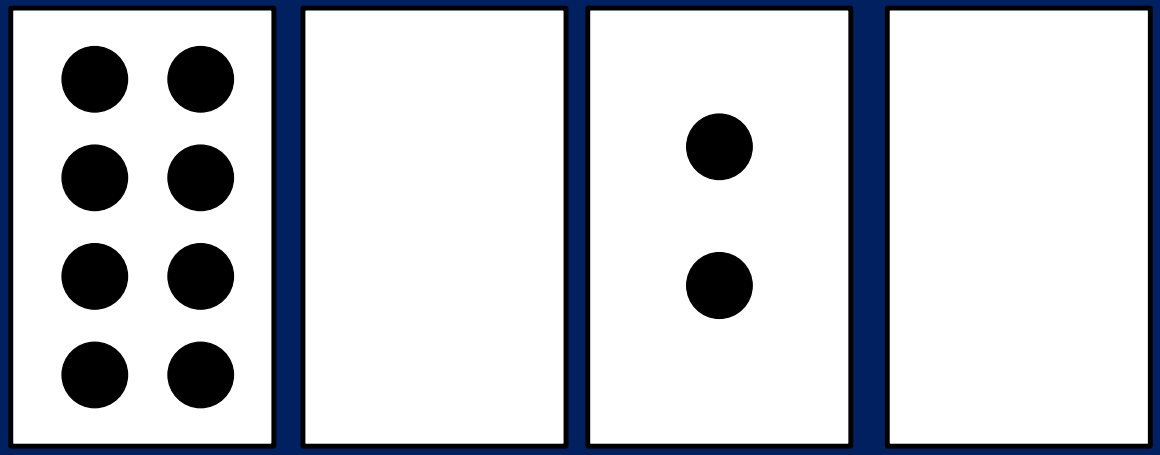
- Explain ***my*** title

Numbers vs. *Numerals*

- CRITICAL IDEA: The type of **symbol** used to *represent* a number is **completely arbitrary**!
- We could design *our own* set of symbols to represent numbers if we wanted:
  Φ =
  Δ = ●
  ‡ = ● ●
  Ξ = ● ● ●
  □ = ● ● ● ●
  Λ= ● ● ● ● ●
  Π = ● ● ● ● ● ●
  Γ= ● ● ● ● ● ● ●
- Note that there are only eight (8) symbols. How would you represent
  ● ● ● ● ● ● ● ● ●
  using these symbols?

From Cards to Numerals

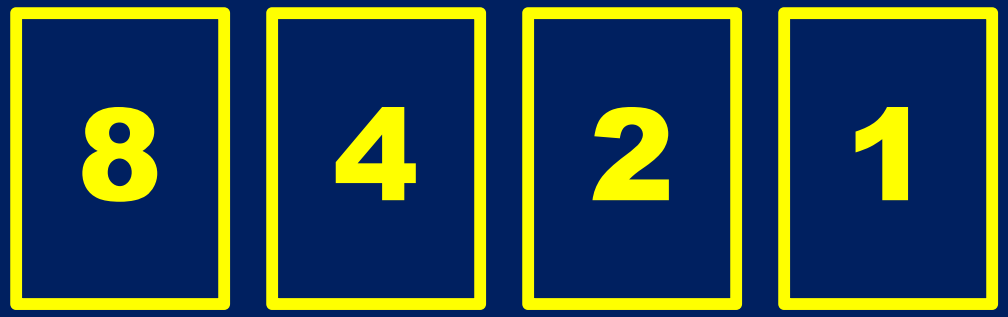
- So now, going back to our cards and poker chips
- Instead of using poker chips to represent whether a card is flipped or not, let's simplify things and just use the **numerals** one (1) and zero (0). *These are the **symbols**, not the **numbers**!*
- Thus, one (1) represents a card that is turned *up*
- And zero (0) represents a card that is turned *down*.



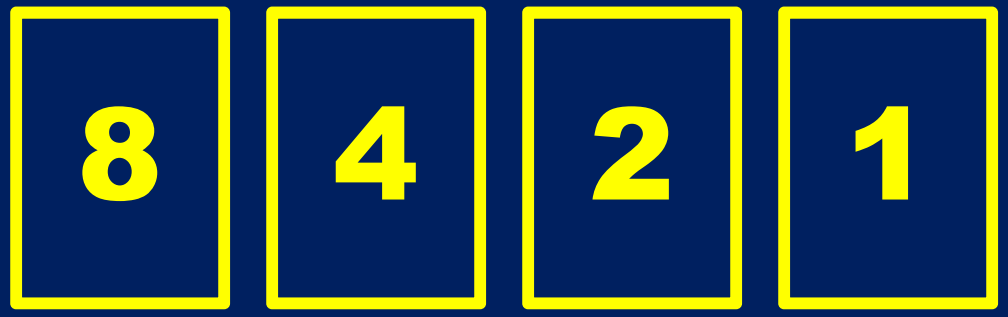
Thus: 1010

The Road to Binary Numbers

- Now, let's put that all together and label each card/box with the *number of dots* each card/box contains.
- To simplify things, we will stop calling them "cards" or "boxes", and start calling them columns or *places*.



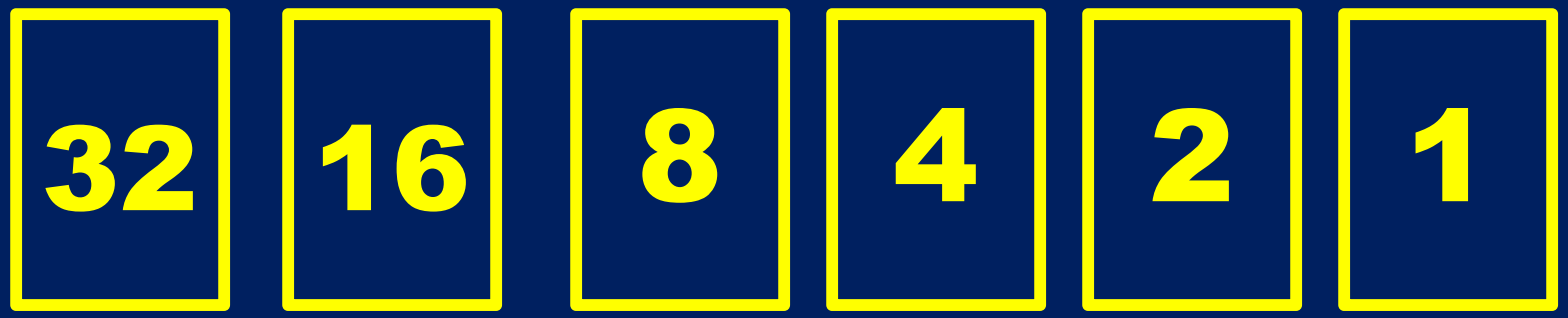
- This makes it easier to translate the 1's and 0's representation:



0 0 1 0

Binary Number System (cont.)

- We already discovered that the largest *number* we can represent with *four* cards, or with *four* binary symbols, is 15 (1111). However, the existing columns can easily be *extended:*

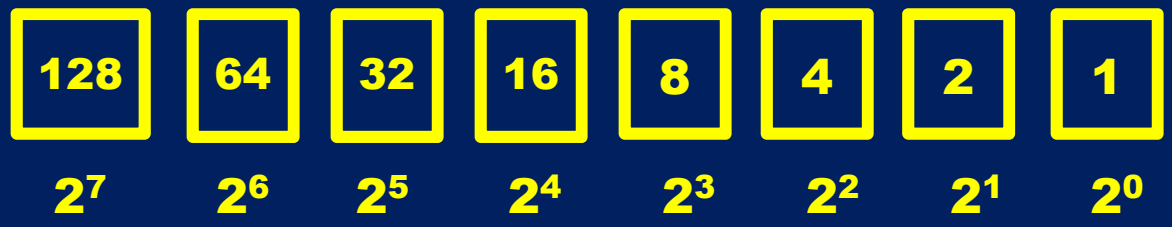


- And we can keep right on going, until we have eight columns or places.

- We will henceforth refer to these as “the eights place”, or “the sixty-fours place”
- Can anyone explain how the *values* for the new columns were figured out?

Binary Number System (cont.)

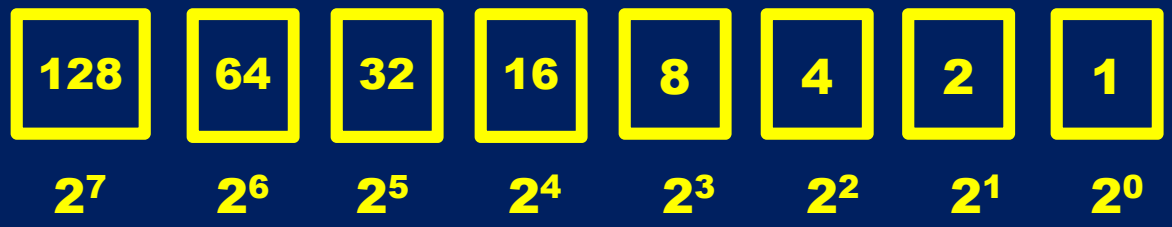
- The “secret decoder ring” is that each column or place *is a power of two*.
- *So this is how the place values for BASE2, or the Binary Number System, are formally written in Mathematics and Computer Science:*



- Notice that there are eight columns
- And remember that a binary digit, a one or a zero, is called a bit
- So what we have here is eight bits. What is the term for eight bits or eight binary digits in computer science and programming?
- And this one should be easy because you already proved it yourselves with Arduino code: what is the largest number that can be represented with eight bits?

Binary Number System (cont.)

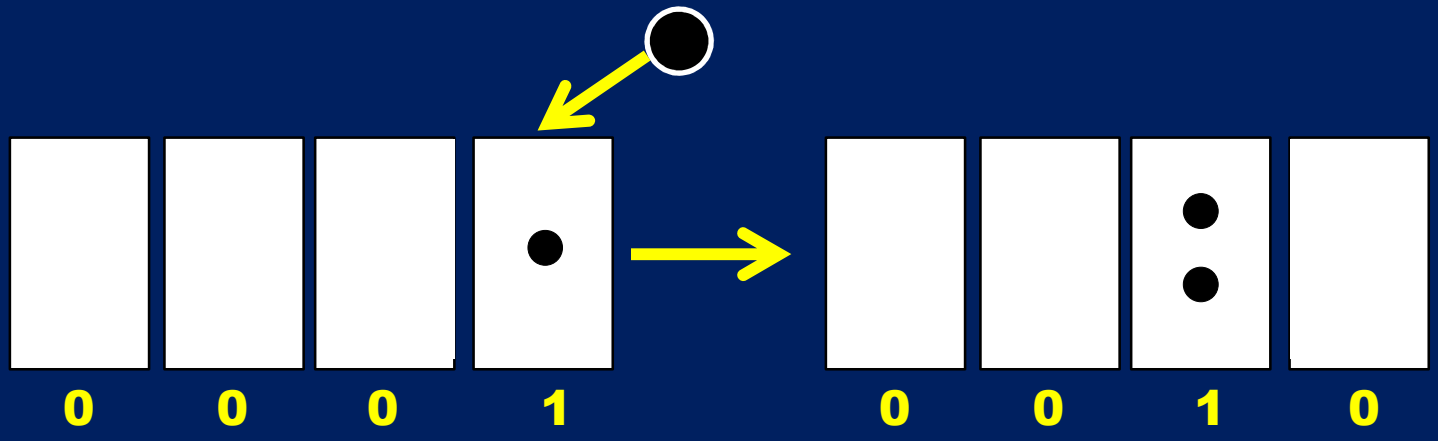
- Let's review everything so far.
- We have learned how binary number system place values are assigned



- And, using the binary place values, we have learned how to *represent* numbers using the two binary *symbols* one (1) and zero (0)
- We have rediscovered that single binary digits, a 1 or a 0, are called ***bits***
- And we have rediscovered that eight bits constitute a ***byte***
- A new term is that four (4) bits– since they are *half of a byte*— are called a ***nybble***. Get it? Gotta love that programmer humor!
- Now let's revisit addition and explore subtraction

Adding Binary Numbers

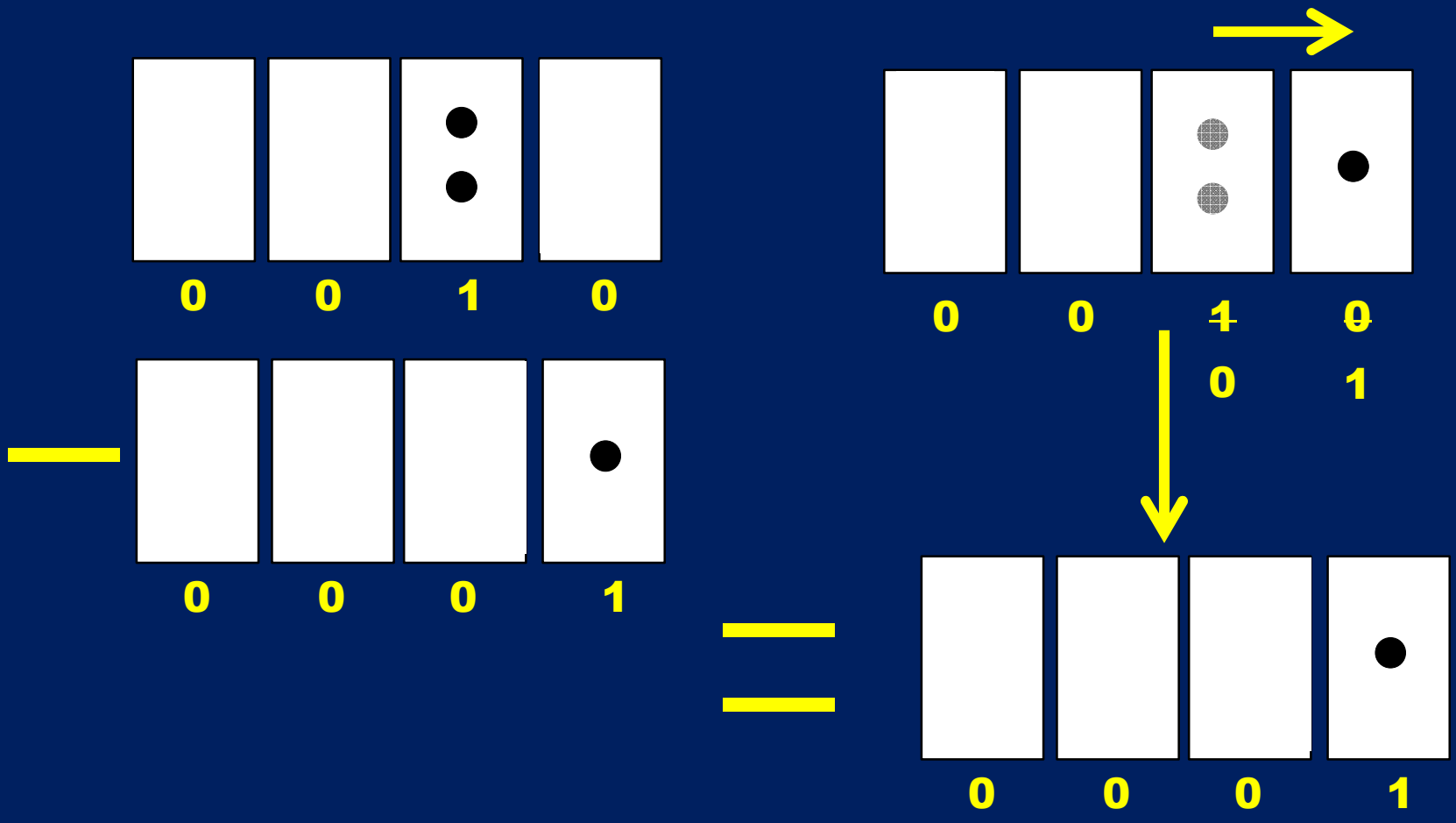
- **We already covered basic addition: when we go to add a third dot to a column, instead we have to add it to the *next* place *to the left,* and then clear the place to the right.**



- What we are doing is “carrying” the extra one to the next column. This works exactly like it does in our decimal number system, only “the next column” is a power of two instead of a power of ten.

Subtracting Binary Numbers

- It should come as no surprise that binary subtraction works the same way. If there is a zero in a column, we have to "borrow" a one *from the next higher place value*.



Subtracting Binary Numbers

- **Class exercise?**
- **1101 – 0101**
- **1110 – 0111**
- **1010 – 0010**

Of course, feel free to "cheat" by converting the binary number to decimal then back again (but no calculators, apps, online utilities, Siri, or Alexa; abaci are permitted ☺)

Multiplication and division work as you would expect, but they are fraught from a paper desk check perspective due to so many digits flying around. It's easier to convert to decimal or hex, then back. ☺

Formal End of Lesson 7

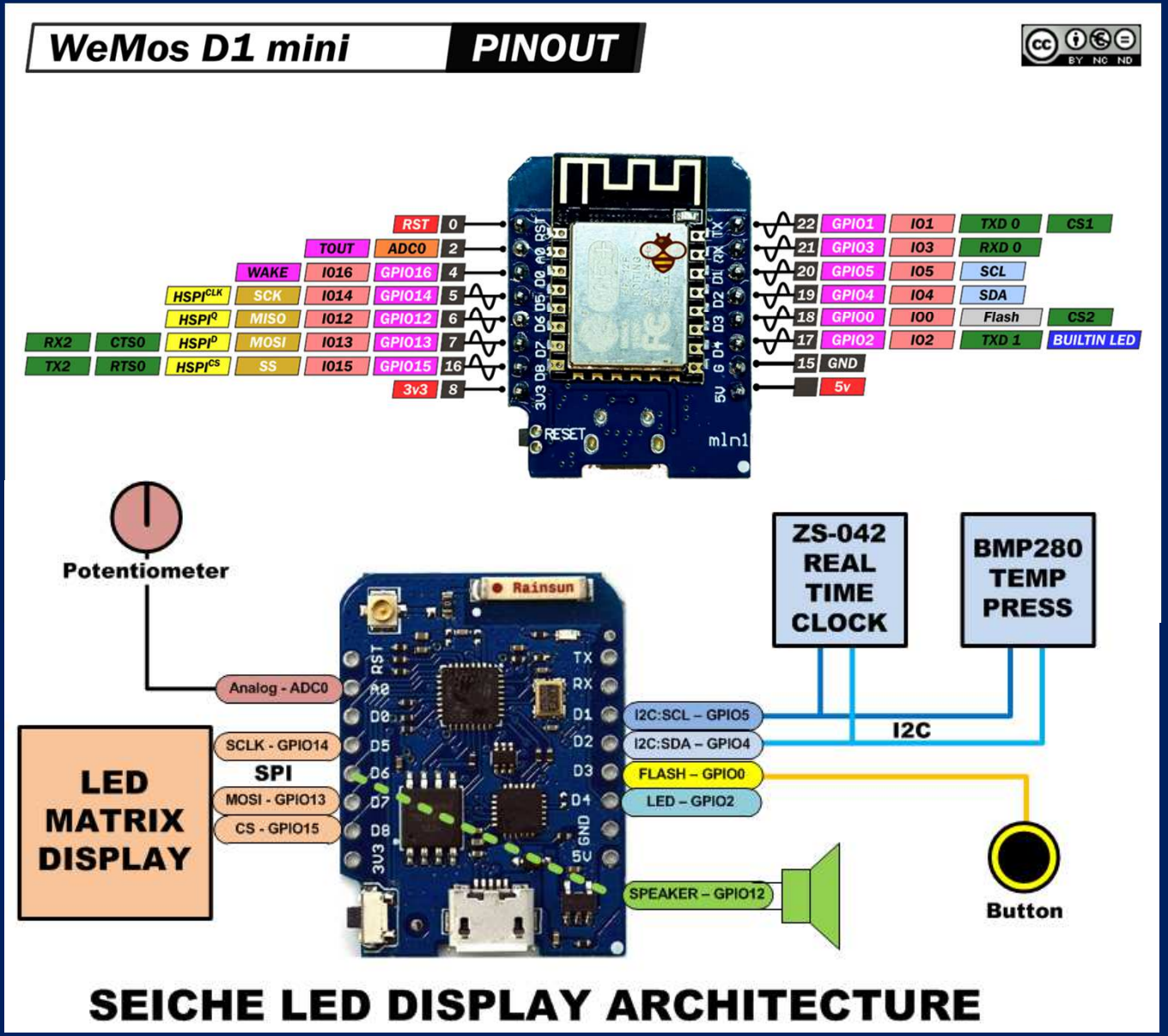
HOMEWORK!

- **Hackspace – Get Started With Arduino – pp. 24 through 39**
- **Arduino Cookbook – You can continue reading Chapter 4; we will not be covering all of Chapter 4 in class**

In next week's exciting episode

- Binary number system – bitwise operations
- Displaying text on the LED matrix display
- Making sounds with Arduino (and your displays)

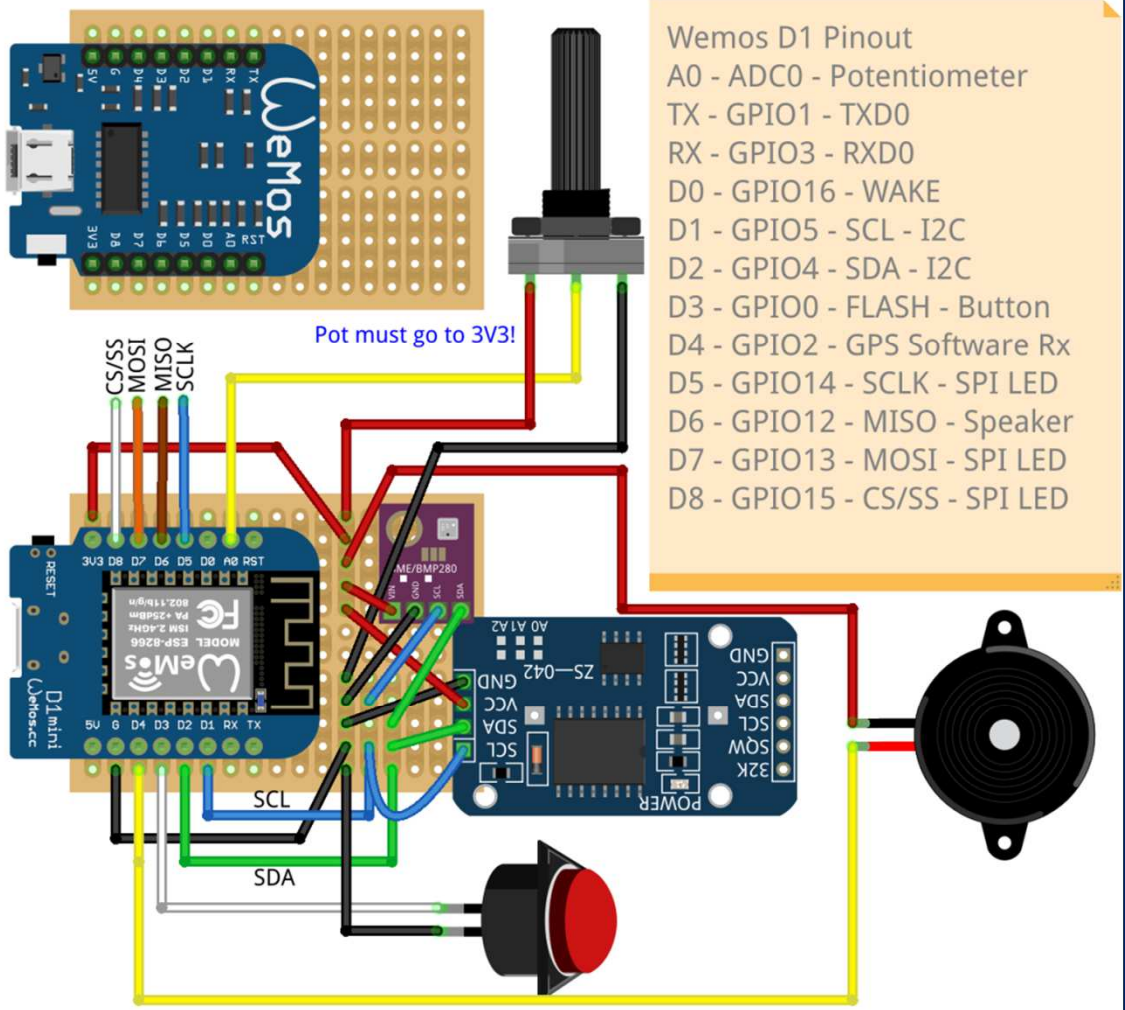
LESSON REFERENCE



LESSON REFERENCE

Pin Assignment Notes
GPIO16/D0 - HIGH at boot - No interrupt, no PWM or I2C - Unused
GPIO2/D4 - HIGH at boot - Input pulled up, output to onboard LED - - probable GPS RX software serial
GPIO12/D6 - Piezo Speaker (not used in SPI LED Matrix)
GPIO[12],13,14,15/D6,D7,D5,D8 - MISO,MOSI,SCLK,CS - SPI
GPIO4,5/D2,D1 - SDA,SCL - I2C
ADC0/A0 - Analog Input - Potentiometer 3.3V divider
GPIO0/D3 - Input pulled up - FLASH button, boot fails if pulled low - button to ground

SPI - LED matrix : 12,13,14,15
I2C - RTC,BMP280 : 4,5
Serial RX - GPS : 16 SS
Input pullup with interrupt - Button : 0
Piezo Speaker : 2
Analog Input - Potentiometer : ADC0



Wemos D1 Pinout
A0 - ADC0 - Potentiometer
TX - GPIO1 - TXD0
RX - GPIO3 - RXD0
D0 - GPIO16 - WAKE
D1 - GPIO5 - SCL - I2C
D2 - GPIO4 - SDA - I2C
D3 - GPIO0 - FLASH - Button
D4 - GPIO2 - GPS Software Rx
D5 - GPIO14 - SCLK - SPI LED
D6 - GPIO12 - MISO - Speaker
D7 - GPIO13 - MOSI - SPI LED
D8 - GPIO15 - CS/SS - SPI LED