임베디드컴퓨팅

Embedded Computing (0009488)

OLED, I2C

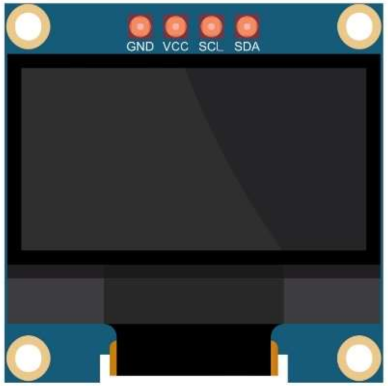
2022년 2학기

정보기술대학 정보통신공학과
김 영 필
ypkim@inu.ac.kr

인천대학교
INCHEON NATIONAL UNIVERSITY

OLED

- **Organic Light Emitting Diode (OLED)?**
  - a light-emitting diode that uses a film of organic compound and electric current to emit light by itself
- **Features**
  - High contrast ratio
    - A ratio of Luminance of the brightest parts and Luminance of darkest parts
  - Fast response time
    - How fast screen can change



128 x 64 I2C OLED

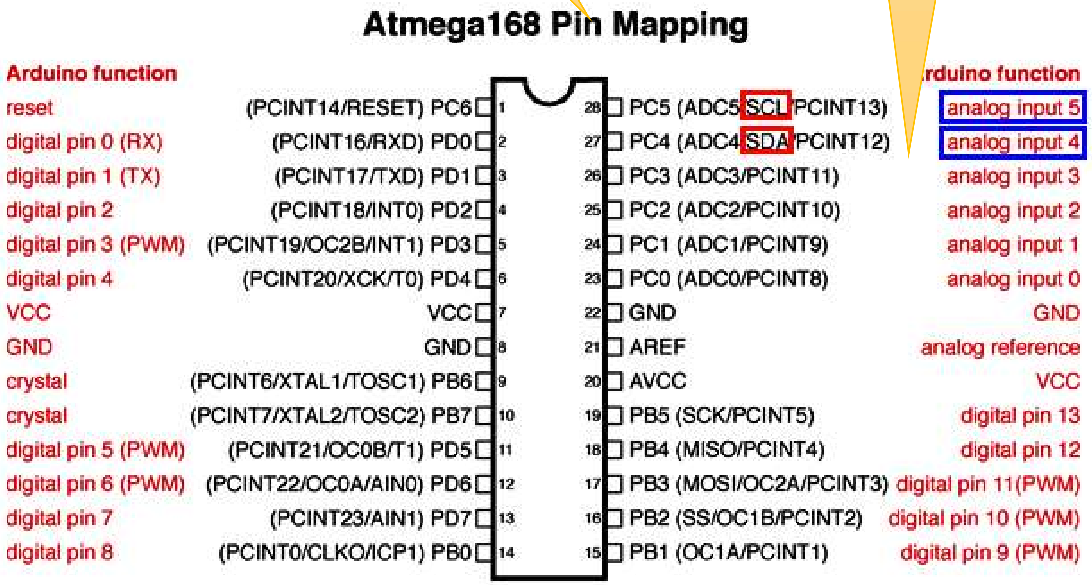
OLED pins

- Communication Interface
  - **I²C**
- **Four Pins**
  - **VCC:** A pin that applies + power.
    - Operating voltage = 1.65 ~ 3.3V, connect it to the **3.3V pin** on the Arduino.
  - **GND:** A grounding pin
    - Connected to the GND of the Arduino.
  - **SCL:** A pin that generates a clock signal
    - connected to **A5** of the Arduino.
  - **SDA:** A pin that transmits/receives data
    - connects to **A4** of Arduino.

Arduino A5
=
SCL for I2C

Arduino A4
=
SDA for I2C

Arduino Uno (Atmega168) Pin mapping



Atmega168 Pin Mapping

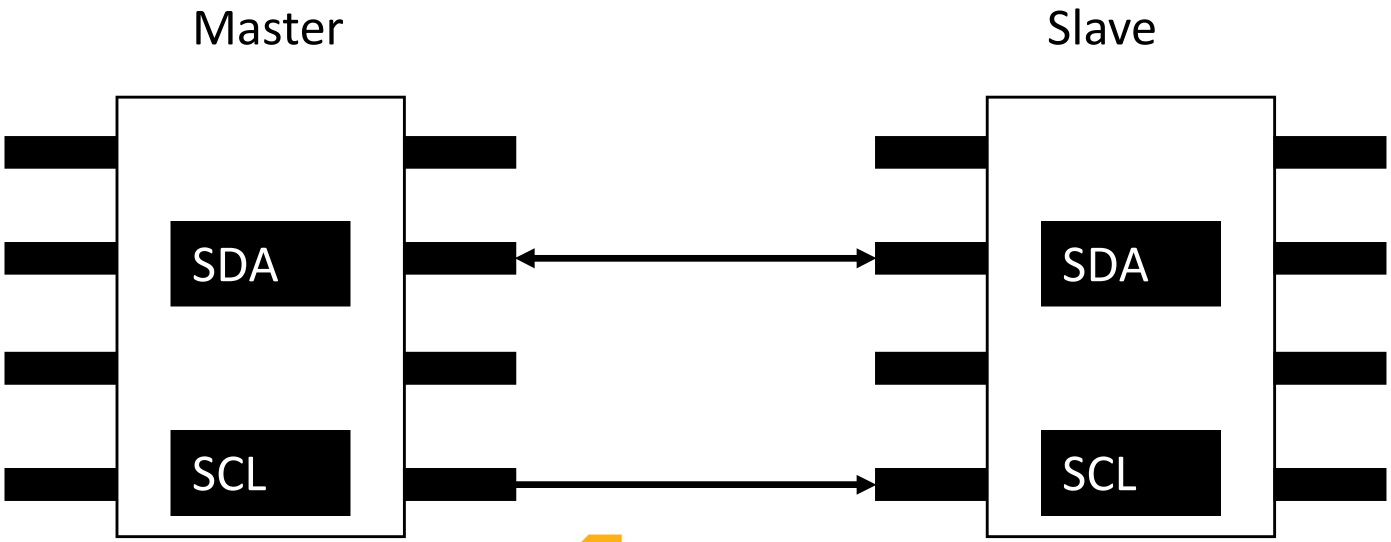
Arduino function					Arduino function
reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22	GND	GND
GND	GND	8	21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)

What is I^{2}C?

- Advanced Serial Communication
  - I2C or IIC
  - a synchronous, multi-controller/multi-target, packet switched, single-ended, serial communication bus
    - invented in 1982 by Philips Semiconductors.
  - Widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication.
- Can connect multiple slaves to a single master; or
- Can have multiple masters controlling single, or multiple slaves.
- Useful when you want to have more than one microcontroller logging data to a single memory card or displaying text to a single LCD.

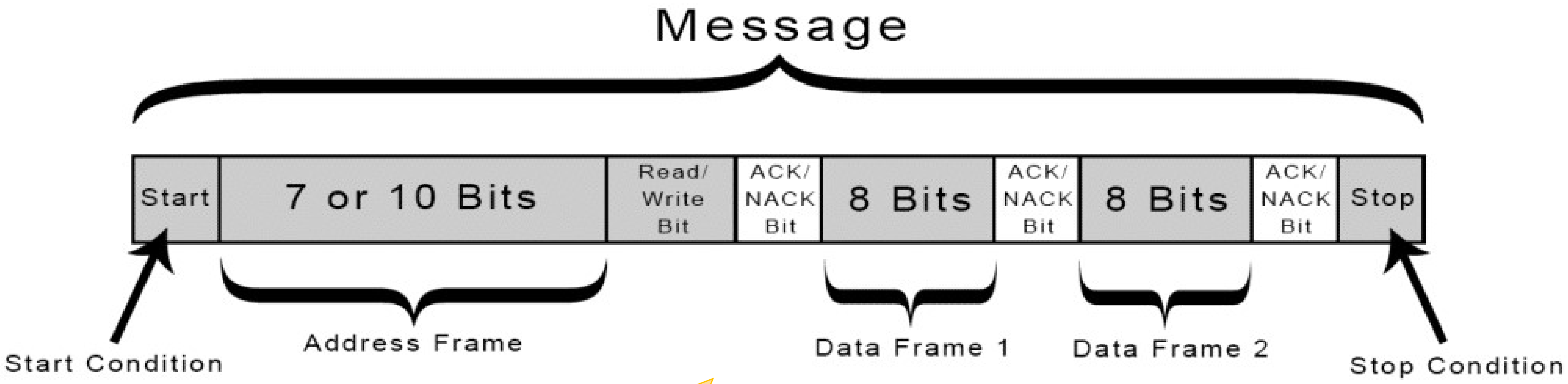
I^{2}C

- I2C uses two wires to transmit data between devices
  - SDA (Serial Data): The line for data transmission.
    - Data transferred bit by bit along SDA line
    - Max. speed 100kbps ~ 5Mbps. Max slaves = 1008
  - SCL Serial Clock: The line for clock signal.
    - Only master controls a signal clock



How I2C Works

- I2C transfer data in **messages**



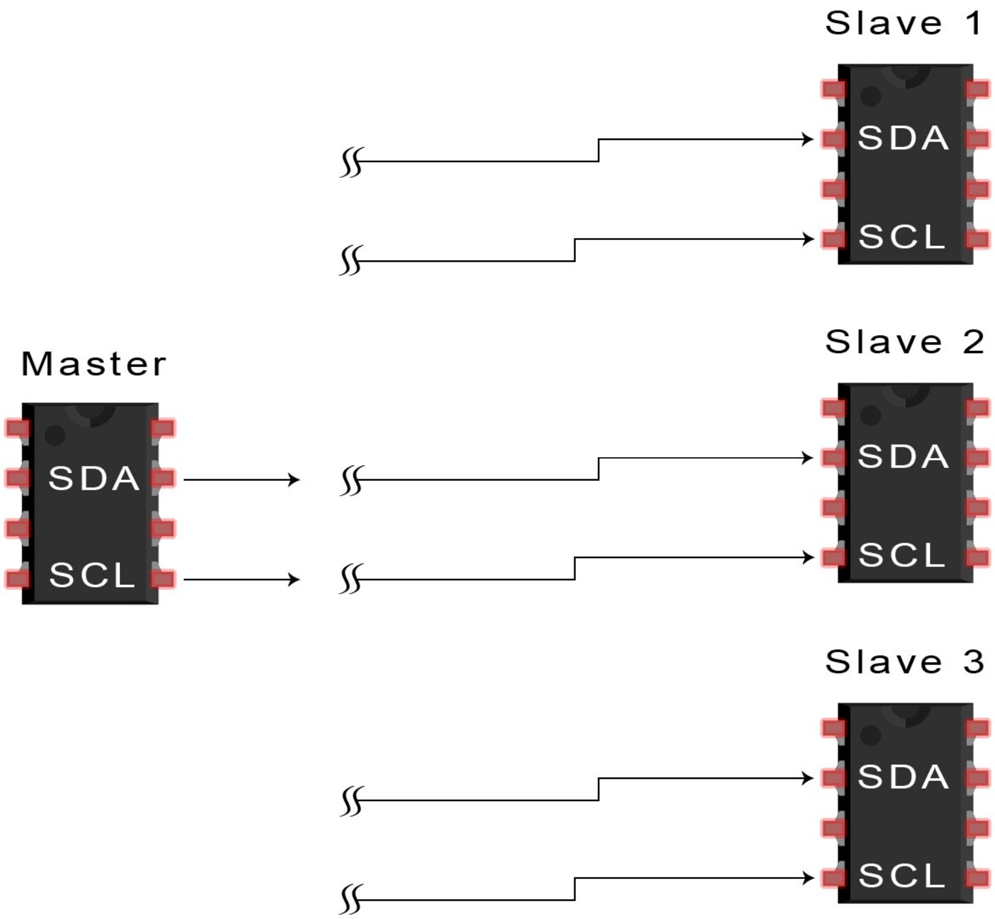
Ref- https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Start condition

Master –(SDA: H-L, SCL: H-L)→ Slaves

Switching SDA High->Low

Switching SCL High ->Low

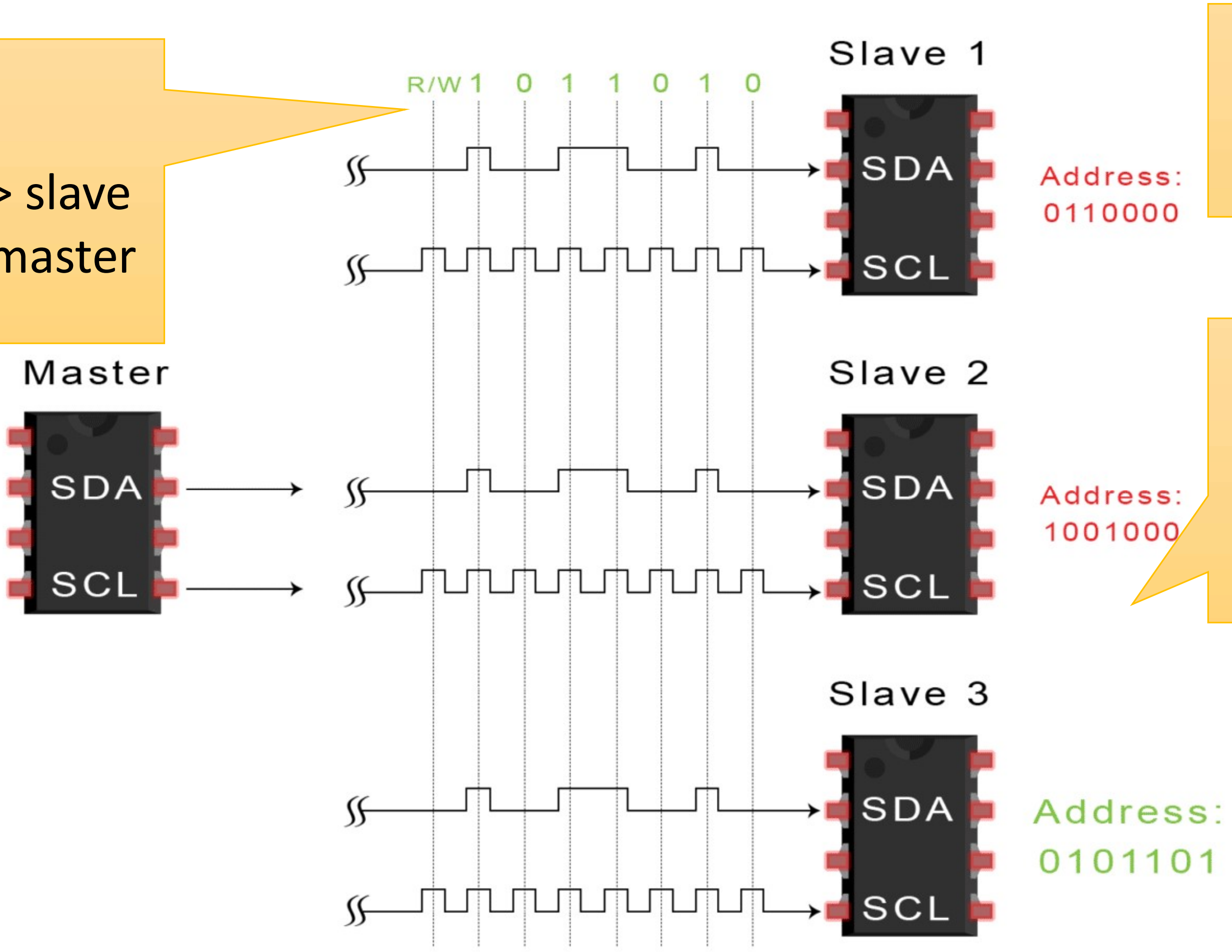


Ref- https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Address bits and R/W bit

Master –(SDA: 7 or 10 bits and R/W bit)→ Slaves

R/W bit:
LOW = master -> slave
HIGH = slave -> master



Address = identifier for slave

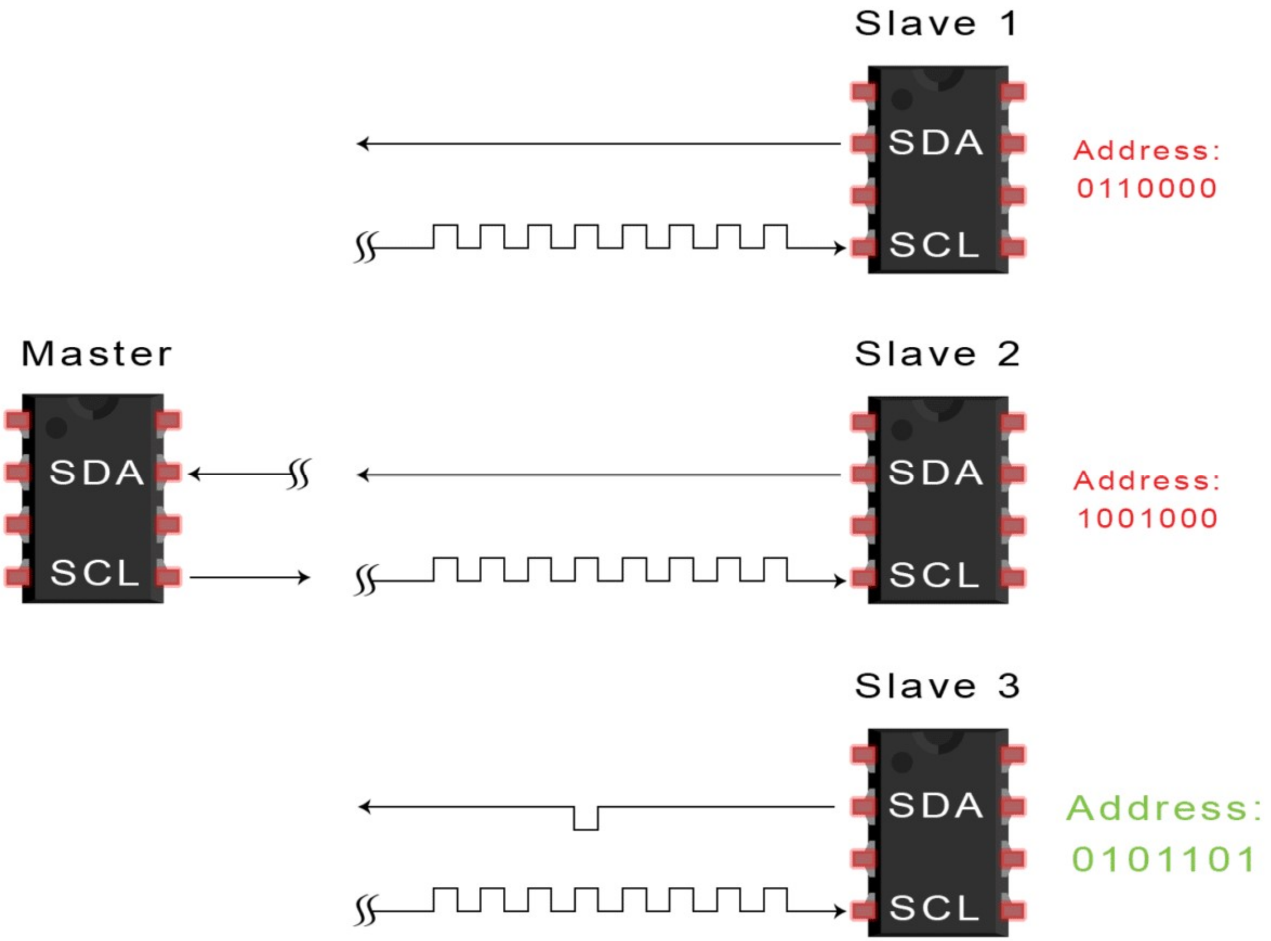
Each slave compares address frame with own address

Slave 3 is matched

Ref- https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Ack bit

Master ← (SDA: a LOW bit)– Selected Slave



Other slaves keep SDA: High

Slave 3 send an ack bit (SDA:Low)

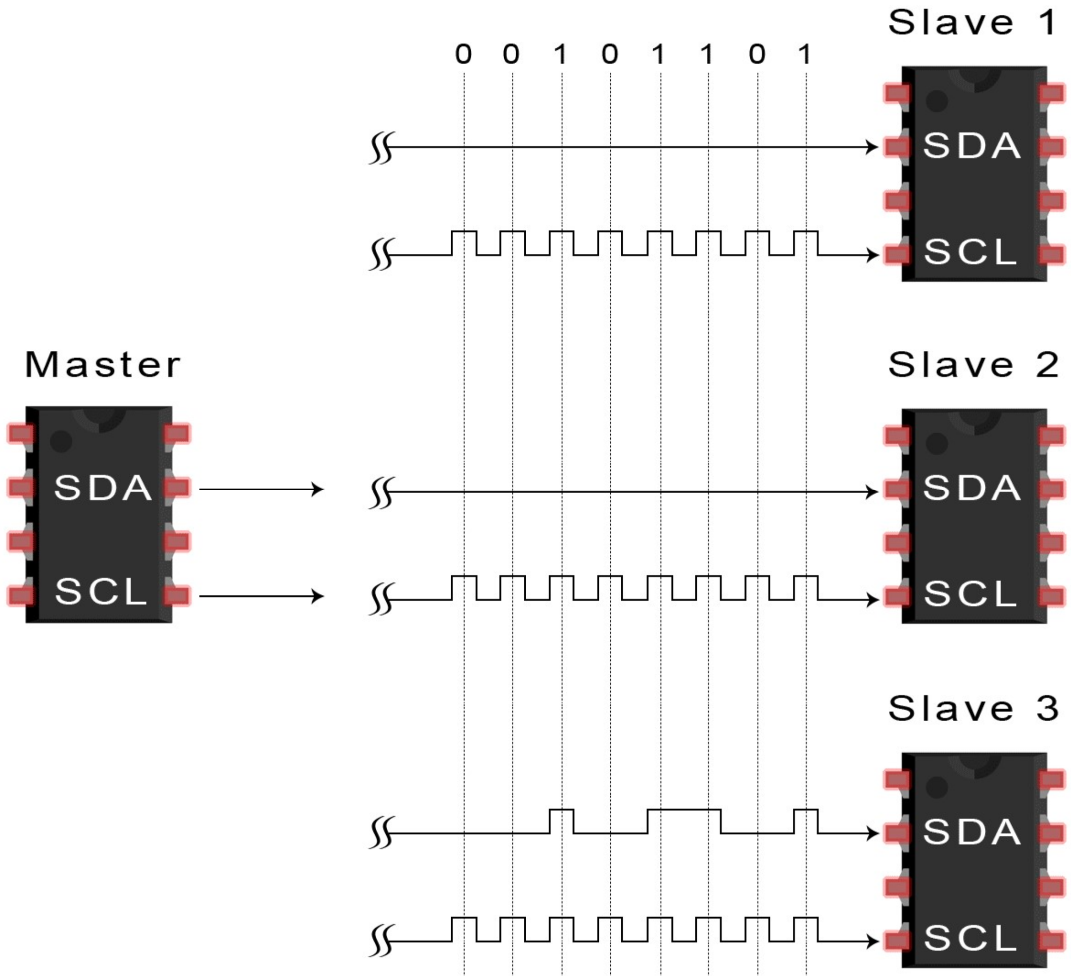
Ref- https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Send/Receive a data frame

Master – (SDA: data frame) → Selected Slave

R/W bit = LOW,

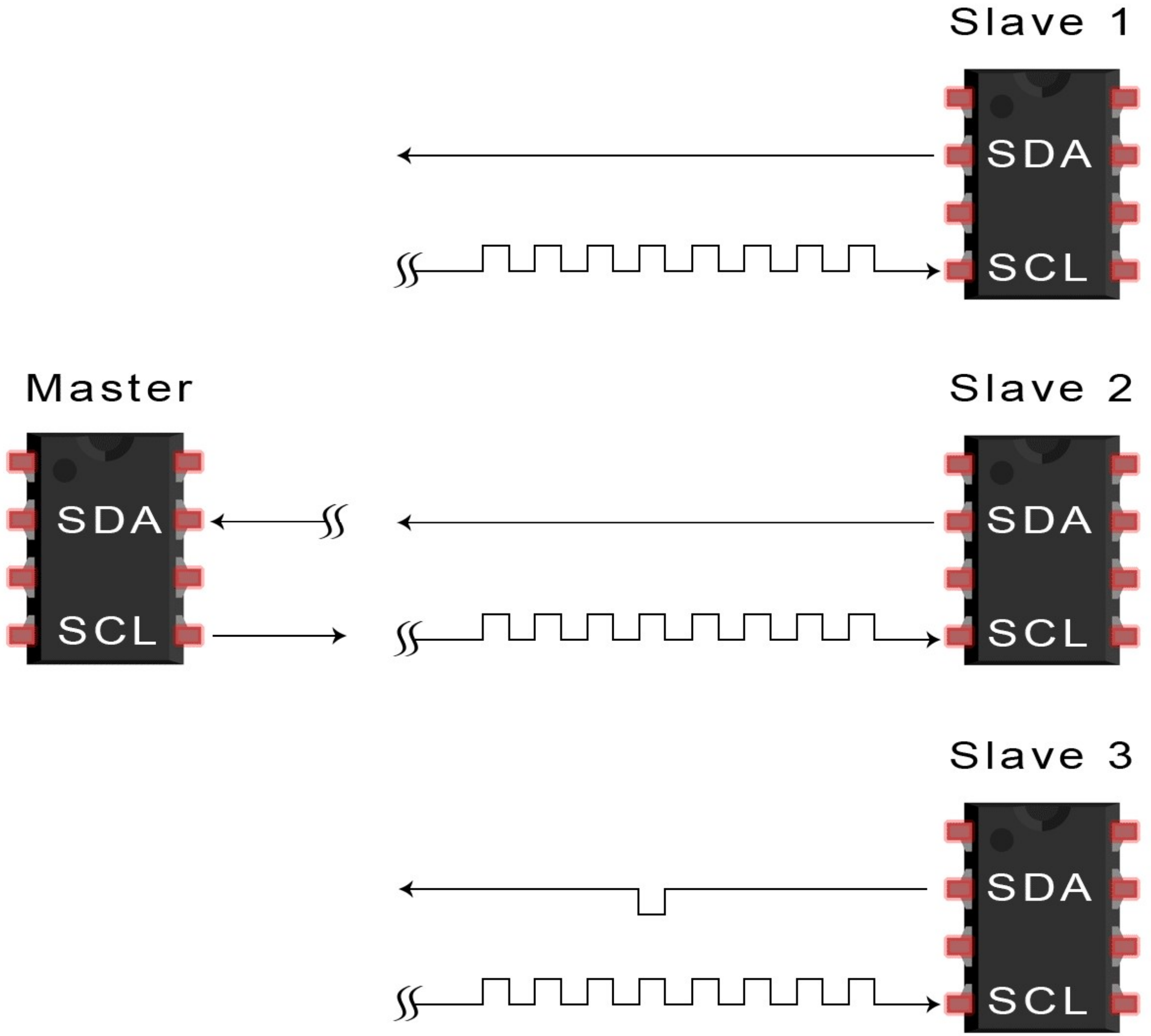
Master -> Slave



Ref- https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Ack bit for data frame

Master ← (SDA: a LOW bit)– Selected Slave



Slave 3 send an ack bit (SDA:Low)

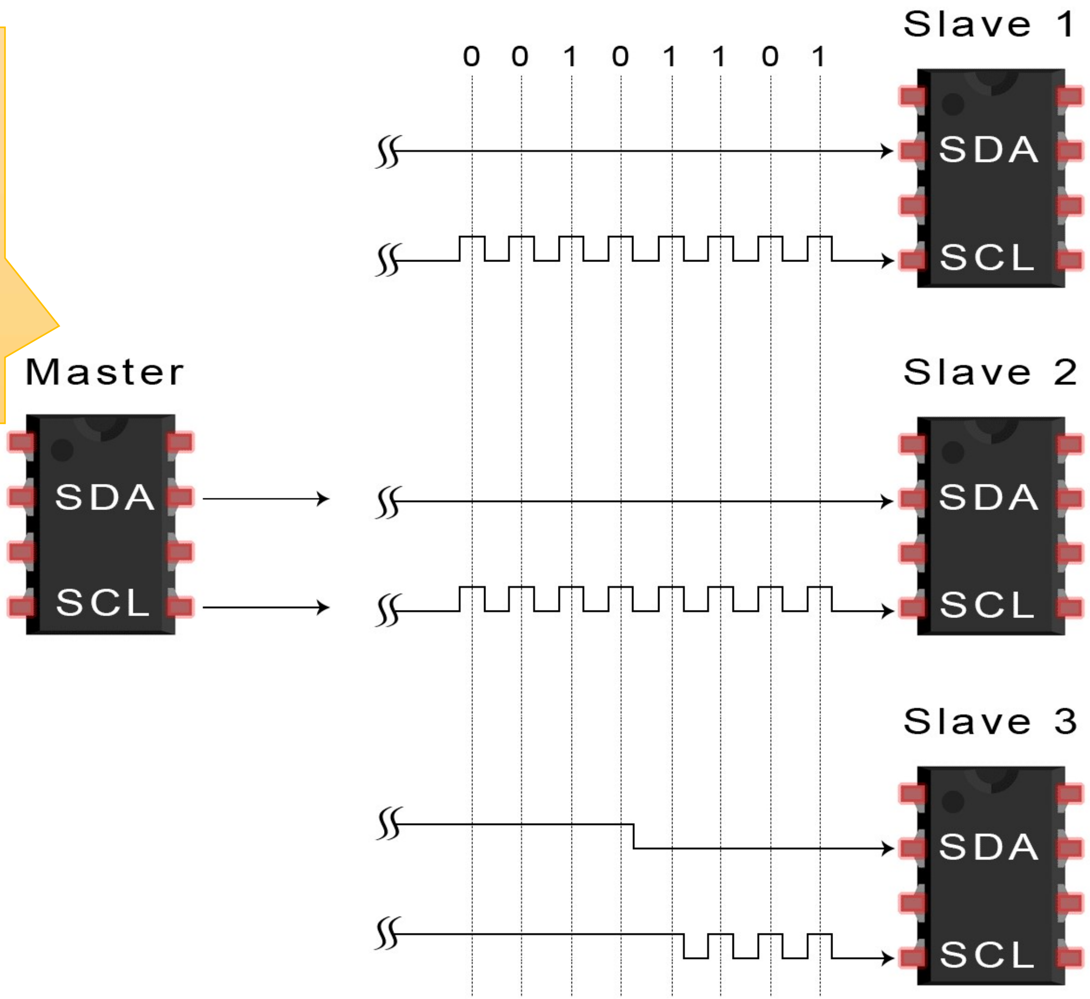
Ref- https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

Stop condition

Master –(SCL: L-H, SDA: L-H)→ Slaves

Switching SDA
Low -> High

Switching SCL
Low -> High



Ref- https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

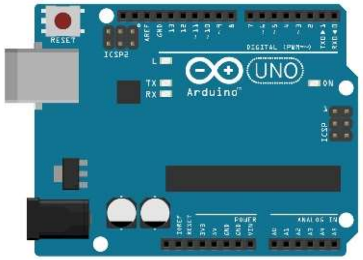
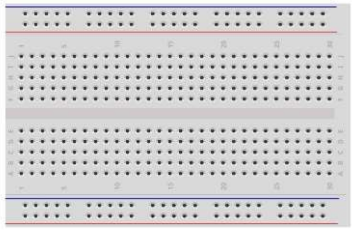
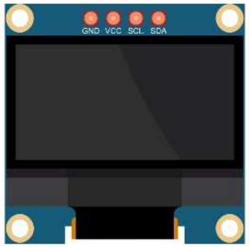
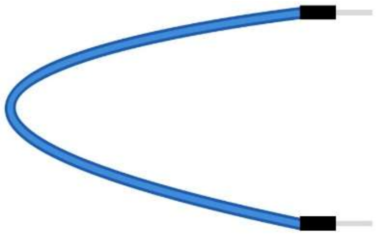
Library for OLED

- **Adafruit_SSD1306**
  - A library for Monochrome OLEDs based on SSD1306 drivers
  - https://github.com/adafruit/Adafruit_SSD1306
- **Adafruit Bus IO**
  - A helper libary to abstract away I2C & SPI transactions and registers
  - https://github.com/adafruit/Adafruit_BusIO
- **Adafruit_GFX_Library**
  - The core graphics library for displays, providing a common set of graphics primitives (points, lines, circles, etc.).
  - https://github.com/adafruit/Adafruit-GFX-Library
- To use a library in a sketch, select the aboves from [Sketch] > [Import Library].
  - If you see a warning about dependent libraries, install them, also.

Lab: Hello OLED, Hello Arduino

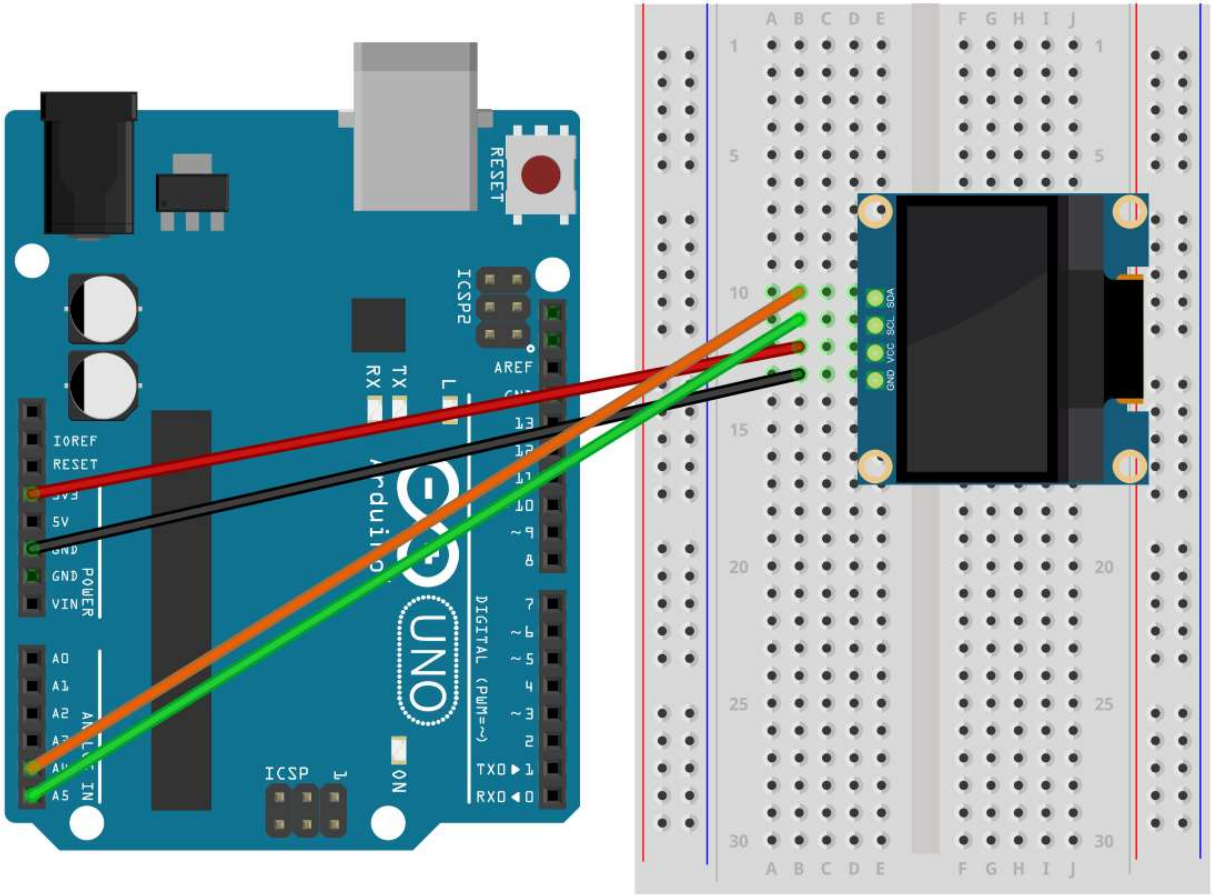
- Let’s write ‘Hello OLED, Hello Arduino’ on OLED display screen

- Required H/W components

Arduino Uno board	Breadboard	OLED (128 x 64 I2C)	Jumper cable (Male-Male)
			
x 1	x 1	x 1	x 4

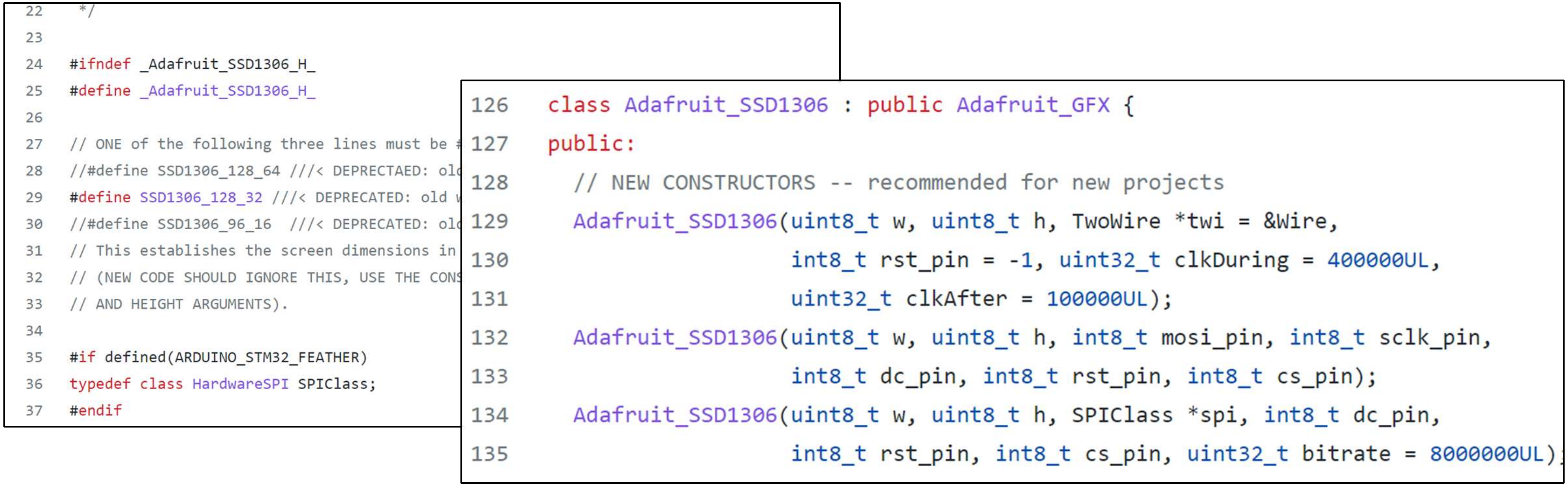
Circuit wiring setup

OLED	Arduino board
VCC	3.3V
GND	GND
SCL	analog A5
SDA	analog A4



Preliminary work for OLED size

- In older version (before v 1.2), you need to edit a header file “Adafruit_SSD1306.h” to specify your OLED display size
- In recent version (v 2.5 or later), you can setup display size using by a constructor



```
  */

#ifndef _Adafruit_SSD1306_H_
#define _Adafruit_SSD1306_H_

// ONE of the following three lines must be #
//#define SSD1306_128_64 ///< DEPRECTAED: ol
#define SSD1306_128_32 ///< DEPRECATED: old w
//#define SSD1306_96_16  ///< DEPRECATED: ol
// This establishes the screen dimensions in
// (NEW CODE SHOULD IGNORE THIS, USE THE CONS
// AND HEIGHT ARGUMENTS).

#if defined(ARDUINO_STM32_FEATHER)
typedef class HardwareSPI SPIClass;
#endif
```

```
class Adafruit_SSD1306 : public Adafruit_GFX {
public:
  // NEW CONSTRUCTORS -- recommended for new projects
  Adafruit_SSD1306(uint8_t w, uint8_t h, TwoWire *twi = &Wire,
                   int8_t rst_pin = -1, uint32_t clkDuring = 400000UL,
                   uint32_t clkAfter = 100000UL);
  Adafruit_SSD1306(uint8_t w, uint8_t h, int8_t mosi_pin, int8_t sclk_pin,
                   int8_t dc_pin, int8_t rst_pin, int8_t cs_pin);
  Adafruit_SSD1306(uint8_t w, uint8_t h, SPIClass *spi, int8_t dc_pin,
                   int8_t rst_pin, int8_t cs_pin, uint32_t bitrate = 8000000UL);
```

Basic setup for Hello OLED

```
#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>


#define SCREEN_WIDTH  128

#define SCREEN_HEIGHT 64

#define SCREEN_ADDRESS  0x3C


Adafruit_SSD1306 display(SCREEN_WIDTH, SSCREEN_HEIGHT)
```

Use library ➔ Include library header files

Macro for functionality extension

SSD1306 object initialization

Basic setup for Hello OLED

```
void setup() {
  display.begin(SSD1306_SWITCHCAPVCC,
SCREEN_ADDRESS);
  display.display();
  delay(2000);
}
void loop() {
}
```

Setup operation voltage 3.3V
Setup address of frame buffer
(our device has 0x3C)

IMPORTANT:
we need **display()**
to flush the buffer data into screen

Delay time is required if you change the screen later

loop do nothing in this example

Basic setup for Hello OLED

```
void setup() {
display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
  display.display();
  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.println("Hello OLED");
  display.display();
  delay(2000);
  display.println("Hello Arduino");
  display.display();
  delay(2000);
  display.println("Hi, Prof. Kim!");
  display.display();
  delay(2000);
}
void loop() {
}
```

use clearDisplay(); to get a blank screen

use **setTextColor(color)** to set various text color;

Unfortunately, our OLED is a monochrome display (WHITE or BLACK)

println(str) works!

don’t forget **display()** to update your changes on screen

Lab: Stylish text

- Let’s write ‘Hello OLED, Hello Arduino’ **with various text styles** on OLED display screen

- Change text color
- Change text size
- Change text position
- Scroll texts

Basic setup for Stylish text

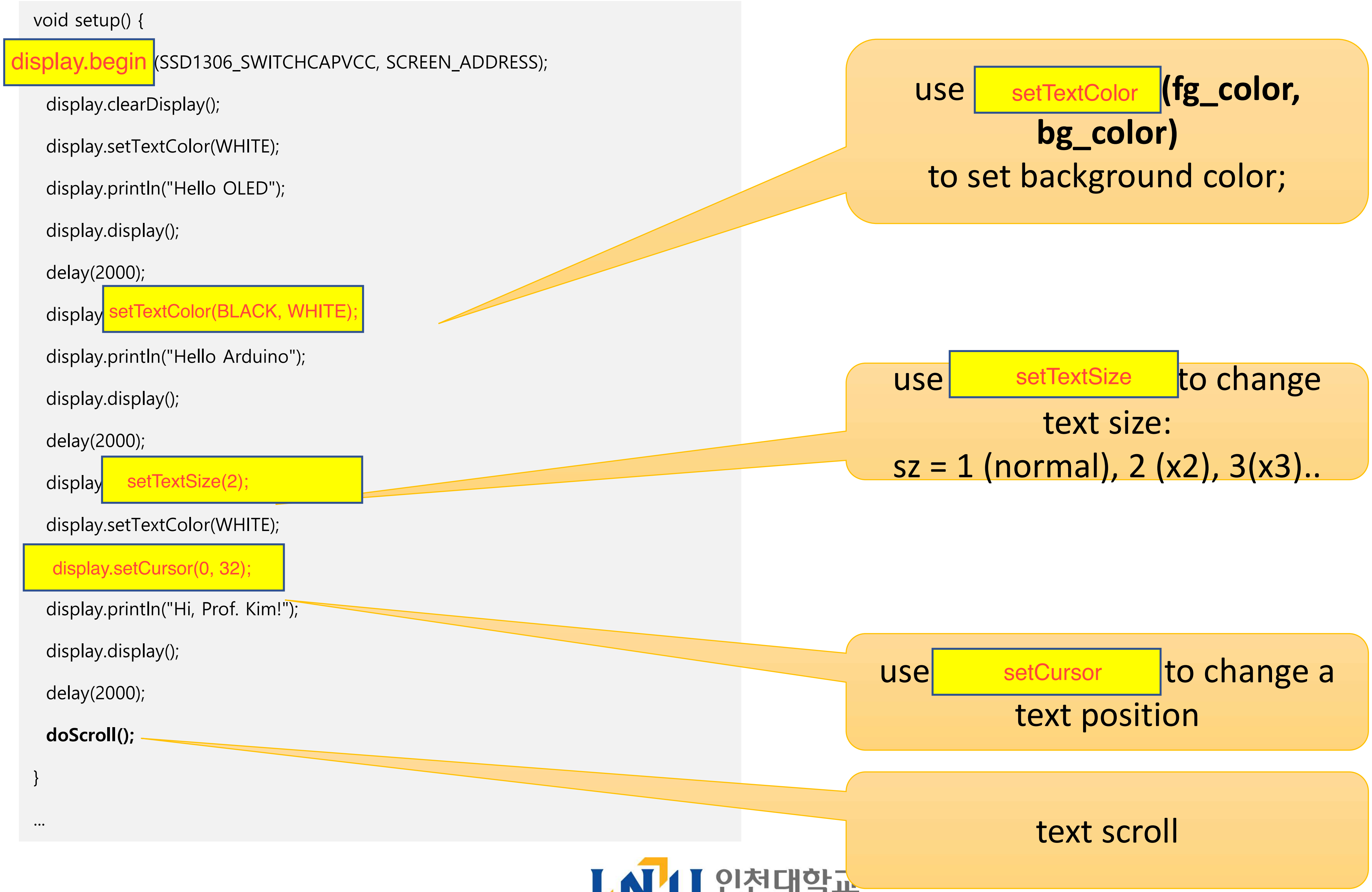
```
#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>


#define SCREEN_WIDTH   128

#define SCREEN_HEIGHT 64

#define SCREEN_ADDRESS   0x3C


Adafruit_SSD1306 display( SCREEN_WIDTH, SCREEN_HEIGHT
```

Use library ➔ Include library header files

Macro for functionality extension

SSD1306 object initialization

Basic setup for Stylish text



```
void setup() {
display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.println("Hello OLED");
  display.display();
  delay(2000);
  display setTextColor(BLACK, WHITE);
  display.println("Hello Arduino");
  display.display();
  delay(2000);
  display setTextSize(2);
  display.setTextColor(WHITE);
 display.setCursor(0, 32);
  display.println("Hi, Prof. Kim!");
  display.display();
  delay(2000);
  doScroll();
}
...
```

use setTextColor(fg_color, bg_color) to set background color;

use setTextSize to change text size: sz = 1 (normal), 2 (x2), 3(x3)..

use setCursor to change a text position

text scroll

Scroll function for Stylish text

```
void doScroll() {
  display.startscrollright(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrolldiagright(0x00, 0x07);
  delay(2000);
  display.startscrolldiagleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
}
…
```

use **startscrollright(start, stop)**
to start scroll the screen
to **right** direction

to scroll whole display:
0x00, 0x0F

use **stopscroll()**
to stop scroll the screen;

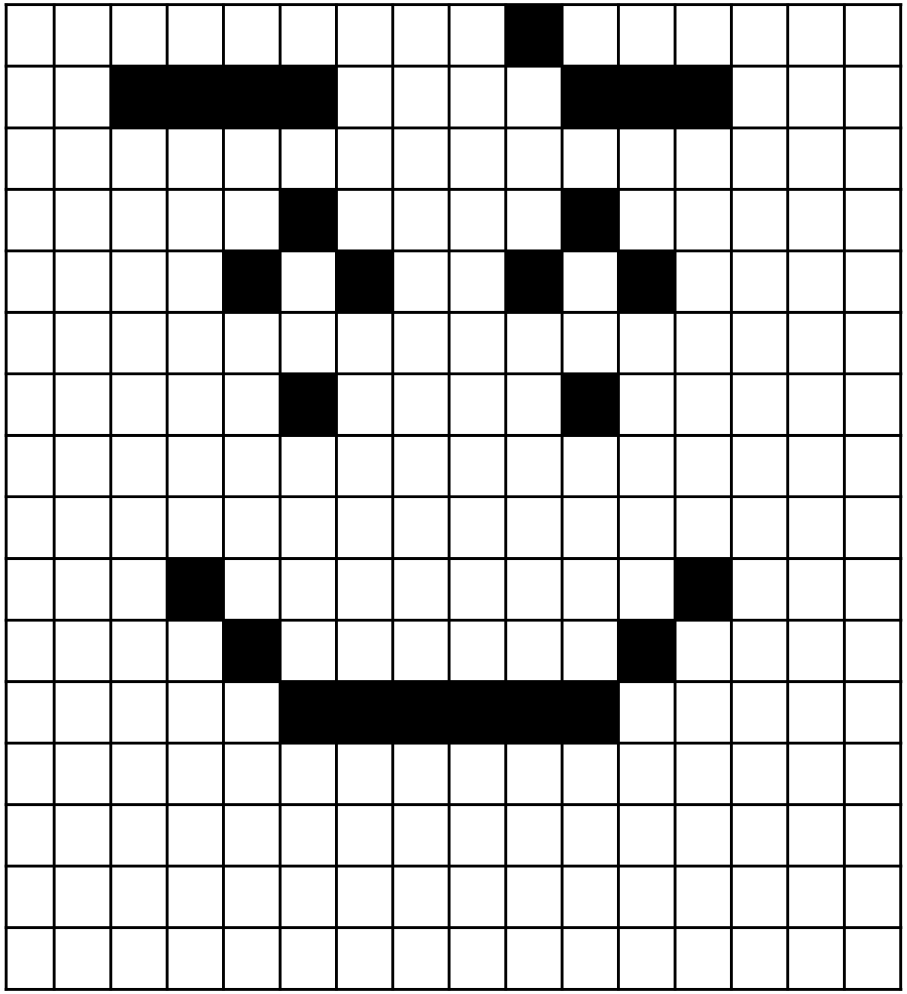
use **startscrollleft(start, stop)**
to start scroll the screen
to **left** direction

use **startscrolldiagright/left (start, stop)**
to start scroll the screen
to **diag-right or left** direction

Lab: drawBitmap

- Let’s draw our own bitmap image on OLED screen!

- Adafruit GFX library provides drawBitmap() function
    - *display.drawBitmap(x, y, bitmap, width, height, color);*
    - *x, y = x, y positions*
    - *bitmap = a ptr to a buffer for bitmap*
    - *width, height = a bitmap size*
    - *color = WHITE (1) or BLACK (0)*



Basic setup for drawBitmap

```
#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
static const unsigned char PROGMEM logo_bmp[] =
{0b00000000, 0b01000000,
0b00111100, 0b00111000,
0b00000000, 0b00000000,
0b00000100, 0b00100000,
0b00001010, 0b01010000,
0b00000000, 0b00000000,
0b00000100, 0b00100000,
0b00000000, 0b00000000,
0b00000000, 0b00000000,
0b00010000, 0b00001000,
0b00001000, 0b00010000,
0b00000111, 0b11100000,
0b00000000, 0b00000000,
0b00000000, 0b00000000,
0b00000000, 0b00000000,
0b00000000, 0b00000000};
```

16 x 16 size bitmap

PROGMEM :
Read large const data from flash memory; instead of SRAM

Usage:

const <datatype> <var_name>[] PROGMEM ={};
or
const PROGMEM <datatype> <var_name>[] ={};
or
const <datatype> PROGMEM <var_name>[] ={};

Function for drawBitmap

```
void drawMybitmap(void) {
  display.clearDisplay();
  display.drawBitmap(
    (display.width()  - LOGO_WIDTH ) / 2,
    (display.height() - LOGO_HEIGHT) / 2,
    logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, 1);

  delay(1000);
}
```

Calculate x, y position of center point
Then, set them to the bitmap start position

don’t forget

Assignment: Smart Namecard

- Requirements
  - Based on the today examples, write a sketch program as follows.
    - Display your student number, your full name, your city, your feelings on OLED with various style
      - Different text size, inversed text, different text position.. Use each at least once.
    - Decorate your namecard using your own 16x16 bitmap image (your logo or favorite icon or image or Korean name etc.)
    - Scroll them!
      - Right, Left, DiagLeft or DiagRight, .. Use each at least once.
  - Write block-type comments in the top of your source code, which includes "your student no., your name, writing date, what you feel about this assignment, etc."
- Results
  - (a source code file) sketch source code (***"sketchfilename.ino"***)
  - (a Arduino board capture file) a photo capture showing how you setup your circuit (max. 1GB file).