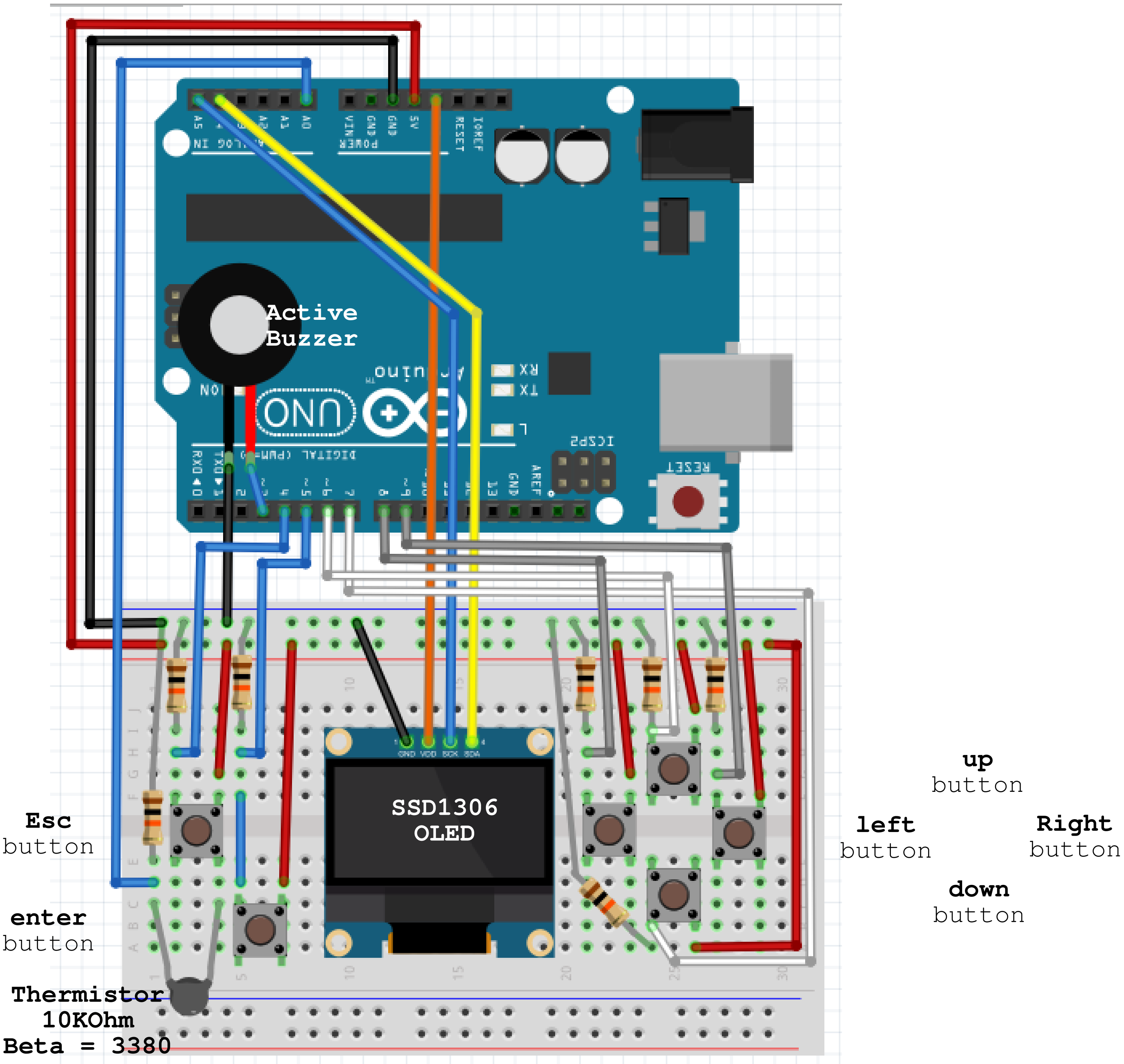
ATMEGA328/ARDUINO - PROJECT - DIGITAL ALARM CLOCK - OLED

https://github.com/teaksoon/p_daco



HARDWARE
1x Computer with Arduino IDE Software
1x USB 2.0 Type A/B Data Cable
1x Arduino Uno Board
1x Solderless Breadboard
Nx Jumper wires

1x Active Buzzer
6x Tactile Switch with 6x 10KOhm Resistor
1x SSD1306 OLED Module i2c 64x128 pixel

- Modular Design Extension -

1x 10Kohm Thermistor(Beta=3380) with 1x 10KOhm Resistor

SSD1306 OLED Module is an Output device where it can be used to show text and images from our PROGRAM. SSD1306 OLED Module comes in two interface design, SPI and i2c

The SPI interface - will have 6 pins on the device, GND, VCC, MOSI, MISO, CHIP-SELECT and CLOCK

The i2c interface - will have 4 pins on the device, GND, VCC, DATA and CLOCK

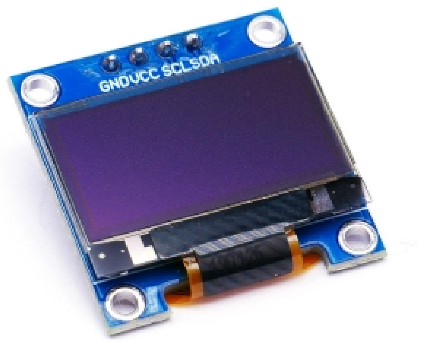
We will use the i2c interface, that is the only ones I have. Both should work the same, difference is just how to send data into the device

Normally people will just use the ARDUINO GRAPHICS Library like the ADAfruit OLED Library, u8g2 OLED or something else. Here however, we use or own coding, without those Libraries

So that we can learn exactly how this thing work and we can have better control over our devices

Do not worry about the Graphics Libraries, to get something displayed on the SSD1306 OLED Module screen is quite easy

All we need to do is to change the content of the GRAPHICS MEMORY in the SSD1306 OLED Module



For the 64x128(8192) pixel SSD1306 OLED module, it has 1024 bytes of MEMORY. We will just focus on this model from now onwards

1-BYTE = 8-BIT, so 1024 bytes is 1024x8 = 8192 BIT. Each BIT in the GRAPHICS MEMORY is a pixel(dot) on the OLED Screen

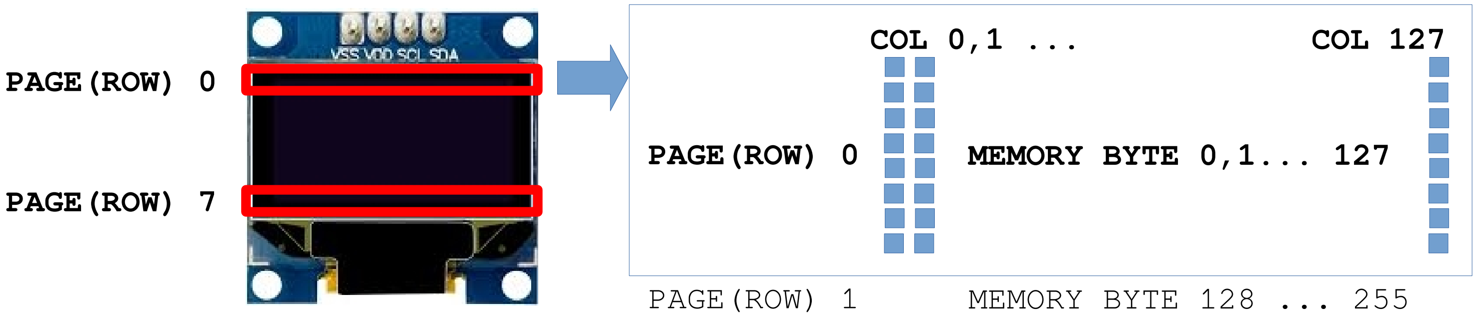
When a BIT = 1, the pixel(dot) for that BIT on the OLED Screen is turned ON
When a BIT = 0, the pixel(dot) for that BIT on the OLED Screen is turned OFF

In order to know which BIT to turn ON or OFF, we need to know where the BIT is located on the OLED Screen

1024-BYTES of GRAPHICS MEMORY are arranged in a sequence, starting BYTE 0 to BYTE 1023.

The OLED Screen is divided into 8-PAGE(ROW)

Each PAGE(ROW) is consists of 128-COL (128-BYTES of MEMORY)



PAGE(ROW) 0 , COL 0 to 127 = MEMORY BYTE 0 to 127 (first 128 BYTES)
PAGE(ROW) 1 , COL 0 to 127 = MEMORY BYTE 128 to 255 (next 128 BYTES)
PAGE(ROW) 2 , COL 0 to 127 = MEMORY BYTE 256 to 383 (next 128 BYTES)
PAGE(ROW) 3 , COL 0 to 127 = MEMORY BYTE 384 to 511 (next 128 BYTES)
PAGE(ROW) 4 , COL 0 to 127 = MEMORY BYTE 512 to 639 (next 128 BYTES)
PAGE(ROW) 5 , COL 0 to 127 = MEMORY BYTE 640 to 767 (next 128 BYTES)
PAGE(ROW) 6 , COL 0 to 127 = MEMORY BYTE 768 to 895 (next 128 BYTES)
PAGE(ROW) 7 , COL 0 to 127 = MEMORY BYTE 896 to 1023 (next 128 BYTES)

1 COL = 1 BYTE = 8 VERTICAL PIXEL(DOT)

We can only change one BYTE at a time, means the entire 8 VERTICAL PIXEL

ATMEGA328/ARDUINO – PROJECT – DIGITAL ALARM CLOCK – OLED
https://github.com/teaksoon/p_daco

Source code: **p_daco_oled_basic**
Download from:
https://github.com/teaksoon/p_daco/blob/main/2022_01_03_p_daco_source.zip
Upload PROGRAM, watch the OLED Screen

This PROGRAM will use the i2c (the ATMEGA328 micro-controller hardware twi interface) to change BYTE-0 of the OLED GRAPHICS MEMORY

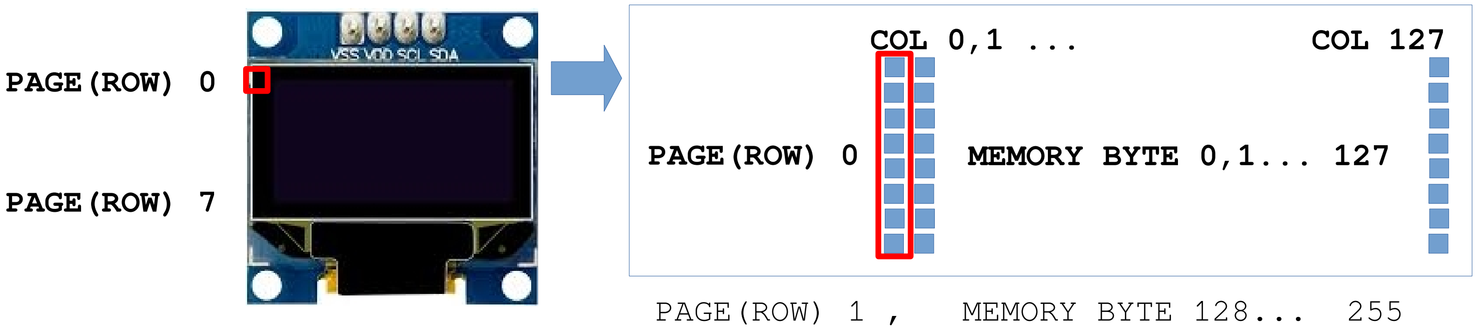
Example: We want to change the bits in BYTE-0

BYTE-0 has 8-BITS, it is located at ROW 0, COL 0 of the OLED Screen

0b10011111 (8-BIT) is binary representation of a BYTE

When the bits inside BYTE-0 is changed to 0b10011111, Pixels on ROW 0, COL 0 of the OLED Screen will be changed

You can try change update different BYTE to see where the OLED is being updated



PAGE(ROW)	1 ,	MEMORY BYTE 128...	255
PAGE(ROW)	2 ,	MEMORY BYTE 256...	383
PAGE(ROW)	3 ,	MEMORY BYTE 384...	511
PAGE(ROW)	4 ,	MEMORY BYTE 512...	639
PAGE(ROW)	5 ,	MEMORY BYTE 640...	767
PAGE(ROW)	6 ,	MEMORY BYTE 768...	895
PAGE(ROW)	7 ,	MEMORY BYTE 896...	1023

Source code: **p_daco_oled_bufffer**
Download from:
https://github.com/teaksoon/p_daco/blob/main/2022_01_03_p_daco_source.zip
Upload PROGRAM, watch the OLED Screen

The SSD1306 i2c OLED GRAPHICS MEMORY is a “write-only” MEMORY

When we do graphics, we need to read from the GRAPHICS MEMORY because we need to manipulate individual BITS. Since we cannot read from this “write-only” OLED GRAPHICS MEMORY, we will use a duplicate GRAPHICS MEMORY in our micro-controller SRAM (1024 byte buffer memory)

With a buffer memory in our micro-controller SRAM, we simply do all our work in our SRAM buffer memory. Once everything is completed, we simply send the entire SRAM buffer memory to replace the entire GRAPHICS MEMORY on the OLED (an entire frame 1024 byte to the OLED)

The entire Screen, frame is changed

This entire screen replacement is measured in “fps” or “frames per seconds” that is commonly used in videos, games or animations

Micro-controllers are normally capable of sending many frames into the OLED within 1 second, we will see animation. More “fps” is better because more “fps” means smoother animation